Change impact analysis for service based business processes

Yi Wang
University Of Technology Sydney

Jian Yang
General Research Institute for Non Ferrous Metals, Ministry of Science & Technology, China, Macquarie University

Weiliang Zhao
University of Wollongong, wzhao@uow.edu.au


Publication Details
Abstract—Change management is critical and challenging in
the development and maintenance of service-based applications
and information systems due to the distributed and dynamic
nature of services. This paper proposes an approach for
facilitating the change impact analysis in a service oriented
environment. This research focuses on a typical scenario that
multiple services are supported by a single business process.
The change impact is analyzed based on the study of the
dependency between services and business processes. Types
of changes and change impact patterns are identified on the
foundation of a service oriented business process model. These
change types and the impact patterns can be used to enable
the analysis of change propagation of the business process and
associated services. Algorithms for computing impact scopes of
changes are provided.

Keywords-service oriented computing; web service; change
management; service evolution; business process

I. INTRODUCTION

Change management is a challenging issue in the service
oriented environment due to the distributed and dynamic
characteristics of services [1]. In the service oriented com-
puting (SOC) paradigm, business processes and services
are coupled with each other when services expose business
functions of business processes [2]. Due to various reasons
such as business regulations and application environments,
services and business processes may change from time to
time. A specific service change usually affect the associated
business processes and services and a change occurred in
a business process often causes various levels of impact on
the associated services.

Let us consider a purchase scenario as an example. A
purchase process receives an order from a buyer, checks the
stock availability, and sends confirmation to the buyer. If an
order has been received, the purchase process sends the bill
to the buyer. The payment is processed by a finance institute.
The buyer is issued with an invoice after the payment. The
purchase process handles the shipment of the goods with the
support of a shipping company. In this scenario, the purchase
process interacts with three partners as a buyer, a finance
institute, and a shipping company. In the SOC environment,
these three partners take part in the purchase process by
invoking the corresponding services exposed by the purchase
process. Each service is an external view of the purchase
process from a specific partner. Private tasks of the purchase
process, such as checking stock availability and processing
an invoice are hidden from its partners.

The above scenario exemplifies a typical case of the
coupling relation between services and business processes
when multiple services are supported by a single business
process. If a change occurs in any of the services, it
will definitely affect the business process and may have
impact on the other services associated with this business
process. If a change occurs in the business process, the
change may affect the services that are associated with this
business process. For supporting the change management,
it is necessary to identify different types of changes and
change impact patterns and have effective mechanisms to
deal with them.

Current researches about service change management
are mainly concentrated on managing changes for BPEL
processes [3], [4] and Web services [5], [6], [7] respectively.
The complex dependencies between services and business
processes are neglected by the existing works when dealing
with changes for service-based applications and information
systems. As the first step to study on these dependencies,
the research focuses on the change impact analysis in service
oriented context and highlights one of typical cases that
multiple services are supported by a single business process.

The goal of this research is to manage the various types
of changes associated with services and business processes
by developing effective mechanisms for controlling changes
and minimizing their impact on other services and business
processes. The proposed approach is based on the service
oriented business process model which captures a typical
type of dependency between services and business processes.
A number of change impact patterns are specified on the
basis of the identified various types of service changes and
business process changes. The change propagation within
service based business processes can be analysed with the
help of these identified change impact patterns.

This research specifies change impact patterns based on
the proposed service oriented business process model and the
various types of changes of services and business processes.
Each change impact pattern provides an understanding of
the direct impact scope of a change, the cause of the
change, the effect of the change on the services and the
associated business process. The change impact patterns
provide intermediate results in the analysis process and they can be reused in the development and maintenance of service based information systems. This paper also provides how to analyse the change propagation of services and business processes with the support of the change taxonomy and the change impact patterns. The algorithms are defined to calculate the impact scope of a specific change.

The remainder of this paper is structured as follows. Section 2 describes the service oriented business process model. In Section 3, the identified change types relating to services and business processes are presented. Based on these change types, Section 4 presents the ten change impact patterns and the functions for calculating direct impact scopes of a service change and a process change. Section 5 discusses the change propagation and the actual impact scopes of a service change and a process change. Section 6 reviews the related work. Section 7 concludes the paper.

II. SERVICE ORIENTED BUSINESS PROCESS MODEL

This section presents the service oriented business process model. The proposed model contains two layers as a process layer and a service layer. The two layers and their relations will be described as follows.

A. Process Layer

The process layer contains internal processes. An internal process consists of a control flow schema and an information flow schema.

Control Flow Schema

The control flow schema consists of a set of activities and the control relations associated with them. Activities are categorized into private activities (p-activities) and communication activities (c-activities) [8], [9]. P-activities are invisible to partners. C-activities exchange information with partners. C-activities are further categorized into four types: receive, send, receive/reply, invoke/receive.

Definition 1 (Control flow schema) The control flow schema of an internal process is defined as a 3-tuple: 

\[ CFS = (A, C, E) \]

where:

- \( A = \{a_1, \ldots, a_n\} \) is a set of activities. For \( a \in A \), if \( a \) is a c-activity, \( a.\text{partner} \) denotes the partner that \( a \) intends to interact with;

- \( C = \{\oplus\text{split}, \oplus\text{join}, \otimes\text{split}, \otimes\text{join}\} \) is the set of control connectors, where \( \oplus \) represents the and connector while \( \otimes \) denotes the xor connector;

- \( E = \{e_1, \ldots, e_m\} \) is a set of directed edges associated activities and connectors.

Figure 1(a) shows the control flow schema of a purchase process which intends to interact with two partners: a buyer and an financial institute.

Information Flow Schema

The information flow schema defines how data is transferred between activities. The information flow is the key for understanding the data dependency between activities which is indispensable for analyzing change impact. We define the information flow schema which is similar to the data flow schema defined in [10].

Let \( D = \{d_1, \ldots, d_n\} \) be a set of data elements associated with the internal process. Every activity \( a \) has input and output parameters, denoting as \( \text{InPARs}(a) \) and \( \text{OutPARs}(a) \) parameters respectively. A data connection is defined as \( dc = (d, a, \text{par}, \text{mode}) \) where \( d \in D \), \( a \in A \), \( \text{par} \in \text{InPARs}(a) \cup \text{OutPARs}(a) \), and \( \text{mode} \in \{\text{read}, \text{write}\} \).

Definition 2 (Information flow schema) Let \( \text{CLF} = (A, C, E) \) be the control flow schema of an internal process, the information flow schema is the set of all data connections \( \text{IFS} = \{dc_1, \ldots, dc_m\} \).

Figure 1(a) shows the part of the information flow schema of the purchase process. The dashed arrows are the data connections. After receiving the order from a buyer, the activity receive order writes \( d_1 \) with the information: customer order. The data connection is \( (d_1, \text{receive order}, \text{customer order}, \text{write}) \). Then send acknowledgement reads from \( d_1 \) as input parameter and sends an acknowledgement to the buyer. The data connection is \( (d_1, \text{send acknowledgement}, \text{customer order}, \text{read}) \). Data dependency between activities is derived from data connections. There is a data dependency between receive order and send acknowledgement as the input of the latter is retrieved from the output of the former. We say receive order depends on send acknowledgement in terms of data.
Definition 3 (Activity data dependency) Let \( CLS = (A,C,E) \) be the control flow schema of an internal process, \( IFS = \{dc_1, \ldots, dc_m\} \) be the information flow schema, and \( D = \{d_1, \ldots, d_n\} \) be the set of data elements associated with the internal process. For \( a_i, a_j \in A \), \( a_j \) depends on \( a_i \) in terms of data, denoting as \( a_i \rightarrow^{D} a_j \) iff: (1) \( \exists dc_x, dc_y \in IFS \) such that \( dc_x = (d,a_j,par_x,write) \), \( dc_y = (d,a_i,par_y,read) \), where \( d \in D \), \( par_x \in OutPARS(a_j) \) and \( par_y \in InPARS(a_i) \), and (2) \( a_j \) precedes \( a_i \) in \( CLS \).

To sum up, an internal process is defined by a 2-tuple: \( IP = (CLF,IFS) \), where \( CLF \) is the control flow schema while \( IFS \) the information flow schema.

B. Service Layer

The service layer contains services supported by an internal process. Every service is an external view of the internal process from the viewpoint of a partner. Observable behavior rather than a list of operations needs to be provided in a service interface [11], [12]. We define a service as a set of operations and the invocation relations associated with the operations.

Definition 4 (Service) A service is defined by a 2-tuple \( s = (O,T) \), where:

- \( O = \{o_1, \ldots, o_n\} \) is a set of operations. Each operation \( o_i \in O \) is associated with a set of messages;
- \( T \subseteq O \times O \) is a set of control relations between operations. Each transition \( t = (o_i, o_j) \in T (o_i, o_j \in O) \) denotes the invocation from operation \( o_i \) to operation \( o_j \). We call \( o_i \) the origin operation of \( t \) while \( o_j \) the destination operation. For \( t \in T \), \( c(t) \) denotes the transition constraint on \( t \). \( t \) happens immediately after the execution of the origin operation. If \( c(t) \neq \emptyset \), \( t \) occurs when \( c(t) \) is evaluated to be true.

Figure 1 shows two services supported by the purchase process. Figure 1(b1) is the service \( s_b \) for the buyer and (b2) is the service \( s_f \) for the financial institute.

C. Relations Between Process Layer and Service Layer

The internal processes and the services are coupled with each other. An internal process may support multiple services. Each activity is associated with an operation that implements the task specified by the activity. Operations that are associated with c-activities are exposed to the corresponding partners. The operations relating to a same partner are grouped as a service. A service is an external view of the internal process from the viewpoint of a partner. Transition sequences of operations reflect the control relations between corresponding activities. For example, in Figure 1(b1), there is a transition sequence \( receive \ PayInfo, send \ invoice \) in service \( s_b \). The activity \( Receive \ PayInfo \) must preceed \( Send \ Invoice \) in the purchase process.

III. TAXONOMY OF CHANGES

On the basis of the proposed model, two major types of changes are identified as: the service change and the process change. The classification of changes provides the foundation for change impact analysis. This section briefly introduces the identified changes.

Two major types of service changes are identified, i.e., the operation change and the transition change (cf. Figure 2). The operation change is further classified into operation existence change and operation granularity change. Operation existence change occurs due to adding or removing operations from a service. Operation granularity change refers to the change that existing operations are reorganized into different grained operations. The operation granularity change is classified into three sub types: asynchronous operation granularity change (AOGC), synchronous operation granularity change (SOGC) and complex operation granularity change (COGC). A transition change refers to the modifications of transitions between operations. Rather than discussing primitive changes, such as adding or removing a transition, we identify seven types of high level transition changes which can be accomplished by applying primitive changes. We believe high level transition changes are more meaningful for describing real world transition changes in a service.

Figure 3 shows the classification of process changes based on the proposed service oriented business process model. The change classification is identified with the consideration to facilitate the impact analysis in the service oriented environment.

IV. CHANGE IMPACT

This section presents the change impact patterns. A change impact pattern captures the effect of a specific change. The change impact patterns provide a rich intermediary results which help reduce the complexity of change
effect are identified: (i) serially inserting a c-activity between two successively executed c-activities with conditions; (ii) inserting a c-activity in parallel to existing c-activities without conditions; (iii) inserting a c-activity in parallel to existing c-activities with conditions; and (iv) inserting a c-activity between two successively executed c-activities without conditions.

A. Change Impact Patterns

A change impact pattern includes: (1) the description of the pattern, (2) the cause of the impact, (3) the direct impact scope, and (4) the change effect on the process or the services. The first five impact patterns describe the impact on the internal process caused by service changes. We describe the impact on the internal process using the abstract control relations, which specify the required structures between c-activities. Three types of abstract control relations are defined as: abstract precedence relation, abstract conditional relation and abstract parallel relation (cf. Figure 4).

Impact pattern 1: Insert a C-activity The Insert a C-activity pattern describes that a c-activity needs to be added to the internal process. This type of impact is caused by adding an operation to a service. Four sub types of effect are identified: (i) serially inserting a c-activity between two successively executed c-activities without conditions; (ii) serially inserting a c-activity between two successively executed c-activities with conditions; (iii) inserting a c-activity in parallel to existing c-activities without conditions; and (iv) inserting a c-activity in parallel to existing c-activities with conditions. Figure 5 is an example of the impact described in (iv). Figure 5(a) is a service change and (b) is the impact on the internal process caused by the service change. The required control relations between the affected activities: $a_i$, $a_j$, $a_k$ and $a_x$ are specified by the abstract control relations.

Impact pattern 2: Remove a C-activity The Remove a C-activity pattern describes that c-activities need to be removed from the internal process. This type of impact is caused by deleting operations in a service.

Impact pattern 3: Replace C-activities The Replace C-activities pattern describes that c-activities need to be replaced by another c-activity or a set of structured c-activities. This type of impact is caused by changing operation granularity. The effect on the internal process is complicated due to the various cases of operation granularity change. The effect is classified into four sub types: (i) replacing a c-activity by another c-activity; (ii) replacing a c-activity by a set of activities. (iii) replacing a set of c-activities by another c-activity; and (iv) replacing a set of activities by another set of c-activities. Figure 6 shows an example of the impact described in (ii). The synchronous operation $a_x$ needs to be changed to asynchronous operations $a_{y1}$, $a_{y2}$ and $a_{y3}$. In such a case, the associated activity $a_x$ must be replaced with a set of structured c-activities (cf. Figure 6(b)).

Impact pattern 4: Move C-activities The Move C-activities pattern describes that existing c-activities need to be reordered. This type of impact is caused by transition sequence change, such as TSOC, SPTSC and PSTSC. The effect is classified into two sub types: (i) serially moving c-activities; and (ii) parallel moving c-activities. Figure 7
Deleting c-activities or replacing c-activities require changes to services. The cause of this impact is the deletion of c-activities. The impact described in (i). The activity a\(_2\) is replaced by two send activities a\(_{y1}\) and a\(_{y2}\) and one receive activity a\(_{w3}\). This process change makes the complex granularity change in the corresponding service (cf. 10).

Impact pattern 6: Add Operations

The Add Operation pattern describes that operations need to be added to the corresponding services. The impact is caused by inserting a c-activity or replacing an existing c-activity in the internal process. The insertion of a c-activity or replacement of existing c-activities with new c-activities requires that operations are added to the corresponding services. The effect is classified into four sub types: (i) sequentially adding an operation in between operations without constraints; (ii) adding an operation sequentially in between operations with constraints; (iii) adding an operation in parallel to existing operations without constraints; and (iv) adding an operation in parallel to existing operations with constraints. Figure 9 is an example of the impact described in (ii). An activity a\(_x\) relating to partner p1 is inserted between two activities in the xor structure. This change requires that the operation a\(_x\) associated with a\(_x\) is added between a\(_x1\) and a\(_x2\) with constraints in service s\(_{p1}\) (cf. Figure 9(b)).

Impact pattern 7: Remove Operations

The Remove Operations pattern describes that operations need to be removed from services. The cause of this impact is the deletion of c-activities or the replacement of c-activities in the internal process. Deleting c-activities or replacing c-activities require changes to services. Replacement of c-activities may incur various type of operation granularity change in the services. The impact is categorized into three sub types: (i) AOGC; (ii) SOGC; and (iii) COGC. Figure 10 is an example of the impact described in (ii). The send/receive type activity a\(_x\) is replaced by two send activities a\(_{y1}\) and a\(_{y2}\) and one receive activity a\(_{w3}\). This process change makes the complex granularity change in the corresponding service (cf. 10).

Impact pattern 8: Change Transition Sequence

The Change Transition Sequence pattern describes that transition sequences of the corresponding services need to be reordered. The impact is caused by moving activities, parallelizing activities or sequencing activities in the internal process. The impact is classified into three sub types: (i) TSO; (ii) SPTSC; and (iii) PSTSC. Figure 11 is an example of the impact described in (i). The activity a\(_x\) is serially moved into a conditional branch. The process change causes the transition sequence to be reordered in the corresponding services.

Impact pattern 9: Change Operation Granularity

The Change Operation Granularity pattern describes that operation granularity needs to be modified. The impact is caused by replacing c-activities in the internal process. Replacement of activities may incur various type of operation granularity change in the services. The impact is categorized into three sub types: (i) AOGC; (ii) SOGC; and (iii) COGC. Figure 12 is an example of the impact described in (ii). The send/receive type activity a\(_x\) is replaced by two send activities a\(_{y1}\) and a\(_{y2}\) and one receive activity a\(_{w3}\). This process change makes the complex granularity change in the corresponding service (cf. 10).
where a service fragment changes S services IP, S, pchange change with o in process elements: change. is a set of service fragments O in the change region before any reaction is taken to handle change schange that belong to the service IP, S, schange scope of a change. Two functions: FuncDISS and FuncDISP are defined, which compute the direct impact scope of a service change and a process change respectively.

**Definition 5** FuncDISS is the function: FuncDISS : IP, S, schange PE. The input of the function includes: (i) an internal process IP with CFS = (A, C, E) as the control flow schema and IFS = {dc1, ..., dc_m} as the information flow schema, (ii) the set of services S = {s1, ..., sn} supported by IP, and (iii) a service change schange with a set of involved operations O_c = {o1, ..., o_r}. The output of the FuncDISS is a set of process elements: PE = {pe1, ..., per}, where pe_i (i = 1, ..., r) consists of: (i) the c-activity a that is associated with o_i, (ii) the set of activities: A_depend = {a1, ..., a_s} C A, where a_1 ∈ A_depend such that a → D a_j, and (iii) ∀dc_j ∈ IFS such that dc_j is associated with a and A_depend.

Algorithm 1 calculates the direct impact scope of a service change.

**Definition 6** FuncDISP is the function: FuncDISP : IP, S, pchange SF. The input of the function includes: (i) an internal process IP = (A, C, E), (ii) the set of services S = {s1, ..., sn} supported by IP, and (iii) a process change pchange, with a set of directly affected operations. As the operations in O_c may belong to different services, we use O_c[i] ⊆ O_c to denote the set of operations that belong to the service s_i. The output of the FuncDISP is a set of service fragments SF = {sf1, ..., sf_r} (r ≤ n), where a service fragment sf_j consists of: (i) all operations in O_c[i] are in sf_j, (ii) a transition t if t takes any operation in O_c[i] as the origin operation or the destination operation, and (iii) an operation o_x if o_x is the origin operation or the destination operation of transitions in sf_j but is not included in O_c[i].

Algorithm 2 calculates the direct impact scope of a process change.

V. CHANGE PROPAGATION

This section discusses the issue of change propagation. Change propagation refers to the ripple effect of change between associated services and business processes. Figure 13(a) is the change propagation of a service change. If a service change c happens, c is mapped to an impact pattern which captures the change effect on the internal process. Based on the impact pattern, a reaction is taken to handle c. Consequently further process changes pce_1, ..., pce_r happen. Each process change pce_i is mapped to an impact pattern which shows the effect on the services. Change reactions are taken and thus cause further changes on the services. Figure 13(b) is the change propagation of a process change. If a process change c occurs, c is mapped to the impact patterns that describe the change effect on corresponding services. For each impact pattern, a reaction is taken and thus causes further changes on the services. The following example shows the change propagation of a service change.
Algorithm 2 DISP

Input \( IP, S = \{s_1, \ldots, s_n\}, pchange \)
Output \( SF = \{s_{f_1}, \ldots, s_{f_r}\}(r \leq n) \)
Let \( A_c \) be the set of activities involved in \( pchange \)
\( O^i \leftarrow \emptyset (i = 1, \ldots, n) \)
for all \( a \in A_c \) do
  if \( a \) is the c-activity relating to \( p_i \) then
    \( O^i \leftarrow O^i \cup \{o\} \) (\( o \) is associated with \( a \))
  end if
end for
for all \( s_{f_i}(i = 1, \ldots, n) \) do
  for all \( a \in O^i \) do
    for all \( t_j \) that associated with \( o \) do
      if \( o_x \) is associated with \( t_j \& a \neq o \) then
        \( s_{f_i} \leftarrow s_{f_i} \cup \{a_x\} \)
      end if
    end for
  end for
end for
\( SF \leftarrow \emptyset \)
for all \( s_{f_i}(i = 1, \ldots, n) \) do
  if \( s_{f_i} \neq \emptyset \) then
    \( SF \leftarrow SF \cup \{s_{f_i}\} \)
  end if
end for
return \( SF \)

Example Change propagation

Figure 14 exemplifies the change propagation of a service change. A service change, Adding Conditional Transition Sequence (ACTS), occurs in service \( s_{p3} \), where the operation \( o_2 \) is invoked conditionally. Figure 14(b) is the internal process. Based on the impact pattern, a c-activity \( a_2 \) needs to be embedded in a conditional branch with conditions specified by the service change. The reaction for handling the change ACTS is to add an xor structure which includes activity \( a_2 \) and a process fragment 1.2. The process fragment 1.2 includes a c-activity \( a_y \) relating to service \( s_{p2} \). In addition, the data required by the xor connector is obtained from service \( s_{p2} \) and thus an activity \( a_x \) is inserted before the xor structure. As shown in the Figure 14(c), the reaction causes further changes to service \( s_{p2} \) and \( s_{p3} \).

Based on the change propagation, the actual impact scope of a change can be derived. The actual impact scope of a process change is its direct impact scope. The function FuncAISS is defined to calculate the actual impact scope of a service change.

Definition 7 FuncAISS is the function: Func\( AISS : IP, S, schange, PCHANGE \rightarrow PE, SFS \). The input of the function includes: (i) an internal process IP, (ii) the set of services \( S = \{s_1, \ldots, s_n\} \) supported by IP, (iii) a service change \( schange \), and (iv) a set of process changes \( PCHANGE = \{pchange_1, \ldots, pchange_r\} \) that are caused by reactions for handling the service change \( schange \). The output of Func\( AISS \) includes: (i) a set of process elements \( PE \), where \( PE = FuncDISP(IP, S, schange) \), and (ii) a list of service fragments \( SFS = \{SF_1, \ldots, SF_h\} \), where \( SF_i = FuncDISP(IP, S, pchange_i)(i = 1, \ldots, h) \).

VI. RELATED WORK

Change management has been widely studied in the context of workflow and information systems [13], [14], [10], [15]. Many existing works focus on the evolution of process schemata and the strategies of process instance migration. Process adaptation and flexibility are also studied for reacting to changes [16], [17]. Existing works about service change management cover service interface and business
protocol adaptation [18], [19], [20], change management for BPEL process orchestration and choreography [3], [4], virtual enterprise [7], and business protocol evolution [21] and service evolution [5], [6]. These works only concentrate on the change issues of services or business processes respectively. They are inadequate to address the issues of change management in the SOC environment where change management becomes more critical and challenging due to the distributed and dynamic natures of services and business processes.

There are various types of dependencies between services and business processes in service based applications and information systems. Change analysis and change reactions are difficult due to the possible complex dependencies between services and business processes. This work proposes an approach of the change impact analysis for a typical case that multiple services are supported by a single business process.

VII. CONCLUSIONS

This paper reports our work on change impact analysis for service based business processes. A typical case of the dependencies between services and business processes is highlighted when multiple services are supported by a single business process. Ten change impact patterns are specified and the functions for calculating the impact scope of a specific change are defined. Change propagation between associated services and processes is studied on the foundation of the identified change impact patterns. These change impact patterns are intermediate results in the analysis process and they can be reused in development and maintenance of applications and information systems. They are helpful to reduce the complexity of change analysis. For the future work, we will carry out extensive investigation about complicated structures and dependencies between services and business processes.

REFERENCES