A SOA governance pattern based approach with change impact analysis for maintenance of service oriented applications

By NGUYEN, Huong Lan

A thesis submitted to
School of Information Science,
Japan Advanced Institute of Science and Technology,
in partial fulfillment of the requirements
for the degree of
Master of Information Science
Graduate Program in Information Science

Written under the direction of
Associate Professor Masato Suzuki

September, 2012

A SOA governance pattern based approach with change impact analysis for maintenance of service oriented applications

By NGUYEN, Huong Lan (1010221)

A thesis submitted to
School of Information Science,
Japan Advanced Institute of Science and Technology,
in partial fulfillment of the requirements
for the degree of
Master of Information Science
Graduate Program in Information Science

Written under the direction of
Associate Professor Masato Suzuki

and approved by
Associate Professor Masato Suzuki
Associate Professor Toshiaki Aoki
Associate Professor Kazuhiro Ogata

August, 2012 (Submitted)

Acknowledgement

First and foremost I would like to express my sincerest gratitude to my supervisor, Associate Professor Masato Suzuki, who has supported me throughout my study with his knowledge, guidance, useful criticism and encouragement. Without his consistent help this thesis would not have been completed or written.

I wish to say grateful thanks to Professor Koichiro Ochimizu, Associate Professor Toshiaki Aoki and Associate Professor Kazuhiro Ogata for their useful comments for this dissertation and on my defense day.

I would like to thanks to Ministry of Education and Training for their financial support for my study time in Japan.

I would like to show my sincere appreciation to Japan Advanced Institute of Science and Technology for supporting me during all the time I study here.I also want to thank all members in Software Engineering Laboratory for their kindness and friendliness.

Last but not least, I would like to thank my family who always courage, love and support me to complete my degree.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Software maintenance has been recognized as the most challenge and costly phase in the software life cycle. Software systems always change over time not only for error correction, performance improvement but also new requirements to remain the competitive strategies among organizations.

A software system is generally designed as a set of logically related components. When a component changes, it might affect to other components in the system. Therefore, changes to requirements need to be captured, controlled to ensure the evolution of the system.

Software systems are usually large and complex. In traditional software development architectures, software systems are generally developed as a set of logically related objects or components. When a component changes, it might affect to other components in the system. So, we need a method or process to specify what effects by a proposed change. This process is called change impact analysis.

Change impact analysis determines the consequences or effects on the system if a change is made. Change impact analysis can be used during software maintenance to keep the system at a high level of quality and reduce the cost to maintain the system.

When building a new system, enterprises always want to reuse existing components rather than build a wrapper. This is one of the most reason why service oriented architecture is gaining much attention and can be widely accepted nowadays. SOA support a methodology to design and develop applications based on Web services. It enhances the agility and cost-effectiveness for enterprises.

In service architecture (SOA), a service is a unit of solution logic to perform a business task or process. Unit of logic can be an object or a component. Business applications are usually built by aggregating a set of services to automate a business process within the organization. When a business process changes, it might affect to its corresponding services and versa. Therefore we need methods to specify the impact of a proposed change.

1.1 Problem

In Service oriented architecture, business applications encode various business processes within an organization [9]. Business processes are a set of logically related activities that are performed to achieve business objects. Business processes are automated by implementing services. Each service has many capabilities or operations. Due to compete strategies, organizations tend to re-engineer their business processes. When a business process is changed, such as adding a task, or removing a task, the services that implement the changed task need to be updated in order to support the new business requirements. However, determining the impact of a business process change is not trivial since business analysts might not understand the whole system and the relation between services in the system are very complex. This is one of the cause of expensive cost for maintenance. Also, with an unmanaged change,software systems can run appropriately or even deterioration.

1.2 Objective

In order to overcome above problems, we introduce a new approach using change impact analysis on design layers and applying some SOA patterns.

Though change impact analysis are mainly used at the business layer and source code layer., we extend them at the design layer for modeling services and business processes and applying some SOA patterns to analyse the impacts for maintenance service oriented applications.

1.3 Dissertation organization

In chapter 2, we will discuss about the background knowledge using in this research. In this chapter, we will deal with some basic concepts related to service oriented architecture, services, business processes, SOA layers and the relationship between services and business processes. We will present about the software maintenance process, especially in services oriented applications. The last content in this chapter are some backgrounds about change impact analysis, change process and change impact analysis process.

In chapter 3, we will present a model for services and business processes. In this chapter, we will characterize some kind of changes for services. Finally, we will present some SOA patterns used for change impact analysis.
The next chapter, chapter 4, we will apply our approach to analysis an example of actual application 'Virtual Travel Agency'.We will introduce services of the travel agency and then applying some requirements change to change impact analysis. Chapter 5 is the discussion about the conclusion and the future works.

Chapter 2

Background Concepts

2.1 Service Oriented Concepts - SOA

2.1.1 Service Oriented Architecture (SOA)

According to Thomas Erl in [8], service-oriented architecture represents an architectural model that aims to enhance the agility and cost-effective of an enterprise while reducing the burden of IT on the over all organization. It accomplishes this by positioning services as the primary means to develop the system.

2.1.2 Services

From a business prospective, services are IT assets that correspond to real-world business activities. Nowadays, most organizations provides services to customers, clients, employee or business partners. Here are some examples:

- Travel Agency: Airlines reservation, hotel reservation, annual summary of business travel expenditures.
- Bank: Saving accounts, checking accounts, payment account, loan, mortgages, credit verification.
- Insurance Agency: Health insurance, car insurance, accident insurance.

These services are generally implemented by the IT systems. Customers or business parties can access these services according to services policies. The service policies define who can access the service, when the service is available to use, the cost of using services, security levels for the service and so on.

From a technical perspective, services are coarse-grained, reusable software program. Services are exposed to interfaces from the viewpoint of customers.

Each service is assigned its own distinct functional context and consist of one or more operations. So, a service can be considered as a container of operations associated with a specified purpose. Figure 2.1 shows the service with its operations.

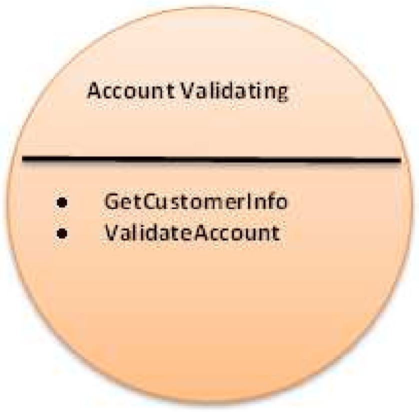


Figure 2.1: An example of a service

Service Models
Depending on the type of logic which they encapsulate, the reuse potential and the logic domains within enterprise, services are classified into two types:

- **Business Service Model**: A business service automates a parent business task or process that the organization deliver to its customers or business partners. This type of service tend to be specific to a particular service domain such as finance, sales, shipment and so on. Therefore, it has less reuse potential and is generally positioned as the controller of a service composition.

- **Utility Service Model**: This type of service stands for any generic service or service agent designed for reuse potential. So utility services are completely generic and non-application specific in nature, such as data accessing, logging and identity management.

Service Composition
A service composition is an aggregate of services together to automate a particular task or business process. Service composition allows us to built new applications and processes using existing services from the service inventory. Figure 2.2 shows an example of a service composition. We assume that company A already has two services:

- Customer Information Service: Gather customer information(GetCustomerInfor) and change customer information (ChangeCustomerInfor).

- Account Validating: Validate account information of customers (ValidateAccount operation).

Now the company is planning to built a new service which creates new accounts for customers. The process for opening a new account is as belows:

- Gather customer and account information.

- Validate customer and account information.
- If the information is valid, open a new account. If not, decline the customer's request.

The new service will invoke these two existing services without building new functionalities.

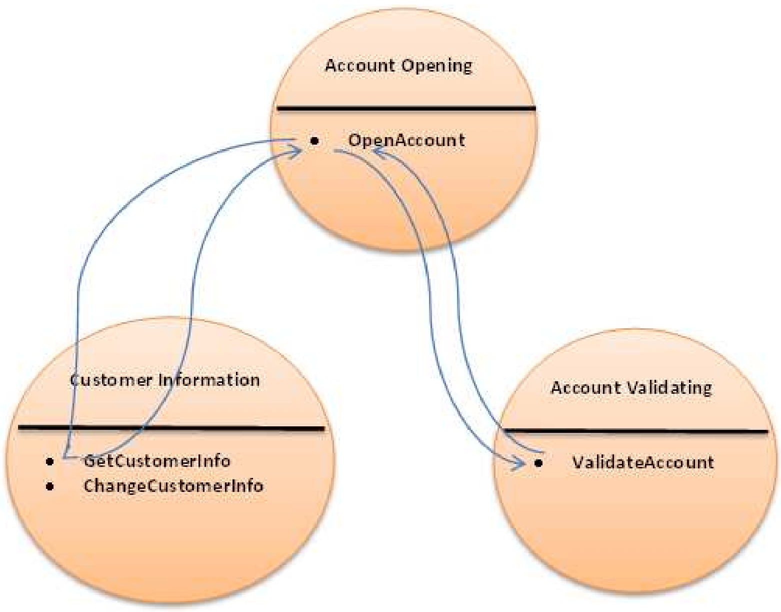


Figure 2.2: An example of Service Composition

Service Consumer
When a program invokes and interacts with a service it is called as a service consumer. Programs can be a desktop application, a web portal or even other services. For example, we can create a desktop application that is capable of exchanging data with a service. When this application invokes the service, it is considered as a service consumer.

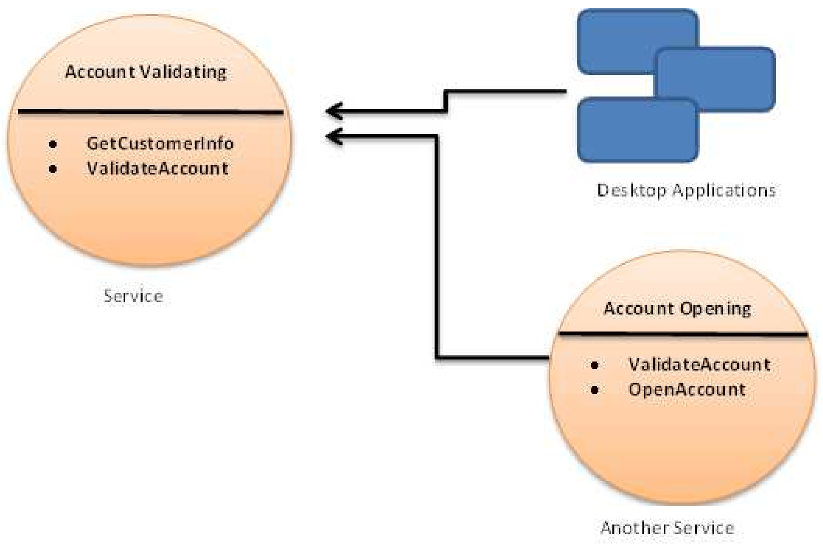


Figure 2.3: An example of Service Consumer

SOA and Web service
As we mentioned, SOA accomplishes logic solutions by using services. One of the most important things in adopting SOA is that it is neutral to any technology platforms. This characteristic allows enterprises leveraging technology advancements. In SOA, a service can be built and implemented as a component or a web service. In this research, we consider services as Web services.

2.1.3 Business Processes

A business process is a set of related logically tasks that are performed in the appropriate sequence.
A business process include:

- Tasks that perform a specific function.
- The links between the tasks that determine the order in which tasks can be performed.
- The data that is passed between tasks.

Let's take an example about a company provide a service which allows customers opening a new account. The business process for this service is showed in the figure 2.4). The steps in the business process are:

Step 1. Receive account information
In this step, the account information of the customer can be gathered by many methods, such as:

- Customer enters data via a Website.
- Customer sends an email or make a call to a a clerk of the company.
- Customer has an account, so the customer information is recorded in a database.

Step 2. Validate account information
This step is used for validating the account information of the customer that is gathered in the previous step. If the account information is valid according to the business rules of the company, step 3a is performed.

Step 3a. Open account
In this step, a new account is created for the customer. Besides, some task might be performed such as update customer information system, update order management system and so on.

Step 3b. Decline the customer's request If the account information does not satisfy all the business rules for opening a new account, then decline the customer's request to open a new account.

Step 4. Send confirmation
After the customer account has been successfully opened, a confirmation is sent to the customer along with details of his or her new account.

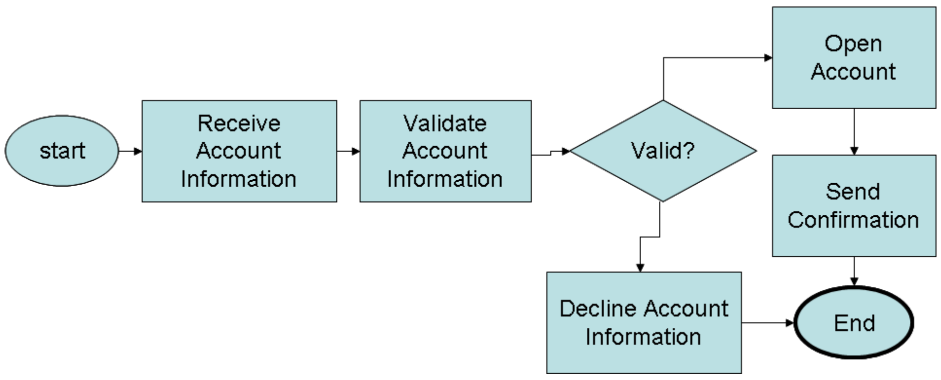

start
Receive Account Information
Validate Account Information
Valid?
Open Account
Send Confirmation
Decline Account Information
End

Figure 2.4: An example of a business process

2.1.4 SOA Layers

Figure 2.5 shows the layers of SOA [18]

- GUI- Graphical user interface is used by customers, such as Web portal or desktop applications.
- Business processes that coordinate tasks of the organizations in a predefined sequence.

- Services that model and define individual tasks of business processes in a reusable and technology-neutral manner.
- Components that are used to implement services.
- Business data can be a data stores, data warehouses or existing legacy applications.

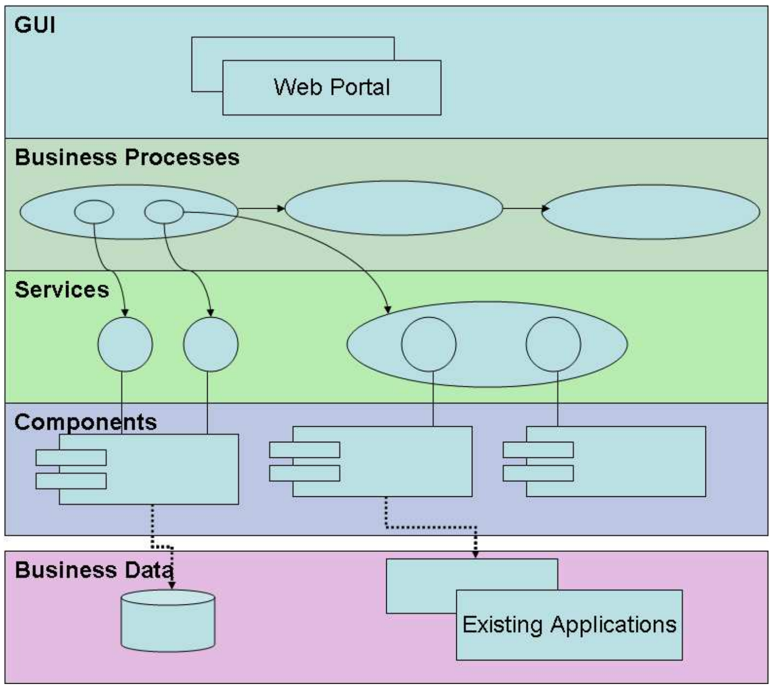


Figure 2.5: SOA layers

2.2 Software Maintenance

Software maintenance is a phase in software life cycle. Software maintenance is the modification of a software product after the deployment of that product. Software maintenance purpose is correcting failures, improving the performance or adapting new requirements from customers. The IEEE definition is as below:

> "Software maintenance is the process of modifying a software system or component after delivery to correct faults, improve performances or other attributes, or adapt to a changed environment."

Software maintenance can be classified into three categories: Corrective maintenance, adaptive maintenance and perfective maintenance [3].

- Corrective maintenance: The maintenance involves the failure repairs in the software, after software is deployed to the customer.

- Adaptive maintenance: The maintenance involves modifications in data and processing environments or new requirements from customers.
- Perfective maintenance: The maintenance is performed to eliminate processing inefficiencies, enhance performance or improve maintainability.

Software maintenance has recognized as the most costly, difficult and time-consuming phase in the software life cycle. The effort in which enterprises have to pay for software maintenance more than 50 percent of the total life cycle.

Unlike many other types of products, software products are progressive change. Even though software neither deteriorates nor changes with age if its media are well-presented, software maintenance is an expensive process where an existing program is modified for a variety of reasons, including correcting errors, adapting to different data or processing environments, enhancing to add functionality, and altering to improve efficiency.

For programs with many interacting modules, modifying and then revalidating a program is complex: analysis, testing, and debugging may be required for each module individually and for the interactions among modules. The problem is further compounded because the maintainers are rarely the authors of the code and usually lack a complete understanding of the program. Even worse, maintainers often do not have access to specifications or design documents just the code. As software ages and evolves, the task of maintaining it becomes more complex and more expensive. Some of the other causes of software maintenance problems are:

- Software maintainability is often not a major consideration during design and implementation.
- Maintenance has been largely ignored in software engineering research.
- Maintenance activities are not well understood

2.3 Change Impact Analysis

The two most expensive activities in software maintenance are the understanding of problems or other expressed needs for change, in relation with the understanding of the maintained software system, and the mastering of all the ripple effects of a proposed change. A small change can ripple throughout the system to have major unintended impacts elsewhere. As a result, software developers need mechanisms to understand how a change to a software system will impact the rest of the system. This process is called change impact analysis.

2.3.1 Process

To put change impact analysis in perspective, we first need to understand the process of change. Madhaji [17] defines the process of change as:

(1) Identify the need to make a change to an item in the environment

(2) Acquire adequate change related knowledge about the item

(3) Assess the impact of a change on other items in the environment

(4) Select or construct a method for the process of change

(5) Record the details of the changes for future reference, and release the changed item back to the environment

One key problem in accommodating changes in an environment is to know all the factors that impact a given change, and the consequences of this change.

2.3.2 Analysis

Impact analysis(IA), as dened by Arnold and Bohner in [5], "is the activity of identifying what to modify to accomplish a change, or of identifying the potential consequences of a change,".

In SOA, change impact analysis is used to specify what will be impacted in services and related service functionality if a proposed change is maded. It is defined as the process of determining the effects on other components of the services resulting from the proposed changed.

There are two main approaches to impact analysis: dependency analysis and traceability analysis.

- Dependency analysis is the analysis of relationships between functionalities or operations inside a service and among services.
- Traceability analysis, on the other hand, is the analysis of relationships among architectural components.

Typical Change Impact Analysis Process

The process for impact analysis for a proposed change consists of following steps:

Stage 1. Convert proposed change into a system change specification.

Stage 2. Extract information from information source and convert into Internal Representation Repository.

Stage 3. Calculate change impact for these change proposals. Do Stage 1-3 again for other competing change proposals.

Stage 4. Develop resource estimates, based on considerations such as size and software complexity.

Stage 5. Analyze the cost and benefits of the change request, in the same way as for a new application.

Stage 6. The maintenance project manager advises the users of the implications of the change request, in business rather than in technical terms, for them to decide whether to authorize proceeding with the change.

Chapter 3

Our method

3.1 Software Change Impact Analysis Overview

Figure 3.1 describes the basic software change impact analysis process[1].

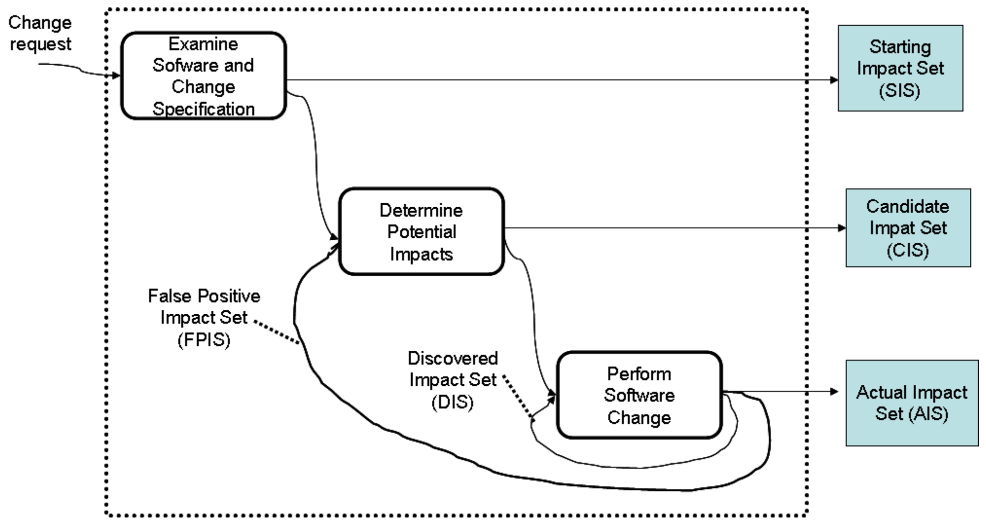

Change request
Examine Sofware and Change Specification
Starting Impact Set (SIS)
Determine Potential Impacts
Candidate Impat Set (CIS)
False Positive Impact Set (FPIS)
Discovered Impact Set (DIS)
Perform Software Change
Actual Impact Set (AIS)

Figure 3.1: Change Impact Analysis

Based on the requests from users and current systems, we specify the change and determine what parts will be directly impacted by this change. The Starting Impact Set (SIS) is the initial set of objects or components thought to be affected by a change. The SIS is normally determined when examining the change specification. The Candidate Impact Set (CIS) is the set of objects estimated to be affected. The CIS is produced while conducting the impact analysis. The Actual Impact Set(AIS) is the set of the objects actually modified.
The impact analysis process is iterative and discovery in nature. While a change is being performed, there are likely to be more impacts discovered. The discovered impact set (DIS) represents an under-estimate of impacts. The false-positive impact set (FPIS)

represents the over-estimate of impacts in the analysis. The CIS plus additions of the DIS and minus deletions of FBIS should represent the AIS. The accuracy (error) is obtained by adding the number of impacts in the DIS and FPIS then dividing them by the number of impacts in the CIS. This represents the number of errors (DIS and FBIS) divided by the estimate. The objective of the analysis is to have the Candidate Impact Set produced from tracing potential impacts (manually or with automation) as close to the Actual Impact Set as possible by identifying true impacts while eliminating false-positives.

3.2 Related Works

Currently, researches focus on change impact analysis for object-oriented or component-based software systems. A number of impact analysis methods have been developed and can be classified into two categories: code-based and model-based.

3.2.1 Code-based Change Impact Analysis

Current works for change impact anlysis of object oriented software systems often base on the dependency between classes, methods and data members to impact analysis. In [14], they investigate the change impact analysis of object-oriented software in the distributed environment. They first categorize the types of change elements in object-oriented software into three types: data, method, and class level changes. They then analyze the impact of each set of changes and represent it in the form of a Distributed Program Dependency Graph(DPDG). The DPDG is a graph showing relationship of object oriented software - with data elements, classes, design documents, servers and its minimized firewall in the distributed environment. Thus, the DPDG reduce the effort and element of software to retest. In [19], they also proposed a technique for determining the effects of source code changes for object-oriented software systems.

The advantages of code-based methods is that we often gather an accurate analysis of the change impact set, but it is extremely time consuming.

3.2.2 Model-based Change Impact Analysis

In order to determine change impacts in earlier phase, and to make proper decision before considering any change implementation details, there is a need of a method to identify change impacts without using program code. These methods are called model-based methods.

Kung et al. [13] describes how change impact analysis can be performed from a class diagram, introducing the notion of class firewall (i.e., classes that may be impacted by a change in a given class), and discusses the impact of object-oriented characteristics (e.g., encapsulation, inheritance, polymorphism, dynamic binding) on such an analysis.

Briand [6] proposed a UML model-based approach to impact analysis that can be applied before any implementation of the changes. They first verify that the UML diagrams are consistent. Then changes between two different versions of a UML model are identified according to a change taxonomy, and model elements that are directly or indirectly impacted by those changes are determined using formally defined impact analysis rules. A measure of distance between a changed element and potentially impacted elements is also proposed to prioritize the results of impact analysis according to their likelihood of occurrence. They also present a prototype tool that provides automated support for our impact analysis strategy, that we then apply on a case study to validate both the implementation and methodology.

In [11], they proposed an approach to apply a light-weight proactive requirement change impact analysis based on Use Case Maps. This approach identifies and assesses the change impact proactively at the requirement level before a change is actually performed on the source code.

3.3 Our Change Impact Analysis for SOA

Nowaday, Service oriented architecture is adopted widely because of the reuse and cost-effective of using services. Service oriented applications automate business processes by a set of services. Each service contain many operations to perform different functions. Like traditional applications, service oriented application are always change over time. When a change request is proposed, such as change in data type of it may affect to other services or operations in each service. Therefore, we need methods to change impact analysis for these applications. Our methods is proposed to solve this problem. Figure 3.2 depicts the proposed model to change impact analysis. Our approach contain 4 steps:

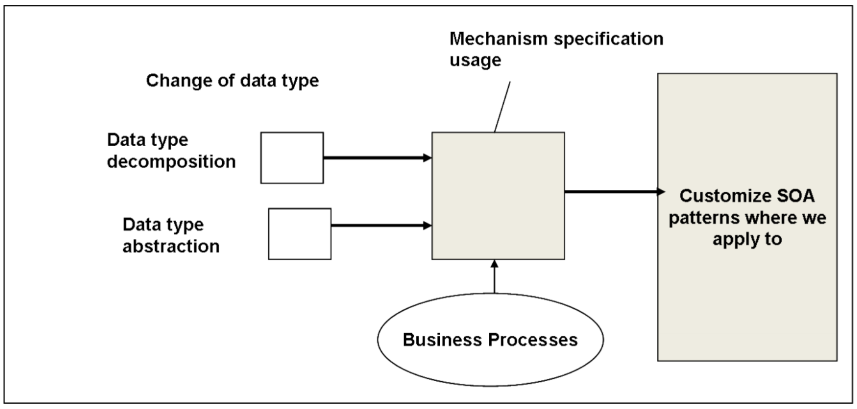


Figure 3.2: Our model for Change Impact Analysis

- Model Services and Business Process
- Characterize kind of change by change in input/output of each service
- Apply impact analysis on the model which represents relationship among input and output data for all of services

- Check the condition for 4 SOA patterns are applicable or not: Operation Abstraction, Operation Decomposition, Service Decomposition and Service Abstractions

3.3.1 Service Oriented Business Process Model

In service oriented architecture, business applications encode business processes to achieve the business objectives. A business process is a set of logically related tasks to achieve the business objectives [20]. Tasks are usually implemented by services.

When a business process is changed (e.g., adding a task, and removing a task), the services which are invoked by a step of the business process need to be updated to support the new requirements. To determine which part need to be changed, we first model the relationship between business processes and services

Let us consider a opening account scenario for a banking system as an example. An account opening process receives an request information from a customer, validate if the account information is valid or not and then sends confirmation to the customer. So, the business process contain the following steps:

Step 1. Get information from the customer

Step 2. Validate the account and customer information. If the account is valid: go to step 3. If not, cancel the customer's request.

Step 3. Open a new account

Step 4. Send confirmation to the customer.

These tasks are performed by invoking the corresponding services. Figure 3.3 shows the relations between the business process and its services. For example, to perform the task "validate account and customer information", the business process might invoke the corresponding service "AccountValidating".

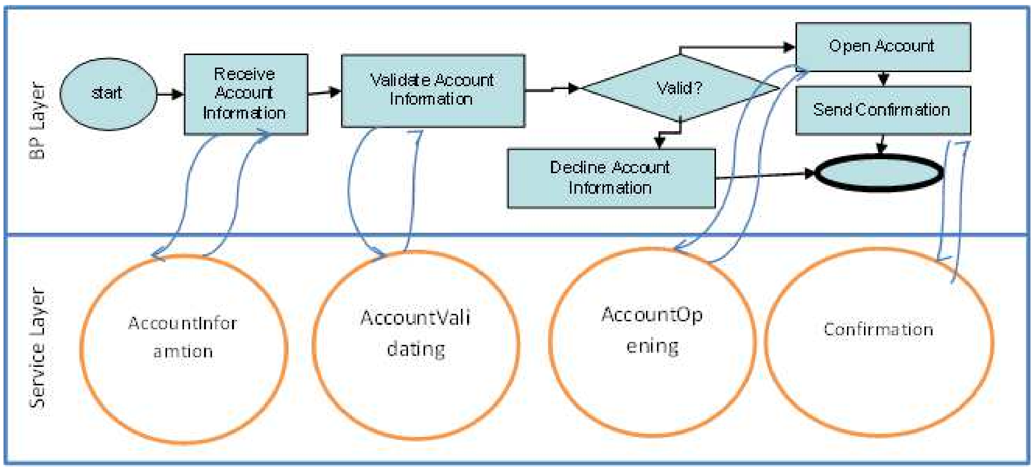


Figure 3.3: An example of the model for BPs and Services

Service model

A service is built and implemented to perform one or more tasks in the business process. Each service contains a set of operations and each operation receives the data input, process requirements and return the output to the customer.

3.3.2 Classification of Service Change

We know that a service oriented application consist of many services. Each service has its operations and it can relate to other services. Therefore, the change also has many different types. In our research, we categorize change into four types:

- Operation abstraction
- Operation decomposition
- Service abstraction
- Service decomposition

Operation Abstraction
In the case O_1 and O_2 in the figure 3.4 may contain some common information which can be abstracted, we can combine these two operations and get a new operation. This is called Operation Abstraction.
After applying operation abstraction, we get a new operation as showed in the figure 3.5.

Change specification

Before Change
Service A has two operations: O_1 and O_2
O_1 =(in_1, out_1)
O_2 =(in_2, out_2)

After Change
Service A has 3 new operations:
$O_{new} = (in_{new}, out_{new})$
$O_1^{'} = (in_1^{'}, out_1^{'})$
$O_2^{'} = (in_2^{'}, out_2^{'})$
In which,
$in_{new} = in_1 \cap out_1$
$out_{new} = out_1 \cap out_2$
$in_1^{'} = in_1 - in_{new}$
$out_1^{'} = out_1 - out_{new}$
$in_2^{'} = in_2 - in_{new}$
$out_2^{'} = out_2 - out_{new}$

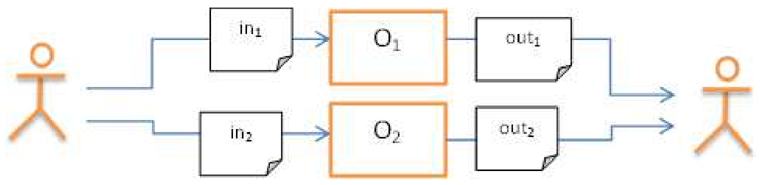


Figure 3.4: Type 1: Operation abstraction before change

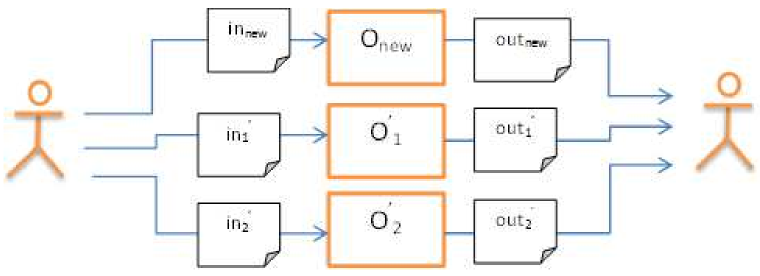


Figure 3.5: Type 1: Operation abstraction after change

Operation Decomposition

Operation decomposition means to decompose one operation into a set of operations.

Change specification

Before Change
Service A has a new operation: O_1
O_1 =(in_1,out_1)

After Change
Service A has two operations: O_{11} and O_{12}
O_{11} =(in_{11}, out_{11})
O_{12} =(in_{12}, out_{12})
in_1 =in_{11}+ in_{12}
out_1= out_{11} + out_{12}

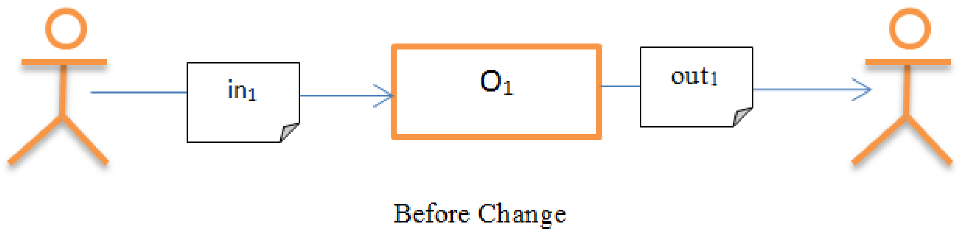


Figure 3.6: Type 2: Operation decomposition before change

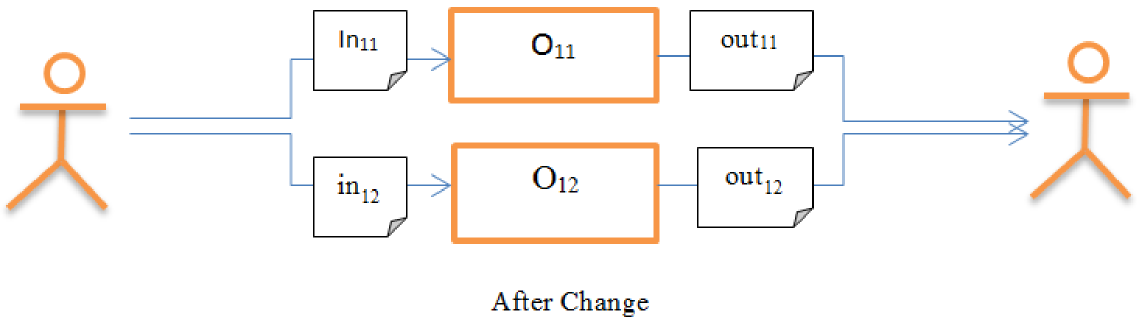


Figure 3.7: Type 2: Operation decomposition after change

Let take an example. We have an web portal which allows customer sign up a mobile phone when they buy a phone.Figure 3.8 depicts the signup service with one operation allocatePhoneNumber. The business process of signup service consist of the following steps:

- Step 1: Randomize a phone number
- Step 2: Validate the phone number
- Step 3: Assign the phone number to customer

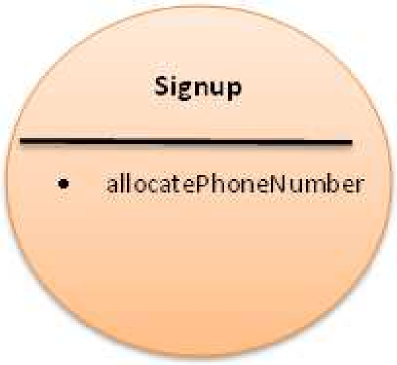


Figure 3.8: Signup service before change

Now we have a change for the sign up process. The customer should be able to choose a phone number he/she prefers. The new process is:

- Step 1a: Send a list of available phone number to the customer
- Step 1b: Customer choose one of them
- Step 2: Validate the phone number
- Step 3: Assign the phone number to the customer

The operation allowcatePhoneNumber of the signup service is decomposed into two operations as showed in the figure 3.9.

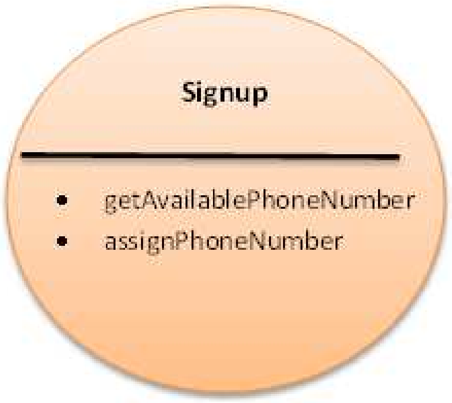


Figure 3.9: Signup service after change

Service Abstraction
In service oriented architecture, services are often aggregated into a composited service to perform a specific some business objective. So, there are many services and the business logics may become overlapped or duplicated among these services. It would be nice if we can abstract these business logics and make them as separated services. There are two types of service abstraction:

- Service simple combination: Business logics of the same domain separated in different services, it would be better to simply combine them to centralize them as one service.
- Service layering: Business logics are overlapping and duplicated in different services, we can abstract the overlapping part into separated services, usually by putting them into different service layers.

Change specification for combing two simple services:

Before Change
Service S_1 has k operations O_1, O_2...O_k
Service S_2 has m operations $O_{1'}$, $O_{2'}$...$O_{m'}$

After Change
A new service $S^{'}$ has m+k operations: O_1, O_2...O_k, $O_{1'}$, $O_{2'}$...$O_{m'}$

Change specification for layering services:

Before Change
Service S_1 has k operations O_1,...O_j, O_{j+1}...O_k
Service S_2 has m operations O_1,...O_j, $O^{'}_j$...$O^{'}_m$

After Change
System has 3 new service:
S_c has j operations: O_1, O_2...O_j
$S_{1'}$ has $(k - j)$ operations: O_{j+1}...O_k
$S_{2'}$ has $(m - j)$ operations: $O^{'}_{j+1}$...$O^{'}_m$

For example, we have the 2 services which are dealing with payment. The logics of payment are separated into two services:

- 'checkCreditCard' service receives credit card information from a customer and validates the credit card is valid or not (Figure 3.10).
- 'Refund' service allows customer to check the amount of their credit card account and perform payment online (Figure 3.11).

Both of these services provide functionalities related to credit card account. Therefore, we should combine them into a new service 'checkCreditCard' (Figure 3.12).

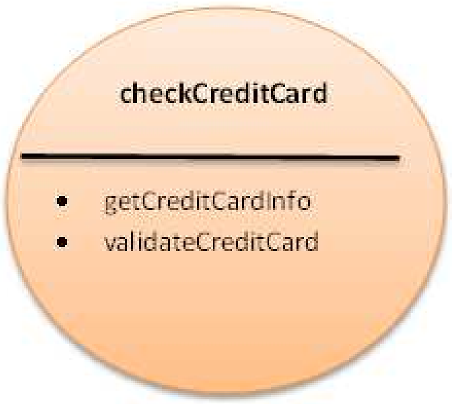


Figure 3.10: checkCreditCard service before change

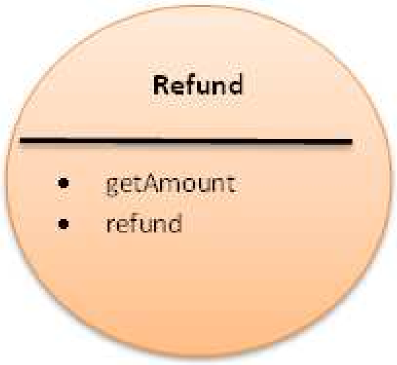


Figure 3.11: Refund service before change

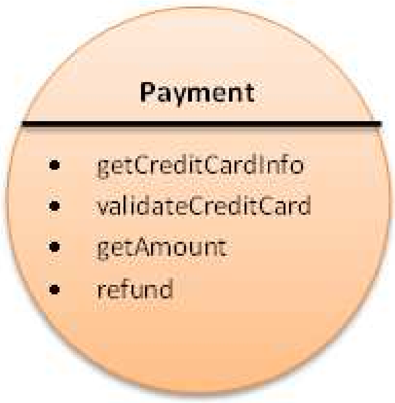


Figure 3.12: Payment service after change

Service Decomposition

Service decomposition simply means to split a large service into several smaller services.

Change specification for decomposing a large service:

Before Change
Service S has k operations O_1, $O_2 ... O_k$
After Change
Two new services
S_1 has j operations O_1, $...O_j$
S_2 has $(k - j)$ operations O_{j+1}, $...O_k$

Chapter 4

Case study

The main focus of this chapter is to show the effect of our approach for detecting the part need to change by impact analysis with an example of actual application 'virtual travel agency'. This example describes travel agency system for booking travels to its customers. It is expected that a Web Application provides these services automatically. For simplicity reasons, we will concentrate just on accommodation service and transportation service which can be carried out using planes.

4.1 Virtual Travel Agency

We use Virtual Trave Agency (VTA) as an example by the following reasons:

(1)This system has a simple structure of services. VTA has two services for "booking transportation" and 'making an accommodation' and these services are independent.

(2) The input and output of each services are clearly defined by a collection of 'String' or a date type. This is easy to specify data types.

4.1.1 System: Example Overview

The travel agency would like to build a system which provides its customers the following functions (Figure 4.1)

- Search flights for a city pairs on a specified day.
- Request a flight availability for a specified flight with a particular cabin type.
- Get the fare information about a given flight with a predefined fare type.
- Book a specific flight with a predefined fare type.
- Find the hotels in a specified city
- Make a reservation to a specified hotel in a duration time.

- Send an invoice for payment.

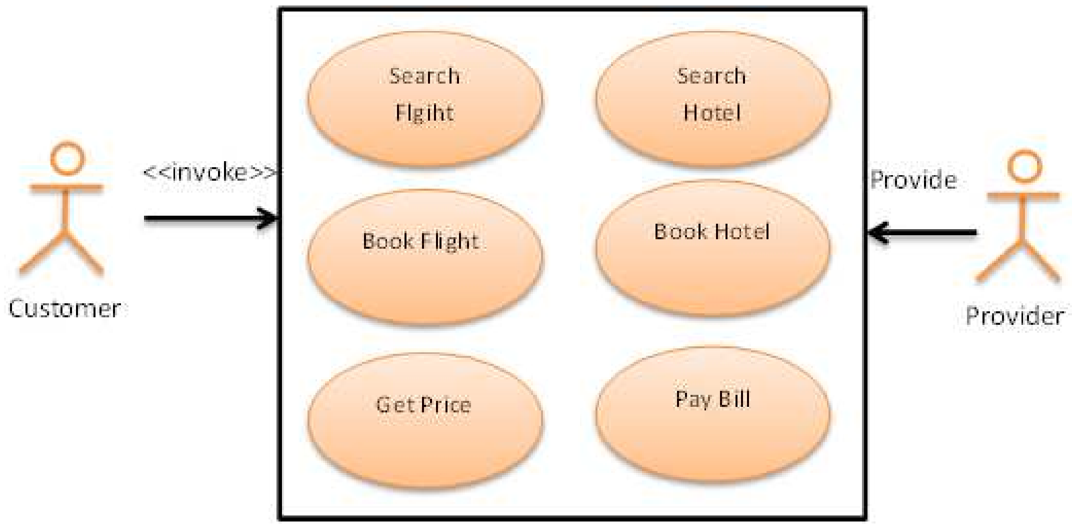


Figure 4.1: VTA Functions

To satisfy these functionalities, the IT engineers of the travel agency is planning to develop the system with a collection of services. Figure 4.2 shows services used to accomplish the logic solution.

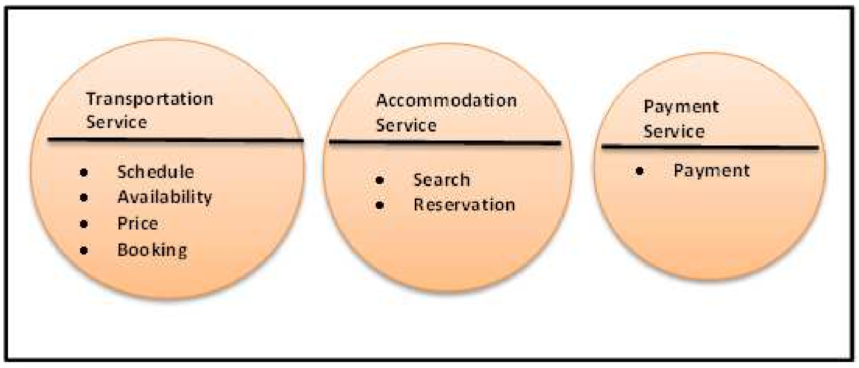


Figure 4.2: VTA services

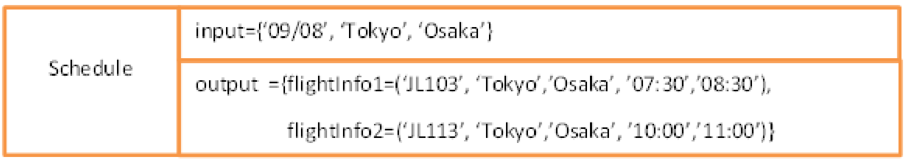
4.1.2 Services

Transportation Service

This service is used for booking the transport.At first, transportation with airlines is considered, but in the future it can be extended more choices such as railways, cars, ships and so on. Transportation Service contains the following operations:

(1) Schedule: This operation shows the flight schedule for connecting two specified cities. The input of the operation contains (Date, Porg, Pdst) which Porg and Pdst are the location for origin and destination of transportation. The output contains the information of a list of flights. The details of the output is described later.

Let us take an example. A passenger want to travel from Tokyo to Osaka for a business meeting. He would like to take a flight that will leave on August 09, 2012. He will receives a list of options as follows:

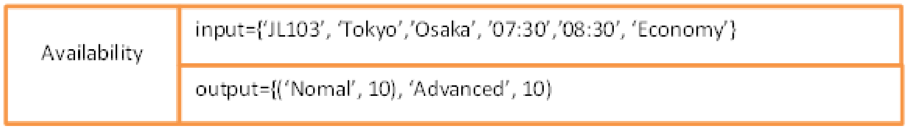


Schedule	input={‘09/08’, ‘Tokyo’, ‘Osaka’}
	output ={flightInfo1=(‘JL103’, ‘Tokyo’,’Osaka’, ’07:30’,’08:30’), flightInfo2=(‘JL113’, ‘Tokyo’,’Osaka’, ’10:00’,’11:00’)}

Figure 4.3: An example of checking flight schedule

(2) Availability: This operation shows the number of available seat for the specified flight. The input contains (FlightInfo, CabinTyle) and the output contains (FareType, Vacancy).

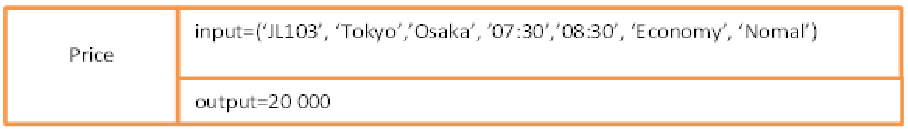
We assume that Mr Smith would like to fly from Tokyo to Osaka on August 09, 2012. He want to fly on the flight 103 of Japan Airline, depart at 9:00 and in the business cabin. Mr Smith requests availability for this flight. He will receive information as showed in the figure 4.11.



Availability	input={‘JL103’, ‘Tokyo’,’Osaka’, ’07:30’,’08:30’, ‘Economy’}
	output={(‘Nomal’, 10), ‘Advanced’, 10)

Figure 4.4: An example of checking availability

(3) Price: This operation shows the prices of specified seat (FareType) and flight information. The input contains (FlightInfo, FareType) and the output contains the price.

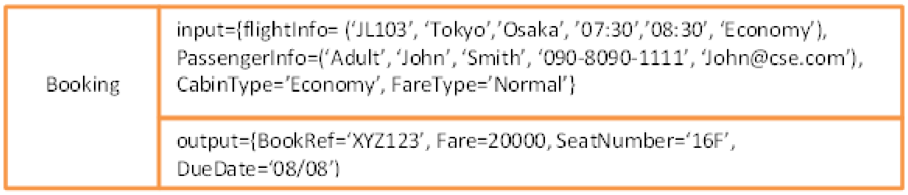


Price	input=(‘JL103’, ‘Tokyo’,’Osaka’, ’07:30’,’08:30’, ‘Economy’, ‘Nomal’)
	output=20 000

Figure 4.5: An example of checking price

(4) Booking: This operation shows the result of booking (succeed or not). The input contain (FlightInfo, PassengerInfo, CabinType, FareType) and the output contains (RefInfo, Fare, Tax, DueDate).

Let us take an example. John Smith wants to fly from Tokyo to Osaka on the flight 103 of Japan Airline. on Agust 09, 2012, at 9:00 AM. He also wants to book in business cabin and with a normal fare. He makes a request via the web portal of the Travel Agency and then receive an email consisting of the information as showed in the figure 4.6.



Booking	input={flightInfo= ('JL103', 'Tokyo','Osaka', '07:30','08:30', 'Economy'), PassengerInfo=('Adult', 'John', 'Smith', '090-8090-1111', 'John@cse.com'), CabinType='Economy', FareType='Normal'}
	output={BookRef='XYZ123', Fare=20000, SeatNumber='16F', DueDate='08/08'}

Figure 4.6: An example of booking transportation

FlightInfo contains:

- AirName: Name of airline (e.g. Japan Airlines)
- FlightNum: Flight number (e.g. Flight 103)
- Porg and Pdst: The original and destination location (e.g. Tokyo to Osaka)
- DepTime and ArrTime: The departure time and the arrival time of the flight (e.g. DepTime 07:30, ArrTime 08:30)

CabinType: The type of cabin of the flight. CabinType is one of a string "Economy"/"Business"/"First".
FareType is one of a string "Normal"/"Advanced".
PassengerInfo contains:

- PassengerType is one of a string "Adult"/"Child"
- FirstName is a string (e.g. "John")
- LastName is a string (e.g. "Smith")
- PhoneNumber is a string (e.g. "090-8090-1111")
- Email is a string(e.g. "johnSmith@gmail.com")

BookRef is a string used to uniquely identify your booking (e.g. "XYZ123")

Schedule Operation Input	**Schedule Operation Output**
- Date: DateTime - Porg: String - Pdst: String	**FlightInfo** - AirName: String - FightNum: String - Porg: String - Pdst: Integer - DepTime: Integer - ArrTime: String

Table 4.1: The I/O of Schedule Operation

Availability Operation Input	Availability Operation Output
FlightInfo - AirName: String - FlightNum: String - Porg: String - Pdst: String - DepTime: String - ArrTime: Stirng **CabinType**: String	**FareType**: String **Vacancy**: Number

Table 4.2: The I/O of Availability Operation

Price Operation Input	Price Operation Output
FlightInfo - AirName: String - FlightNum: String - Porg: String - Pdst: String - DepTime: String - ArrTime: Stirng **FareType**: String	**Fare**: Number

Table 4.3: The I/O of Price Operation

Reservation Operation Input	Price Reservation Output
FlightInfo - AirName: String - FlightNum: String - Porg: String - Pdst: String - DepTime: String - ArrTime: Stirng **PassengerInfo** - PassengerType: String - FirstName: String - LastName: String - PhoneNumber: String - Email: String **FareType**: String **SeatNumber**: String	**BookRef**: String **Fare**: Number **Tax**: Number **DueDate**: DateTime

Table 4.4: The I/O of Booking Operation

Accommodation Service

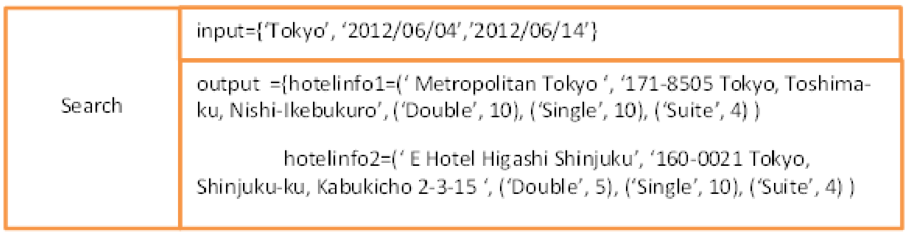
This service is used for searching and booking hotel. This service provides a wide range of functionalities, including:

(1) Search Operation provides the ability to search for hotels. The input contains (City, CheckinDate, CheckoutDate). The output contains a list of HotelInfo which is described later.

Let us take an example. We assume that Ms Minamoto wants to travel to Tokyo for 10 days (from June 04 to June 14).SHe doesn't know any hotels in Tokyo and want to get a list of hotels corresponding to her criteria:

- City: Tokyo
- Check in date:2012 June 04
- Check out date: 2012 June 14

A list of hotels is returned including information hotel name, hotel address and the pair (Room Type, Available rooms)as showed in the figure 4.7.



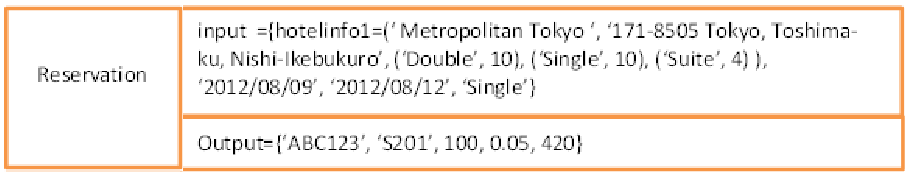
Search	input={'Tokyo', '2012/06/04','2012/06/14'}
	output ={hotelinfo1=(' Metropolitan Tokyo ', '171-8505 Tokyo, Toshima-ku, Nishi-Ikebukuro', ('Double', 10), ('Single', 10), ('Suite', 4)) hotelinfo2=(' E Hotel Higashi Shinjuku', '160-0021 Tokyo, Shinjuku-ku, Kabukicho 2-3-15 ', ('Double', 5), ('Single', 10), ('Suite', 4))

Figure 4.7: An example of searching accommodation

(2) Reservation
This operation provides customers to reserve an accommodation. The input contains (HotelInfo,ChkinDate, ChkoutDate, RoomType). The output contains (BookingRef,RoomName PriceInfo) that is described later.

For illustrating, we assume that Mr John Smith is planning to travel to Tokyo and has decided to book his hotel reservation online for 4 days from August 09 to August 12. Before booking, he made an availability request to determine which hotel he would like to book. He found that the Metropolitan Tokyo hotel has 5 available double rooms 3 available single rooms. Therefore, he decided to make a reservation for one single room to this hotel.

After that, he received a booking reference number and price information as showed in the figure 4.9.



Reservation	input ={hotelinfo1=(' Metropolitan Tokyo ', '171-8505 Tokyo, Toshima-ku, Nishi-Ikebukuro', ('Double', 10), ('Single', 10), ('Suite', 4)), '2012/08/09', '2012/08/12', 'Single'}
	Output={'ABC123', 'S201', 100, 0.05, 420}

Figure 4.8: An example of booking accommodation

HotelInfo contains:

- HotelName is a string, the name of the hotel (e.g. "Shinjuku Prince Hotel")
- HotelAddress is a string, the address of the hotel (e.g. "1-30-1 Kabuki-cho, Tokyo")
- The pair (RoomType, AvailRooms) is the type of room and the corresponding available rooms for this type. RoomType is one of a string "Single"/"Double"/"Suite"

PriceInfo contains:

- RatePerNight: is the rate per night (e.g 100USD)
- TaxInfo: The information about the tax (e.g. TaxInfo 0.05)
- TotalAmount is the total amount which customer has to pay for accommodation.

Search Operation Input	**Search Operation Output**
- City: String - CheckinDate: Date - CheckoutDate: Date	- HotelName: String - HotelAddress: String - (RoomType, AvailRooms): String

Table 4.5: The request and response message of HotelSearch operation

Reservation Input	**Reservation Output**
HotelInfo - HotelName: String - HotelAddress: Date CheckinDate: Date CheckoutDate:Date RoomType	BookingRef: String PriceInfo - RatePerNight: Number - TaxInfo: Number -TotalAmount: Number RoomName

Table 4.6: The request and response message of HotelAvailability operation

Payment Service Payment service (refer to figure 4.2) is used to pay the bill online via a credit card account. This service has only one operation: Payment Operation

The input contains (AccInfo, PayDetail, CusInfo) and the output contains a receipt number and total amount paid. The detail of data input and ouput are described later.

AccInfo contains

- CCNumber: is a string, the credit card number of the customer (e.g. ”4111 1111 1111 1111”).
- CCHolderName: is a string, the holder of the credit card account.
- ExpDate: is the expiration date of the credit card (e.g. ”2012-09-06”).

PayDetail contains:

- PaymentType: is a string, decribes the type of payment in which customers want to pay. (e.g. ”Airline ticket payment”).
- Amount: The total amount is paid.

CusInfo contains customer information such as the first name, last name, address, phone number and email.

Payment Input	**Payment Output**
AccInfo - CCNumber: String - CCHolderName: String - ExpDate: String **PayDetail** - PaymentType:String - Amount: Number **CusInfo** - FirstName: String - LastName: String - Address: String - PhoneNumber: String	BookingRef: String Amount

Table 4.7: The I/O of Payment Operation

4.1.3 Business processes

To understand how the travel agency automate its business process, we consider some business processes, the transportation booking process and accommodation booking process.

Transportation Booking Process
We assume that a customer named 'John Smith' want to book a flight from Tokyo to Osaka on August 09 for his business. The process of his booking contains the following steps:

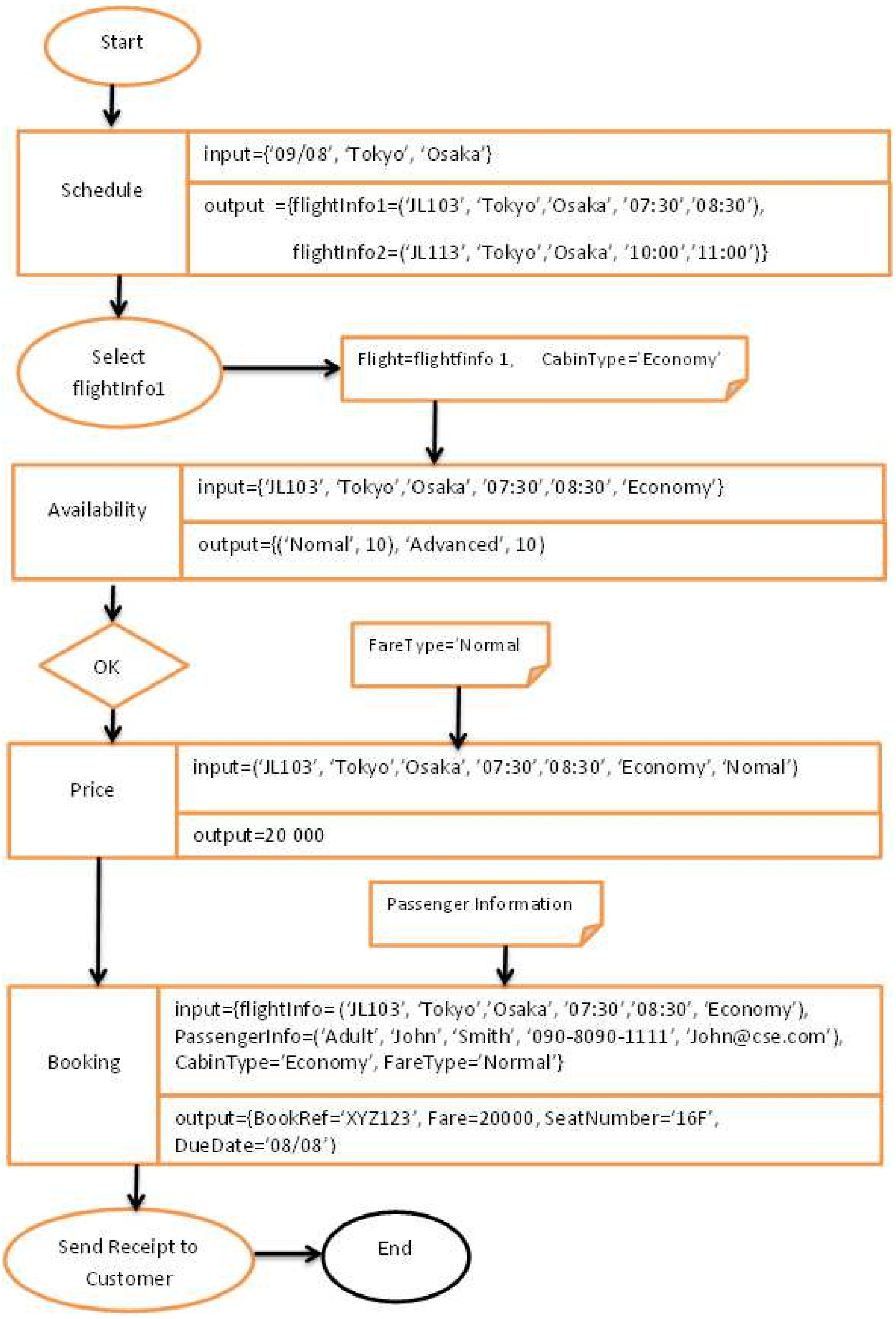
Step 1. Search flights which match his requirements
This step is performed by invoking 'Schedule' operation of 'Transportation' service.

Step 2. Check if the flight has available seats or not
After receiving a list of flights in the step 1, he will select a flight and check if this flight has tickets. This step is performed by invoking 'Availability' operation of 'Transportation' service.

Step 3. Check the price of the ticket. This step is performed by invoking 'Price' operation of 'Transportation' service.

Step 4. Book the flight This step is performed by invoking 'Booking' operation of of 'Transportation' service.

The booking process is showed in the figure 4.9.



Start
Schedule
input={'09/08', 'Tokyo', 'Osaka'}
output ={flightInfo1=('JL103', 'Tokyo','Osaka', '07:30','08:30'),
flightInfo2=('JL113', 'Tokyo','Osaka', '10:00','11:00')}
Select flightInfo1
Flight=flightfinfo 1, CabinType='Economy'
Availability
input={'JL103', 'Tokyo','Osaka', '07:30','08:30', 'Economy'}
output={('Nomal', 10), 'Advanced', 10)
OK
FareType='Normal
Price
input=('JL103', 'Tokyo','Osaka', '07:30','08:30', 'Economy', 'Nomal')
output=20 000
Passenger Information
Booking
input={flightInfo= ('JL103', 'Tokyo','Osaka', '07:30','08:30', 'Economy'),
PassengerInfo=('Adult', 'John', 'Smith', '090-8090-1111', 'John@cse.com'),
CabinType='Economy', FareType='Normal'}
output={BookRef='XYZ123', Fare=20000, SeatNumber='16F',
DueDate='08/08')
Send Receipt to Customer
End

Figure 4.9: Transportation booking process

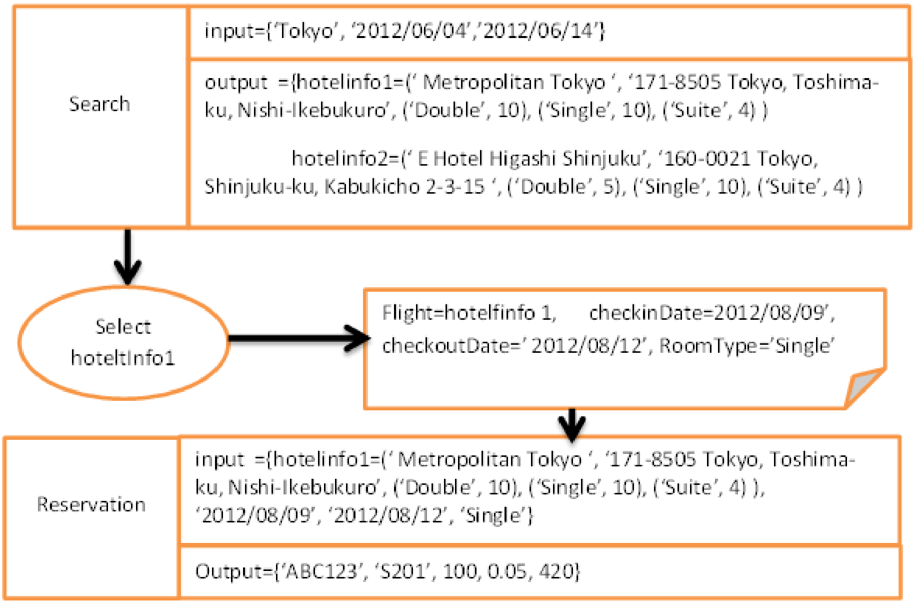
Accommodation Booking Process

We assume that Mr 'John Smith' want to book a hotel in Tokyo from August 09 to August 12 for his accommodation. The process of his booking contains the following two steps:

Step 1. Search hotels which match his requirements
This step is performed by invoking 'Search' operation of 'Accommodation' service.

Step 2. Make an reservation to the hotel which he chooses from the list of hotels in the step 1. This step is performed by invoking 'Reservation' operation of 'Accommodation' service.

The booking process is showed in the figure 4.10.



Search
input={'Tokyo', '2012/06/04','2012/06/14'}
output ={hotelinfo1=(' Metropolitan Tokyo ', '171-8505 Tokyo, Toshima-ku, Nishi-Ikebukuro', ('Double', 10), ('Single', 10), ('Suite', 4))
hotelinfo2=(' E Hotel Higashi Shinjuku', '160-0021 Tokyo, Shinjuku-ku, Kabukicho 2-3-15 ', ('Double', 5), ('Single', 10), ('Suite', 4))
Select hotelInfo1
Flight=hotelfinfo 1, checkinDate=2012/08/09', checkoutDate=' 2012/08/12', RoomType='Single'
Reservation
input ={hotelinfo1=(' Metropolitan Tokyo ', '171-8505 Tokyo, Toshima-ku, Nishi-Ikebukuro', ('Double', 10), ('Single', 10), ('Suite', 4)), '2012/08/09', '2012/08/12', 'Single'}
Output={'ABC123', 'S201', 100, 0.05, 420}

Figure 4.10: Accommodation booking process

4.2 Change analysis in VTA System

We assume that customers want to change some requirements in the VTA system. We need to specify what things will be affected by these proposed changes.

Case 1. A customer requires to know the prices of more than one flights.
Change of input/output

- **Before Change**
  Operation 'Availability'

in =(flightInfo, CabinType)
out=(list of (FareTypye, vacancy))

Operation 'Price'

in =(flightInfo, CabinType, FareType)
out=(Fare)

- **After Change**
  New operation: CheckFlightPrice

in =(list of flightInfo, CabinType)
out=(list of (flightInfo, list of (FareType, Vacancy, Fare)

Here is an example. assume that:
flightinfo1=(JL,103, Tokyo, Osaka, 07:30, 08:30)
flightinfo2=(JL,107, Tokyo, Osaka, 08:30, 09:30)

Before change

[**Specification for Operation 'Availability'**]

input = ('August 09', 'Tokyo', 'Osaka')
output=((JL,103, Tokyo, Osaka, 07:30, 08:30),
(JL,107, Tokyo, Osaka, 08:30, 09:30))

[**Specification for Operation 'Price'**]

input=(flightinfo1, Economy, normal) output = 16000

input=(flightinfo1, Economy, advanced) output = 13000

input=(flightinfo2, Economy, normal) output = 16000

input=(flightinfo2, Economy, advanced) output = 11000

After change

input = ((flightinfo1, Economy), (flightinfo2, Economy))

output = ((flightinfo1, ((normal, 10, 16000), (advanced, 10, 13000))), (fliightinfo2, ((normal, 3, 16000), (advanced, 6 11000))))

Change in datatype: input of 'Availability'

Impact to : input of 'Price'
Applicable Patterns : Operation Abstraction
The operation 'Availability' and operation 'Price' are combined into the new operation 'CheckFlightPrice'
Reason: a set of info conbining input of 'Availability' and 'Pricing' are identical between before and after changing.

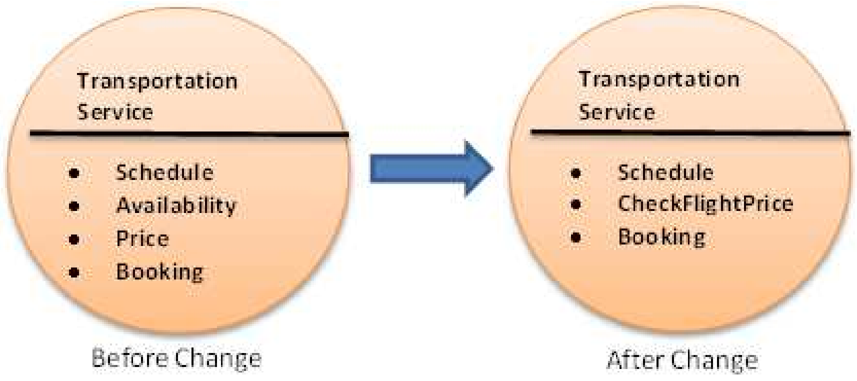


Figure 4.11: An example of operation abstraction

Case 2. A customer want to know schedule of all flights in the duration from August 10 to August 20 because his/her schedule are flexible.
Change of input/output

- **Before Change**
  Operation 'Schedule'

in =(Date, Porg, Pdst)
out=(FlightInfo)

- **After Change**

in =(From, To, Porg, Pdst)
out=(list of (flightInfo, Date)

Here is an example. assume that:
flightinfo1=(JL,103, Tokyo, Osaka, 07:30, 08:30)
flightinfo2=(JL,107, Tokyo, Osaka, 08:30, 09:30)

Before change

[**Specification for Operation 'Schedule'**]
input = ('August 09', 'Tokyo', 'Osaka')
output=(JL,103, Tokyo, Osaka, 07:30, 08:30)
(JL,107, Tokyo, Osaka, 08:30, 09:30)
After change
input = ('August 09', 'Agust 15', 'Tokyo','Osaka')
output = ((JL,103, Tokyo, Osaka, 07:30, 08:30),'August 09'),
(JL,103, Tokyo, Osaka, 08:30, 09:30),'August 09'),
(JL,105, Tokyo, Osaka, 07:30, 08:30),'August 10'), (JL,106, Tokyo, Osaka, 11:30, 12:30),'August 12')

Change in datatype: input of 'Schedule'
Impact to : 'Schedule' operation
Applicable Patterns : Operation Decomposition
The operation 'Schedule' is decomposed into two coarse-grained: 'ScheduleofDate' and 'ScheduleofDuration'.
Reason: a set of info conbining input of 'Availability' and 'Pricing' are identical between before and after changing.

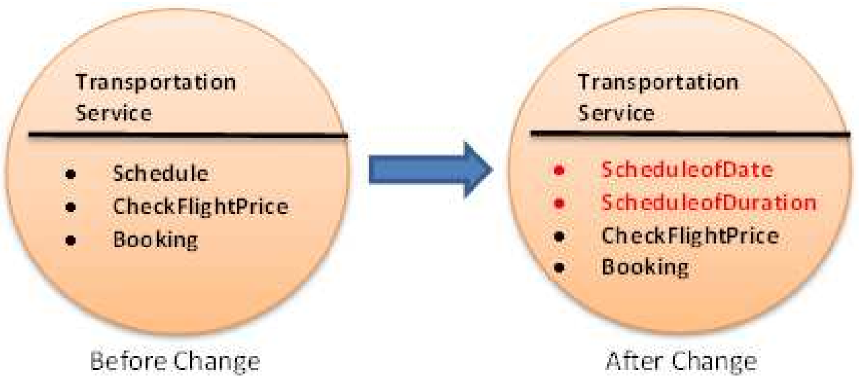


Figure 4.12: An example of operation decomposition

Chapter 5

Conclusion and future works

This thesis presented an approach for change impact analysis for services at the design level. In this thesis, we presented a model for business processes and services. Based on the relationship between business processes and services. We also presented four types of service governance patterns used for change impact analysis: operation abstraction, operation decomposition, service abstraction and service decomposition. we also evaluated the effectiveness of our approach with a case study, a virtual travel agency system. We proposed some change of requests and apply to each pattern.

However there are still some limitations in our approach. In the future, we would like to develop an executable model for change impact analysis that can be built in the real world.

Bibliography

[1] Software change impacts - an evolving perspective. In *Proceedings of the International Conference on Software Maintenance (ICSM'02)*, ICSM '02, pages 263–, Washington, DC, USA, 2002. IEEE Computer Society.

[2] N. Bieberstein, S. Bose, L. Walker, and A. Lynch. Impact of service-oriented architecture on enterprise systems, organizational structures, and individuals. *IBM Systems Journal*, 44(4):691–708, 2005.

[3] H. Bilal and S. Black. Computing ripple effect for object oriented software.

[4] S. Bohner. Impact analysis in the software change process: a year 2000 perspective. *Software Maintenance, IEEE International Conference on*, 0:42, 1996.

[5] S. A. Bohner. Software change impact analysis. 1996.

[6] L. C. Briand, Y. Labiche, and L. O'Sullivan. Impact analysis and change management of uml models. In *Proceedings of the International Conference on Software Maintenance*, ICSM '03, pages 256–, Washington, DC, USA, 2003. IEEE Computer Society.

[7] T. Erl. *Service-Oriented Architecture: Concepts, Technology, and Design.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005.

[8] T. Erl. *SOA Design Patterns.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2009.

[9] G. Feuerlicht. Service-oriented computing — icsoc 2008 workshops. chapter Design of Composable Services, pages 15–27. Springer-Verlag, Berlin, Heidelberg, 2009.

[10] J. Han. Supporting impact analysis and change propagation in software engineering environments. *Software Technology and Engineering Practice, International Workshop on*, 0:172, 1997.

[11] J. Hewitt and J. Rilling. A light-weight proactive software change impact analysis using use case maps. In *Proceedings of the IEEE International Workshop on Software Evolvability*, SOFTWARE-EVOLVABILITY '05, pages 41–48, Washington, DC, USA, 2005. IEEE Computer Society.

[12] M. N. Huhns and M. P. Singh. Service-oriented computing: Key concepts and principles. *IEEE Internet Computing*, 9(1):75–81, Jan. 2005.

[13] D. Kung, J. Gao, P. Hsia, F. Wen, Y. Toyoshima, and C. Chen. Change impact identification in object oriented software maintenance. pages 202–211, 1994.

[14] M. Lee, A. J. Offutt, and R. T. Alexander. Algorithmic analysis of the impacts of changes to object-oriented software. In *Proceedings of the Technology of Object-Oriented Languages and Systems (TOOLS 34'00)*, TOOLS '00, pages 61–, Washington, DC, USA, 2000. IEEE Computer Society.

[15] D. Leffingwell and D. Widrig. *Managing software requirements: a unified approach.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2000.

[16] D. Leffingwell and D. Widrig. *Managing Software Requirements: A Use Case Approach.* Pearson Education, 2 edition, 2003.

[17] N. Madhavji. Environment evolution: The prism model of changes. *IEEE Transactions on Software Engineering*, 18:380–392, 1992.

[18] E. Newcomer and G. Lomow. *Understanding SOA with Web Services (Independent Technology Guides).* Addison-Wesley Professional, 2004.

[19] B. G. Ryder and F. Tip. Change impact analysis for object-oriented programs. In *Proceedings of the 2001 ACM SIGPLAN-SIGSOFT workshop on Program analysis for software tools and engineering*, PASTE '01, pages 46–53, New York, NY, USA, 2001. ACM.

[20] H. Xiao, J. Guo, and Y. Zou. Supporting change impact analysis for service oriented business applications. In *Proceedings of the International Workshop on Systems Development in SOA Environments*, SDSOA '07, pages 6–, Washington, DC, USA, 2007. IEEE Computer Society.