2011

Service change analyzer: An enabling tool for change management in 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

Follow this and additional works at: <https://ro.uow.edu.au/engpapers>

Part of the [Engineering Commons](#)

<https://ro.uow.edu.au/engpapers/5073>

Recommended Citation

Wang, Yi; Yang, Jian; and Zhao, Weiliang: Service change analyzer: An enabling tool for change management in service-based business processes 2011, 237-244.
<https://ro.uow.edu.au/engpapers/5073>

Service Change Analyzer: An Enabling Tool for Change Management in 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 the Service Change Analyzer (SCA) as an enabling tool for analyzing change impact in services and associated business processes. The SCA is built up based on our change management approach focusing on the analysis of dependencies between services and their supporting business processes. Our change management approach includes a service-oriented business process model, a change taxonomy, change impact patterns, and the algorithms for calculating the impact scopes of a specific change. The SCA provides developers a standard practice to change the complicated tasks of change management into a series of simple and standard procedures. The reported results in this paper make a step progress to achieve the automation of change management in the service based environment.

Keywords—Service-oriented computing; Web service; Change management

I. INTRODUCTION

In the Service-Oriented Computing (SOC), services and business processes are subject to changes and variations from time to time [1]. A service change usually requires changes applied to its supporting business processes and consequently to other services the same business processes support. Similarly a change in a business process can cause changes to the services supported by this business process. Let us consider a sales scenario as an example. A sales process receives an order from a buyer, checks the stock availability, and sends confirmation to the buyer. If an order has been received, the sales process sends bill to the buyer. The payment is processed by a finance institute. The buyer is issued with an invoice after the payment. The sales process handles the shipment of the goods through a shipping company. In this scenario, the sales process interacts with three business partners: *buyer*, *finance institute*, and *shipping company*. In the service based environment, these three partners interact with the sales process by invoking the relevant services exposed to them by this sales process. Each service is an external view of the sales process from the view point of a specific partner. This scenario exemplifies a case of the coupling relations between services and business processes when multiple services are supported by a single business process. The changes of services and their internally supporting business processes can affect each

other. Change management is critical and challenging in the context described above.

As a traditional problem in IT, change management has been studied in a wide range of research areas such as software engineering [2], distributed systems [3], database management systems [4], and information systems [5]. In particular, the change management for business processes has been extensively studied since mid 1990s [6], [7], [8], [9]. These researches mainly concentrate on business processes without considering too much on their relationships with services. They are inherently inadequate to support the change management in the service based environment. Current works on service change management are mainly focused on managing changes for BPEL processes [10], [11] and Web services [12], [13], [14]. The complex dependencies between services and business processes have not been fully investigated in the existing works, nor does any effective tool that supports change impact analysis in services and their internally supporting business processes is provided. As the first step towards study on these dependencies, this paper focuses on the change impact analysis in the context of SOC and highlights the dependent relationships between services and business processes when multiple services are supported by a single business process.

This paper presents a systematic methodology for analyzing change impact in the above described context and an enabling tool called Service Change Analyzer (SCA). The proposed methodology provides a change taxonomy for services and business processes based on the developed service-oriented business process model. A set of change impact patterns are also specified for capturing the types of change effect in services and business processes [15]. The functions for calculating the direct impact scopes of a service change and a process change are also defined. Based on these change analysis mechanisms, the SCA is built up which accepts a service change as its input and it provides the detailed analyzed results for the change impact scope and suggestions for potentially used change impact patterns. With the help of the SCA, the impact of a specific change becomes transparent and it is not necessary to analyze the impact of changes manually. The time and cost of change

management tasks can be dramatically reduced. Our change analysis approach and the enabling tool provide developers a standard practice to change the complicated change management tasks into a series of simple standard procedures. The results in this paper contribute a step progress to achieve the automation of change management in the service based environment.

The remainder of this paper is structured as follows: Section 2 reviews the related work. In Section 3, we present the change analysis approach. Section 4 provides the design details of the SCA and two running examples. In Section 5, we conclude this paper.

II. RELATED WORK

Change management has been extensively studied in the context of workflow based processes since mid 1990s [6], [7], [8], [9]. These studies mainly concentrate on evolving business processes and enabling the flexibility of business processes. Works on the evolution of workflow processes aim to allow business processes to evolve in a disciplined, controlled, and dynamic manner. The verification techniques developed in these work can be used as the theoretical foundation for the proposed approach reported in this paper. The researches on process flexibility are focused on dynamically modifying process schemas and instances at runtime in order to cope with both expected and unexpected changes. These research works of change management focus on business processes without taking services into consideration. They are inherently inadequate to support the complex tasks of change management in the service-oriented environment.

Current research works on service change management focus on the compatibility of Web services [16], [17], [18], [19], [20], [21], change management for Web services [22], [14], [10], [11], and service evolution [12], [13], [1], [21]. Most of the existing works on service compatibility investigate the incompatibilities of Web services from the aspects of service signatures and business protocols. For example, Benatallah et al. [16] study the problem of the compatibility between service protocols for the purpose of service adaptation. The incompatible types (mismatches) of service protocol are identified. Templates for designing adapters for incompatible service protocols are specified for each type of mismatch pattern. The change management for Web services is at its early stages and existing researches provide only partial solutions for change issues in relation to Web services. Wombacher [11] proposes an approach for aligning the choreographies and the orchestration automatically when there are occurrence of changes in process choreographies. The author aims to solve the problem: if the choreography of a partner changes (this change may originated from the associated orchestration of this partner), how this change affects the choreographies of other partners and in turn their orchestrations. The studies of change management for Web service compositions concentrate on the issues of detecting

Web service changes and designing effective change reactions [22], [14]. Service evolution and service versioning control are still not fully supported by the current Web service technologies. Important theorems and guidelines for service evolution management that abstracts from current Web service standards are proposed in the recent works [12], [13], [1]. Service adaption is an important area closely related to the change management in the SOC paradigm. Current researches on service adaptation are mainly focused on overcoming mismatches of service interfaces, service protocols, and behaviors of BPEL processes [16], [23], [24].

The above mentioned research works on change management in the SOC paradigm concentrate only on either service changes or process changes separately. Normally, business processes and services are coupled with each other. There may be complicated dependencies between business processes and services. Changes of a business process or a service will affect a set of other business processes and services. Change analysis and change reactions are difficult due to the possible complex dependencies between services and business processes. Unfortunately, the dependencies between services and business processes have not been fully addressed in the existing works. A number of tools for managing changes in Web service protocols [24], [21], [25], mediating service mismatches [23], and supporting business process flexibility [26] are found in the existing works. However, effective tools are still lacking for analyzing change impact in services and business processes in the above described context. The research reported in this paper presents an approach for filling the gaps mentioned above. The proposed change management methodology and the built up tool highlight a case of dependencies between services and business processes when a single business process supports multiple services.

III. CHANGE IMPACT ANALYSIS

This section presents our approach for change impact analysis in service-based business processes when multiple services are supported by a single business process.

A. Service-Oriented Business Process Model

Two layers: a process layer and a service layer are defined in the service-oriented business process model. The process layer contains business processes referred to as internal processes. The service layer consists of services supported by internal processes.

An internal process is defined by a control flow schema and an information flow schema. A 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) [17]. 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*. A control flow schema is defined as a 3-tuple: $CFS = (A, C, E)$, where $A = \{a_1, \dots, a_n\}$ is a set of activities. For $a \in A$, if a is a c-activity, $a.partner$ denotes the partner that a intends to interact with; $C = \{\oplus_{split}, \oplus_{join}, \otimes_{split}, \otimes_{join}\}$ is the set of control connectors, where $\oplus$ represents the *and* connector while $\otimes$ denotes the *xor* connector; and $E = \{e_1, \dots, e_m\}$ is a set of directed edges associated activities and connectors.

The information flow schema defines how data is transferred between activities. Let $D = \{d_1, \dots, d_n\}$ be a set of data elements associated with the internal process. Each activity a has input parameters, denoting as $InPARs(a)$, and output parameters, denoting as $OutPARs(a)$. A data connection is defined as $dc = (d, a, par, mode)$, where $d \in D$, $a \in A$, $par \in InPARs(a) \cup OutPARs(a)$, and $mode \in \{read, write\}$. An information flow schema is the set of all data connections $IFS = \{dc_1, \dots, dc_m\}$. Data dependency between activities can be derived based on the data transferring among them. For $a_i, a_j \in A$, a_i depends on a_j in terms of data, denoting as $a_i \xrightarrow{D} a_j$ iff: (1) $\exists dc_x, dc_y \in IFS$ such that $dc_x = (d, a_j, par_s, write)$, $dc_y = (d, a_i, par_t, read)$, where $d \in D$, $par_s \in OutPARs(a_j)$ and $par_t \in InPARs(a_i)$, and (2) a_j precedes a_i in CFS .

A service is described by a 2-tuple $s = (O, T)$, where: $O = \{o_1, \dots, o_n\}$ is a set of operations, and $T \subseteq O \times O$ 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.

Internal processes and 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 this 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. Transition sequences in services are based on the control relations between the corresponding activities in the internal process.

B. Change Taxonomy

Based on the proposed model, two major types of changes are identified as: service change and process change.

Two major types of service changes are identified, i.e., *operation change* and *transition change* (cf. Figure 1). 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

Figure 1. Classification of service changes.

Change Impact Patterns		
Name	Pattern description	Cause
Impact pattern 1 Insert a c-Activity	a c-activity needs to be added to the internal process	adding an operation to a service
Impact pattern 2 Remove a c-Activity	c-activities need to be removed from the internal process	deleting operations in a service
Impact pattern 3 Replace c-Activities	c-activities need to be replaced by another c-activity or a set of structured c-activities	changing operation granularity in a service
Impact pattern 4 Move c-Activities	c-activities need to be reordered	operation transition sequence change, such as TSOC, SPTSC and PSTSC
Impact pattern 5 Add, Remove or Modify Conditional Branches	xor structures need to be modified or new xor structures need to be created	transition sequence change, such as ACTS, RCTS, ALTS, and RLTS
Impact pattern 6 Add Operations	operations need to be added to corresponding services	inserting a c-activity or replacing a c-activity in the internal process
Impact pattern 7 Remove Operations	operations need to be deleted from services	deletion of c-activities or the replacement of c-activities in the internal process
Impact pattern 8 Change Operation Granularity	operation granularity needs to be modified	replacing c-activities in the internal process
Impact pattern 9 Change Transition Sequence	transition sequences of the corresponding services need to be reordered	moving activities, parallelizing activities or sequencing activities in the internal process
Impact pattern 10 Add Conditional or Looping Transition Sequence	constraints and extra transition sequences need to be added between operations	embedding activities in conditional branches

Figure 2. Change impact patterns.

different grained operations. 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.

We have defined nine major types of process changes as: insert an activity, remove an activity, move an activity, replace activities, parallelize activities, sequence activities, embed in conditional branches, embed in loop, and update conditions.

C. Change Impact Patterns

We use change impact patterns to capture the effect of service changes and process changes [15]. Figure 2 shows an overview of our identified change impact patterns. The impact patterns 1-5 describe the effect on internal processes by service changes, and the impact patterns 6-10 describe the effect on 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. To provide

Figure 3. Change impact pattern 1: Insert a c-Activity.

an example, Figure 3 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.

Definition 1 *FuncDISS* is the function: $FuncDISS : IP, S, schange \rightarrow PE$. The input of the function includes: (i) an internal process $IP = (A, C, E)$, (ii) the set of services $S = \{s_1, \dots, s_n\}$ supported by IP , and (iii) a service change $schange$ with a set of involved operations $O_c = \{o_1, \dots, o_r\}$. The output of the *FuncDISS* is a set of process elements: $PE = \{pe_1, \dots, pe_r\}$, where pe_i ($i = 1, \dots, r$) consists of: (i) the c-activity a that is associated with o_i , (ii) the set of activities, denoting as A_{depend} , that a depends on in terms of data.

Definition 2 *FuncDISP* is the function: $FuncDISP : IP, S, pchange \rightarrow SF$. The input of the function includes: (i) an internal process $IP = (A, C, E)$, (ii) the set of services $S = \{s_1, \dots, s_n\}$ 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 $O_c^i \subseteq O_c$ to denote the set of operations that belong to the service s_i . The output of the *FuncDISP* is a set of service fragments $SF = \{sf_1, \dots, sf_r\}$ ($r \leq n$), where a service fragment sf_i consists of: (i) all operations in O_c^i are in sf_i , (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_i but is not included

Figure 4. Hierarchy diagram.

in O_c^i .

IV. SERVICE CHANGE ANALYZER

The SCA is a JAVA based tool that implements the above described change management approach for analyzing the change impact when a change happens. This tool enables the change analysis in service-based business processes where multiple services are supported by a single business process. A service change is accepted by the SCA as its input and it provides detailed results for the impact scopes and suggestions of potentially used change impact pattern. These analysis results for a specific service change help developers understand the direct impact and the cascading impact in services and their supporting business processes. These results also provide the foundation for developers to determine proper change reactions for handling service changes. With the help of the SCA, the impact of a specific change becomes transparent and the process of change impact analysis can be realized automatically. The time and cost of change management tasks can be dramatically reduced. This general methodology and the enabling tool provide developers a standard practice to change the complicated change management tasks into a series of simple standard procedures. More importantly, the results in this paper make a step progress to achieve the automation of change management in the SOC paradigm. In this section, we provide the design details of the SCA including its architecture and its functional components. We also present two running examples to show the effectiveness of this tool.

A. Architecture

As shown in Figure 4, the SCA is realized by two major modules as **operation based analysis** and **transition based analysis**.

1) *Operation Based Analysis*: The operation based analysis module is divided into two modules: **change operation existence** and **change operation granularity**.

The **change operation existence** module is realized by three sub modules as: sequentially add an operation, add an operation in parallel to an existing operation, and delete an operation. The sequentially add an operation module accepts a service change with the type of sequentially adding an operation and generates the impact analysis results of this change. The add an operation in parallel to an existing operation module accepts a service change with the type of parallel adding an operation and generates the impact analysis results of this change. The delete an operation module accepts a service change with the type of deleting an operation and generates the impact analysis results of this change.

The **change operation granularity** module consists of three major sub modules as Asynchronous Operation Granularity Change (AOGC), Synchronous Operation Granularity Change (SOGC), and Complex Operation Granularity Change (COGC). The AOGC module is further divided into four sub modules as: AOGC one-to-one operation granularity change, AOGC one-to-many operation granularity change, AOGC many-to-one operation granularity change, and AOGC many-to-many operation granularity change. These modules deal with the change analysis for asynchronous operation granularity changes. The SOGC module is classified into four sub modules as: SOGC one-to-one operation granularity change, SOGC one-to-many operation granularity change, SOGC many-to-one operation granularity change, and SOGC many-to-many operation granularity change. These modules deal with the impact analysis for synchronous operation granularity changes. The COGC module is categorized into three sub modules as: COGC asynchronous-to-synchronous operation granularity change, COGC synchronous-to-asynchronous operation granularity change, and COGC mixed operation granularity change. These modules handle the impact analysis for operation granularity changes involving both asynchronous operations and synchronous operations.

2) *Transition Based Analysis*: As shown in Figure 4, the transition based analysis is divided into seven sub modules as: change transition sequence order, change sequential transitions to parallel transitions, change parallel transitions to sequential transitions, add conditional transition sequences, remove conditional transition sequences, add looping transition sequences, and removed looping transition sequences. These seven modules accept the corresponding service transition changes and provide the change impact analysis results. For instance, the change transition sequence order module accepts a service change with the type of transition sequence order change and generates the impact analysis results for this service change.

B. Running Examples

In this section, we provide two running examples to show the effectiveness of our change analyzer. The two examples cover the **operation based analysis** and **transition based analysis** respectively.

1) *Example for Operation Based Analysis*: First, a user needs to select the service that he/she wants to change. The SCA provides an interface for users to browse existing services, which contains a service list and a tabbed panel: Operation and Transition. The service list shows the existing services that are retrieved from the databases. In our example, three services as: buyer service, payment service, and shipper service exist and are listed. When a service is selected, its operations and transitions will be displayed in the tabbed panel below as trees. The root node is the selected service and the operations are displayed as children nodes. Each operation node has three children nodes as operation type, input messages, and output messages. For example, the operation **send acknowledgement** is an asynchronous operation denoted as "A". The input message of this operation is "order acknowledgement".

When a user wants to change the selected service, he/she can right click the mouse in the area of panels showing service operations and transitions. When the user right clicks the mouse in the service operation area, the defined types of service change related to operations are popped out as menus (cf. Figure 5(a)).

Based on the change taxonomy for services, the change types associated with service operations provided for users are: Sequentially add an operation, Add an operation in parallel to an existing operation, Delete an operation, Change granularity of asynchronous operation, Change granularity of synchronous operation, and Complex operation granularity change. The Change granularity of asynchronous operation menu has four submenus as: AOGC one-to-one granularity change, AOGC one-to-many granularity change, AOGC many-to-one granularity change, and AOGC many-to-many granularity change. Similarly, the Change granularity of synchronous operation menu also has four submenus as: SOGC one-to-one granularity change, SOGC one-to-many granularity change, SOGC many-to-one granularity change, and SOGC many-to-many granularity change. The Complex operation granularity change has three submenus as: COGC asynchronous-to-synchronous operation granularity change, synchronous-to-asynchronous operation granularity change, and mixed operation granularity change.

A user can choose a change type he/she wants to apply on the selected service. When a specific change type is chosen, the SCA provides the corresponding interface for specifying the change. In this example, the *Add an operation in parallel to an existing operation* change is selected and the corresponding interface is presented (cf. Figure 5(b)).

Suppose a user wants to add an operation **send dispatch notification** to the buyer service. The new operation must

Figure 5. (a) Choose change types of service operations; (b) specify operation change.

be invoked after the operation **send acknowledgement** and before the operation **receive PayInfo**. In addition, the new operation can be executed parallel with the operation **send bill**. The constraints for executing the new operation **send dispatch notification** is “order is confirmed and the goods are dispatched”. The operation **send dispatch notification** is an asynchronous operation and its input messages include “customer order” and “dispatch notice from shipper”. To make the above operation change, the user needs to choose the item *Add an operation in parallel to an existing operation* from the popped up menu. Then an interface for specifying this change will be provided. There are two parts of information that need to be specified by the user: indicating where the new operation needs to be inserted and specifying the details of the new operation. As shown in Figure 5(b), the user can select the origin operation and destination operation from the drop-down lists. In addition, the user must specify which operation the new operation can be executed in parallel. If the invocation of this new operation is conditional, the user can specify the conditions in the constraints textfield. The name, type, input and output messages of the new operation need to be specified in the corresponding textfields.

After the user input the information of a specific change, he/she can click the “Analyze Change Impact” button at the bottom of the frame. The function of analyzing service change impact is based on our proposed change analysis approach. The results of the change impact analysis of the SCA include the direct impact scope of the input service change, the potentially used impact pattern, and a description for the change effect based on the specified change information. Figure 6 shows the output: the results of the change impact analysis for the specified change “Add an operation in parallel to an existing operation”.

2) *Example for Transition Based Analysis:* Transitions of a service are displayed as a tree, of which the root node is the selected service and the children nodes are transitions of that service. Each transition node has three

Figure 6. Impact analysis results for *add operation in parallel to an existing operation*.

children nodes as origin operation, destination operation, and constraints. When the user wants to make a change to transitions, he/she can right click the mouse in the area of service transitions. Figure 7(a) shows the popped out menu of transition change types. Based on the taxonomy of service changes, we have designed seven menu items as: Change transition sequence order, Change sequential transitions to parallel transitions, Change parallel transitions to sequential transitions, Add conditional transition sequences, Remove conditional transition sequences, Add looping transition sequences, and Remove looping transition sequences.

Suppose a user wants to change the order of the transition sequence in the buyer service: (**send bill**, **receive PayInfo**,

Figure 7. (a) Choose change types of service transitions; (b) specify transition change.

Figure 8. Impact analysis results for change transition sequence order.

send invoice) to (send invoice, send bill, receive PayInfo). To make this transition change, the user needs to choose the item *Change transition sequence order* from the change menu. Then an interface for specifying this change will be provided. The user can select the transition sequence he/she wants to change from the drop-down list. The selected transitions will be displayed in the below testarea. The user must specify the new order of the selected transition sequence in the corresponding textarea (cf. Figure 7(b)). Figure 8 shows the results of the change impact analysis for the specified change “Change transition sequence order”.

V. CONCLUSION

In this paper, we present a systematic methodology and an enabling tool: Service Change Analyzer (SCA) for change impact analysis in services and their internally supporting business processes. A case of the dependencies between services and business processes is highlighted when multiple services are supported by a single business process. The SCA accepts service changes as its input and it can give the detailed analyzed results for the change impact scopes and suggestions for potentially used change impact patterns. The theoretical foundations of this tool are provided by our proposed service-oriented business process model, the identified types of changes that can happen to services and business processes, the specified change impact patterns that capture the various types of impact on services and business processes, and the functions for calculating the direct impact scopes of service changes as well as process changes. The functionalities of this tool are realized by two major modules as: operation based analysis and transition based analysis. Two running examples are presented which show how to analyze impact of an operation change and a transition change through the SCA. With the help of the SCA, the impact of a specific change becomes transparent. The time and cost of change management tasks can be dramatically reduced. For the future work, we will carry out extensive investigation on complex structures and dependencies between services and business processes and possibility of change automation.

REFERENCES

- [1] M. P. Papazoglou, “The challenges of service evolution,” in *Proceedings of the 20th International Conference Advanced Information Systems Engineering*, Montpellier, France, June 16-20 2008, pp. 1–15.
- [2] M. M. Lehman, “Program evolution,” *Information Processing & Management*, vol. 20, no. 1, pp. 19–36, 1984.
- [3] J. Kramer and J. Magee, “The evolving philosophers problem: Dynamic change management,” *IEEE Transactions on Software Engineering*, vol. 16, no. 11, pp. 1293–1306, 1990.

- [4] C. Yu and L. Popa, "Semantic adaptation of schema mappings when schemas evolve," in *Proceedings of the 31st VLDB Conference*, Trondheim, Norway, 2005, pp. 1006–1017.
- [5] C. Dorn and S. Dustdar, "Interaction-driven self-adaptation of service ensembles," in *Proceedings of the 22nd International Conference on Advanced Information Systems Engineering*, Hammamet, Tunisia, June 6-9 2010, pp. 393–408.
- [6] W. M. P. van der Aalst and T. Basten, "Inheritance of workflows: an approach to tackling problems related to change," *Theor. Comput. Sci.*, vol. 270, no. 1-2, pp. 125–203, 2002.
- [7] F. Casati, S. Ceri, B. Pernici, and G. Pozzi, "Workflow evolution," *Data & Knowledge Engineering*, vol. 24, pp. 211–238, 1998.
- [8] G. Joeris and O. Herzog, "Managing evolving workflow specifications," in *Proceedings of the 3rd IFCIS International Conference on Cooperative Information Systems*, New York City, New York, USA, August 20-22 1998, pp. 310–321.
- [9] M. Reichert and P. Dadam, "Adept_{flex}-supporting dynamic changes of workflows without losing control," *J. Intell. Inf. Syst.*, vol. 10, no. 2, pp. 93–129, 1998.
- [10] S. Rinderle, A. Wombacher, and M. Reichert, "Evolution of process choreographies in dychor," in *Proceedings of the 2006 OTM Confederated International Conferences on CoopIS, DOA, GADA, and ODBASE*, Montpellier, France, October 29 - November 3 2006, pp. 273–290.
- [11] A. Wombacher, "Alignment of choreography changes in BPEL processes," in *Proceedings of the International Conference on Services Computing (SCC)*, Bangalore, India, 2009, pp. 1–8.
- [12] V. Andrikopoulos, S. Benbernou, and M. P. Papazoglou, "Managing the evolution of service specifications," in *Proceedings of the 19th International Conference on Advanced Information Systems Engineering (CAiSE)*, Montpellier, France, 2008, pp. 359–374.
- [13] —, "Evolving service from a contractual perspective," in *Proceedings of the 20th International Conference on Advanced Information Systems Engineering (CAiSE)*, Amsterdam, the Netherlands, 2009, pp. 290–304.
- [14] X. Liu, C. Liu, M. Rege, and A. Bouguettaya, "Semantic support for adaptive long term composed services," in *Proceedings of the 2010 IEEE International Conference on Web Services (ICWS 2010)*, Miami, Florida, USA, July 5-10 2010, pp. 267–274.
- [15] Y. Wang, J. Yang, and W. Zhao, "Change impact analysis for service based business processes," in *IEEE International Conference on Service-Oriented Computing and Applications, SOCA 2010, 13-15 December 2010, Perth, Australia*, 2010, pp. 1–8.
- [16] B. Benatallah, F. Casati, D. Grigori, H. R. Nezhad, and F. Toumani, "Developing adapters for web services integration," in *Proceedings of the 17th International Conference on Advanced Information Systems engineering (CAiSE)*, Porto, Portugal, 2005, pp. 415–429.
- [17] M. Dumas, B. Benatallah, and H. R. M. Nezhad, "Web service protocols: Compatibility and adaptation," *IEEE Data Eng. Bull.*, vol. 31, no. 3, pp. 40–44, 2008.
- [18] A. Martens, S. Moser, A. Gerhardt, and K. Funk, "Analyzing compatibility of bpel processes," in *Proceedings of the Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services*, Guadeloupe, French Caribbean, 19-25 February 2006, p. 147.
- [19] H. R. M. Nezhad, B. Benatallah, A. Martens, F. Curbera, and F. Casati, "Semi-automated adaptation of service interactions," in *Proceedings of the 16th International Conference on World Wide Web*, Banff, Alberta, Canada, May 8-12 2007, pp. 993–1002.
- [20] S. Ponnekanti and A. Fox, "Interoperability among independently evolving web services," in *Proceedings of the 2004 ACM/IFIP/USENIX International Middleware Conference*, Toronto, Canada, October 18-20, 2004, pp. 331–351.
- [21] S. H. Ryu, F. Casati, H. Skogsrud, B. Benatallah, and R. Saint-Paul, "Supporting the dynamic evolution of web service protocols in service-oriented architectures," *ACM Transactions on the Web*, vol. 2, no. 2, p. Article 13, April 2008.
- [22] M. S. Akram and A. Bouguettaya, "Managing changes to virtual enterprises on the semantic web," in *Proceedings of the 5th International Conference on Web Information Systems Engineering (WISE)*, Brisbane, Australia, 2004, pp. 472–478.
- [23] M. Dumas, M. Spork, and K. Wang, "Adapt or perish: Algebra and visual notation for service interface adaptation," in *Proceedings of the 4th International Conference on Business Process Management*, Vienna, Austria, 2006, pp. 65–80.
- [24] W. Kongdenfha, R. Saint-Paul, B. Benatallah, and F. Casati, "An aspect-oriented framework for service adaptation," in *Proceedings of the 2nd International Conference on Service-Oriented Computing (ICSOC)*, New York, USA, 2006, pp. 15–26.
- [25] H. Skogsrud, B. Benatallah, F. Casati, and F. Toumani, "Managing impacts of security protocol changes in service-oriented applications," in *Proceedings of the 29th International Conference on Software Engineering (ICSE 2007)*, Minneapolis, MN, USA, May 20-26 2007, pp. 468–477.
- [26] A. Hallerbach, T. Bauer, and M. Reichert, "Capturing variability in business process models: The provop approach," *Software Process: Improvement and Practice*, vol. (Accepted for Publication), 2010.