Managing changes 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
Managing Changes for Service Based Business Processes

Yi Wang, Jian Yang and Weiliang Zhao
Department of Computing, Faculty of Science,
Macquarie University
NSW, 2109 Australia
yi.wang, jian.yang, weiliang.zhao@mq.edu.au

Abstract—In this paper, we propose an approach to deal with the change management for service oriented business processes. Beyond existing work, the proposed approach highlights the dependencies between services and business processes. A service oriented business process model is devised for capturing the major characteristics of change management in service oriented context. The taxonomy for the changes associated with services and business processes is presented. A set of change impact patterns are specified and the functions for calculating impact scopes of a change are defined. With the help of the change taxonomy and the change impact patterns, the ripple effect of changes of the business processes and services can be clearly analyzed. This research provides a step progress for change management in the service oriented environment.

Keywords—Service oriented computing; Web services; change management; service evolution; business process

I. INTRODUCTION

The Service Oriented Computing (SOC) facilitates the low-cost and rapid composition of loosely coupled software applications. Service oriented models are introduced to replace or extend traditional process models in order to develop flexible business processes and to realize inter-organization integration [1]. Heterogenous services can be integrated in distributed applications across organization boundaries. The business processes and services are subject to change and variation arising from both the external and internal requirements of organizations. A specific change usually forces a ripple effect of changes in business processes and services due to various types of dependencies among business processes and services. The change management, which is a traditional problem in IT, is becoming more challenging in the service oriented paradigm [2].

There are a lot of researches about change management in the context of workflow and process change [3], [4], [5], [6]. In particular, the process flexibility is studied in details [7], [8]. These researches focus on the processes only without considering the characteristics of services. They are inherently inadequate to support change management goals in service oriented environment. Quite a few researches have been published about service evolution [9], [10], service adaptation [11], [12], change management for service protocols [13], [14] and BPEL processes [15], [16]. These researches only consider the features of services without considering the associated business processes.

The services and business processes are coupled with each other in the real world. The dependencies between services and business processes will be crucial for the change management in the service environment because changes may introduce ripple effects for services and business processes. In particular, a single business process may support multiple services. The change management becomes complicated due to the dependencies between the business process and different services. In next section, an example will be given about how a business process supports multiple services. The example shows readers the motivation of this research.

In this paper, we propose an approach to deal with the change management for service oriented business processes. A service oriented business process model is proposed for capturing the major characteristics of change management in the service oriented context. This research targets techniques for understanding and identifying various types of changes, analysing the impact of changes, and facilitating the evolution of services and business processes in a service oriented environment. This paper makes the following contributions:

- The taxonomy for the changes associated with services and business processes is presented. The operation changes and transition changes are identified as the two major types of service changes. The operation changes are further classified into existence changes and granularity changes. The process changes are classified based on the activities and the constraints and relationships of involved activities.
- A set of change impact patterns are specified and the functions for calculating impact scopes of a service change and a process change is defined. With the help of identified impact patterns and specified impact scope, a ripple effect of changes in the business process and services can be clearly analyzed.
- The complexity of the dependency network of business processes to invoked services can be managed with the support of the proposed set of definitions about service oriented business process model, the taxonomy of changes for service oriented business processes, the identified change impact patterns, and the change impact scope.

The rest of the paper is organized as follows. Sec.2
provide a motivating example. Sec.3 introduces the service oriented business process model. In Sec.4 the identified various types of changes associated with services and business processes are presented. Sec.5 briefly discusses the change impact patterns. Sec.6 reviews the related work and Sec.7 concludes the paper.

II. A Motivating Example

Let us consider a typical purchase scenario. A purchase process can receive an order from a buyer, check the stock availability, and send confirmation to the buyer. If an order is accepted, the purchase process will send the bill to the buyer. The payment is processed by a finance institute. The buyer will be issued with an invoice after the payment. In the meantime, the purchase process handles the shipment of the goods with the support of a shipping company. The buyer will be notified with a shipping schedule. 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 interact with the purchase process by invoking the correspondent services exposed by the process. The three services are exposed for interacting with the buyer, the financial institute, and the shipping company (Figure 1). 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.

In the real world, there are cases which are similar to the above scenario where multiple services are supported by a single business process. The dependencies between the services and the process make change management complicated and challenging. On one hand, when a change occurs in any of the services, the change may impact on the business process and the other services. On the other hand, when a change occurs in the business process, the change may impact on the services that are associated with this business process. The changes of a business process and multiple services will affect with each other. The above scenario provides the basic requirements and motivation for our approach about the change management for service oriented business processes. The typical case that multiple services are supported by a single business process will be highlighted in this research.

III. SERVICE ORIENTED BUSINESS PROCESS MODEL

This section describes a service oriented business process model. This model contains two layers as a process layer and a service layer. The details of the two layers and the relationships between them will be discussed.

A. Process Layer

The process layer contains business processes, which will be referred to as internal processes in the following part of this paper. An internal process consists of a set of activities and the control relations associated with them. Following the convention in [17], [18], activities of internal processes are categorized into private activities (p-activities) and communication activities (c-activities). P-activities belong to internal processes only and they are invisible to partners. C-activities look after communication tasks for exchanging information with partners. C-activities are further categorized into four types: receive, send, receive/reply, invoke/receive.

Definition 1 (Internal process) An internal process is defined as a 3-tuple: $IP = (A, C, E)$, where:
- $A = \{a_1, \ldots, a_n\}$ is a set of activities. Each activity $a \in A$ is associated with an operation $o$ that implements the activity. If $a$ is a c-activity, $a.partner$ refers to the trading partner that $a$ intends to interact with;
- $C = \{\oplus_{split}, \oplus_{join}, \otimes_{split}, \otimes_{join}\}$ is a 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 2(a) is the purchase process which intends to interact with two partners: a buyer and a financial institute.

B. Service Layer

The service layer contains services that are supported by the internal process. Every service is an external view of the internal process from the view point of a particular partner. A service interface exposes the observable behavior rather than a list of operations [19], [20]. We define a service as a set of operations and the invocation relations associated with the operations.

Definition 2 (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 c-activity. Every operation has 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 $t$. If $c(t) = \emptyset$, $t$ happens immediately after the execution of the origin operation. Transition $t$ occurs only if $c(t)$ is evaluated to be true.

Figure 2 shows two services supported by the purchase process. Figure 2(b1) is the service $s_b$ for the buyer which contains six operations and five transitions. Among the transitions, $t_2$ and $t_3$ are governed by constraints $c(t_2)$ and $c(t_3)$ respectively, which means that after the invocation of operation send acknowledgement, whether send reject order or send bill will be executed depends on the value of $c(t_2)$ and $c(t_3)$. For simplicity, messages associated with these operations are not shown in the figure.

C. Relations Between Process Layer and Service Layer

In this sub section, we discuss how services and internal processes are related with each other. The relations between the two layers are indispensable to understand the impact of a particular change. 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 correspondent partners. The operations that are related to one partner are grouped as a service. For example, the service $s_b$ contains six operations relating to buyers. Transitions between operations are based on the control flows associated with correspondent activities. For example, transition $t_3$ in $s_b$ is obtained from the control flow between activities send acknowledgement and send bill, which are both c-activities for interacting with a buyer. As the activity send bill is in the conditional branch, $t_3$ is governed by constraint $c(t_3)$ that is obtained from the conditions of the xor connector. Thus, the service for a particular partner is abstracted from the internal process by exposing operations associated with the c-activities relating to the partner and generating transitions between operations from the control relations between corresponding activities.

A service is an external view of the internal process from the view point of a particular partner. Transition sequences of operations reflect the abstract control relations between associated activities in the internal process. For example, in Figure 2(b1), there is a transition sequence receive PayInfo $t_5$ send invoice in service $s_b$. The activity Receive PayInfo must precede Send Invoice in the purchase process. There are other activities between the two activities but are invisible to the buyer. We identify three types of abstract control relations: the abstract precedence relation, the abstract conditional relation, and the abstract parallel relation (cf. Figure 3).

IV. TAXONOMY OF CHANGES

Services changes and process changes are categorized into various types in this section. These change types will be the foundation for the analysis of change impact discussed in the next section.

A. Service Changes

Two major types of service changes are identified: operation change and transition change. Operation change is further classified into operation existence change and operation granularity change.

1) Operation Existence Change: An existence change occurs due to adding or removing operations in a service. There are four possible ways of adding an operation as shown in Figure 4: sequentially adding an operation without constraints, sequentially adding an operation with constraints, adding an operation in parallel to existing operations without constraints, and adding an operation in parallel to existing operations with constraints.
2) Operation Granularity Change: Operation granularity change refers to the change that existing operations are re-organized into different grained operations. Changing granularity of operations is a service design concern in order to meet business requirements from both the organization and the partners. We consider asynchronous operations with only input messages and synchronous operations with both input and output messages [18], [15]. We call the input and output messages of an operation the input and output parameters. We assume that two operations having the same input and output parameters perform the same functionalities [21]. Based on the assumption, the operation granularity change is discussed by analyzing the changes on input and output parameters. We focus on the change of information structure that an operation processes. The information structure of an operation refers to the basic data types an operation can handle. Two functions are defined to retrieve the basic data types from the operation parameters. Suppose `dataType` is the basic XML data types used by operation definition, the functions are defined as: `InInfo : O \rightarrow \phi(dataType)` and `OutInfo : O \rightarrow \phi(dataTypes)`. `InInfo` takes an operation as the input and generates the set of basic data types of the input parameter, whereas `OutInfo` takes an operation as the input and generates the set of basic data types of the output parameter.

We identify three major types of operation granularity changes as: asynchronous operation granularity change (AOGC), synchronous operation granularity change (SOGC) and complex operation granularity change (COGC). The three types granularity change will be described in details in the follows.

**AOGC** refers to the granularity change of asynchronous operations. We classify AOGC into three sub types: **AOGC type 1 one-to-one change**, **AOGC type 2 one-to-many/many-to-one change**, and **AOGC type 3 many-to-many change**.

**AOGC type 1 one-to-one change** describes that one asynchronous operation `o_x` is modified to `o'_x`. The following relations between `o_x` and `o'_x` exist:

1. `InInfo(o_x) \subset InInfo(o'_x)`, which means that `o_x` is modified by accepting more data types as its input parameter;
2. `InInfo(o_x) \supset InInfo(o'_x)` which means that `o_x` is modified by requiring less data types as its input parameter;
3. the above two conditions do not hold and `InInfo(o_x) \cap InInfo(o'_x) \neq \emptyset`.

**AOGC type 2 one-to-many/many-to-one change** defines the granularity change between an asynchronous operation and a set of asynchronous operations. **One-to-many change** covers the case that an operation is split into a set of operations. **Many-to-one change** covers the case that multiple operations are merged into one operation. We discuss the one-to-many change in details. The many-to-one change is similarly defined. Let `o_x` be an asynchronous operation, `o_x` is split into a set of operations `O_Y = \{o_{y_1}, \ldots, o_{y_i}\}`, where \(\forall o_{y_i} \in O_Y, \text{InInfo}(o_{y_i}) \cap \text{InInfo}(o_x) \neq \emptyset\). The possible relations between `o_x` and `O_Y` are:

1. `InInfo(o_x) = InInfo(O_Y)`, which means `o_x` is split into functionally equivalent finer operations;
2. `InInfo(o_x) \subset InInfo(O_Y)`, which means `o_x` changes to a set of operations `O_Y` which process more information structures;
3. `InInfo(o_x) \supset InInfo(O_Y)` which means `o_x` changes to a set of operations `O_Y` which accept less information structures;
4. the above three conditions do not hold and `InInfo(o_x) \cap InInfo(O_Y) \neq \emptyset`. The relation in (4) describes that `O_Y` covers only part of the functionality of `o_x`. Moreover, `O_Y` processes information that is not accepted by `o_x`.

**AOGC type 3 many-to-many change** describes the granularity change between two sets of asynchronous operations. Let `O_X = \{o_{x_1}, \ldots, o_{x_i}\}` be a set of asynchronous operations, `O_Y` can be reorganized into a set of operations `O_Y = \{o_{y_1}, \ldots, o_{y_i}\}`, where \(\forall o_{x_i} \in O_X, \exists o_{y_j} \in O_Y\), such that `InInfo(o_{x_i}) \cap InInfo(o_{y_j}) \neq \emptyset`. There are the following possible relations between `O_X` and `O_Y`:

1. `InInfo(O_X) = InInfo(O_Y)`, which means that operations in `O_X` are redesigned into a set of operations `O_Y`. Although `O_X` and `O_Y` remain functionally equivalent, each operation in `O_X` is modified by its input parameters;
2. `InInfo(O_X) \subset InInfo(O_Y)` which means that the set of operations `O_X` are changed to the set of operations `O_Y` and `O_Y` process more data types;
3. `InInfo(O_X) \supset InInfo(O_Y)` which means the set of operations `O_X` are changed to the set of operations `O_Y` and `O_Y` process less data types;
4. the above conditions do not hold and `InInfo(O_X) \cap InInfo(O_Y) \neq \emptyset`. The relation in (4) describes that `O_Y` retains only part of the functionality of `O_X` and has functionality that is not provided by `O_X`.

**SOGC** refers to granularity change of synchronous operations. SOGC is classified into three types: **SOGC type 1 one-to-one change**, **SOGC type 2 one-to-many/many-to-one change**, and **SOGC type 3 many-to-many change**.

![Figure 4. Operation existence change](image-url)
SOGC type 1 one-to-one change describes the granularity change between two synchronous operations. Let \( o_x \) be a synchronous operation, \( o_x \) is changed to another synchronous operation \( o'_x \) by modifying its input and output parameters. For instance, \( \text{InInfo}(o_x) = \text{InInfo}(o'_x) \) and \( \text{OutInfo}(o_x) \subseteq \text{OutInfo}(o'_x) \), which indicates that \( o'_x \) accepts the same input as \( o_x \) but generates output with more data types in its information.

SOGC type 2 one-to-many/many-to-one change describes the granularity change between a synchronous operation \( o_x \) and a set of synchronous operations \( O_Y = \{o_{y_1}, \ldots, o_{y_l}\} \), where \( \forall o_{y_j} \in O_Y \) such that \( (\text{InInfo}(o_x) \cup \text{OutInfo}(o_x)) \cap (\text{InInfo}(o_{y_j}) \cup \text{OutInfo}(o_{y_j})) \neq \emptyset \). For instance, the relation between \( o_x \) and \( O_Y \) during the change is: \( \text{InInfo}(o_x) = \text{InInfo}(O_Y) \) and \( \text{OutInfo}(o_x) \subseteq \text{OutInfo}(O_Y) \). This relation indicates that \( o_x \) and \( O_Y \) accept the same input parameters whereas \( O_Y \) generates output with more data types in its information than \( o_x \).

SOGC type 3 many-to-many change defines the granularity change between two sets of synchronous operations. Let \( O_X = \{o_{x_1}, \ldots, o_{x_s}\} \) be a set of synchronous operations, \( O_X \) can be reorganized into a different set of synchronous operations \( O_Y = \{o_{y_1}, \ldots, o_{y_l}\} \) with different granularity, where \( \forall o_{x_i} \in O_X, \exists o_{y_j} \in O_Y \) such that \( (\text{InInfo}(o_{x_i}) \cup \text{OutInfo}(o_{x_i})) \cap (\text{InInfo}(o_{y_j}) \cup \text{OutInfo}(o_{y_j})) \neq \emptyset \).

COGC refers to the change that involves both synchronous and asynchronous operations. COGC is classified into: COGC type 1 asynchronous-to-synchronous/ synchronous-to-asynchronous change and COGC type 2 mixed change.

**COGC type 1 asynchronous-to-synchronous/ synchronous-to-asynchronous change** refers to the granularity change from asynchronous operations to synchronous operations and vice versa. We define the grain change from asynchronous operations to synchronous operations. Let \( O_X = \{o_{x_1}, \ldots, o_{x_s}\} \) be a set of asynchronous operations and \( O_Y = \{o_{y_1}, \ldots, o_{y_l}\} \) be a set of synchronous operations. There is a granularity change of COGC type 1 iff: \( \forall o_{x_i} \in O_X, \exists o_{y_j} \in O_Y, \) such that \( \text{InInfo}(o_{x_i}) \cap (\text{InInfo}(o_{y_j}) \cup \text{OutInfo}(o_{y_j})) \neq \emptyset \). Various relations between \( O_X \) and \( O_Y \) exist. For instance, \( \text{InInfo}(O_X) \subseteq \{\text{InInfo}(O_Y) \cup \text{OutInfo}(O_Y)\} \), which means \( O_Y \) covers all the functionality of \( O_X \) and provides extra functionality than \( O_X \).

**COGC type 2 mixed change** describes the change between two sets of operations, each set contains both synchronous and asynchronous operations. Let \( O_X = \{o_{x_1}, \ldots, o_{x_s}\} \) and \( O_Y = \{o_{y_1}, \ldots, o_{y_l}\} \) be sets of operations. \( O_X \) is categorized into two sets: \( O_X^S \) and \( O_X^A \), where \( O_X^S \) consists of the asynchronous operations while \( O_X^A \) contains the synchronous operations. Similarly \( O_Y \) is classified into \( O_Y^S \) and \( O_Y^A \). There is a granularity change of COGC type 2 iff all the following conditions are satisfied:

1. \( \forall o_{x_i} \in O_X^A, \exists o_{y_j} \in O_Y, \) such that \( \text{InInfo}(o_{x_i}) \cap (\text{InInfo}(o_{y_j}) \cup \text{OutInfo}(o_{y_j})) \neq \emptyset \) (if \( o_{y_j} \in O_Y^S \) or \( \text{InInfo}(o_{x_i}) \cap (\text{InInfo}(o_{y_j}) \cup \text{OutInfo}(o_{y_j})) \neq \emptyset \) (if \( o_{y_j} \in O_Y^A \));

2. \( \forall o_{x_i} \in O_X^S, \exists o_{y_j} \in O_Y \), such that \( (\text{InInfo}(o_{x_i}) \cup \text{OutInfo}(o_{x_i})) \cap (\text{InInfo}(o_{y_j}) \cup \text{OutInfo}(o_{y_j})) \neq \emptyset \) (if \( o_{y_j} \in O_Y^S \) or \( (\text{InInfo}(o_{x_i}) \cup \text{OutInfo}(o_{x_i})) \cap (\text{InInfo}(o_{y_j}) \cup \text{OutInfo}(o_{y_j})) \neq \emptyset \) (if \( o_{y_j} \in O_Y^A \)).

3. **Transition Change:** Transition change refers to the modifications of transitions. Rather than discussing primitive changes, such as adding or removing a transition, we identify seven types of high level transition changes. We believe the high level transition changes are more meaningful for describing real world service behavioral changes. As shown in Figure 5, the seven types of transition changes are: Transition Sequence Order Change (TSOC), Sequential to Parallel Transition Sequence Change (SPTSC), Parallel to Sequential Transition Sequence Change (PSTSC), Adding Conditional Transition Sequence (ACTS), Removing Conditional Transition Sequence (RCTS), Adding Looping Transition Sequence (ALTS), and Removing Looping Transition Sequence (RLTS).

### B. Process Changes

A broad variety of change patterns have been proposed in the workflow systems for managing process changes [22], [6]. The classification of process changes shown in Figure 6 is based on our proposed service oriented business process model described in Sec.3. It modifies the classification proposed by [6] according to the specific requirements for the change management for service oriented business processes. This classification will be used for facilitating the change impact analysis. The basic element of change pattern defined in [6] is process fragment. A process fragment is defined as a sub process that consists of structured activities with a single node in and a single node out. Different from [6], we use activity as the basic element when identifying process change. We believe that activities are linked with services more closely than the concepts of process fragments. Let us consider a process change insert a process fragment. If the process fragment contains no c-activities, such insertion causes no impact on the associated services. The change is private and invisible to the partners. If the process fragment
includes more than one c-activities that associated with different partners, the insertion causes correspondent services to change accordingly.

V. CHANGE IMPACT PATTERN

We will use change impact patterns to capture the effect of service changes and process changes. A change impact pattern can provide a solid foundation for judging possible reaction to changes. The separation of change impact and change reaction is helpful to reduce the complexity of challenging change management tasks for service oriented business processes and more intermediate results in the analysis process can be reused.

Figure 7 shows an overview of our identified change impact patterns. The impact patterns 1-5 describe the impact on the internal process by service changes, and the impact patterns 6-10 describe the impact of the services by process changes. Each impact pattern includes: (1) the description of the impact, (2) the cause of the impact, (3) the direct impact scope, and (4) the change effect on the services or the internal process. Due to the page limitation we can not discuss the ten impact patterns in details. To provide an example, Figure 8 shows the impact pattern 1: Insert a C-activity.

In order to specify the impact of a specific change, we define FuncDISS for calculating the direct impact scope of a service change and FuncDISP for calculating the direct impact scope of a process change. The direct impact scope of a service change includes the affected activities of the internal process. The direct impact scope of a process change includes the affected operations and transitions of associated services.

Definition 3 FuncDISS is the function: FuncDISS : IP, S, $s_{change}$ $\rightarrow$ PE. The input of the function includes: (i) an internal process $IP = (A, C, E)$, (ii) the set of services $S = \{s_1, \ldots, s_n\}$ supported by $IP$, and (iii) a service change $s_{change}$ with a set of involved operations $O_c = \{o_1, \ldots, o_r\}$. The output of the FuncDISS is a set of process elements: $PE = \{p_{e_1}, \ldots, p_{e_r}\}$, where $p_{e_i} (i = 1, \ldots, r)$ consists of: (i) the c-activity $a$ that is associated with $o_i$, (ii) the set of activities, denoting as $A_{depends}$, that $a$ depends on in terms of data.

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

Definition 4 FuncDISP is the function: FuncDISP : IP, S, $s_{change}$ $\rightarrow$ SF. The input of the function includes: (i) an internal process $IP = (A, C, E)$, (ii) the set of services $S = \{s_1, \ldots, s_n\}$ supported by $IP$, and (iii) a process change $s_{change}$, with a set of directly affected operations. As the operations in $O_r$ may belong to different services, we use $O_r \subseteq O_1$ to denote the set of operations that belong to the service $s_i$. The output of the FuncDISP is a set of service fragments $SF = \{s_{f_1}, \ldots, s_{f_r}\}$ ($r \leq n$), where a service fragment $s_{f_i}$ consists of: (i) all operations in $O_r$ are in $s_{f_i}$, (ii) a transition $t$ if $t$ takes any operation in $O_r$ as the origin operation or the destination operation, and (iii) an operation $o_r$ if $o_r$ is the origin operation or the destination operation of transitions in $s_{f_i}$ but is not included in $O_r$.

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

VI. RELATED WORK

Without being related to SOC, change management has been studied in a wide range of research areas such as

---

The text continues with more detailed explanations and examples, but the key points are set out in a clear and structured manner, focusing on the impact of change on services and processes, and the methodologies for calculating and defining these impacts.
Figure 8. Change impact pattern 1

Algorithm 1: Change impact pattern 1

<table>
<thead>
<tr>
<th>Input: $P, S = {s_1, \ldots, s_n}$, $\rho_{change}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output: $P_E$, $PE = \emptyset$</td>
</tr>
</tbody>
</table>

1. Let $O_i = \{a_1, \ldots, a_m\}$ be the set of operations involved in $\rho_{change}$.
2. If $a_i \in A$ do:
   
   - $PE = PE \cup \{a_i\}$

3. If $a_i$ depends on $a_j$ in terms of data then:
   
   - $PE = PE \cup \{a_j\}$

4. If $a_i$ is the e-activity associated with $o_i$ do:
   
   - $PE = PE \cup \{o_i\}$

5. if $a_i$ depends on $a_j$ do:
   
   - $PE = PE \cup \{a_j\}$

6. return $PE$

Algorithm 2: Change impact pattern 2

<table>
<thead>
<tr>
<th>Input: $P, S = {s_1, \ldots, s_n}$, $\rho_{change}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output: $S_F$, $SF = \emptyset$</td>
</tr>
</tbody>
</table>

1. Let $O_i = \{a_1, \ldots, a_m\}$ be the set of operations involved in $\rho_{change}$.
2. If $a_i \in A$ do:
   
   - $SF = SF \cup \{a_i\}$

3. If $a_i$ depends on $a_j$ in terms of data then:
   
   - $SF = SF \cup \{a_j\}$

4. If $a_i$ is the e-activity associated with $o_i$ do:
   
   - $SF = SF \cup \{o_i\}$

5. if $a_i$ depends on $a_j$ do:
   
   - $SF = SF \cup \{a_j\}$

6. return $SF$

change management has been studied from different aspects such as workflow systems [7], [8], [9], and process oriented organization [2]. A change management framework has been studied in [10].

change management in service oriented environment. The research reported in this paper shows our approach for filling the gaps described above. Our change management solution...
aims to control the ripple effect of changes of business processes and services. In particular, our approach highlights the typical case that a business process supports multiple services from view points of different partners of a business process.

VII. Conclusion

Beyond existing work, our proposed approach for change management focuses the dependencies between business process and services in service oriented environment. The taxonomy for changes of business processes and services has been established based on the service oriented business model. A set of change impact patterns have been identified. Functions for deriving impact scopes of a service change and a process change have been defined. The proposed approach can be used as the foundation to analyze and control the ripple effect of changes of business processes and services. This research targets guidelines and a generic solution for the change management mechanisms. We have highlighted the typical case that one business process supports multiple services. We are still working to identify more typical types of dependencies between business processes and services and develop corresponding change management mechanisms.

References


