Service based Approach for Adaptability of Workflow Models
The Subcontracting Architecture

Saida Boukhedouma[1, 2], Mourad Oussalah [2], Zaia Alimazighi[1] and Dalila Tamzalit[2]
[1]USTHB- FEI- Department of Computer Science- LSI Laboratory – ISI Team,
El Alia BP n°32, Bab Ezzouar, Algiers, Algeria
[2]University of Nantes - LINA Laboratory - MODAL Team
2, Rue de la Houssinière, BP 92208, 44322 Nantes, Cedex 3, France

Keywords: IOWF, Service, Cooperation Pattern, Subcontracting, Orchestration Function, Adaptability, Evolutivity.

Abstract: The environment of businesses is naturally unstable and dynamic due to increasing market constraints and events. Thus business processes are frequently subject to changes which must be supported by process models and systems that implement them. This paper deals with adaptation of IOWF (Inter-Organizational Workflow) process models based on services. It states conceptually, the most frequent adaptations that can be operated on IOWF models described through the concept of orchestration function which abstracts the control flow of the process. Thus, operations of adaptation turn to modification of services and transformation of the orchestration function describing the model. We particularly distinguish evolutive adaptation leading to expansion of the cooperation and/or the global functionality of the process.

1 INTRODUCTION

In our research works, we are interested in structured cooperation supported by concepts and tools of *Inter-Organizational workflow* (IOWF) (Aalst, 99). In structured cooperation, the steps of the business process and interactions between business partners are well defined resulting in an IOWF model clearly defined; so all process instances follow the same IOWF model implemented. In (Aalst, 99), generic architectures of IOWF have been defined. These are 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 cooperate together. However, in their initial form these architectures were subject to criticisms (Chebbi, 2007) because of their rigidity and the difficulty to adapt business processes.

This paper deals with *adaptability* of IOWF process models. An adaptation is due to various reasons such as the improvement of the process, the occurrence of new constraints imposed by the environment or the correction of errors in the process model. Another reason of adaptation can be the evolution of process models called *evolutive adaptation* that we perceive through two perspectives: expansion of process *functionalities* and expansion of *cooperation*; we globally talk about *evolutivity of process models*.

For that, we propose *cooperation patterns* based on services corresponding to the basic architectures defined in (Aalst, 99), using a *SOA based approach* because 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. We state that the basic architectures considered can be implemented as *global orchestration* or *distributed local orchestrations* of services, according to constraints relative to each architecture.

This paper focuses on the *subcontracting*; it states conceptually, the most frequent adaptations and evolutions to be done on IOWF process models based on services and describes some basic operations applied. The *orchestration function* abstracts the structure (control flow) of the IOWF process; it orchestrates internal and external services using basic operators of control flow.

In the following, Section 2 presents some related works and explains the motivation of our work.

DOI: 10.5220/0004008702240231
In *Proceedings of the 14th International Conference on Enterprise Information Systems* (ICEIS-2012), pages 224-231
ISBN: 978-989-8565-12-9

Section 3 synthesizes the necessary background to understand the paper. Section 4 describes the *cooperation pattern* suitable to the subcontracting architecture and illustrates the concept of *orchestration function*. Sections 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

The use of WF technology and SOA paradigm had a great impact in the promotion of the B2B cooperation. Hence, several approaches such as CoopFlow (Chebbi, 2007), CrossFlow (Grefen and al, 2001), CrossWork (Mehandjiev et al., 2005), Pyros (Belhajjame et al., 2005), e-Flow (Casati et al., 2001) have been proposed.

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. Otherwise, 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 (Sadiq et al., 2001). (ii) At the *level of process models* that defines the ability of a process model to be adaptable, evolvable and reusable; many research works have been proposed describing different techniques such as adaptation patterns (He et al., 2008), (Döhring et al., 2011), (Tragatcshnig et al., 2011), rule-based adaptation patterns (Döhring et al., 2010) and constraint-based modeling (Pesic et al., 2007).

The goal of this paper is to deal with *adaptability* of *IOWF process models* based on services especially obeying to the *subcontracting* architecture. First, we introduce the concept of *cooperation pattern*. Then, we express this cooperation pattern using *SOA approach* in order to deal with IOWF models easily adaptable and evolvable. The cooperation pattern based on services is defined using the concept of *orchestration function* that abstracts the structure of the process; thus, all adaptations and evolutions turn to modification of this function.

3 BASIC DEFINITIONS AND CONCEPTS

3.1 Definition and Dimensions 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.

In (Aalst, 99), generic architectures of IOWF have been defined. These architectures are characterized according to two main dimensions: the *partitioning of the process* and the *control of execution*. Regarding to the first dimension*,* two types of partitioning are distinguished: *process schema partitioning* and *instance partitioning*. Process schema partitioning means that the IOWF process model is implemented as fragments at the partner's sites. Instance partitioning means that the execution of a process instance is distributed among the different sites in a disjoint manner (at each moment, an instance is located at one site).

For the second dimension, 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 the 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 other secondary systems.

3.2 IOWF Meta-model, Adaptability and Evolutivity

An IOWF process model is defined by a set of *WFs* and a *cooperation pattern* that links two or more WFs through a set of *interaction points.* Each WF is attached to a *partner*, manipulates *data* and is submitted to a *condition* (see Figure 1). A cooperation pattern is defined through the two dimensions of IOWF: the *partitioning of the process* and the *control of execution*. We can affirm that the constraints of flexibility in IOWF model are not limited to one axis, but cover all axes that define it (process, organization, data and interaction). We focus on flexibility reflected at the *process* and *interaction* axes.

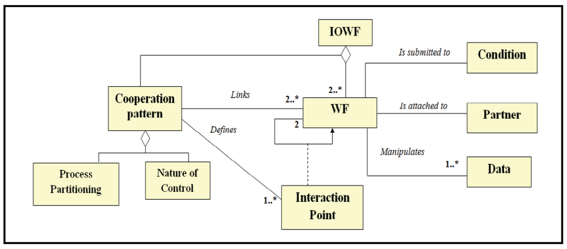


Figure 1: Generic meta-model of IOWF.

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.

The *evolutivity (evolutive adaptability)* of a process model defines its capacity to accept *expansion* of its functionalities and/or cooperation (additional business partners and so additional WF fragments) where maintaining the coherence of the process.

3.3 The Subcontracting Architecture

The *subcontracting* architecture supports a model of cooperation that connects two or more business partners, each of which implements its own workflow process. There is *one main workflow* attached to the main partner which subcontracts some activities not implemented locally to *one or more secondary workflow*s. The UML activity diagram of Figure 2 describes an IOWF process related to the design and realization of integrated circuits (PCB) to potential customers; the process involves a main partner and a secondary partner. When the customer's order is received, the main partner studies the schema of PCB, if it is a mono-layer PCB, it is entirely designed and implemented locally; otherwise in case of multi-layer PCB, its design is subcontracted to an external partner because the main partner has not enough competencies and resources to design multi-layer PCB. The result of processing (the design of multi-layer PCB) is returned to the main partner. Figure 2 shows the most important phases of the process; after *studying the schema* of the process, *electrical parameters* are fixed, then *mechanical* and *thermal conditions* are established simultaneously. After that, all criteria established are *validated* according to specified norms. Then, the PCB is *designed* and *implemented*. The WF of the secondary partner seems to be atomic from the main partner but in reality it is complex and contains several phases.

In the next section, we define the *subcontracting pattern* based on services and we introduce the concept of *orchestration function.*

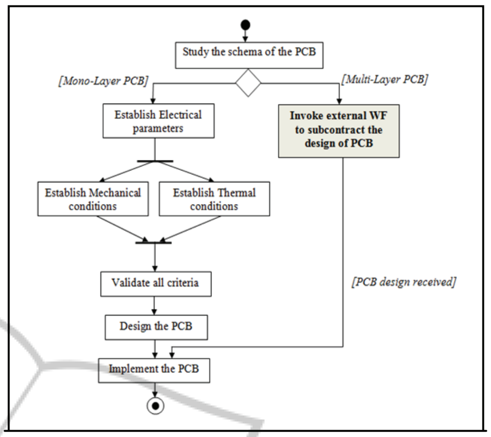

Study the schema of the PCB
[Mono-Layer PCB]
[Multi-Layer PCB]
Establish Electrical parameters
Invoke external WF to subcontract the design of PCB
Establish Mechanical conditions
Establish Thermal conditions
Validate all criteria
[PCB design received]
Design the PCB
Implement the PCB

Figure 2: Example of IOWF process "Realization of PCB" - The subcontracting architecture.

4 BASIS OF OUR APPROACH

The question is to decide which parts of the global 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 where maintaining interaction points in the initial IOWF* (Boukhedouma et al., 2011).

4.1 The "Subcontracting" Pattern based on Services

For this architecture, we propose to entirely encapsulate *each secondary WF within a service*. On Figure 3, partner 1 hosts the main WF and partner 2 provides his secondary WF as a global *service S2*. Thus, Partner 1 invokes the service of partner 2 for subcontracting. To obtain an IOWF entirely based on services, the whole WF can be implemented as an *orchestration* of *local* services encapsulating sub-processes or activities of the main WF and *external* services provided by secondary partners. In the subcontracting architecture, the control of execution is *hierarchized* because the main WF manages the control of the whole process and controls invocation of external services. The *subcontracting* pattern is described through the meta-model of Figure 3; given a set of local and external services attached to several partners and an *orchestration function* implemented by the main partner, we can define the IOWF obeying to this pattern.

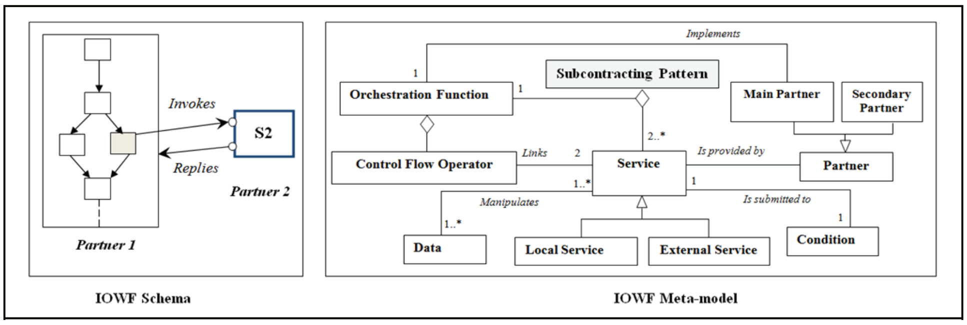
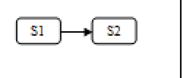
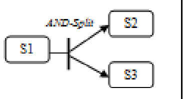
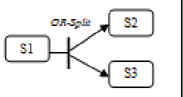
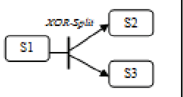
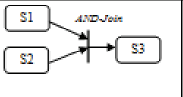
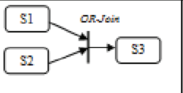
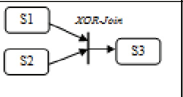


Figure 3: Schema and meta-model of the subcontracting pattern.

4.2 Orchestration Function and Control Flow

The orchestration function is defined using basic control flow operators. In table 1, we introduce these basic operators that we express using a general notation independently from any language or platform.

Table 1: Basic operators of control flow.

Operator	*Schema*	*Description*	*Orchestration function*
Seq		*Sequential execution* S1 followed by S2	*Seq* (S1,S2)
Par		*Simultaneous execution* of S1 and S2	*Par* (S1,S2)
Alt		*Inclusive choice* between S1 and S2	*Alt* (S1,S2)
Exl		Describes an *exclusive choice* of S1 and S2	*Exl* (S1,S2)
		*Synchronous merge* of S1 and S2 after parallism Expressed using *Par* and *Seq* operators	*Seq* (*Par* (S1,S2), S3)
		*Simple merge* of S1 and S2 after inclusive choice Expressed using *Alt* and *Seq* operators	*Seq* (*Alt* (S1,S2), S3)
		*Simple merge* of S1 and S2 after exclusive choice Expressed using *Exl* and *Seq* operators	*Seq* (*Exl* (S1,S2),S3)

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)).

To illustrate the concept of orchestration function, let's consider the example of the IOWF process "Realization of PCB" described on Figure 2. If we consider that local activities "*study the schema of PCB*", "*establish electrical parameters*", "*establish mechanical conditions*", "*validate criteria*", "*design PCB*" and "*implement PCB*" are implemented as local services named respectively S11, S12, S13, S14, S15, S16 and S17 and the secondary WF is implemented as external service S2, the schema of the IOWF process and the corresponding orchestration function are described like shown on Figure 4.

For more readability and in order to reduce the complexity of the orchestration function, we can structure the process into blocs (expressing composite services) of sequential, parallel or alternative services. In a hierarchical manner, a bloc can be expressed using other blocs.

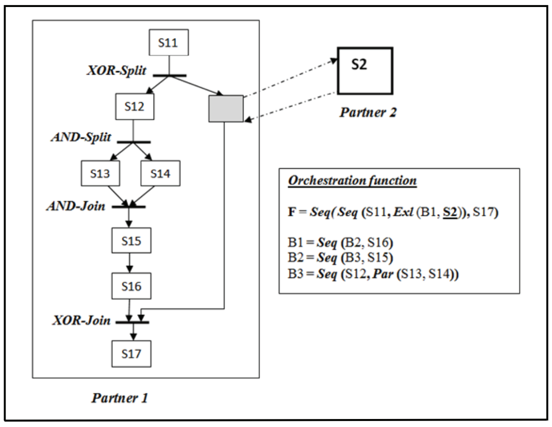


Figure 4: IOWF Schema and Orchestration function of the process "Realization of PCB".

4.3 Formal Definition of IOWF

An IOWF is defined by a pair **<S, F>** where S is a set of local and external services *Sij***.** Local services are attached to a main partner and external services are attached to one or more secondary partners. **F** is

an *orchestration function* where
F (Si1, Si2, … , Sin) = Si1 ***op1*** Si2 ***op2***…***opn-1*** Sin
op1, op2, … opn-1 are operators of control flow.

5 ADAPTABILITY OF IOWF MODELS

According to the previous definition, adaptation of process models turns to modifications of the entities composing it that means the *services* or the *orchestration function*. 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 Figure 3). Also, other operations of adaptation can affect only the control flow in the process that means the *orchestration function*.

5.1 Adding, Removing, 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 Figure 5 describes the basic operations of *adding* of services illustrated by generic schemas, the corresponding *orchestration functions* and the sequence of operations allowing the transformation of the orchestration function. The adding of a service in an exclusive choice (or parallel) bloc is not represented in the figure because it is done in the same manner as in a bloc of inclusive choice.

Adding a service is done in order to insert additional steps in the process. If we consider the example "Realization of PCB" (Figure 4), the designer of the IOWF process can decide to add a service "*validate electrical parameters*" named **S'** after service S12 in order to add a step of validation that can be necessary for the design of complex PCB; then we obtain the schema shown on Figure 6 (Adding Service S'). 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. The part on the bottom of Figure 5 shows typical operations of removing of services (service ***S2*** for example). 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 Figure 5. Another basic operation of adaptability concerns the substitution (*replacing)* of services. This is typically a *removing* followed by an *adding* of services.

5.2 Fusion and Decomposition of Services

The operation of *fusion* can concern two services linked by a sequence, an alternative or a parallel execution, in order to simplify the process model. If the services to merge are in the same bloc, the operation of fusion becomes easy since it consists to replace the bloc that is considered as a single *composite service*. 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 orchestration 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 we don't maintain it; this information should be provided as an additional parameter. In both cases, this must be decomposed into elementary operations of adding and removing.

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 bloc. The decomposition of services can be done to improve the parallelism in the process (parallelization of services) or to add condition (inclusive/exclusive choice) due to new constraints or to have more control on process execution (sequence of services). The decomposition of a service consists to remove a single service and to add a bloc composed by two services.

5.3 Adapting the Control Flow

Another category of adaptation on IOWF models concerns modification of the *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

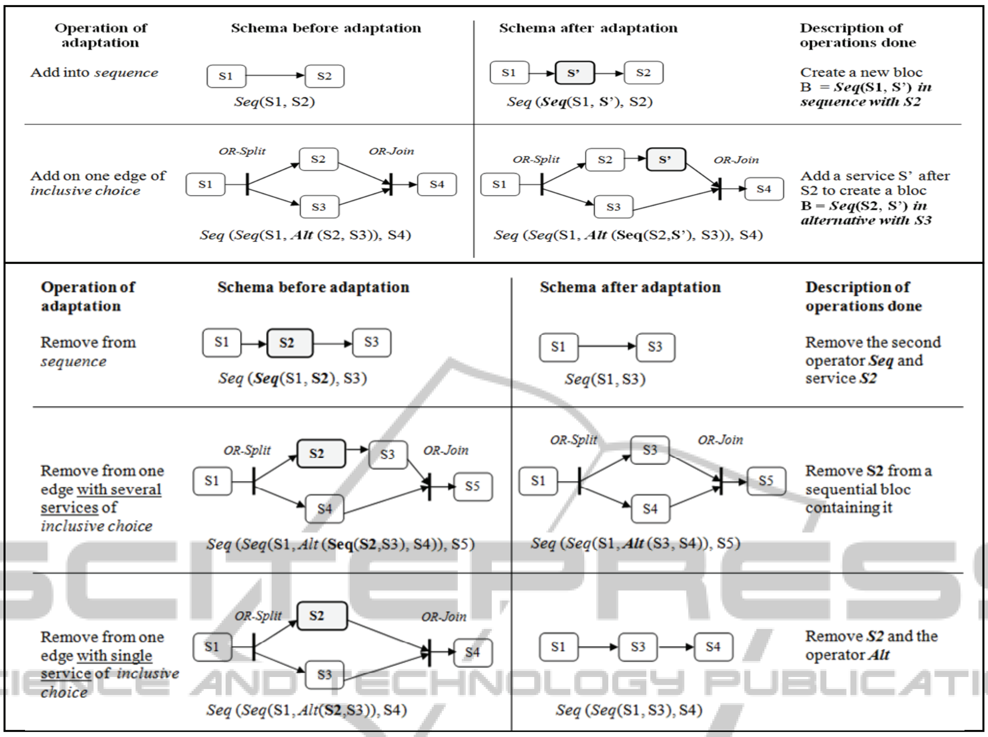

Operation of adaptation
Schema before adaptation
Schema after adaptation
Description of operations done
Add into sequence
Seq(S1, S2)
Seq (Seq(S1, S'), S2)
Create a new bloc B = Seq(S1, S') in sequence with S2
Add on one edge of inclusive choice
OR-Split
OR-Join
Seq (Seq(S1, Alt (S2, S3)), S4)
Seq (Seq(S1, Alt (Seq(S2,S'), S3)), S4)
Add a service S' after S2 to create a bloc B = Seq(S2, S') in alternative with S3
Remove from sequence
Seq (Seq(S1, S2), S3)
Seq(S1, S3)
Remove the second operator Seq and service S2
Remove from one edge with several services of inclusive choice
Seq (Seq(S1, Alt (Seq(S2,S3), S4)), S5)
Seq (Seq(S1, Alt (S3, S4)), S5)
Remove S2 from a sequential bloc containing it
Remove from one edge with single service of inclusive choice
Seq (Seq(S1, Alt(S2,S3)), S4)
Seq (Seq(S1, S3), S4)
Remove S2 and the operator Alt

Figure 5: Adding and removing of services.

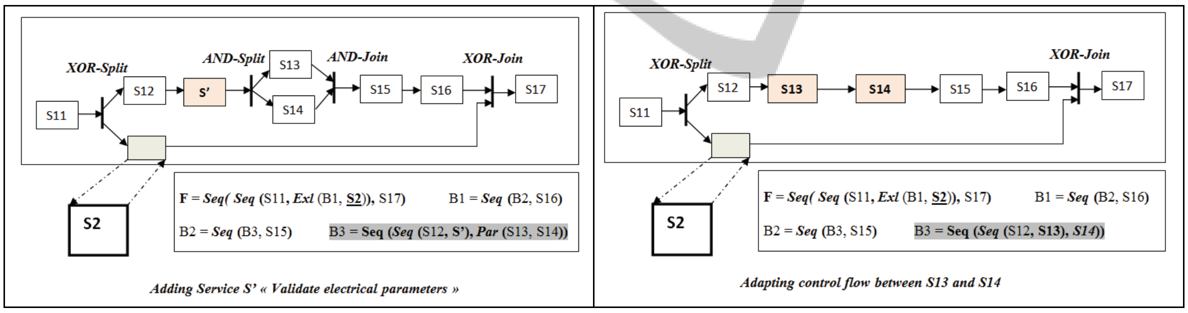

XOR-Split
AND-Split
AND-Join
XOR-Join
S11
S12
S'
S13
S14
S15
S16
S17
S2
F = Seq(Seq (S11, Exl (B1, S2)), S17)
B1 = Seq (B2, S16)
B2 = Seq (B3, S15)
B3 = Seq (Seq (S12, S'), Par (S13, S14))
Adding Service S' « Validate electrical parameters »
B3 = Seq (Seq (S12, S13), S14))
Adapting control flow between S13 and S14

Figure 6: Examples of adaptation on the IOWF process "Realization of PCB".

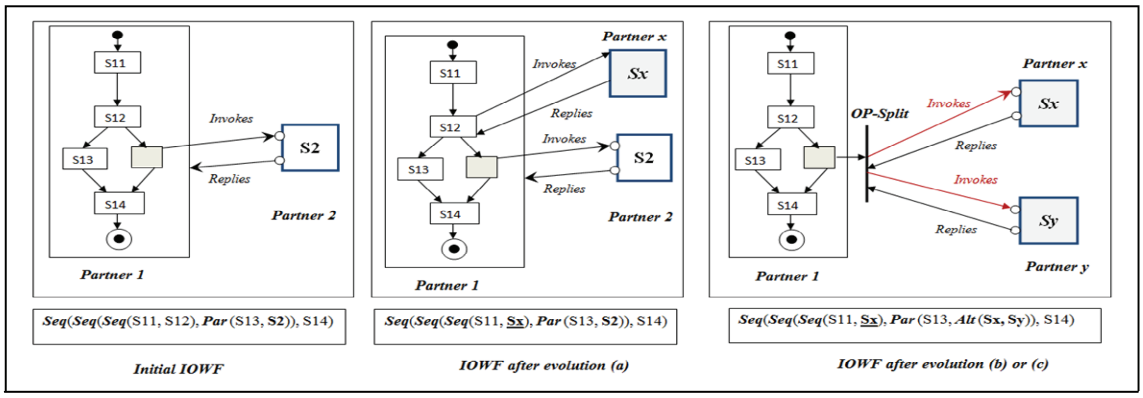

S11
S12
S13
S14
Invokes
Replies
S2
Partner 2
Partner 1
Partner x
Sx
Sy
Partner y
OP-Split
Seq(Seq(Seq(S11, S12), Par(S13, S2)), S14)
Seq(Seq(Seq(S11, Sx), Par(S13, S2)), S14)
Seq(Seq(Seq(S11, Sx), Par(S13, Alt(Sx, Sy)), S14)
Initial IOWF
IOWF after evolution (a)
IOWF after evolution (b) or (c)

Figure 7: Expansion of the subcontracting.

another service.

Let's consider our example of the process "*Realization of PCB*", the designer can decide to reorganize the control flow in the process by restructuring services S13 and S14 in sequence instead of parallel because he notices that in some cases there is a dependence between mechanical conditions and thermal conditions of the circuit. This adaptation and the corresponding orchestration function are shown on Figure 6 (Adapting control flow between S13 and S14).

When services to be reorganized (restructured) are in the same bloc, the operation of adaptation can be easily done by substituting operators. For example, in the orchestration function *seq* ***(seq (S1,S2)***, *S3),* if we want to link services *S1* and *S2* by parallel operator, 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)*, if we want to link (S2,S3) by parallel operator, we cannot do this by direct substitution of operator ***seq*** 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 function seq (seq (S1, **par (S2, S3)**), S4)*.

6 EVOLUTIVITY OF IOWF MODELS

The *evolutivity* of IOWF process models is reflected at two perspectives: the global *functionality* and the *cooperation* of the IOWF. Hence, an IOWF model evolves if it can be extended to additional functionalities or if it allows expansion to more partners and more external services. The two perspectives are not exclusive.

6.1 Expanding Functionalities

Expansion of functionalities in 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 functionality. To do that, we can refer to operations described in section 5.1, the only difference is that the new services implement additional functionalities. At external level, the expansion of functionalities can be realized by replacing an external service *Si* encapsulating a WF fragment by another external service that covers additional functionalities.

6.2 Expanding Cooperation

According to the *cooperation* perspective, it is the capacity to open the IOWF to more partners. This can occur when the main partner subcontracts other activities to external partners, this is what we call "*expanding the subcontracting*" or when a secondary partner in turn subcontracts part of its WF to other partners, this results in what we call "*multi-level subcontracting*".

6.2.1 Expanding the Subcontracting

Expansion of subcontracting can be done according to one of these configurations (Fig. 7):

a) Replacing an internal service of the main WF by an external service. b) Replacing an external service by an *alternative* branch composed by two external services *Sx* and *Sy* provided by two partners where for some cases (according to a condition) , *Sx* is invoked and for other cases *Sy* is invoked. c) Replacing an external service by a *parallel* branch composed by two external services *Sx* and *Sy* provided by two partners; *Sx* and *Sy* are executed simultaneously. Changes obviously described can be expressed through operations of substitution and decomposition explained in section5.The only difference is that evolutivity concerns *external* services. In our example of the process "Realization of PCB", the main partner can subcontract the task "*validate all criteria*" to another partner which provides it as *external service **S3***; then evolution consists to substitute the internal service ***S15*** by the external service ***S3***.

6.2.2 Multi-level Subcontracting

The configuration of multi-level subcontracting is obtained when the main WF invokes a secondary WF through the external service provided, and the secondary partner in turn operates changes to subcontract part of its own WF to another partner; this is invisible for the main WF but the overall IOWF implies additional partners at different levels.

Changes relative to this configuration are done at the secondary partner by substituting one or more of its local services by external services.

7 CONCLUSIONS AND FUTURE WORKS

This paper deal with adaptability of IOWF process models suitable to structured cooperation. To explain our approach, we have focused on process models obeying to the *subcontracting* architecture which describes a model of cooperation fairly common in B2B relationship. 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; then, we introduce the concept of *orchestration function* that abstracts the structure of the process in terms of control flow. Also, we distinguish operations of adaptation from operations of evolution basis on two perspectives the *overall functionality* of the IOWF process and the *cooperation*. The operations of adaptation and evolution of process models are described at a conceptual level and turn to changes operated on the orchestration function.

We are currently working to implement these operations of adaptation and evolution as generic adaptation patterns using a specific language of business process definition like BPEL or jPDL. Furthermore, we must provide mechanisms to check the correctness of models after adaptation.

After that, we intend to deal with reusability of IOWF process models which is another aspect of flexibility allowing the combination of several IOWF in order to build more complex business processes based on existing ones. In our view, this is possible because integration, composition and reuse are well supported in SOA paradigm.

REFERENCES

Aalst, W. V. D., 1999.Process oriented architectures for electronic commerce and interogranizational workflow**,** *Journal of Information systems, volume 24 issue 9*.

Belhajjame K., Vargas G. Solar, Collet C.: Pyros *2005*- an environment for building and orchestrating open services. In *Proceedings of the 2005 IEEE International Conference on Services Computing, pages 155–164, Washington, DC, USA,. IEEE Computer Society.*

Boukhedouma S., Alimazighi Z., Oussalah M., Tamzalit D., 2011, SOA basedapproach for interconnecting workflows according to the subcontracting architecture. In *proceedings of MCCSIS- IADIS' International Conference, CT'2011. Italy. Pp 3-12. ISBN: 978-972-8939-40-3*

Casati F., Shan M., 2001.Dynamic and adaptive composition of e-services. *Information Systems*, 26(3) :143–163

Chebbi I., 2007, CoopFlow: an approach for ascendant cooperation of workflows in virtual enterprises. *Phd Thesis, National Institute of Telecom,* France.

Döhring M., Zimmermann B., Godehardt E., 2010. Extended workflow flexibility using rule-based adapatation patterns with eventing semantics. In *proceedings. of INFORMATIK'10, pp.* 216,226.

Döhring M., Zimmermann B., Karg L., 2011. Flexible Workows at design- and Runtime using BPMN2 Adaptation Patterns. In *proceedings of BIS'2011.*

Grefen P., Aberer K., Hoffer Y., Ludwig H. , 2001. Crossflow: Cross-organizational workflow management for service outsourcing in dynamic virtual enterprises. *IEEE Data Engineering Bulletin*, 24(1):52–57.

He Q., Yan Y., Jin H., 2008, Adaptation of web service composition based on WF patterns**.** In *proceedings of Service Oriented Computing,* ICSOC'08.

Mehandjiev N., Stalker I., Fessl K., Weichhart G. 2005, Interoperability contributions of crosswork. In invited short paper to *Proceedings of INTEROP-ESA'05 Conference, Geneva, February 2005. Springer-Verlag.*

Papazoglou M. P., Heuvel W. J. van den, 2007, Service Oriented Architectures: approaches, technologies and research ***issues***, the VLDB Journal, vol.16, pp 389-415.

Pesic M., Schonenberg M. H., Sidorova N., Aalst W. V. D, 2007. Constraint-based workflow models: Change made easy. In *Proceedings of the OTM Conference CoopIS'07. In vol 4803 of Lecture Notes in Computer Science, pp 77–94.Springer-Verlag, Berlin*

Sadiq S. W., Orlowska M. E., 2001. On capturing Exceptions in workflow process models. In *Proceedings of ER'2001.*

Tragatschnig S., Zdun U., 2011. Runtime business adaptation for BPEL Process execution engines. In *the 15th IEEE international enterprise object computing conference workshops.*