



Cooperation Patterns and Adaptation Patterns for Service-Based Inter-Organizational Workflows

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

► To cite this version:

Saida Boukhedouma, Mourad Chabane Oussalah, Zaia Alimazighi, Dalila Tamzalit. Cooperation Patterns and Adaptation Patterns for Service-Based Inter-Organizational Workflows. Uncovering Essential Software Artifacts through Business Process Archeology, Ricardo Perez-Castillo, pp.250-283, 2013, 10.4018/978-1-4666-4667-4.ch010 . hal-01067303

HAL Id: hal-01067303

<https://hal.science/hal-01067303>

Submitted on 23 Sep 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Cooperation Patterns and Adaptation Patterns for Service-Based Inter- organizational Workflows

Boukhedouma S.^(1,2), Oussalah M.⁽²⁾, Alimazighi Z.⁽¹⁾, Tamzalit D.⁽²⁾

⁽¹⁾ *USTHB- FEI- Department of Computer Science- LSI Laboratory – ISI Team*

✉: *El Alia BP n°32, Bab Ezzouar, Algiers, Algeria.*

{sboukhedouma; zalimazighi}@usthb.dz ;

⁽²⁾ *University of Nantes - LINA Laboratory – MODAL Team*

✉: *2, Rue de la Houssinière, BP 92208, 44322 – Nantes, cedex 3- France*

{mourad.oussalah; dalila.tamzalit}@univ-nantes.fr;

ABSTRACT

Modernization is an effective approach to making existing mainframe and distributed systems more responsive to business needs. SOA (service-oriented architecture) is an adequate paradigm that allows companies to tap into the business value in their current systems and position IT for rapid future changes to the business model. In our research works, we focus on the use of SOA to implement Inter-Organizational WorkFlows (IOWF). The goal is to take benefits from the advantages offered by the SOA paradigm like interoperability, reusability and flexibility in order to deal with workflow models easily adaptable, evolvable and reusable. This paper focuses on two specific architectures of IOWF which are the “chained execution” and the “subcontracting”; the first issue of this work is to define Service-Based Cooperation Patterns (SBCP) suitable to the two architectures considered. A SBCP is based on SOA; it is defined through three main dimensions: the *distribution* of services among the partner’s sites, the *control* of instance execution and the structure of *interaction* between the workflows involved in the cooperation. The second issue of the paper consists of adaptation and evolution of IOWF process models obeying to the defined SBCP. Then, we state the main operations of adaptation that can be applied on these models; we focus on adaptation at process and interactional levels. Conformably to the three dimensions of SBCP, we define three classes of adaptation patterns: “service adaptation”, “control flow adaptation” and “interaction adaptation” patterns. Also, we particularly distinguish some operations of adaptation called evolution of process models based on two perspectives: the expansion of the global functionality of the process and the expansion of the cooperation; we show that some evolutions are realized by reuse of existing IOWF models. For implementation, we consider IOWF process models specified with BPEL.

KEY WORDS

IOWF, Service, Structured cooperation, Cooperation pattern, Flexibility, Adaptation pattern, Evolution, Chained Execution, Subcontracting.

1. INTRODUCTION

SOA is a paradigm that supports software modernization (Rotibi & al, 2012) by providing flexible distributed and collaborative applications easily adaptable and reactive to changes. All companies that aim to stay competitive in their field of activity should increase their production and must be quickly responsive to change in order to best satisfy their potential customers. For that, they transform their legacy information systems into business processes based on SOA by rebuilding them or by using techniques of business process archeology (Pérez-Castillo & al, 2011). In our research works, we focus on flexibility of business to business (B2B) processes based on SOA.

Since many years, B2B applications has been promoted with the appearance of business oriented technologies such as workflow (WF) (Van Der Aalst, 2002) and web services (Alonso & al, 2004) supported by service oriented architectures (SOA) (Papazoglou & al, 2007). We are interested with structured cooperation (Eder & al, 2002) supported by the concept of inter-organizational workflow (IOWF), allowing a cooperation among several business processes attached to business partners, in order to reach a common objective according to a “winner-winner” policy.

In (Van Der Aalst, 99) and (Van Der Aalst, 2000), generic architectures of IOWF have been defined in order to support structured cooperation. These architectures are the *capacity sharing*, the *chained execution*, the *subcontracting*, the *case transfer*, the *extended case transfer* and the *loosely coupled WF*. These architectures implement different schemas of cooperation that can link business partners in B2B relationship and cover a large number of existing processes in several domains. In our works, we consider these generic architectures as *basic patterns* of structured cooperation; however in their initial form, these architectures were subject to criticisms because of their rigidity and the difficulty to adapt to changes (Chebbi, 2007).

Also, in a context of a dynamic and unstable environment, businesses are often faced with stressful situations like a breach of contract with a partner, a failure of the business process and the need of additional resources. Faced with these situations and others, companies must review their systems, their business processes and their cooperation with other business partners in order to make the required changes. Consequently, the WF process and the system implementing it must be *flexible enough* to support the necessary adjustments. These adjustments can cover four complementary aspects of IOWF process definition: *data*, *process*, *interaction* and *organization*. Here, we focus on flexibility on process and interaction aspects. Also, we define the flexibility of process models through three perspectives: *adaptability*, *evolutivity* and *reusability*.

Consequently, we have to reach two objectives: the first one is to define *cooperation patterns* supporting flexible models of IOWF corresponding to the basic architectures considered. The second objective is to implement adaptation mechanisms dedicated to support changes on IOWF models obeying to the cooperation patterns defined.

In order to deal with IOWF models flexible enough, we adopt an SOA-based approach to define a set of service-based cooperation patterns (SBCP); each of which is suitable to a specific IOWF-architecture among those considered. SOA allows organizations to overcome technical obstacles to change by improving interaction with other platforms and simplifying application architectures, thanks to the characteristics of services which are loosely coupled components, easily invoked through their interfaces, business oriented and platform independent; also the SOA paradigm supports integration, reuse and composition of services. We state that the basic IOWF-architectures considered can be implemented as *global orchestration* or *distributed local orchestrations* of services, according to constraints relative to each architecture. The current paper focuses on two specific IOWF-architectures which are the “chained execution” and the “subcontracting” for which we define two service-based cooperation patterns: SBCP2 and SBCP3.

For the second issue of our work, we state and we implement usual operations of adaptation that can be applied on IOWF process models obeying to the cooperation patterns defined. We particularly distinguish some operations of adaptation that we qualify as evolution of process models according to two

perspectives: the expansion of the overall functionality of the IOWF process and the expansion of the cooperation. For this second perspective, we show that in some cases, an IOWF model can evolve respecting the initial cooperation pattern and in other cases it can evolve and falls into the combination of two different cooperation patterns. We illustrate these two kinds of evolution by describing evolution patterns applied to IOWF models obeying to SBCP2 and SBCP3.

The rest of the paper is structured as follows: Section 2 explains the motivations of our research and presents some related works attached on one hand, to IOWF approaches and on the other hand, to WF adaptability approaches. Section 3 synthesizes the background necessary to understand the paper mainly the IOWF process concepts and aspects of flexibility of IOWF models. Section 4 lays the basis of our approach for WF interconnection using services; here, we explain the concepts of SBCP and orchestration function. Section 5 describes the two cooperation patterns proposed SBCP2 and SBCP3 respectively suitable to the “chained execution” and the “subcontracting” architectures. Section 6 describes the basic adaptation patterns proposed. Section 7 describes evolution patterns applied to IOWF models obeying to SBCP2 and SBCP3. Section 8 gives some implementation details and Section 9 summarizes the paper and talks about future works.

2. RELATED WORKS AND MOTIVATIONS

With the emergence of SOA and web services standards, many research works deal with orchestration and choreography of web services (Peltz, 2003), (Amireza, 2009), especially based on BPEL4WS (Jordan & al, 2006) in order to build business processes by service composition. Other research works such as (Leymann & al, 2002), (Gorton & al, 2009) show the interest of combining BPM (business process management), workflow and SOA for the re-use of services to construct dynamic business processes. This had a great impact in promoting B2B relationships since several approaches and platforms have been developed to support B2B cooperation using WF and SOA. In *structured* cooperation for example, we can cite some approaches like CoopFlow (Chebbi, 2007), CrossFlow (Grefen & al, 2001) CrossWork (Mehanjić & al, 2005), Pyros (Belhajjame & al, 2005) and e-Flow (Casati & al, 2001).

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: (1) at the *system level*, the flexibility defines the ability of a WFMS (WF management system) to face unexpected and erroneous situations (Sadiq & al, 2001), (Meng & al, 2006). (2) 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 based on different techniques such as adaptation patterns (He & al, 2008), (Döhring & al, 2011), (Weber & al, 2008), rule-based adaptation patterns (Muller & al, 2004), (Döhring & al, 2010) constraint-based modeling (Pesic & al, 2007) and interactive adaptation (Dadam & al, 2009) to support flexibility of process models. For example, in (Weber & al, 2008), the authors identify the most important process change patterns and change features for PAIS (process aware information systems). In (Tragatschnig & al, 2011), a framework was described using adaptation patterns and aspect-programming in order to support process adaptation for BPEL engines.

In the current work, we introduce the concept of service-based cooperation pattern (SBCP) to define an IOWF process model based on SOA paradigm where a central concept is a service; this allows a preservation of autonomy and confidentiality between business partners since services are provided with a certain degree of flexibility and published via their interfaces. The idea of using services to build collaborative business applications is not new. The motivations behind this come from three main points: (1) the relevance of service orientation, (2) the benefits of service orientation for the information system and (3) the benefits of service orientation for the cooperation. For the first point, the concept of service

(particularly web services) provides credible answers to constraints and problems attached to the information system like the lack of flexibility, the reluctance to openness and those attached to the cooperation like the need to preserve the autonomy and the confidentiality. For the second point, the service-based approach provides a certain degree of flexibility to the information system by easing the participation in new business opportunities and meeting new market demands. For the third point, the cooperation between business partners is realized by service composition. Thus, SOA can help enterprises liberate core business assets by making it easier to enrich, modernize, extend and reuse those assets well beyond their original scope of design.

Regarding the choice of the basic IOWF-architectures, we have considered those proposed in (Van Der Aalst, 99) and (Van Der Aalst, 2000) because they define different ways in which businesses can cooperate together and cover a wide range of existing business processes. Then, our approach of WF interconnection (respectively, adaptation and evolution) can be applied to a large number of existing IOWF process models.

Also, for conceptual aspects of our solution, we adopt a pattern-based approach to define the different schemas of WF interconnection and to define the different adaptations applied on these schemas. The concept of pattern was initially used in software engineering as the abstraction from a concrete form which keeps recurring in specific non-arbitrary context. In the workflow area, this concept has been usually used for business process modeling (Van Der Aalst & al, 2003), business process improvement or changes (Weber & al, 2008), (Tragatschnig & al, 2011) or exception handling (Russel & al, 2006).

Also, the pattern-based approach allows the enumeration of all recurrent and structurally well defined situations that can occur repeatedly to interconnect and to adapt IOWF processes. From the implementation perspective, the pattern-based approach allows modular and reusable implementation of the proposed patterns starting with elementary patterns and going to more complex ones by reuse of the first ones.

In the next section, we introduce the necessary definitions and concepts to ease the understanding of the paper.

3. SYNTHESIS OF BACKGROUND

3.1. IOWF Definition and Dimensions

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 (Van Der Aalst, 99), generic architectures of IOWF have been defined in order to support structured cooperation which must obey, depending on needs of partners, to a schema clearly defined. These architectures are 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*.

Regarding 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 of the global model 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).

Since IOWF are distributed systems, the control of instance execution can be *centralized*, *decentralized*, *mixed* or *hierarchized*. 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; this is suitable for the *capacity sharing*. 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, this is appropriate for the *chained execution*, the *loosely coupled* and the *(extended)case transfer* architectures. 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, like in the *subcontracting* architecture.

Let's recall that in the current work, we deal with two specific IOWF-architectures: the “chained execution” and the “subcontracting” that we describe in the following.

3.2. The chained Execution Architecture

The *chained execution* architecture supports a model of cooperation that connects two or more business partners, each of which implements its own workflow process. Workflows implied in the cooperation are executed in *sequence*. The results of execution of WFi are input data of $WFi+1$. In this architecture, we have *process schema 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*. The UML activity diagram of Figure 1 shows an example of an IOWF process obeying to the “chained execution” architecture that implies two partners in the design and the implementation of integrated circuits (PCB). We suppose that partner 1 has competencies and resources for the design of the PCB and partner 2 has competencies and resources to implement the PCB designed by partner 1. The overall process is executed in sequence and each WF is composed by a set of activities.

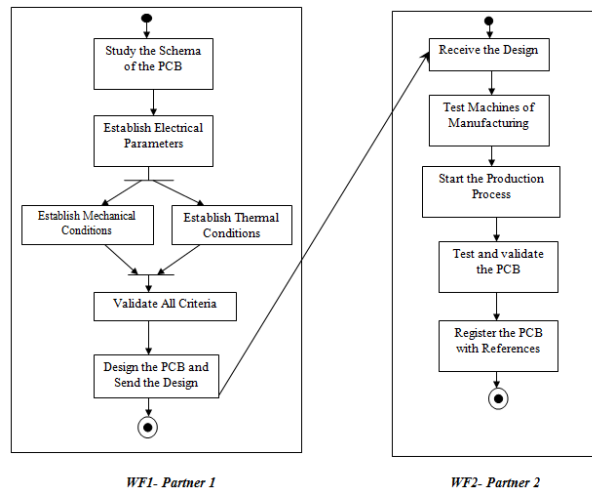


Figure 1. Example of IOWF process “Realization of PCB” according to the “chained execution” architecture

3.3. The Subcontracting Architecture

This 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 workflows* implemented by other partners involved in the subcontracting relationship. The subcontracting cooperation obeys to *process schema partitioning* and *hierarchized* control of execution.

We can see that an IOWF process model is defined by a set of *WF* (fragments of the global IOWF) and a *cooperation pattern*. Each *WF* is attached to a *partner*, manipulates *data* and is submitted to a *condition* of invocation. A given cooperation pattern is attached to a specific architecture of IOWF; it links two or more workflows and is based on three main dimensions: the partitioning of the process, the control of execution and the structure of interaction which is defined by a set of interaction points between *WF* fragments; interaction between *WF* fragments obeys to an interaction pattern that can be synchronous, asynchronous, one-a-way, etc. Intuitively a cooperation pattern defines the manner in which *WF* fragments are distributed among the partner's sites, how the execution of instances is managed and how *WF* fragments interact together.

3.5. Flexibility of IOWF models

As already evoked in the introduction, the environment of businesses and the business processes describing their behavior are naturally dynamic, because they are continually submitted to new market constraints and unexpected events. Indeed, a business process is perpetually subject to changes affecting its structure and its validity. So, a business process should be flexible enough in order to support these changes.

Through the concepts exhibited on the meta-model of Figure 3, we can see that an IOWF model covers four main axes: *process* (concepts of IOWF, *WF*, condition and cooperation pattern), *organization* (concept of partner), *data* and *interaction* (concepts of interaction structure and interaction point). Consequently, we can affirm that the constraints of flexibility in a IOWF model are not limited to one axis, but cover the four axes composing it. Also, we perceive the flexibility of process models through three main perspectives: adaptability, evolutivity and reusability that we define as follows:

The *adaptability* of an IOWF process model defines its capacity to easily support changes while maintaining the coherence of the process after changes, the overall functionality and the cooperation (the set of partners). Hence, an IOWF model is *adaptable* if one or more of the entities (*WF*, condition, data, interaction points) composing it can be modified without affecting the global functionality of the process and the cooperation.

The *evolutivity* (called *evolutive adaptability*) of an IOWF process model is its capacity to accept *expansion* of its global *functionality* and/or expansion of *cooperation* inducing additional business partners and so additional *WF* fragments where maintaining the coherence of the process, we say that the IOWF model is *evolvable*.

The *reusability* of a model defines its capacity to be easily integrated with another model in order to build more and more complex models. Then, an IOWF model is *reusable* if it can be manipulated as a separate entity (*IOWF*) and to be integrated to other models in order to build more complex IOWF processes which cover more functionalities and services.

Let's notice that in our work, we focus on flexibility reflected at process and interaction axes (although it involves and also draws on other levels – data and organization) and in the current paper we mainly exhibit two aspects of flexibility which are the adaptability and the evolutivity of process models obeying to two specific IOWF-architectures.

4. OUR GLOBAL APPROACH FOR WF INTERCONNECTION

Regarding the issue of *WF* interconnection, we use a SOA-based approach as already evoked, our main idea is to encapsulate each *WF* fragment into a single service or a set of services according to the interaction points existing in the basic IOWF-architecture so as that interactions between *WF* fragments turn to invocation of services.

4.1. Encapsulation of a WF Process into Services

The encapsulation of a WF process (or a sub-process) into a service is possible due to conceptual and technical similarities between the concept of WF and the concept of service. Figure 4 exhibits these conceptual similarities.

- Conceptual Aspects

A WF process is attached to a business partner exactly as a business service. A service is eventually composed by other services (components), in the same manner a WF process is eventually composed by sub-processes having the same structure as the global WF. At the lower level of decomposition, a WF process is hierarchized into activities; an activity uses/produces data, it is submitted to a transition condition and can invoke external applications. Also, a service is hierarchized into operations (activities); each operation uses/produces data, it is submitted to a pre-condition (analog to transition condition) and can invoke external services (applications).

In addition, a WF process covers a global business functionality that can be decomposed into sub-functionalities performed by sub-processes. Service in turn, has a global business functionality that can be decomposed into sub-functionalities performed by the service components. Therefore, we can say that a WF process is conceptually similar to a business oriented service.

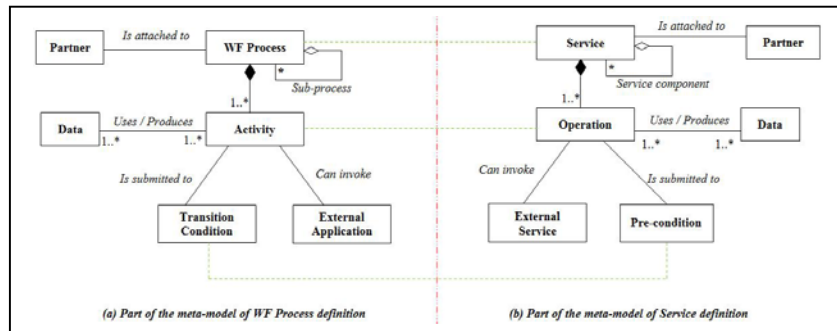


Figure 4. Correspondence of Concepts – WF vs Service

- Technical Aspects

Technically, a service has an interface and a description allowing its invocation in accordance with syntactic, semantic and QoS constraints. Similarly, a WF process has a description and an interface (set of API) for its invocation from another WF through the interface 4 of the reference model proposed by the WFMC coalition. Thus a WF process (or sub-process) can be considered as a business service performing a well defined functionality and that should be invoked through an interface, under some constraints. Hence the idea of encapsulating a WF process in a service which presents the advantages of a loosely coupled, interoperable and platform independent component.

4.2. Structuring of the IOWF into Services

Globally, in order to structure an IOWF schema into services, our approach is to cut each WF process provided by each partner into sub-processes that must be encapsulated into services. For that, we consider *interaction points* between the workflows involved in the cooperation as *markers* allowing the cutting of the process schema into sub-processes. A sub-process is composed of activities implementing part of the process and covering a sub-functionality, it requires input data and produces output ones. According to the interaction points in the IOWF, we can envisage two configurations: (1) the interaction points are located only at the beginning and the end of the WF invoked; (2) the interaction points are located at several points of the WF invoked. However, the two IOWF-architectures considered in this work (the “chained execution” and the “subcontracting”) obey to the first configuration, then we propose to entirely

encapsulate each WF invoked into a service in such a manner that all interactions between the workflows implied in the cooperation turn into invocation of services (see Figure 5).

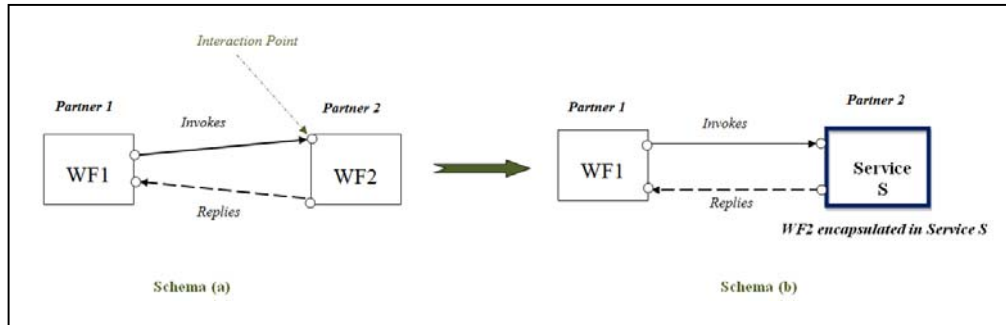


Figure 5. Generic schemas of IOWF

Depending on the IOWF architecture, the operations of invocation are interpreted differently. Indeed, for a chained execution, invocation consists to forward the instance (partially performed by partner 1) to partner 2 in order to complete its execution; for a subcontracting, the invocation consists to delegate part (one activity or more) of a principal WF to a secondary WF.

Remarks

- On Figure 5, service S is not necessarily atomic; it can be composed by several services. But, even if a service is composite, it seems to be atomic from outside since it requires a single interaction point.
- In order to deal with IOWF models completely based on services, the WF of each partner can be structured at internal level, as a set of services encapsulating activities or sub-processes and linked with control flow operators. Also, activities that require human intervention can be abstracted into specific services (for example, in Oracle-BPEL, these activities are implemented through the concept of *Task manager*).

In the following section, we introduce the concept of SBCP that exhibits the characterization of specific IOWF-architectures using SOA-based approach.

4.3. Service Based Cooperation Pattern (SBCP)

Figure 6 below describes the meta-model of SBCP definition.

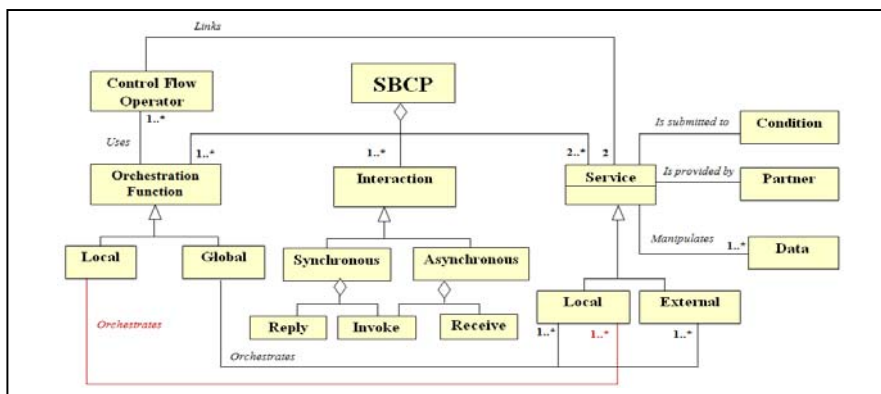


Figure 6. Meta-model of a SBCP Definition

In addition to the cutting of the WF process into services, we should decide about the appropriate mode of control of execution at runtime and the structure of interaction between services. This leads us to

three main questions: (1) How to *structure* the WF process into services? (2) How to *control* the execution of instances? (3) how to *define interactions* between services provided by different partners? These three questions exhibit three main dimensions on which is based the concept of SBCP as shown on figure 6. here, we define a sbcp in a generic manner (covering all the IOWF-architectures defined in (Van Der Aalst, 99), (Van Der Aalst, 2000)). In section 5 below, we exhibit the specificities of the two SBCP related to the IOWF-architectures considered in this work.

Regarding the first dimension which is the *distribution of services* among the partner's sites, we consider that each service encapsulates part or all of the WF process and is implemented at the partner's site that provides it. This dimension corresponds to the dimension *Process partitioning* which is defined for the initial IOWF-architectures (see the meta-model of Figure 3). From the perspective of a given partner, a service can be implemented locally (local service) or provided by an external partner (external service).

The second dimension which is the *control of execution* (centralized, decentralized or hierarchized) is expressed through the concept of orchestration function that abstracts the structure of the process in terms of control flow and interactions between services composing the IOWF process. Hence, in case of centralized control, there is one global orchestration function implemented at the site of one partner that controls the execution of the whole IOWF. In contrast, in case of decentralized control, there is a set of local orchestration functions. Each orchestration function is implemented at one partner site and allows the control of the fragment implemented at the same partner site. In case of hierarchized control, there is one global orchestration function that controls the invocation of internal and external services and a set of local orchestration functions that control the execution of secondary workflows implied in the “*subcontracting*” cooperation. The concept of orchestration function is defined and illustrated in Section 4.4 and Section 5 below.

The third dimension defines the interactions between services of several partners implied in the IOWF process. This dimension is expressed via interactional activities (invoke-receive for asynchronous or one - a- way communication and invoke-receive-reply for synchronous communication).

4.4. Orchestration Function and Control Flow

Like shown on the meta-model of Figure 6, the concept of *orchestration function* describes the control flow between services composing the IOWF using basic control flow operators. On Figure 7, we introduce these basic operators and we express them using a general notation independently from any language or platform.

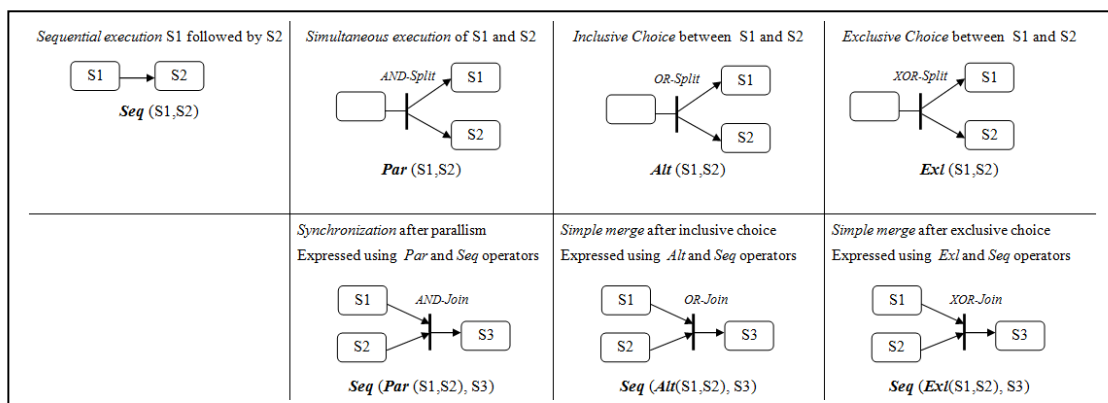


Figure 7. Basic control flow operators

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

In the following, we describe the two SBCP related to the “chained execution” and the “subcontracting” architectures. For each cooperation pattern, we give some descriptive details (reference, name, structure, control, type of interaction, use in practice), a generic schema and a meta-model. We refer to the “chained execution” pattern and the “subcontracting” pattern by SBCP2 and SBCP3 respectively, in order to keep the same references as in our other works.

5. DESCRIPTION OF THE TWO SBCP

5.1. The “Chained Execution” Pattern –SBCP2

For the SBCP obeying to the chained execution architecture (SBCP2), the WF of each partner is *entirely* encapsulated within a service that means service S_i encapsulates WF_i provided by partner i . Process instances are executed according to the *sequence* of services implemented. The first service ($S1$) of the sequence is triggered by an external event (the occurrence of a new instance), the other services of the sequence, each of which is triggered by the service that precedes it in the sequence. In a general way, a service S_{i+1} is invoked by service S_i that precedes it once S_i terminates its execution.

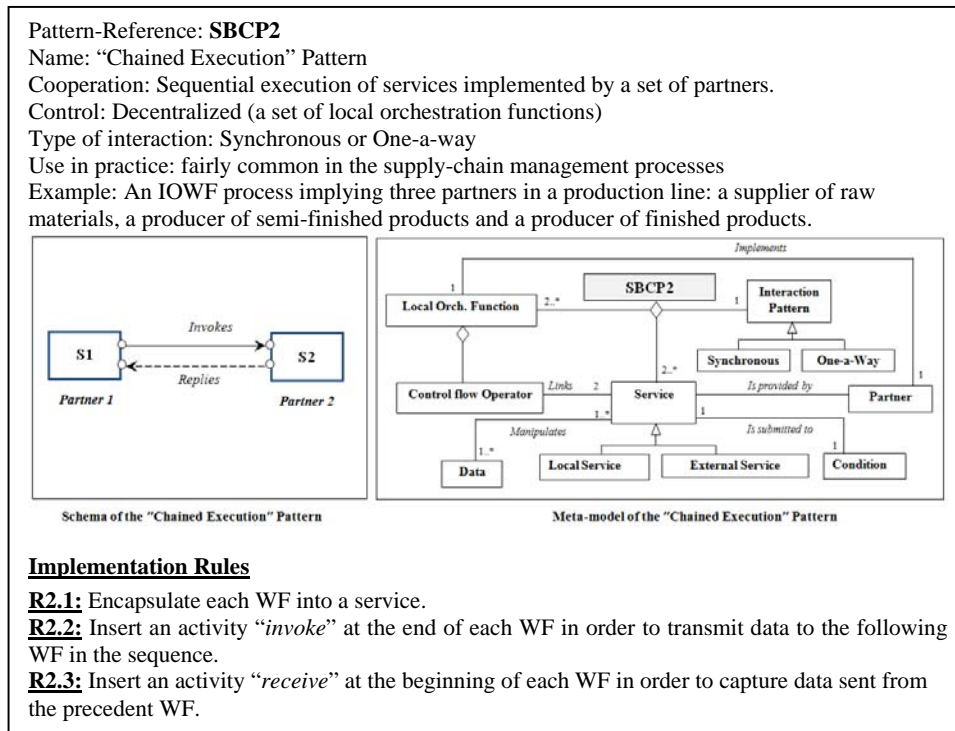


Figure 8. Description of the “Chained Execution” Pattern- SBCP2

In Figure 8, we give general characteristics of the pattern (reference, name, cooperation, control mode, use in practice); we provide a generic schema and a meta-model of the pattern definition, the dotted arrow on the schema indicates a facultative reply to the service invoker (for notification); at the bottom of the same figure, we give the set of rules (R2.1, R2.2,...) that should be used in our framework of cooperation, to implement an IOWF obeying to the SBCP2 pattern.

SBCP2 uses a *decentralized control* of execution; we implement it as a set of local orchestrations of services (Boukhedouma & al, 2012a). At internal level, services S_i can be implemented as composite services since they respectively encapsulate the WF of each partner; each internal activity of WFi is implemented as a *local service* S_{ij} . Local services are orchestrated using a local *orchestration function* implemented at each partner where maintaining a *decentralized control* of execution in the IOWF. The local orchestrator (see Figure 9) of partner i has to receive *input data* from another orchestrator and to *invoke its local composite service* (S_i) with this input data and then to invoke service S_k of the next partner by sending *results* (output) of its local service.

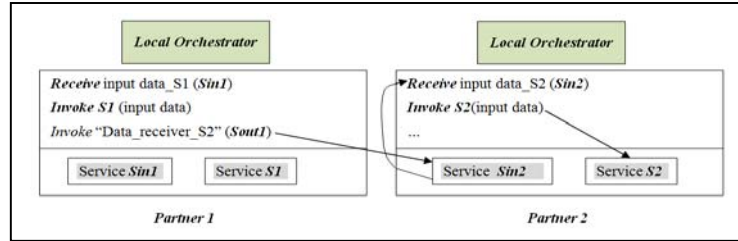


Figure 9. Illustration of local orchestrators

For this cooperation pattern (SBCP2), the interaction between services obeys to a “one-a-way” interaction pattern if no reply (to the invoker) is necessary or a synchronous interaction pattern if we consider a reply for notification. In a one-a-way interaction, or fire and forget, the client sends a message to the service and does not wait for a response. In BPEL (Figure 10(a)), this interaction pattern is implemented using an *invoke* activity from the client (WFi) and a *receive* activity at the service ($WFi+1$) that becomes in turn a client when it invokes the next service ($WFi+2$). In a synchronous interaction pattern, the client process invokes the service and waits for a reply in order to perform the execution of the rest of the client process. In BPEL, this pattern is implemented using an *invoke* operation from the client and a *reply* from the service (see Figure 10 (b)); of course, a *receive* activity is also implemented at each partner.

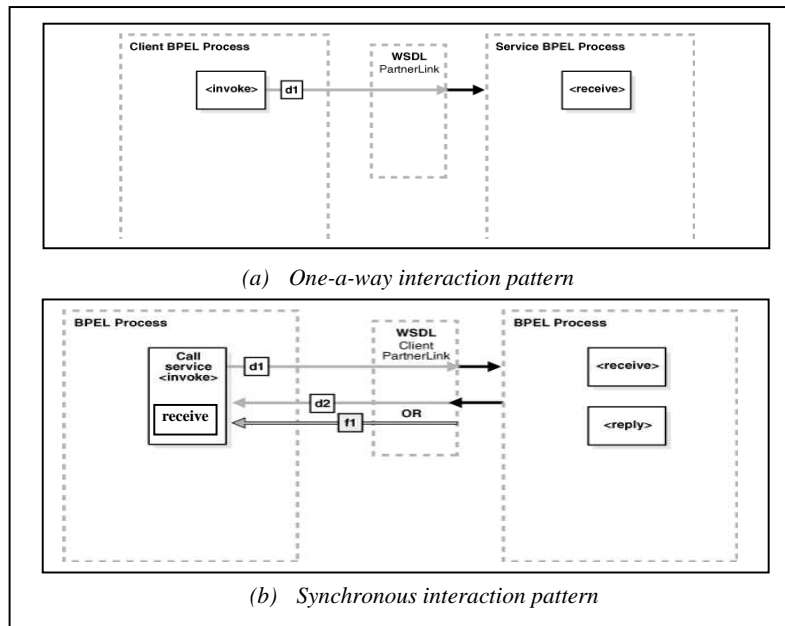


Figure 10. Interaction patterns in BPEL

Figure 11 below illustrates the concept of orchestration function using our notation on an example of IOWF model obeying to the SBCP2. 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*. For more readability and less complexity of the orchestration function, we can structure the process fragments into blocks *Bij* of sequential, parallel or alternative services. In a hierarchical manner, a block can be expressed using other blocks. *Sout1* corresponds to an activity “*invoke*” of service *S2* and *Sin2* corresponds to an activity “*receive*”.

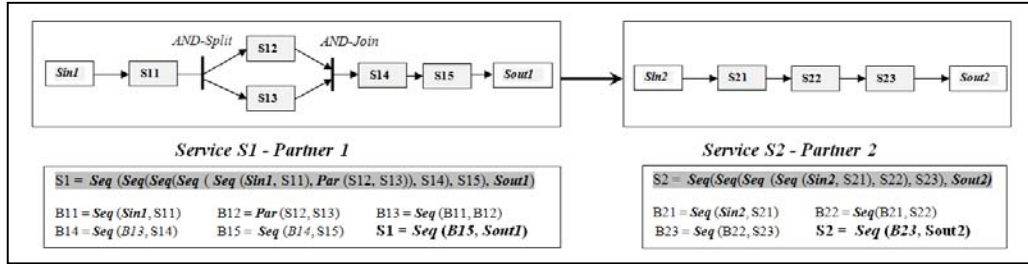


Figure 11. Illustration of orchestration functions - SBCP2

5.2. The “Subcontracting” Pattern –SBCP3

To realize a service-based subcontracting pattern (SBCP3), we propose to entirely encapsulate *each secondary WF* involved in the cooperation within a *service* (Boukhedouma & al, 2012b). On Figure 12, partner 1 hosts the main WF and partner 2 provides his secondary WF as a global *service S2* which can be composite but from the perspective of the main partner, it is abstracted to a single entity; 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 SBCP3, the control of execution is *hierarchized* because the main WF manages the control of the whole process and controls invocation of external services. SBCP3 is described through the meta-model of Figure 12. For this pattern, the interaction between services is synchronous.

To illustrate the concept of orchestration function on SBCP3, we give a simple example of IOWF like shown on Figure 13. The process schema describes an IOWF implying two partners, partner 1 and partner 2. Partner1 provides the main WF composed by *internal* services *S11*, *S12*, *S13*, *S14* and an invocation of *S2* which is the *external* service provided by partner 2.

Pattern-Reference: **SBCP3**

Name: “Subcontracting” Pattern

Cooperation: Externalization of services to other partners

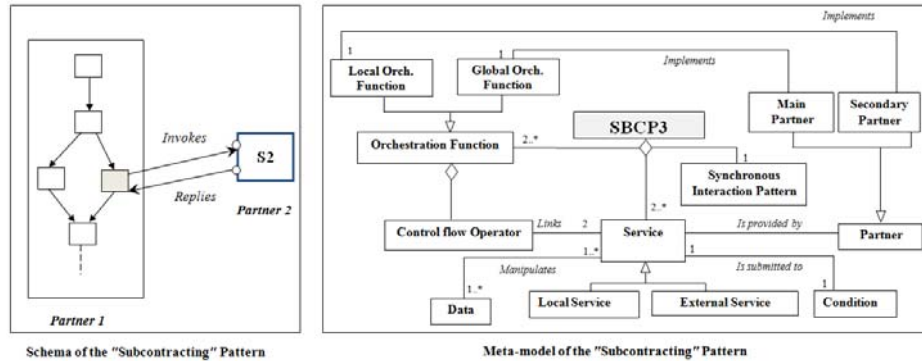
Structure: A set of internal and external services orchestrated by a global orchestration function implemented at the main partner and a set of local orchestration functions, each of which implemented at the corresponding secondary partner.

Control: Hierarchized

Type of interaction: Synchronous

Use in practice: Fairly common between business partners with complementary skills and competencies.

Examples: Processes of pharmaceutical production, automotive processes, manufacturing and assembly of integrated circuits.



Implementation Rules

R3.1: Encapsulate each secondary WF into a service.

R3.2: Insert an activity “invoke” into the main WF in order to invoke the service encapsulating the secondary WF.

R3.3: Insert an activity “receive” at the beginning of the secondary process to be invoked, in order to receive the input data sent by the main workflow.

R3.4: Insert an activity “reply” at the end of the secondary WF in order to return results to the main WF.

R3.5: Insert an activity “receive” into the main workflow after the corresponding activity “invoke” in order to receive results from the secondary WF.

Figure 12. Description of the “Subcontracting” pattern – SBCP3

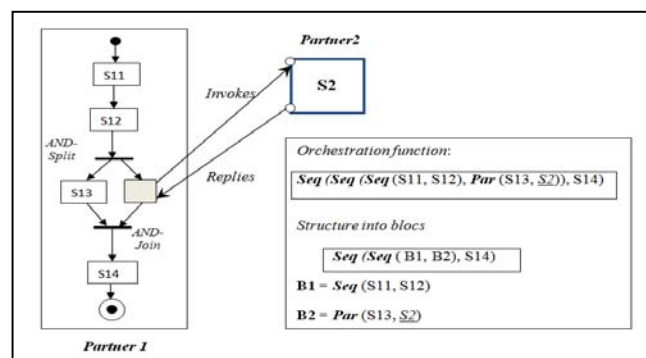


Figure 13. Illustration of orchestration functions - SBCP3

In the next section, we focus on the second issue of our work which is the adaptability and the evolutivity of IOWF models obeying to the SBCP previously described.

6. ADAPTABILITY OF IOWF MODELS

According to the previous descriptions and the meta-model of Figure 6, adaptation of process models turns to modifications of the entities attached to the three dimensions defining a SBCP that means *services, orchestration functions and/or interactions*. Consequently, we classify our adaptation patterns into three main: *Service* adaptation patterns, *Control Flow* adaptation patterns and *Interaction* adaptation patterns.

6.1. Service Adaptation Patterns

These patterns concern the modifications that can be applied on the services composing the IOWF process; these modifications are typically adding, removing, replacing, merging two services (in a sequential, parallel or alternative block) and decomposing a service into a block of two services expressing sequential, parallel or alternative execution. An adaptation of a service usually induces modification on the *orchestration function* using it or a modification of closely attached attributes like *condition* or *data*.

6.1.1 Adding, Removing and Substituting Services

Adding a service is done in order to insert an additional step in the process. The reverse operation of adding is the *removing* of services. For *adding* or *removing* services, it is to distinguish adding or removing of a service on *one edge* composed by sequential services or in a block composed by *two edges* expressing parallel or alternative execution. Table 1 describes the basic patterns of *adding* services illustrated by generic process schemas and the corresponding *orchestration functions*. We can see that there are elementary patterns named AP1.1, AP1.2, respectively for adding a new service before or after a given service in the process, and there are more elaborated patterns like AP1.3, AP1.4 and AP1.5 which are implemented using elementary patterns AP1.1 or AP1.2, depending on the location of the service to add.

Table 1. Description of “Adding Service” Patterns

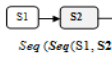
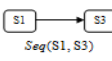


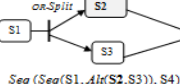

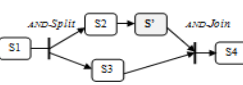

AP1: “Adding Service” Patterns				
Pattern Reference	Pattern Description	Before adaptation	After adaptation	Patterns used
AP1.1	Add into sequence before a service	 $Seq(S1, S2)$	 $Seq(Seq(S', S1), S2)$	None (Elementary pattern)
AP1.2	Add into sequence after a service	 $Seq(S1, S2)$	 $Seq(Seq(S1, S'), S2)$	None (Elementary pattern)
AP1.3	Add on one edge of inclusive choice	 $Seq(Seq(S1, Alt(S2, S3)), S4)$	 $Seq(Seq(S1, Alt(Seq(S2, S'), S3)), S4)$	AP1.1 AP1.2
AP1.4	Add on one edge of exclusive choice	 $Seq(Seq(S1, Ext(S2, S3)), S4)$	 $Seq(Seq(S1, Ext(Seq(S2, S'), S3)), S4)$	AP1.1 AP1.2
AP1.5	Add on one edge of parallel execution	 $Seq(Seq(S1, Par(S2, S3)), S4)$	 $Seq(Seq(S1, Par(Seq(S2, S'), S3)), S4)$	AP1.1 AP1.2

Table 2 shows typical operations of removing services (service S2 for example). Let's notice that two configurations are possible when removing a service S from a block with two edges: (1) service S is in sequence with other services, (2) service S is alone on the edge; this results on two different scenarios of adaptation. These two configurations are represented only for inclusive choice, but in our implementation,

they are also considered for exclusive choice and parallel execution. For the removing patterns, we can see that AP2.1 is an elementary pattern and AP2.2, AP2.3, AP2.4, AP2.5, etc. are implemented using AP2.1.

Another basic operation of adaptation concerns the substitution (*replacing*) of services. This is typically a *removing* of the service to replace followed by an *adding* of the new service. Then, the pattern **AP3** (called “Service Substitution” Pattern) is implemented using patterns PA1.x and PA2.x for respectively adding and removing, depending on the location in the process schema (in sequence, parallel or alternative) of the service to be replaced.

Table 2. Description of “Service Removing” Patterns

AP2: “Service Removing” Patterns				
Pattern Reference	Pattern Description	Before adaptation	After adaptation	Patterns used
AP2.1	Remove from sequence	 $Seq (Seq(S1, S2), S3)$	 $Seq(S1, S3)$	None (Elementary pattern)
AP2.2	Remove from one edge with several services of inclusive choice	 $Seq (Seq(S1, Alt(Seq(S2, S3), S4)), S5)$	 $Seq (Seq(S1, Alt(S3, S4)), S5)$	AP2.1
AP2.3	Remove from one edge with single service of inclusive choice	 $Seq (Seq(S1, Alt(S2, S3)), S4)$	 $Seq (Seq(S1, S3), S4)$	AP2.1
AP2.4	Add on one edge of parallel execution	 $Seq (Seq(S1, Par(Seq(S2, S'), S3)), S4)$	 $Seq (Seq(S1, Par(S2, S3)), S4)$	AP2.1

6.1.2 Fusion and Decomposition of Services

The operation of *fusion* can concern two services linked 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. Table 3 below describes these basic operations and the corresponding orchestration functions modified after each operation for merging $S2$, $S3$ in a single service S' . We can state that since services to merge are in the same block, they become easier to remove and to replace, because the block ($Alt(S2, S3)$, $Par(S2, S3)$ or $Exl(S2, S3)$) is considered as a single *composite service* to be replaced. More elaborated operations of fusion concern configurations such as services to merge are not in the same block. 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 additional parameter. In both cases, this must be decomposed into elementary operations of removing and adding of single services or blocks. Then, the fusion patterns are implemented using the adding and the removing patterns AP2.5 and AP2.6 which are not represented on Table 2, correspond to removing a service from one edge with a single service of parallel execution and of exclusive choice respectively.

The reverse operation of fusion is the *decomposition* of a service to obtain a block of two services that can be sequential, parallel or alternative block. The decomposition of services can be done to improve the parallelism in the process (parallel decomposition) or to add condition (alternative decomposition) due to new constraints or to have more control on process execution (sequential decomposition). We can see on Table 4 that the decomposition of a service consists to *remove* a single service ($S2$ for example) and to *add a block* composed by two services (S' and S'') linked by a sequence, an alternative or a parallel operator. This explains the use of adding patterns AP1.x and removing Patterns AP2.x.

Remark - For all the adaptation patterns proposed, the assignation of variables (inputs and outputs of the injected services) is done in a semi-automatic manner; a wizard that displays a set of variables is proposed to the designer who selects the appropriate variables.

Table 3. Description of Fusion Patterns

















AP4: Fusion Patterns				
Pattern Reference	Pattern Description	Before adaptation	After adaptation	Patterns used
AP4.1	Fusion of <i>sequence</i>	 $Seq(Seq(Seq(S1, S2), S3), S4)$	 $Seq(Seq(S1, S'), S4)$	AP1.1 AP1.2 AP2.1
AP4.2	Fusion of <i>inclusive choice</i>	 $Seq(Seq(S1, Alt(S2, S3)), S4)$	 $Seq(Seq(S1, S'), S4)$	AP1.1 AP1.2 AP2.3
AP4.3	Fusion of <i>exclusive choice</i>	 $Seq(Seq(S1, Exl(S2, S3)), S4)$	 $Seq(Seq(S1, S'), S4)$	AP1.1 AP1.2 AP2.6
AP4.4	Fusion of <i>parallel execution</i>	 $Seq(Seq(S1, Par(S2, S3)), S4)$	 $Seq(Seq(S1, S'), S4)$	AP1.1 AP1.2 AP2.5

Table 4. Description of Decomposition Patterns

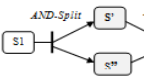
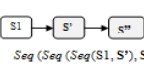
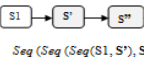
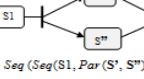
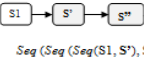


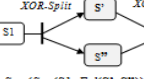
AP5: Decomposition Patterns				
Pattern Reference	Pattern Description	Before adaptation	After adaptation	Patterns used
AP5.1	Decomposition into <i>sequence</i>	 $Seq(Seq(S1, S2), S3)$	 $Seq(Seq(Seq(S1, S'), S''), S3)$	AP1.1 AP1.2 AP2.1
AP5.2	Decomposition into <i>inclusive choice</i>	 $Seq(Seq(S1, S2), S3)$	 $Seq(Seq(S1, Alt(S', S'')), S3)$	AP2.1 AP1.3
AP5.3	Decomposition into <i>exclusive choice</i>	 $Seq(Seq(S1, S2), S3)$	 $Seq(Seq(S1, Exl(S', S'')), S3)$	AP2.1 AP1.4
AP5.4	Decomposition into <i>parallel execution</i>	 $Seq(Seq(S1, S'), S3)$	 $Seq(Seq(S1, Par(S', S'')), S3)$	AP2.1 AP1.5

6.2. Control Flow Adaptation Patterns

This category of patterns concerns modification of the control flow between services composing the IOWF process, without affecting the services themselves. 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*) (parallelization of services) to improve the execution time of process instances, or vice versa (sequentialization of services) if an execution of a service becomes dependant from another service (the

order of services in the sequence should be provided as a parameter), or alternation of services if an execution of a service depends from a given condition that should be specified.

Table 5. Description of “Control Flow” Adaptation Patterns

AP6: “Control Flow” Adaptation Patterns				
Pattern Reference	Pattern Description	Before adaptation	After adaptation	Patterns used
AP6.1	Sequentialization of services	 $Seq(Seq(S1, Par(S', S'')), S3)$	 $Seq(Seq(Seq(S1, S'), S''), S3)$	AP1.1 AP1.2 AP2.3
AP6.2	Parallelization of services	 $Seq(Seq(Seq(S1, S'), S''), S3)$	 $Seq(Seq(S1, Par(S', S'')), S3)$	AP2.1 AP1.5
AP6.3	Inclusive Alternation of services	 $Seq(Seq(Seq(S1, S'), S''), S3)$	 $Seq(Seq(S1, Alt(S', S'')), S3)$	AP2.1 AP1.3
AP6.4	Exclusive Alternation of services	 $Seq(Seq(Seq(S1, S'), S''), S3)$	 $Seq(Seq(S1, Ext(S', S'')), S3)$	AP2.1 AP1.4

Even if there is no modification on services implied in the IOWF, the implementation of the control flow patterns uses other patterns of adding and removing services (see Table 5) because we have to update input and output data of services and also the conditions of invocation.

6.3. Interaction Adaptation Patterns

This category of patterns concerns modification of the interactions between services composing the IOWF process and provided by different partners. Specifically, updating the structure of interaction is done by adding, removing or updating *interactional points* (see Table 6).

Table 6. Description of “Interaction” Adaptation Patterns

AP7: “Interaction” Adaptation Patterns			
Pattern Reference	Pattern Description	Scenarios	Patterns used
AP7.1	Add Interaction Point	<ul style="list-style-type: none"> Add an external service Substitute a local service by an external one 	AP1.x AP3
AP7.2	Remove Interaction Point	<ul style="list-style-type: none"> Remove an external service Substitute an external service by a local one 	AP2.x AP3
AP7.3	Update Interaction Point	<ul style="list-style-type: none"> Update input/output data exchanged Update the interaction mode (synchronous/asynchronous) 	AP3

7. EVOLUTIVITY OF IOWF MODELS

As already explained, the *evolutivity* (or evolutive adaptability) of IOWF process models is reflected at two perspectives: the *functionality* and the *cooperation* of the IOWF. Hence, an IOWF model is evolvable if it can be extended to additional functionalities or if it allows expansion of cooperation to involve more

partners and more external services. The two perspectives are not exclusive; indeed expansion of the cooperation can induce expansion of functionalities and vice versa.

7.1. Expanding Functionalities

Expansion of functionalities of the IOWF can be done by adding internal services S_{ij} (respectively blocks) with novel functionalities into the WF of one or more partner(s) or by replacing a service (respectively block) by another that covers more functionalities. To do that, we can refer to operations described in Section 6.1, the only difference is that the injected services implement additional functionalities in the IOWF process. At external level, the expansion of functionalities can be realized by replacing an external service S_i encapsulating a WF fragment by another external service that covers more functionality.

7.2. Expanding Cooperation

According to the *cooperation* perspective, it is the capacity to open the IOWF to more partners. For this perspective, the evolution patterns depend on the cooperation pattern defining the IOWF process. For this second perspective, we show that in some cases, an IOWF model can evolve while maintaining the initial cooperation pattern and in other cases it can evolve and fall into the combination of two different cooperation patterns. To illustrate these two kinds of evolution, we describe evolution patterns applied to the IOWF models obeying to SBCP2 and SBCP3.

7.2.1 Expanding the “Chained Execution”

An IOWF obeying to the chained execution pattern (SBCP2) can be extended according to one of these two configurations:

- Adding a new external service encapsulating a WF provided by a new partner in order to extend the functionalities of the initial IOWF or to add a new intermediary phase in the IOWF process.
- Replace a given service in the sequence of the IOWF model by an exclusive block of two services provided by two different partners. In all cases, the process instances are exclusively executed according to one sequential edge obeying to the chained execution pattern.

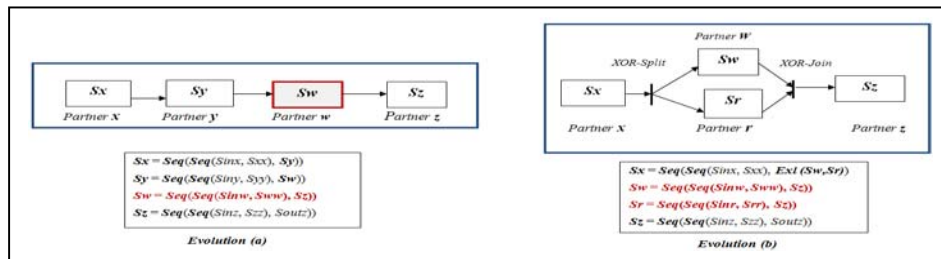


Figure 14. Expanding the chained execution

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, Figure 14 shows the possible configurations of evolution previously described. In case of evolution (a), we have to add the orchestration function of service w and to modify the interactional services $Souty$ and $Sinz$ insuring interaction with the novel service w . In evolution (b), we have to add the orchestration functions of services r and w and to modify the interactional services $Soutx$ to implement the exclusive choice and $Sinz$ insuring interaction with the

novel service r or w . Let's notice that the pattern SBCP2 is preserved since instances are executed according to one path of sequential services (Sx, Sw, Sz) or (Sx, Sr, Sz).

7.2.2 Expanding the “Subcontracting”

Expansion of the subcontracting can occur when the main partner subcontracts other activities to external partners or when a secondary partner in turn subcontracts part of its WF to other partners, this results in what we call “*multi-level subcontracting*”.

Expansion of subcontracting can be done according to one of the configurations (a, b, c) shown on Figure 15. *OP-Split* means OR-Split, XOR-Split or AND-Split.

- (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 Section 6. The only difference is that evolutivity concerns *external* services.

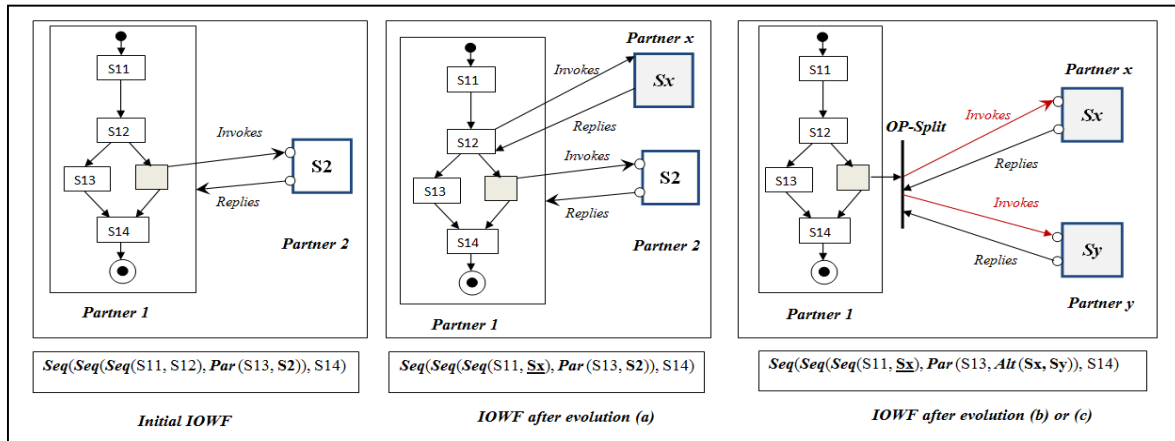


Figure 15. Expanding the subcontracting

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 from the perspective of the main WF but the overall IOWF implies additional partners at different levels. Figure 16 shows a schema of this configuration. Changes relative to this configuration are done at the secondary partner by substituting one or more of its local services by external services.

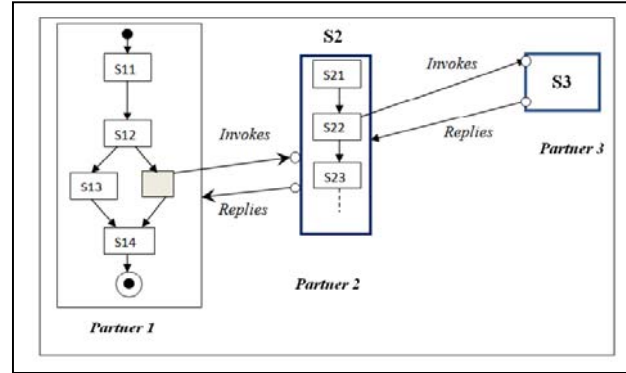


Figure 16. Multi-level subcontracting

7.2.3 Evolution by Combination of the SBCP2 and SBCP3

Describing the WFIO as a composite service using the concept of orchestration function allows the combination of two IOWF obeying to the same or different SBCP. In fact, the IOWF is manipulated itself as a composite service that can be added, removed or substituted with respect of the constraints relative of each IOWF-architecture. In the following, we describe two IOWF evolution patterns based on the combination of SBCP2 and SBCP3; this represents one aspect of reuse of IOWF models and in such architectures, there is cooperation pattern which is predominant compared to the other. Then we talk about a “chained execution in a subcontracting” or “a subcontracting in a chained execution” (SBCP2 in SBCP3 or SBCP3 in SBCP2).

- A “chained execution” in a “subcontracting”

This configuration is obtained when the main partner subcontracts part of its WF to a group of partners which are involved in the “chained execution” cooperation. In this case, the predominant pattern is SBCP3 because the execution of the IOWF is controlled by the main partner that invokes the IOWF obeying to the SBCP2 pattern provided by the group of partners. This last, once invoked is controlled by the group of (secondary) partners. Also, this architecture requires a return of results in a reverse sequence in the IOWF obeying to SBCP2 in order to communicate the results of subcontracting to the main partner. This falls on IOWF-architecture with a mixed control “hierarchized/decentralized”. On Figure17, partner 1 subcontracts part of its WF to the group of partners x, y, z providing a WFIO composed by the sequence of services (S_x , S_y , S_z). Partner 1 invokes the first service (S_x) of the sequence.

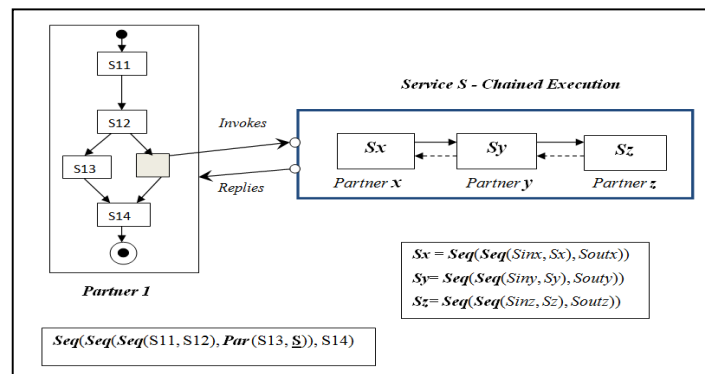


Figure 17. Combination of Cooperation Patterns – SBCP2 in SBCP3

- A “subcontracting” in a “chained execution”

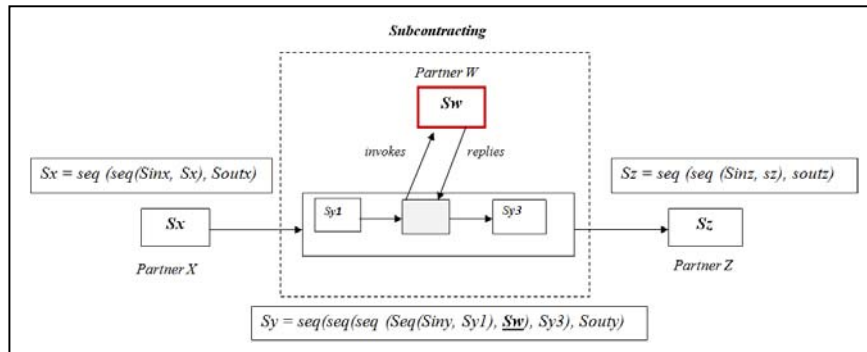


Figure 18. Combination of Cooperation Patterns – SBCP3 in SBCP2

This configuration can be used when a partner involved in an IOWF obeying to SBCP2 decides to subcontract part of its local WF to an external partner. In this case, the predominant pattern is SBCP2 and on a given fragment of the sequence, there is a subcontracting. The control of execution is mixed decentralized/hierarchized. On Figure 18, Partner y subcontracts part of its WF to partner w; a modification is done on the orchestration function of service y in order to invoke the external service Sw.

8. IMPLEMENTATION DETAILS

Since our work focuses on two main issues which are closely linked: (1) the service-based cooperation patterns (SBCP) and (2) the adaptation (respectively evolution) patterns applied to process models obeying to the SBCP defined, our implementation contains two main frameworks: (1) a framework for cooperation that supports the generation of SBCP starting with service-based process models provided by a set of partners, and (2) a framework for adaptation composed by a set of adaptation (respectively evolution) patterns that support the adaptation (respectively evolution) of the generated process models.

The implementation of our frameworks provides a set of wizard functions allowing the guidance of the designer step by step in order to realize a specific IOWF-architecture according to a selected SBCP or a specific operation of adaptation on a given IOWF model obeying to one of the SBCP defined.

For the deployment of the framework of cooperation, we suggest to install the applicative modules at a tier structure accessing to abstracted definition of the WF of each partner that should be implied in the cooperation. Once the IOWF model is realized (based on the implementation rules specified for each SBCP and a control of coherence of the data flow when invoking services), its execution is triggered to check the correct execution for instances conformably to the process model built.

For the deployment of the framework of adaptation, applicative modules can be accessed by all partners for adaptations affecting the internal structure of the process. But, adaptations that affect the cooperation pattern or interactional aspects cannot be done locally; these should be delegated to a tier structure in order to guarantee the coherence of the global IOWF.

8.1. Development Tools

For the development of our frameworks, we have considered process models specified with BPEL and interpreted by the WF engine OPEN ESB 2.2, we also used a plug-in SOA Netbeans. We have developed our frameworks using the Java language and the IDE Netbeans, the application server used is GlassFish server version 2. To implement the cooperation and the adaptation patterns, we have used the API jdom2 that eases the modification on the code BPEL specifying the WF processes since it is based on the XML

language. For example, we simply use the class *Element* implemented in the API jdom to create a new XML tag.

Our frameworks are as modular as possible since we implement a separate class for each elementary operation to perform on process models. For example, for our framework of adaptation, we create a class for adding a service *after* a given service in a sequential branch, another class for adding a service *before* a given service in a sequential branch, another class for adding a service in an alternative block, etc. This eases the reuse of existing classes to implement other ones; thus, the operations of substitution, fusion and decomposition are implemented using elementary operations of adding and removing of services (see Tables 3, 4). For the application of the adaptation patterns, after each operation of adaptation, we run the adapted process in order to check that the adaptation has been successfully done. In the following, we illustrate the implementation of the adaptation patterns AP1.2 and AP4.4.

8.2. Examples of Implementation

- The AP1.2 Pattern

Figure 19 below shows the interface related to implementation of AP1.1 or AP1.2 patterns (add a service in sequence). Once the WF process to adapt is selected, the designer has to introduce some parameters like the name of the service (for example “NewService”) to add, the inputs and outputs, the location (before or after what service). At the right side of Figure 19, we show the code java corresponding to the implementation of the pattern AP1.2 “add in sequence after...”

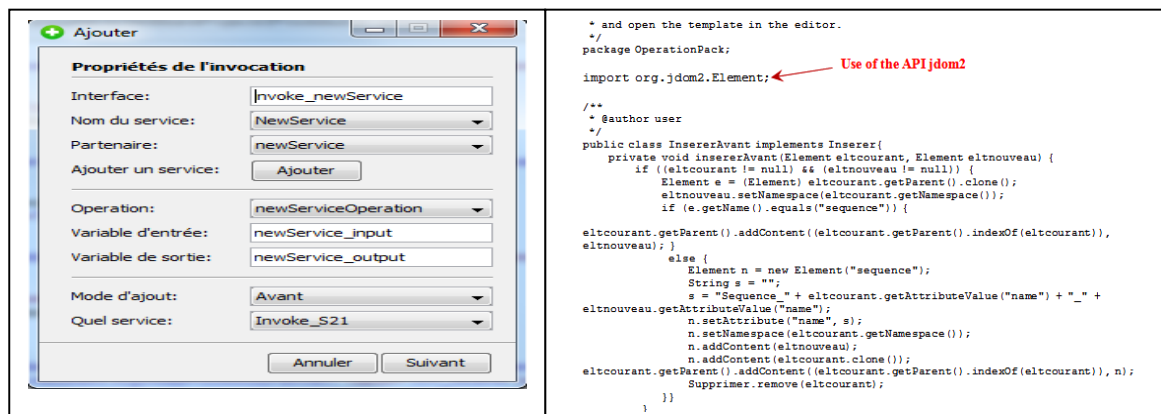


Figure 19. Interface “Adding a service” and Part of the corresponding java code

- The AP4.4 Pattern

Figure 20 shows the interface of fusion and part of the code java corresponding to the implementation of the pattern AP4.4 “Parallel Fusion of services”.

- Update Variables and Conditions

In order to maintain the coherence of the process after realizing a cooperation pattern or an adaptation, our applications provide an interface allowing the update of the data flow in the process. It is to select a service and all input/output variables are displayed to the designer who selects the appropriate input/output variables.

Also, when the adaptation concerns alternative blocks, we have to generate the correct conditions of choice, then our application provides a simple graphical wizard allowing the generation of simple or composite conditions.

<div> <p>Interfaces à fusionner</p> <p>Interface n°1: <input type="text" value="service1"/></p> <p>Interface n°2: <input type="text" value="service2"/></p> <p>Invocation fusion</p> <p>Nom de l'invocation: <input type="text" value="newService"/></p> <p>Nom du service: <input type="text" value="Traducteur"/></p> <p>Partenaire: <input type="text" value="TraducteurPartner"/></p> <p>Ajouter un service: <input type="button" value="Ajouter"/></p> <p>Operation: <input type="text" value="traduire"/></p> <p>Variable d'entrée: <input type="text" value="newServiceIn"/></p> <p>Variable de sortie: <input type="text" value="newServiceOut"/></p> <p><input type="button" value="Annuler"/> <input type="button" value="Suivant"/></p> </div>	<pre> package OperationPack; import IOpack.BpelFile; import java.util.ArrayList; import java.util.Iterator; import java.util.List; import org.jdom2.Element; import org.jdom2.Namespace; /** * @author user */ public class Fusion { public static void fusion(BpelFile bpel, Element firstService, Element secondService, Element newService) { Element parentElement = firstService.getParentElement(); int size = parentElement.getChildren().size(); Inserer insert = new InsererAvant(); if (size > 2) { insert.inserer(firstService, newService, null); Supprimer.supprimer(bpel, "invoke", firstService.getAttributeValue("name")); Supprimer.supprimer(bpel, "invoke", secondService.getAttributeValue("name")); } else if (size == 2) { insert.inserer(parentElement, newService, null); Supprimer.supprimer(bpel, parentElement.getName(), parentElement.getAttributeValue("name")); } } } </pre>
---	---

Figure 20. Interface and part of the java code corresponding the implementation of the AP4.4 pattern

9. CONCLUSION

The current paper focuses on two main issues which are the WF cooperation and the WF adaptation. So, we exhibit two main contributions closely linked. First, in order to deal with IOWF process models flexible enough, we have proposed cooperation patterns based on SOA paradigm called service-based cooperation patterns (SBCP) in order to implement specific IOWF-architectures (here we deal with the “chained execution” and the “subcontracting” architectures). Depending on the IOWF-architecture, we express the process model using a global orchestration function or a set of local orchestration functions.

For the second issue that concerns the adaptation of process models obeying to the SBCP defined, we classify our adaptation patterns in three categories according to the three dimensions (services, control flow and interaction) on which we define the concept of SBCP. Specific operations of adaptation called evolutions affect the global functionality of the IOWF model and/or the cooperation and depend on the SBCP on which the IOWF model is based. Regarding the cooperation perspective, we show that in some cases, an IOWF model can evolve while maintaining the initial cooperation pattern and in other cases it can evolve and fall into the combination of two different cooperation patterns. We have illustrated these two kinds of evolutions on IOWF models obeying to SBCP2 and SBCP3. For the implementation of the proposed patterns, we have considered process models specified with BPEL.

Let's notice that the adaptation patterns can be applied at build time by a designer of the process or at runtime combined with a technique (such as rule-based technique) of dynamic adaptation. At this stage of our work, we have only simulated dynamic adaptation by suspending process instances, applying adaptation and resuming the execution of the suspended instances. Furthermore, 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. This aspect has been briefly illustrated, for example, by the two scenarios of evolution (SBCP2 in SBCP3 and SBCP3 in SBCP2).

We are currently working on the implementation of cooperation patterns suitable to other IOWF-architectures that means the “case transfer” that we have already considered in our previous works (Boukhedouma & al, 2011), (Boukhedouma & al, 2012c) and the “loosely coupled” architecture. Furthermore, we intend to complete our framework of adaptation by implementing adaptation and evolution patterns suitable to these architectures.

ACKNOWLEDGMENT

We would like to thank our students Djamel-Eddine Khelladi, Younes Asma, Ait Belkacem Lyes and Si-Mohamed Mehdi for their participation in the implementation of the frameworks.

REFERENCES

- Alonso G., Casati F., & Kuno H., (2004), Web services: concepts, architectures and applications. *Heidelberg, Germany, Springer Verlag*.
- Amireza T., (2009), Web Service Composition Based Interorganizational Workflows, Sudwestdeutscher Verlag fur Hochschulschriften (Ed.), ISBN 9783838106700.
- Belhajjame K., Vargas-Solar G., & Collet C., (2005), Pyros - an environment for building and orchestrating open services. *In Proceedings of the IEEE International Conference on Services Computing, 155–164*.
- Boukhedouma S., Alimazighi Z., Oussalah M., & Tamzalit D., (2011), Une approche basée SOA pour l'interconnexion de workflows : application au « transfert de cas ». *In proceedings of INFORSID' 2011, 43-58*.
- Boukhedouma S., Alimazighi Z., Oussalah M., & Tamzalit D., (2012 a), Adaptability of service based workflow models : the chained execution architecture. *In proceedings of BIS'2012, Lithuania, 21-23 may. W. Abramowicz et al. (Eds.), LNBP 117, Springer-Verlag Berlin Heidelberg*.
- Boukhedouma S., Oussalah M., Alimazighi Z., & Tamzalit D., (2012b), SOA Based Approach for Adaptability of Workflow Models: The subcontracting architecture. *In proceedings of ICEIS'2012, 224-23.1*
- Boukhedouma S., Alimazighi Z., Oussalah M., & Tamzalit D., (2012c), Interconnecting workflows using services: an approach for case transfer with centralized control. *In proceedings of ICISTM'2012, S. Dua et al. (Eds.): CCIS 285, 396–401, Springer-Verlag Berlin Heidelberg*.
- 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*.
- Dadam, P., & Reichert, M. (2009). The ADEPT project: a decade of research and development for robust and flexible process support. *Computer Science-Research and Development, 23(2), 81-97*.
- Döhring M., Zimmermann B., Godehardt E., (2010), Extended workflow flexibility using rule-based adaptation patterns with eventing semantics. *In proceedings of INFORMATIK'10, 216-226*.
- Döhring M., Zimmermann B., & Karg L., (2011), Flexible Workflows at design and Runtime using BPMN2 Adaptation Patterns. *In proceedings of BIS'2011- Springer Verlag*.
- Eder J., & Gruber W., (2002), A meta model for structured workflows supporting workflow transformations. *In Proceedings of the Sixth East European Conference on Advances in Databases and Information Systems (ADBIS 2002). 326 –339*
- Papazoglou M. P., & Van Den Heuvel W.J., (2007), Service Oriented Architectures : approaches, technologies and research issues. *The VLDB Journal, 16, 389-415*.
- Gorton S., Montangero C., Reiff-Marganiec S., & Semini L., StPowla, (2009), SOA, Policies and Workflows, *ICSOC workshops, LNCS 4907, 351-362*.
- Grefen P., Aberer K., Hoffer Y., & Ludwig H., (2001), CrossFlow: Cross-organizational workflow management for service outsourcing in dynamic virtual enterprise. *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*.
- Jordan D., & Evdemon J., (2006), Web services business process execution language V.2.0, W3C.
- Leymann F., Roller D., & Schmidt M.-T., (2002), Web Services and Business Process Management. *IBM Systems Journal 41(2)*.

- Mehandjiev N., Stalker I., Fessl K., & Weichhart. G. , (2005), Interoperability contributions of CrossWork. *In invited short paper to Proceedings of INTEROP-ESA'05 Conference*, Springer-Verlag.
- Meng J., Su S.Y.W, Lam H., Helal A., Xian J., Liu X., & Yang S., (2006), DynaFlow: a dynamic inter-organisational workflow management system, *Int. Journal of Business Process Integration and Management*, 1(2), 101–115.
- Muller R., Greiner U., & Rahm E., (2004), AGENT-WORK: a workflow system supporting rule-based workflow adaptation. *In journal of Data and Knowledge Engineering* 51 (2) 223-256.
- Peltz C., (2003), Web Services Orchestration and Choreography, *IEEE Computer* 36(10), 46-52.
- Pérez-Castillo, R., de Guzmán, I. G. R., & Piatini, M. (2011). Business process archeology using MARBLE. *Information and Software Technology*, 53(10), 1023-1044.
- Pesic M., Schonenberg MH., Sidorova N., Van Der Aalst W., (2007), Constraint-based workflow models: Change made easy. *In Proceedings of the OTM Conference CoopIS'2007*. LNCS 4803, 77–94. Springer-Verlag, Berlin.
- Rotibi P., Murphy I., & Herzlich P., (2012), Enterprise Modernization: business agility and commercial edge through application reuse and modernization, *Creative Intellect Consulting Ltd.* <http://public.dhe.ibm.com/common/ssi/ecm/en/zsl03182usen/zsl03182usen.pdf>
- Russell N., Van Der Aalst W., ter Hofstede A.H.M., (2006), Exception handling patterns in process-aware information systems. *In CAiSE'06 (Luxembourg)*, pp. 288-302.
- Sadiq S.W., & Orlowska M.E., On capturing Exceptions in workflow process models. *In proceedings of ER'2001*.
- Tragatschnig S., Zdun U., (2011), Runtime Process Adaptation for BPEL Process Execution Engines. *15th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW)*.
- Van Der Aalst W., (1999), Process oriented architectures for electronic commerce and interorganizational workflow. *Journal of Information systems*, 24(9).
- Van Der Aalst W., (2000), Loosely Coupled Inter-organizational Workflows : modeling and analyzing workflows crossing organizational boundaries. *Journal of Information and Management* 37 (2), 67-75.
- Van Der Aalst W., (2002), Workflow Management: Models, Methods and Systems. *The MIT Press. Cambridge, Massachusetts, London, England*.
- Van Der Aalst W., ter Hofstede W.M.P, Kiepuszewski A.H.M., & Barros B. A.P., (2003), Workflow Patterns. *DAPD* 14(1), 5-51.
- Weber B., Reichert M., Rinderle-Ma S., (2008), Change patterns and change support features- Enhancing flexibility in process-aware information systems. *In Journal of Data & Knowledge Engineering* (66), 438-466.

ADDITIONAL READING SECTION

- Andonoff E., Bouaziz W., Hanachi C., Bouzguenda L., (2009), An Agent-Based Model for Autonomic Coordination of Inter-Organizational Business Processes. *INFORMATICA Journal*, 20(3), 323–342.
- Badr Y. (2008), Service oriented workflow. *Journal of digital information management*, 6(1).
- Belhajjame K., Collet C., & Vargas-Solar G., (2001), A flexible workflow model for process oriented applications. *In Web Information Systems Engineering (WISE) (1)*, page 72, 2001.
- Camarinha L. M. & Pantoja-Lima C., (1999) , A framework for cooperation in virtual enterprises. *In DIISM '98, Proceedings of the IFIP TC5 WG5.3/5.7 Third International Working Conference on the Design of Information Infrastructure Systems for Manufacturing II*, pages 305–322, The Netherlands.
- Casati F., & Shan M., (2001), Dynamic and adaptive composition of e-services. *Information Systems*, 26(3), 143–163.
- Chebbi I., Dustdar S., & Tata S. (2006), The view-based approach to dynamic interorganizational workflow cooperation. *Data and Knowledge Engineering Journal*, 56(2) :139–173.

- Chebby I. , & Tata S., (2005), CoopFlow : a framework for inter-organizational workflow cooperation. *In Proceedings of International Conference on Cooperative Information Systems*, 112–129, Agia Napa, Cyprus.
- Chen M., Zhang D., & Zhou L. (2005), Empowering collaborative commerce with web services enabled business process management system. *Decision Support System*, 2005. www.sciencedirect.com
- Fdhila, W, (2011), Décentralisation Optimisée et Synchronisation des Procédés Métiers Inter-Organisationnels. *Thèse de doctorat*, Université Henri Poincaré-Nancy I, France.
- Gerhard K., Retschitzegger W., Bernauer M., and Kappel G. (2002), Specification of interorganizational workflows - a comparison of approaches. *Technical Report 08/02, Institute of Software Technology and Interactive Systems, Business Informatics Group, Vienna University of Technology*.
- Grefen P., Mehandjiev N., Kouvas G., Weichhart G., & Eshuis R., (2009), Dynamic business network process management in instant virtual enterprises. *Computers in Industry*, 60(2), 86–103.
- Hoffner Y., Ludwig H., Gülcü C., & Grefen P. W. P. J., (2000), An architecture for cross-organizational business processes. *In Second International Workshop on Advance Issues of E-Commerce and Web-Based Information Systems*, Milpitas.
- Hofreiter, B., Huemer, C. (2008), A model-driven top-down approach to inter-organizational systems: From global choreography models to executable BPEL. *In: IEEE Joint Conference on E-Commerce Technology (CEC) and Enterprise Computing, e-Commerce, and e-services (EEE)*.
- Huang C. J., & Liao L.M., (2008), Applying intelligent agent technology to develop coordinative workflow platform for inter-organizational applications. *International journal of electronic business management*, 6(4), 185-194.
- Jan Ahn H., Lee H., Kim H., JOO Park S., & Shepherdson J., (2010), Dynamic change handling for inter-organisational workflows in open virtual emarketplaces. *International journal of intelligent information and database systems*, 4 (2).
- Khadka R. Sapkota B., Pires L., van Sinderen M., & Jansen, S., (2011), Model-Driven Development of Service Compositions for Enterprise Interoperability. *In Proceedings of the 3rd International IFIP Working Conference on Enterprise Interoperability (IWEI 2011)*. 177–190, Springer.
- Kiepuszewski A.H.M., Hofstede T., & Bussler. C., (2000), On Structured Workow Modelling. In B. Wangler and L. Bergman (ed), *Proceedings of the Twelfth International Conference on Advanced Information Systems Engineering (CAiSE'2000)*, LNCS 1789, Springer-Verlag, 431-445.
- Kim J. & al., (2004), An Intelligent Assistant for Interactive Workflow Composition. *In Proceedings of 2004 International Conference on Intelligent User Interfaces (IUI-2004)*, Madeira Islands, Portugal.
- Lazcano A., Alonso G., Schuldt H., & Schuler C., (2000), The wise approach to electronic commerce, *International Journal of Computer Systems Science & Engineering, special issue on Flexible Workflow Technology Driving the Networked Economy*, 15(5).
- Meng J., & Helal S., (2000) An ad-hoc workflow system architecture based on mobile agents and rule-based processing. *In: Proceedings of the 2000 international conference on artificial intelligence (ICAI2000)*, Las Vegas.
- Muller R. (2002), Event-oriented dynamic adaptation of workflows. *Ph.D.Thesis, Department of Computer Science, University of Leipzig*.
- Perrin O., & Godart C., (2004), A model to support collaborative work in virtual enterprises. *Data Knowledge*
- Van Der Aalst W., & Weske M., (2001), The P2P approach to inter-organizational workflows. *In proceedings of the 13th international conference on advanced information systems engineering*, Springer-Verlag, 140–156.
- Van Der Aalst W., Adams M., Hofstede A. ter, Pesic M., & H. Schonenberg, (2008), Flexibility as a Service. *BPMcenter.org, Tech. Rep. BPM-08-09*.
- Virdell M., (2003), Business Processes and Workflow in the Web Services World. *IBM, Developer Works.Engineering*, 50(1) :63–86.

Voorhoeve M., & Aalst W.M.P., (1997), Ad-hoc Workflow: Problems and Solutions. In R. Wagner (ed), *Database and Expert Systems Applications, 8th. International Workshop, DEXA'97 Proceedings*, 36–40. *IEEE Computer Society Press, Los Alamitos, California*.

Zhao X., Liu C., & Yang Y., (2004), Web service based architecture for workflow management systems, Book chapter, *Database and expert systems applications*, LNCS 3180, ISSN 0302-9743.

KEY TERMS & DEFINITIONS

WF process: is a set of activities consuming input data and producing outputs and linked between them with control flow operators.

WF process model: is an abstraction of the main aspects (functional, behavioral, interactional, organizational and/or informational) defining a WF process. But, we usually refer to functional and behavioral aspects.

WF sub-process: is part of a WF process composed of a set of activities implementing a sub-functionality, it requires input data and produces output ones.

Structured cooperation: is a form of cooperation where a process model is clearly defined at build time and all process instances follow the same model at runtime.

Ad-hoc cooperation: is appropriate for non-durable cooperation where process models are not completely defined at build time but can be completed in fly at runtime and process instances not necessarily follow the same model.

Planned cooperation: is a form of cooperation where partners are known a priori and agree for cooperation through a contract.

Cross-organizational WF: Inter-organizational WF which is a WF that implies several business partners.

WF composition: is a mechanism that allows a construction of new (usually more complex) WF processes using existing ones (less complex).

Service-based WF: is a WF where activities are implemented as services.