Adaptability of Service Based Workflow Models: the "Chained Execution" Architecture

Saida Boukhedouma, Zaia Alimazighi, Mourad Chabane Oussalah, Dalila Tamzalit

Adaptability of Service Based Workflow Models: The "Chained Execution" Architecture

Saida Boukhedouma[1,2], Zaia Alimazighi[1], Mourad Oussalah[2], and Dalila Tamzalit[2]

[1] USTHB- FEI- Departement of Computer Science, LSI Laboratory, ISI Team
El Alia BP n°32, Bab Ezzouar, Algiers, Algeria
{Sboukhedouma,zalimazighi}@usthb.dz
[2] Nantes University, LINA Laboratory, MODAL Team
2, Rue de la Houssinière, BP 92208, 44322 – Nantes, cedex 3- France
{Mourad.oussalah,Dalila.tamzalit}@univ-nantes.fr

Abstract. Business processes are frequently subject to changes which must be supported by process models and systems implementing them. This paper deals with adaptability of Inter-Organizational Workflow (IOWF) process models based on services. It states conceptually, typical adaptations that can be operated on IOWF models obeying to the chained execution architecture. IOWF models are described through the concepts of service and orchestration function expressed using basic control flow operators. Thus, operations of adaptation turn to modification of services and transformation of orchestration functions describing the model. We particularly distinguish evolvable adaptation leading to expansion of the cooperation and/or the global functionality of the process.

Keywords: IOWF, Chained execution, Service, Cooperation pattern, Orchestration function, Adaptation, Evolution.

1 Introduction

The B2B cooperation was initially supported by concepts and tools of *Inter-Organizational workflow* (IOWF) [1] and more recently by *Service Oriented Architectures* (SOA) and web services [2]. Also, many research works have been directed towards the combination of these technologies for the development of collaborative business applications. These last implement two kinds of cooperation: *ad-hoc cooperation* appropriate for non-durable cooperation and process models not completely defined at build time; or *structured cooperation* which is suitable for durable cooperation and clearly defined process models at build time.

In our research work, we are interested in structured cooperation supported by the concept of IOWF. In [1], generic architectures of IOWF have been defined; we talk about the *capacity sharing*, the *chained execution*, the *subcontracting*, the *case transfer*, the *extended case transfer* and the *loosely coupled WF*. We consider these architectures as basis of our research work because they cover a wide range of business processes since they express the different ways in which businesses can

W. Abramowicz et al. (Eds.): BIS 2012, LNBIP 117, pp. 96–107, 2012.

cooperate together. However in their initial form, these architectures were subject to criticisms because of their rigidity and the difficulty to adapt business processes to support changes. Then, our idea is to propose *cooperation patterns* based on services suitable to the basic architectures defined in [1], using a *SOA based approach*. According to constraints relative to IOWF architecture, this last can be implemented through *global orchestration* or *distributed local orchestrations* of services. *Global orchestration* means that services of different partners are orchestrated using a global WF process implemented at one site; where *distributed local orchestrations* mean that services of each partner are orchestrated by a WF process implemented locally. The goal behind the use of SOA is to obtain process models flexible enough to ease their adaptation because services are loosely coupled components and platform independent.

This paper deals with *adaptability* of IOWF process models suitable to structured cooperation; we focus particularly on adaptability relative to the control flow perspective. Also, according to various reasons of adaptation, we distinguish several types, then we talk about *perfective adaptation* [3] in case of improvement of the process in order to meet the client's requirements, we talk about *adaptive adaptation* [3] in case of new constraints to take into account and we talk about *corrective adaptation* [3] if we need to correct errors in the process model. In our case, we globally talk about *adaptation of process models*. Another reason of adaptation is the evolution of process models called *evolvable adaptation* that we perceive through two perspectives: expansion of process *functionalities* and expansion of *cooperation*; we globally talk about *evolution of process models*.

The present work focuses on the *chained execution* architecture which connects two or more WFs in *sequential* manner. The paper describes the corresponding cooperation pattern based on services, states conceptually typical adaptations that can be operated on IOWF process models and describes the transformation of the orchestration function for each kind of adaptation.

For the rest of the paper, Section 2 presents some related works and explains the motivation of our work. Section 3 synthesizes the necessary background to understand the paper. Section 4 describes the *chained execution pattern* based on services and illustrates the concept of *orchestration function*. Section 5 and 6 describe respectively the different operations of adaptation and evolution of IOWF process models. Section 7 concludes the paper and talks about future works.

2 Related Works and Motivation

Many research works deal with the combination of WF, SOA and web services technologies for the development of flexible business collaborative applications [4], [5], [6]. This had a great impact in promoting B2B relationships since several approaches and platforms have been proposed to support business cooperation using WF and SOA. In *structured* cooperation for example, we can cite some approaches like CoopFlow [7], CrossFlow [8], CrossWork [9], Pyros [10] and e-Flow [11].

Also, flexibility is an important propriety to be satisfied by business processes and their systems allowing them to support changes. Even if some approaches like CoopFlow, Pyros and e-Flow provide *internal adaptation* of workflows without compromising the coherence of the global process, a large number of the proposed solutions are not flexible enough because they are closely coupled with the platforms. So for any changes, they impose to re-adapt the interfaces and to newly build the structure of interaction. Moreover, WF flexibility is perceived at two complementary levels: (i) at the *system level*, the flexibility defines the ability of WFMS (WF management system) to face unexpected and erroneous situations [12], [13], [14]. (ii) at the *level of process models* that defines the ability of a process model to be adaptable, evolvable and reusable. For that, many research works have been proposed describing different techniques such as adaptation patterns [15], [16], [17], rule-based adaptation patterns [18] and constraint-based modeling [19].

The goal of this paper is to deal with *adaptability of IOWF process models* based on services especially obeying to the *chained execution* architecture. First, we introduce the concept of *cooperation pattern* that we define through two dimensions: the partitioning of the process among the partner's sites and the control of execution. Then, we express this cooperation pattern using *SOA approach* in order to deal with IOWF models easily adaptable. The use of SOA is motivated by the fact that services are loosely coupled components, easily invoked through their interfaces, business oriented and platform independent and SOA paradigm supports integration, reuse and composition of services.

3 Basic Definitions and Concepts

3.1 Definition and Architectures of IOWF

An IOWF can be defined as a manager of activities involving two or more workflows *autonomous*, possibly *heterogeneous* and *interoperable* in order to achieve a common business goal [1].

In [1], generic architectures of IOWF have been defined to support structured cooperation; we talk about the *capacity sharing*, the *chained execution*, the *subcontracting*, the *case transfer*, the *extended case transfer* and the *loosely coupled WF*. These architectures are characterized according to two main dimensions: the *partitioning of the process* and the *control of execution*. The partitioning of the process defines the way in which the process fragments of IOWF are distributed among the partner's sites (*process* partitioning) and the location of process instances at runtime (*instance* partitioning). The second dimension which is the *control of execution* defines the manner in which the execution of process instances is managed by the systems of partners. The control is *centralized* if the execution of process instances is delegated to one system that also manages all interactions between the systems of partners. The control is *decentralized* if the execution of instances is distributed among the systems of all partners and each system manages itself its interactions with other systems. We say that a control is *hierarchized* if each system manages its own WF and there is one principal system that controls interactions with

one or more secondary systems. In some cases, the control can be a *mixture* of previous modes. The *chained execution* architecture supports a model of cooperation that connects two or more business partners, each of which implements its own WF process. Workflows implied in the cooperation are executed in *sequential.* The results of execution of *WFi* are input data of *WFi+1*. In this architecture, we have *process partitioning* since each partner implements a fragment of the global WF and *instance partitioning* because at each moment a process instance is at one location; the control of execution is *decentralized*.

3.2 IOWF Meta-model, Adaptability and Evolutivity

An IOWF process model is defined by a set of *WF* fragments and a *cooperation pattern* (see Fig. 1). The cooperation pattern defines a specific architecture; it links two or more WF through a set of *interaction points.* Each WF is attached to a *partner*, manipulates *data* and is submitted to *condition* of control flow. A cooperation pattern is defined through the two dimensions of IOWF: the *partitioning of the process* and the *control of execution*. Through the concepts of the meta-model, the IOWF model covers four main axes: *process* (concepts of *IOWF*, *WF*, *condition* and *cooperation pattern*), *organization (*concept of *partner)*, *data* and *interaction (*concept of *interaction point).* Consequently, we can affirm that the constraints of flexibility in IOWF model are not limited to one axis, but cover all axes that define it. However, the flexibility is mainly reflected in the *process* and *interaction* axes although it involves and also draws on other levels – data and organization.

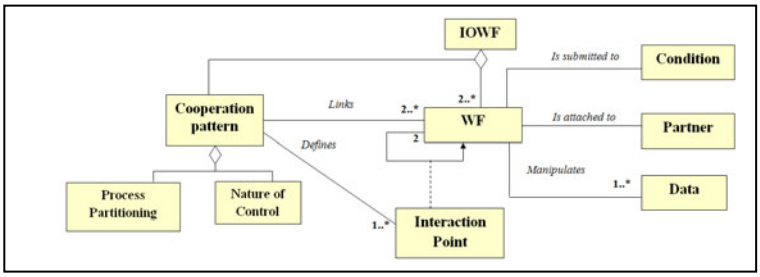

Fig. 1. Generic meta-model of IOWF

IOWF adaptability: An IOWF model is *adaptable* if one or more of the entities -*WF*, *conditions*, *data* and *interaction points* - composing it can be modified without affecting the global functionality and the cooperation (circle of partners).
IOWF evolutivity: An IOWF model is *evolvable* if it allows expansion of the global *functionality* or expansion of *cooperation* (additional business partners and so additional WF fragments).

As already said, we focus on the *chained execution* architecture of IOWF. For that, we describe the corresponding *cooperation pattern* (called "*chained execution pattern*") based on services in order to deal with IOWF models easily adaptable and evolvable. Then, we introduce the concept of *orchestration function.*

4 Cooperation Pattern and Orchestration Function

To define a cooperation pattern suitable to a specific architecture of IOWF, the question is to decide which parts of the WF process should be encapsulated within services in order to abstract them and to invoke them from outside. Specifically, *it is to encapsulate a WF process or a sub-process in a service.* In the following, we present the *chained execution* pattern.

4.1 The "Chained Execution" Pattern Based on Services

For the chained execution architecture, we propose to *entirely* encapsulate WF of each partner within a service that means service *Si* encapsulates *WFi* provided by partner *i*. Process instances are executed according to the *sequence* of services implemented (see Fig. 2). Thus, the first service (*S1*) of the sequence is triggered by an external event (the occurrence of a new instance), the other services (Si+1) of the sequence, each of which is triggered by the service (Si) that precedes it.

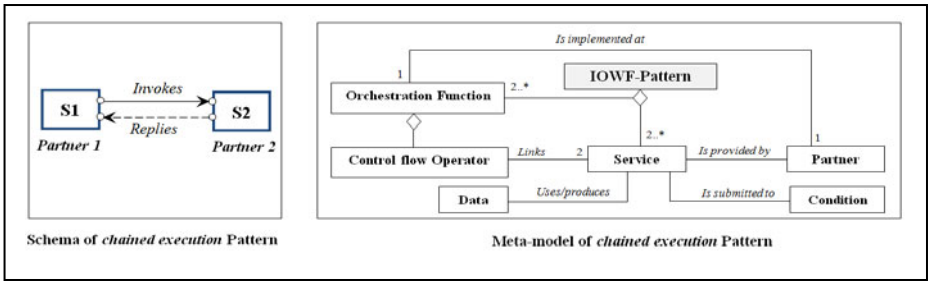

Fig. 2. Schema and meta-model of the "Chained Execution" Pattern

We can say that this architecture is implemented as *choreography* of services with *decentralized control* because services of several partners interact directly together without need to central orchestrator. Also, a reply to the service invoker can be facultative, hence the dotted arrow on the schema. The *chained execution* pattern is described through the meta-model on the right of Fig. 2, using UML notation.

At internal level, services *S1* and *S2* can be implemented as composite services encapsulating WFs of partner1 and partner2; it means that each internal activity of *WFi* is implemented as a *local service Sij*. Then, we propose implementation of a local orchestration function at each partner where maintaining a *decentralized control* of execution in the IOWF (see Fig. 3). The local orchestrator of partner *i* has to receive *input data* (through a service ***Sini***) from another orchestrator to *invoke its local composite service (Si)* with this input data and then to invoke service *S* of the next partner by sending *results* (output) of its local service through service ***Souti***; this is implemented at each partner of the IOWF.

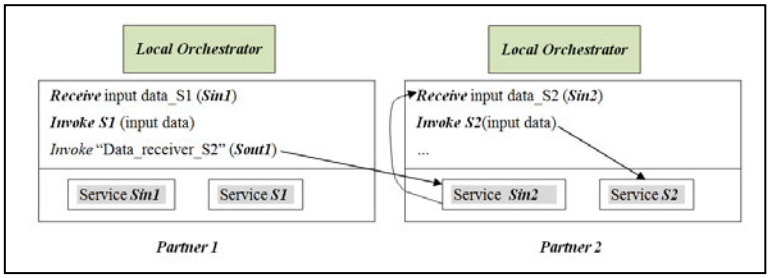

Fig. 3. Illustration of local orchestrators

4.2 Orchestration Function and Control Flow

On the meta-model of Fig. 2, the concept of *orchestration function* describes the control flow between services composing the WF. The orchestration function is expressed using a combination of basic control flow operators. On Fig.4, we introduce these basic operators and we express them using a general notation independently from any language or platform.

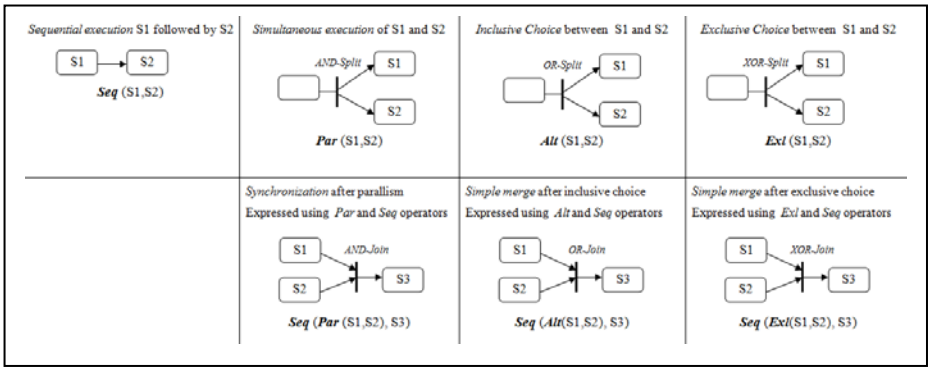

Fig. 4. Basic operators of control flow

Remark. To describe multi-choice – respectively multi-parallel - (more than two edges), we can decompose on several simple choices – respectively several simple parallel blocs. For example, *Alt* (S1, S2, S3) is expressed as *Alt* (*Alt* (S1, S2), S3) or *Alt* (S1, *Alt* (S2, S3)).

Fig. 5 bellow illustrates the concept of orchestration function using our notation; we give an example of IOWF obeying to the *chained execution pattern*. The process schema describes an IOWF implying two partners, partner 1 and partner 2 implementing their WFs as services *S1* and *S2* respectively. Partner 1 provides his WF composed by *internal* services *S11, S12, S13, S14, S15* and partner 2 provides his WF composed by internal services *S21, S22* and *S23*; in this case, the service *Sout1* corresponds to invocation of *S2* from *S1*. For more readability and less complexity of the orchestration function, we can structure the process fragments into blocs *Bij* of

sequential, parallel or alternative services. In hierarchical manner, a bloc can be expressed using other blocs. The orchestration function can be represented by a binary tree with two types of nodes: operators and services.

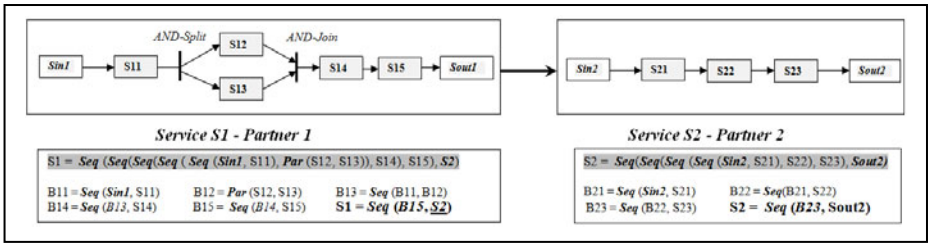

Fig. 5. Illustration of orchestration function

4.3 Formal Definition of IOWF

An IOWF is defined by a pair **<S, F>** where S is a set of services *Si* (or *Sij* for internal level) and F is a set of orchestration functions *f*. f (Si1, Si2, … , Sin) = *Si1* ***op1*** *Si2*… ***opn-1*** *Sin* = ***Si*** op1, … opn-1 are operators of control flow.

For the "*chained execution"* pattern, an orchestration function f implemented at partner ***i*** associates a composite service ***Si*** to a set of internal services ***Sij***. Interactions between services of the same partner define internal interactions and interactions between services of different partners define external ones.

5 Adaptability of IOWF Models

According to the previous definition, adaptation of process models turns to modifications of the entities composing it that means *services* or *orchestration functions*. A modification of a service can be adding, removing, replacing, merging of two services and decomposing a service into a bloc of two services expressing sequential, parallel or alternative execution. Adaptation of a service usually induces modification on the *orchestration function* using it or a modification of closely attached attributes like *condition* or *data* (see Fig. 2). Also, other operations of adaptation can affect only the control flow in the process that means the *orchestration function* while maintaining all services composing the process.

5.1 Adding, Removing and Substituting of Services

For *adding* or *removing* of services, it is to distinguish adding or removing of a service on *one edge* composed by sequential services or in a bloc composed by *two edges* expressing parallel or alternative execution. The part on the top of Fig. 6 describes the basic operations of *adding* of services illustrated by generic schemas, the corresponding *orchestration functions* and the sequence of operations done in order to obtain the new orchestration function from the initial one. Let's notice that the adding of service in a bloc of exclusive choice or parallel execution is not represented in the figure because it is done in the same manner as inclusive choice.

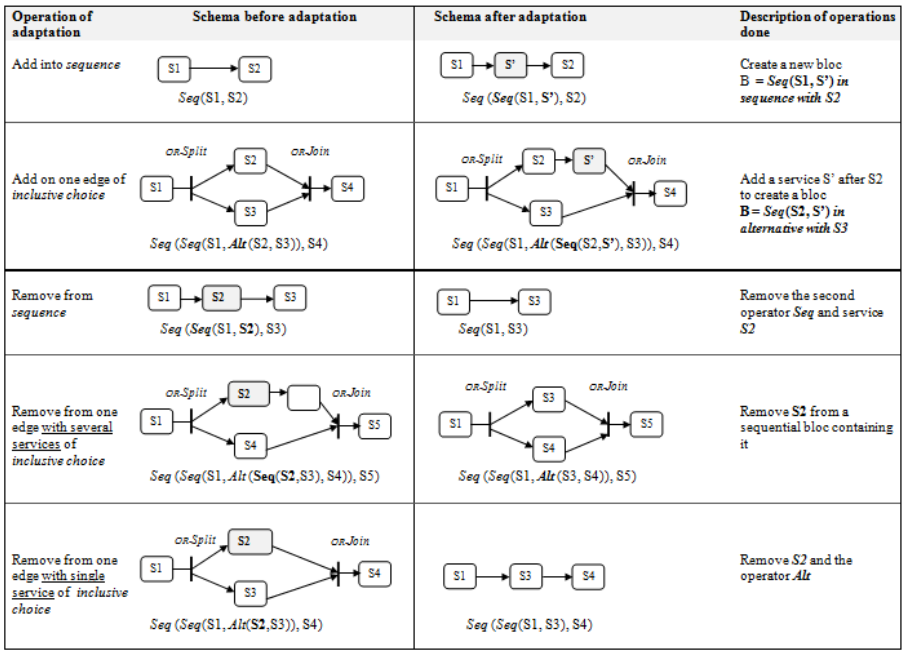

Fig. 6. Adding and Removing of a service

The reverse operation of adding is the *removing* of services. It is also to distinguish the removing of a service from *one edge* composed by sequential services or from a bloc composed by *two edges* according to parallel or alternative execution. Fig. 6 (the part on the bottom) shows typical operations of removing of services (service ***S2*** for example). For non sequential bloc, we only describe the removing from alternative bloc expressing inclusive choice; the same scenario is applied for exclusive choice or parallel execution. Let's notice that two configurations are possible when removing a service *S* from a bloc with two edges: (i) service *S* is in sequence with other services, (ii) service *S* is alone on the edge; this results on two different scenarios for operations done like shown on Fig. 6.

Another basic operation of adaptability concerns the substitution (*replacing)* of services. This is typically a *removing* of service to replace followed by an *adding* of the new service. Particularly, the *replacing* of an interactional service ***Sini*** or ***Souti*** by another is done to adapt the interface of a service ***Si*** implied in the IOWF.

5.2 Fusion and Decomposition of Services

The operation of *fusion* can concern two services related by a sequence, an inclusive choice, an exclusive choice or a parallel execution, in order to simplify the process model and to abstract several services into one. The part on the top of Fig.7 describes

these basic operations, the set of operations done and the corresponding orchestration functions modified after each operation for merging ***S2, S3*** in a single service ***S'***. We can see on Fig. 7, that since services to merge are in the same bloc, they become easier to remove and to replace because the bloc *Alt (S2, S3), Par(S2,S3)* or *Exl (S2, S3)* is considered as a single *composite service to be replaced.*

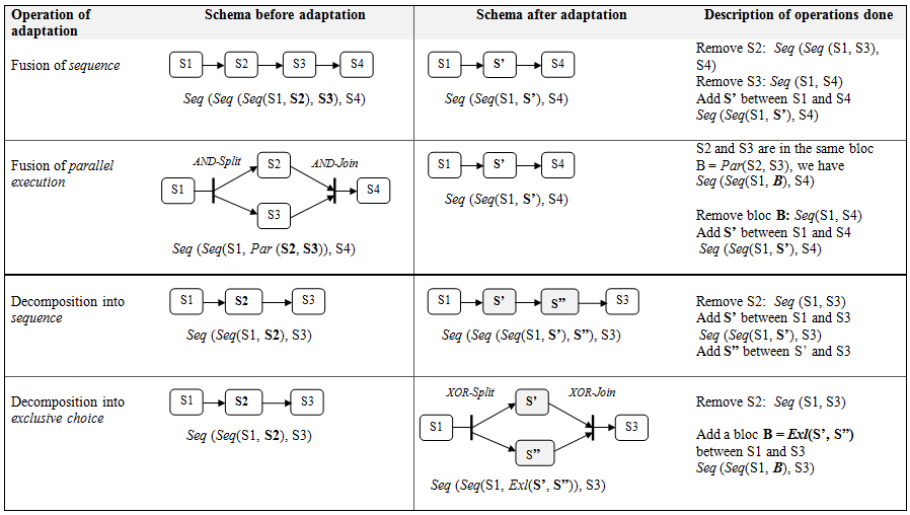

Fig. 7. Fusion and decomposition of services

More elaborated operations of fusion concern configurations such as services to merge are not in the same bloc. For example in a model described by the function *Seq(Seq(**S1**, Par(**S2**,S3)), S4)*, the operation of merging *S1* and *S2* cannot be done directly since we must know if we maintain the parallelism or not; this information should be provided as additional parameter. In both cases, this must be decomposed into elementary operations of removing and adding of single services or blocs.

The reverse operation of fusion is the *decomposition* of a service to obtain a bloc of two services that can be sequential, parallel or alternative. We can see on the bottom of Fig.7 that the decomposition of a service consists to *remove* a single service (***S2*** for example) and to *add a bloc* composed by two services (***S' and S''***) linked by sequence, alternative or parallel operator. The decomposition of services is done in order to improve the parallelism in the process (parallelization of services) or to add condition (inclusive or exclusive choice) due to new constraints or to have more control on the execution of the process (sequence of services).

5.3 Adapting the Orchestration Function

Another category of adaptation on IOWF models concerns modification of *orchestration function* without modifying services, this is typically a replacing of an

operator of control flow by another; we can replace for example a sequence operator (***seq)*** by parallel operator (***par)*** to improve the execution time of process instances, or vice versa if an execution of a service becomes dependant from another service.

When services to be restructured are in the same bloc, the operation of adaptation can be easily done by substituting operators; it is to replace the initial operator by another one in the orchestration function. For example, in the orchestration function *seq (**seq** (**S1**,**S2**), S3),* if we want to parallelize (S1, S2), we just replace the operator ***seq*** by the operator ***par*** to obtain the transformed function ***seq (par(S1**,**S2**), S3).* By contrary, if services to be restructured are not in the same bloc, operations of adaptation are less evident; for example in the orchestration function *seq (seq (seq (S1,**S2**), **S3**), S4)*, the parallelization of (S2,S3) cannot be done directly but we must remove S2 to obtain *seq (seq (S1, **S3**), S4),* then remove S3 to obtain *seq (S1, S4),* and finally add a bloc ***par (S2,S3)*** between S1 and S4 to obtain *the transformed orchestration function seq(seq (S1,**par (S2**,**S3**)), S4).*

6 Evolutivity of IOWF Models

As already explained, the *evolutivity* (or evolvable adaptability) of IOWF process models is reflected at two perspectives: the *functionality* and the *cooperation* of the IOWF. Hence, an IOWF model evolves if it can be extended to additional functionalities and/or it allows expansion of cooperation to involve more partners and more external services; the two perspectives are not exclusive.

6.1 Expanding Functionalities

Expansion of functionalities of the IOWF can be done by adding internal services *Sij* (resp. blocs) with novel functionalities into the WF of one or more partner(s) or by replacing a service (resp. bloc) by another that covers more functionalities. To do that, we can refer to operations of section 5.1, the only difference is that the injected services implement additional functionalities of the IOWF. At external level, the expansion of functionalities can be realized by replacing an external service *Si* encapsulating a WF fragment by another external service.

6.2 Expanding Cooperation

According to the second perspective, it is the capacity to open the IOWF to more partners. This can occur in two cases: (a) an additional external service is added to the sequence of external services composing the IOWF, in order to extend the functionality of the global process or (b) replacing an external service by a bloc of *exclusive choice* of two external services according to new constraints. Starting with an IOWF model initially composed by a sequence of three services *Sx, Sy* and *Sz* provided by partners *x, y* and *z* respectively, Fig. 8 shows the possible configurations of evolution previously described. We assume that each service Sp provided by partner p is composed by a sequence *Sinp, Spp* (the composite business service)

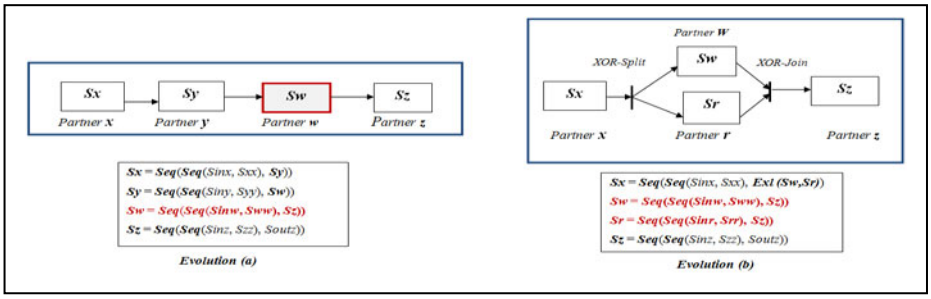

Fig. 8. Expansion of the cooperation

and invocation of the following service in the sequence or *Soutp* for the last service in the sequence. In case of evolution (a), we have to add the orchestration function of service ***w*** and to ensure interaction between the pairs of services (*Sy*,*Sw*) and (*Sw* , *Sz)*. In evolution (b), we have to add the orchestration functions of services S***r*** and S***w***, to implement the exclusive choice in Service *Sx* and to ensure interaction between pairs of services (*Sx, Sw), (Sw, Sz),(Sx, Sr) and (Sr, Sz)***.*** Let's notice that the *chained execution* pattern is preserved since instances are executed according to one path of sequential services (*Sx, Sw, Sz*) or (*Sx, Sr, Sz*).

7 Conclusion and Future Works

In this paper, we focused on the issue of adaptability of IOWF models in case of structured cooperation. We have considered process models obeying to the *chained execution* architecture defined in [1]. In order to deal with process models flexible enough, we have proposed a *cooperation pattern based on services* to implement IOWF obeying to the architecture considered in this paper. So, we have introduced the concept of *orchestration function* that is built on basic operators of control flow to orchestrate internal services composed to build a fragment of WF provided by a partner. To maintain *decentralized* control, each partner implements his orchestration function and interactional services insuring the communication with external services. We distinguish operations of evolution (evolvable adaptation) from other adaptations basis on two perspectives the *functionality* of the IOWF process and the *cooperation*; so, we talk about *evolutivity* if the functionality of the IOWF is expanded and/or the cooperation is expanded. The operations of adaptation and evolution of process models are described at a conceptual level showing the transformation of orchestration functions for each type of adaptation or evolution. Also, with the proposed approach, we can deal with reusability (well supported by SOA) of IOWF process models which is another aspect of flexibility allowing the combination of several IOWF obeying to the same or different architectures, in order to build more complex business processes based on existing ones.

We are currently working to implement these operations of adaptation and evolution as adaptation patterns by translating them to a specific language of business process definition. Furthermore, we must provide mechanisms to check the correctness of models after adaptation.

References

1. Aalst, W.V.D.: Process oriented architectures for electronic commerce and interogranizational workflow. Journal of Information Systems 24(9) (1999)
2. Papazoglou, M.P., Heuvel, W.J.V.D.: Service Oriented Architectures: approaches, technologies and research issues. The VLDB Journal 16, 389–415 (2007)
3. Bastide, G.: SCORPIO - An Approach for Structural adaptation of software components: application to ubiquitous environments. Phd Thesis, University of Nantes (2007)
4. Chen, M., Zhang, D., Zhou, L.: Empowering collaborative commerce with web services enabled business process management system. Decision Support System (2005)
5. Leymann, F., Roller, D., Schmidt, M.-T.: Web Services and Business Process Management. IBM Systems, Journal 41(2) (2002)
6. Gorton, S., Montangero, C., Reiff-Marganiec, S., Semini, L.: StPowla: SOA, Policies and Workflows. In: Di Nitto, E., Ripeanu, M. (eds.) ICSOC 2007. LNCS, vol. 4907, pp. 351–362. Springer, Heidelberg (2009)
7. Chebbi, I.: CoopFlow - an approach for ascendant cooperation of workflows in virtual enterprises. Phd Thesis, National Institute of Telecom, France (2007)
8. Grefen, P., Aberer, K., Hoffer, Y., Ludwig, H.: Crossflow: Cross-organizational workflow management for service outsourcing in dynamic virtual enterprises. IEEE Data Engineering Bulletin 24(1), 52–57 (2001)
9. Mehandjiev, N., Stalker, I., Fessl, K., Weichhart, G.: Interoperability contributions of Crosswork. Invited short paper to Proceedings of INTEROP-ESA 2005 Conference, Geneva. Springer (February 2005)
10. Belhajjame, K., Vargas-Solar, G., Collet, C.: Pyros - an environment for building and orchestrating open services. In: Proceedings of the 2005 IEEE International Conference on Services Computing, pp. 155–164. IEEE Computer Society, Washington, DC (2005)
11. Casati, F., Shan, M.: Dynamic and adaptive composition of e-services. Information Systems 26(3), 143–163 (2001)
12. Sadiq, S.W., Orlowska, M.E.: On capturing Exceptions in workflow process models. In: Proceedings of ER 2001 (2001)
13. Meng, J., Su, S.Y.W., Lam, H., Helal, A., Xian, J., Liu, X., Yang, S.: DynaFlow - a dynamic inter-organisational workflow management system. Int. Journal of Business Process Integration and Management 1(2), 101–115 (2006)
14. Muller, R., Greiner, U., Rahm, E.: AGENT-WORK: a workflow system supporting rule-based workflow adaptation. Journal of Data and Knowledge Engineering 51(2), 223–256 (2004)
15. He, Q., Yan, J., Jin, H., Yang, Y.: Adaptation of Web Service Composition Based on Workflow Patterns. In: Bouguettaya, A., Krueger, I., Margaria, T. (eds.) ICSOC 2008. LNCS, vol. 5364, pp. 22–37. Springer, Heidelberg (2008)
16. Döhring, M., Zimmermann, B., Karg, L.: Flexible Workflows at Design- and Runtime Using BPMN2 Adaptation Patterns. In: Abramowicz, W. (ed.) BIS 2011. LNBIP, vol. 87, pp. 25–36. Springer, Heidelberg (2011)
17. Weber, B., Reichert, M., Rinderle-Ma, S.: Change patterns and change support features-Enhancing flexibility in process-aware information systems. Journal of Data & Knowledge Engineering 66, 438–466 (2008)
18. Döhring, M., ZimmermaSnn, B., Godehardt, E.: Extended workflow flexibility using rule-based adapatation patterns with eventing semantics. In: Proc. of INFORMATIK 2010, pp. 216–226 (2010)
19. Pesic, M., Schonenberg, M.H., Sidorova, N., van der Aalst, W.M.P.: Constraint-Based Workflow Models: Change Made Easy. In: Meersman, R., Tari, Z. (eds.) OTM 2007, Part I. LNCS, vol. 4803, pp. 77–94. Springer, Heidelberg (2007)