Interaction Protocols Based Approach for Reliable Web Services Composition

Mohamed Ali Bouanaka, Djamel Benmerzoug, Nacereddine Zarour
Department of Software Technologies and Information Systems
Faculty of New Technologies of Information and Communication
University Constantine 2, Algeria
bouanaka.mohamedali@yahoo.fr, benmerzoug@gmail.com, Nasro_zarour@umc.edu.dz

ABSTRACT: *This paper deals with one of the key issues in Cooperative Information Systems (CIS): the Business Interaction Protocols (IP) that are interconnecting the different parts involved in collaborative activities. The challenge here is twofold. First, we must provide a formal model that is rich enough to capture interactions characteristics. Second, we must allow designers to combine existing protocols to achieve a new specific need.*

The paper introduces a formal analysis framework allowing the verification of the conformance between IP. This framework is based on our previous work [7][8]. In this paper, we mainly focus on the compositions of IP, where particular protocols may then be selected and composed to support a desired business application.

Keywords: Web Protocols, Web Services, Business Interaction Protocols, Cooperative Information Systems

Received: 1 March 2013, Received 14 April 2013, Accepted 20 April 2013

© 2013 DLINE. All rights reserved

1. Introduction

Contemporary enterprises and their business processes are becoming more dynamic, distributed and complex. Thus, even a simple process (e.g., processing an order) may cause business transactions across boundaries of numerous business units (shipping department, delivery department, sale department, etc.) and trigger interactions of multiple actors and software applications [3].

In fact, enterprises are compelled to adopt new forms of organization that are mobile and reactive, and they are brought to focus on their skills and contract alliances in order to satisfy the needs of customers. These alliances must permit the constitution of teams with members coming from different enterprises to work together on a specific goal, so as to accelerate production, while improving the quality and reducing substantially the costs. This definition is one of the several possible definitions of these alliances, usually called Cooperative Information Systems (CIS). In this environment there is a clear need to coordinate multiple Web services into a multistep business transaction. This requires that several Web services attain transactional properties reflecting business semantics, which are to be treated as a single logical unit of work.

In such an environment, if there were no mechanisms to structure inter-enterprises interaction, the formed software components result in increasingly complex and potentially unreliable systems. This opens new challenges to be tackled. For example, common message formats must be agreed upon and expected interaction sequences must be clearly defined [16]. Also, legal consequences of message exchanges as well as business constraints must be captured [22].

To tackle these challenges, our approach motivates the use of *Business Interaction Protocols* (*IP*) for flexible inter-enterprise application collaboration. IP are a useful way for structuring communicative interaction among business partners, by organising messages into relevant contexts and providing a common guide to the all parts. This approach requires distilling from the structure of a business collaboration the key capabilities that must necessarily be present in a business transaction and specifying them accurately and independently of any specific implementation mechanisms. The IP then become the framework for expressing detailed operational business semantics.

In previous work [8][5], we described the use of interaction protocols to define and manage collaborative processes in B2B relationships where the autonomy of participants is preserved. We demonstrated the practicability of our approach by embedding it in a Web services language for specifying IP, which conducive to reuse, refinement and aggregation of our IP. We also elaborated translation rules from interaction protocols notations used in our approach into Colored Petri Nets (CPN). These rules are implemented in IP2CPN: the tool we developed to automatically generate Petri nets from protocols specifications. Resulting Petri nets can be analyzed by dedicated tools to detect errors as early as possible.

In this paper, we propose a basis for a theoretical approach for aggregating protocols to create a new desired business application. The proposed approach provides the underpinnings of aggregation abstractions for protocols. Our approach is a high-level abstraction that relates to real-world interaction protocols, and hence is easy for protocol designers to understand.

The remainder of the paper is organized as follows: Section 2 overviews our previous work. Based on that, in section 3, we give details about protocols composition. Section 4 focuses on the verification of composite protocol correctness. Section 5 overviews some related work and Section 6 provides some concluding remarks.

2. Our PreviousWork: an Overview

2.1 Introducing the Organic Computing Systems for CIS

Serious challenges like mergers and acquisitions, outsourcing possibilities, rapid growth, the need for regulatory compliance, and intense competitive pressures are overtaxing existing traditional business processes, slowing innovation and making it difficult for an enterprise to pursue and reach its business strategies and objectives. Such challenges require changes at the enterprise-level and thus lead to a continuous business process redesign and improvement effort [1].

Routine process changes usually lead to possible reorganization and realignment of many businesses processes and increase the propensity for error. To control process development one needs to know why a change was made, what are its implications and whether the change is complete. Eliminating spurious results and inconsistencies that may occur due to uncontrolled changes is therefore a necessary condition for the ability of processes to evolve gracefully, ensure stability and handle variability in their behavior. Such kind of changes must be applied in a controlled fashion so as to minimize inconsistencies and disruptions by guaranteeing seamless interoperation of business processes that may cross enterprise boundaries when they undergo changes [13].

Driven by the motivation of change management in the context of CIS, we have introduced the Organic Computing systems on enterprise level [21][10]. Organic Computing systems shall show so called self-x properties, i.e., they should be self-organizing, self-optimizing, self-healing, self-protecting, self-configuring, self-explaining and so on. Like in natural systems these properties shall become apparent on the level of the whole system through the properties and interactions of their components.

The self-x feature which is considered to be the most important for Organic Computing systems is self-organization, i.e., the adaptive and dynamic process that allows systems to establish and maintain structure and function without external control. Consequently, the proposed approach is based on the Observer/Controller architecture [24][11][23]. The observer monitors the state of the system and the properties of the different components and reports an aggregated quantified context (a description of the currently observed situation) to the controller. The controller evaluates this context with respect to a given objective function and takes appropriate control actions whenever it is necessary to influence the underlying system to meet the system goal.

The architecture of the proposed system is composed of the following parts:

• A group of information systems (composed of Web services). Each information system has a Local Controller that sends the list of the Web services to the global controller (through the observer).

• A global controller that receives a list of the available Web services of each information system. This list can be modified at any moment, when a Web service is created, deleted and so on.

• Control components, which are collaborative components, and their role is to make compositions between Web services.

• And finally an observer. When the system has a goal to reach, it tries to reach this goal as following: The global controller has the list of allWeb services of all information systems. This global controller starts to make a set of compositions between Web services. This task of compositions is divided between control components. To improve the system performers, Web services with similar characteristics are put into categories, to decrease the number of combinations of compositions (Web services of each category have similar functionalities). Compositions between two categories are allocated to a group of control components. After making compositions, a set of results is found. The global controller selects the result that fits more with the initial goal.

When a Web service leaves the system, this latter has to omit all compositions where this Web service is present. When a new Web service enters in the system, this latter has to create new compositions with the new Web service. When characteristics (some QoS for example) of aWeb service change, the system has to update compositions where this Web service takes place. In all these cases the system is auto-reconfigurable in order to adapt to the new changes, and it avoids doing all compositions after every modification of Web services, which is an operation that comes with costs. The decision about which control component has to execute which task, are made by the control components their self.

2.2 Putting Interaction Protocols at the Center of CIS

From an abstract point of view of systems modelling, i.e. once we abstract from the nature of the languages and platforms over which services are deployed, the main challenge raised by CIS Application is in the number of autonomic entities involved and the complexity of the interactions within them. That is, the complexity that matters is not so much in the "*size*" of the code through which such entities are programmed but on the number, intricacy and dynamicity of the interactions in which they will be involved.

This is why it is so important to put the notion of interaction at the center of research in CIS modelling. This is also why new methods and formal techniques become necessary. To tackle these challenges, our approach is based on a notion IP through which we specify complex services and break the complexity of running systems by recognising larger chunks that have a meaning in the application domain. This notion of IP, which is inspired by work of Multiagent systems (MAS), supports the modelling of composite services as entities whose business logic involves a number of interactions among more elementary service components.

Our approach [6] [8] motivates the use of IP based on AUML/BPEL4WS. where pre- and post-conditions, rules, guards are specified in OCL[1].

AUML (Agent UML) notation [2][4] is a UML profile dedicated to agents trying to simplify the transition from software engineering to multi-agent system engineering. In other hand, BPEL4WS [18] (Business Process Execution Language for Web Services) is a *de facto* standard for describing Web services composition. In our context, BPEL4WS was used as a specification language for expressing the interaction protocols of the multi-agent system.

For a better interactivity, communication between software components (Web service, private application) should be appropriately regulated. IP provide a formal ground for enabling this regulation. We demonstrated the practicability of our approach by embedding it in a Web services language for specifying business protocols, which conducive to reuse, refinement and aggregation of our business protocols.

2.3 Formalisation of Interaction Protocols

When protocols are employed in open environments, such as the internet, they must be executed by agents that behave more

[1]OCL: Object Constraint Language. (www.omg.org/cgi-bin)

or less autonomously and whose internal designs are not known. In such settings, therefore, there is a risk that the participating agents may fail to comply with the protocol [19]. Without a rigorous means to verify compliance, the very idea of protocols for interoperation is subverted.

The use of formal methods is important because ensuring the correctness of complex protocols is seldom possible via other design approaches. High-level Petri nets are a suitable formal method for the design of IP because of their ability to express concurrency, non-determinism and system concepts at different levels of abstraction.

For this reason, we use Colored Petri Net (CPN) as target notation. CPN allows analysis to check for properties in IP. The semantics of the IP notations used in our approach and its application are described on the basis of translation rules into CPN. Consequently, lifelines, messages, constraints, splitting/merging paths, interaction terminations, and other IP construction elements are translated into a Petri net. Afterwards, the resulting Petri net specification can be analysed by dedicated tools to detect errors as early as possible.

Definition 1. (Colored Petri Net [17])
A Colored Petri Net is a nine-tuple ($Plc, Tr, Ar, N, C, f_c, G, E, I$), where:

• Plc is a finite set of places.
• Tr is a finite set of transitions.
• Ar is a finite set of arcs. The sets of places, transitions, and arcs are pairwise disjoint, that is $Plc \cap Tr = Plc \cap Ar = Tr \cap Ar = \phi$.
• N is a node function. It is defined from Ar into $(Plc \times Tr) \cup (Tr \times Plc)$.
• C is a finite set of non-empty types, also called color sets.
• f_c is a color function. It is defined from Plc into C.
• G is a guard function. It is defined from Tr into expressions such that:

$$\forall t \in Tr : [Type\,(G\,(t)) = \text{Boolean} \wedge Type\,(Var\,(G\,(t))) \subseteq C].$$

• E is an arc expression function. It is defined from Ar into expressions such that: $\forall a \in Ar : [Type\,(E\,(a)) = C\,(p)\,MS \wedge Type(V\text{ar}\,(E(a))) \subseteq \text{C}]$,
where p is the place of $N\,(a)$ and $C\,(p)MS$ denotes the multi-set type over the base type $C\,(p)$.
• I is an initialization function. It is defined from Plc into closed expressions such that: $\forall p \in Plc : [Type\,(I\,(p)) = C\,(p)\,MS]$.

In previous work [7], we elaborated translation rules from interaction protocols notations used in our approach into Colored Petri nets. These rules are implemented in IP2CPN: the tool we developed to automatically generate the Petri net from protocols specification. The resulting Petri net specification can be analyzed by CPN-AMI tool [14] to detect errors as early as possible.

The CPN representation in our approach introduces the use of colors to represent additional information about business processes interaction states and communicative acts of the corresponding interaction. Colors sets are defined in the net declaration as follow: (the syntax follows standard CPN-notation used in [17]).

Color sets :
Communicative Act = **with** *FIPA ACL Communicative Acts*;
Role = *string* **with** *a . . . z*;
Content = *string* **with** *a . . . z*;
Bool = **with** *true* | *false*;
MSG = *record*
    s, *r* : *Role*;
    CA : *Communicative Act*;
    C : *Content*;
Variables : *msg*, *msg*1, *msg*2 : *MSG*;
    x : *Bool*;

The MSG colour set describes the communicative acts and is associated with the net message places. The *MSGs* colored token is a record < *s*, *r*, *ca*, *c* >, where the *s* and *r* elements determine the sender and the receiver of the corresponding message. These elements have the color setRole, which is used to identify business processes or/and Web services participating in the corresponding interaction. The *CommunicativeAct* and the Content color sets represent respectively the FIPA-ACL communicative acts and the content of the corresponding message. We note that the places without color set hold an indistinguishable token and therefore have the color domain token = {•}.

In our CPN representation, each role is considered equivalent to a type of resource, which is represented in a Petri net as a place. Hence, there will be one token in the place for each actor playing this role. As shown in figure 1, each one of these places is labelled $Start_{r_i}$ where r_i is the role name.

The life line of role is represented implicitly by a places and transitions sequence belonging to this role.The net is constituted therefore by one sub-net (Petri net process) for each role acting during the interaction and these nets are connected by places that correspond to the exchanged messages.

Driven by the motivation of reuse, would like to treat protocols as modular components, potentially composed into additional protocols, and applied in a variety of business processes. By maintaining repositories of commonly used, generic, and modular protocols, we can facilitate the reuse of a variety of well-defined, well-understood and validated protocols. For example, a payment protocol can be used in a process for purchasing goods as well as in a process for registering for classes at a university. Further, the repository would expand as newly composed protocols are inserted into it.

In the rest of this paper, we propose a basis for a theoretical approach for aggregating protocols to create a new desired business application. The proposed approach provides the underpinnings of aggregation abstractions for protocols.

3. Towards an Approach for Business Protocols Composition

In traditionally approaches, protocols are either not described formally or are described merely in terms of message order. As states by [15], such approaches give a basis for structuring communicative interaction among business partners, but do not address the fundamental knowledge engineering challenge of reuse and composition at the level of protocols.

Specifically, because protocols address different business goals, they often need to be composed to be put to good use. For example, a company that is interested in selling books could focus on this protocol while outsourcing other protocols such as payment and shipment.

The composition of two or more IP generates a new protocol providing both the original individual behavioral logic and a new collaborative behavior for carrying out a new composite task. This means that existing protocols are able to cooperate although the cooperation was not designed in advance.

Definition 2. (Composite Protocol) A Composite Protocol (*CP*) is a tuple $CP = (P, O_p, Input, Output, P_{init}, P_{fin})$
where:

- P is a non empty set of basic protocols,

- O_p is a non empty set of operators, $O_p \subseteq (P \times P) \cup (P \times CP) \cup (CP \times P) \cup (CP \times CP)$,

- *Input*,*Output* are a set of the elements required (produced) by the composite protocol *CP*,

- P_{init} is non empty set of *initial* protocols, $P_{init} \in P$ and

- P_{fin} is non empty set of *final* protocols, $P_{fin} \in P$.

Modelling protocols composition requires control structures involving loops, choice and parallelism. This will enable complex behaviour of IP, such as concurrent execution, or iteration while a certain condition holds. Consequently, an automated means of assembling complex protocols from atomic ones is essential.

In this section, we introduce a theoretical approach for structural protocols composition that allows the creation of new valueadded business protocols using existing ones as building blocks. Sequence, alternative, iteration, and arbitrary sequence are typical constructs specified in the control flow.

The proposed approach provides the underpinnings of aggregation abstractions for protocols. To achieve this goal we require an agreement between IP in the form of a shared contract.

3.1 Protocol Contracts

For a correct composition of protocols, they must come to an agreement or contract. A contract describes the details of a protocol (participants in the protocol, produced elements, required elements, constraints,...) in a way that meets the mutual understandings and expectations of two or more protocols. Introducing the contract notion gives us a mechanism that can be used to achieve a meaningful composition.

Definition 3. (Protocol Contract) Contract C, is a collection of elements that are common across two interaction protocols. It represents the mutually agreed upon protocol schema elements that are expected and offered by the two protocols.

1. Let us denote by P_k^{in}, P_k^{out} the set of elements required (produced) by the protocol P_k where $P_k^{in} = \{x_i, i \geq 1\}$ and $P_k^{out} = \{y_j, j \geq 1\}$

2. Let us define a function θ, called a contract-mapping, that maps a set of P_k^{out} elements (produced by the protocol P_k) anda set of P_r^{in} (consumed by the protocol P_r), $\theta = \exists y_i \in P_k^{out} \wedge \exists x_j P_r^{in} \mid (x_i, y_j)\}$, which means that the protocol P_r consumes the element y_j provided by the protocol P_k, and $C = (P_k^{out})_\theta (C) \cup (P_r^{in})_\theta (C)$.

3.2 Protocols Compositability relationship

The compositability relationship means that business protocols can be joined together to create a new protocol that performs more sophisticated applications. That composition can then serve as a protocol itself.

Definition 4. (Partial Compositability) Two protocols P_k and P_r meet the Partial Compositability relationship *iff* $\exists x_i \in (P_r^{in})_\theta (C)$, $\exists y_i \in (P_r^{out})_\theta\ (C) \mid (x_i, y_j)$, which means that the protocol P_r can be executed after the protocol P_k but it must wait until all its "*required elements*" will be provided by other protocols.

Definition 5. (Full Compositability) Two protocols P_k and P_r are called Full Compositability iff $\exists x_i \in (P_r^{in})$ (C), $\exists y_i \in (P_k^{out})\,(C) \mid (x_i, y_j)$, which means that the protocol P_r must be (immediately) executed after the protocol P_k because **all** its "*required elements*" are offered by the protocol P_k.

We note here that the partial (full) compositability relationship is not commutative. So, if a protocol P_k has a partial (full) compositability relationship with a protocol P_r, it doesn't means that P_r have a partial (full) compositability relationship with P_k.

Proposition. Let $P = \{P1, P2, \ldots, P_m\}$ a set of business protocols. The set P constitute a meaningful composition if: $\forall P_k \in (P - P_{init}$[2]$)$, $\forall x_i \in (P_k^{in})_\theta (C)$, $\exists y_j \in (P_r^{out})_\theta\ (C) \mid (x_i, y_j)$, where $r, k \in [1, m]$ and $r \neq k$.

This proposition states that, if we have a set of protocols $P = \{P_1, P_2, \ldots, P_m\}$ when all their "*required elements*" are offered (by other protocols), this means that all the protocols of P can be executed.

Justification. By definition 4, a protocol P_k can be executed iff all its "*required elements*" are offered. So, if we have a set of protocols $P = \{P_1, P_2, \ldots, P_m\}$ when all their "*required elements*" are offered (by other protocols), this means that all the protocols of P can be executed.

3.3 Protocols Composition Operators

We describe below the syntax and semantics of the protocols composition operators. The operators were chosen to allow common and advanced protocols combinations. The set of new protocols can be defined by the following grammar in BNF like notation:

$$P ::= P \rightarrow P \mid P \diamond P \mid P \blacklozenge P \mid P \xrightarrow{e} P \mid P \Lsh$$

[2] *$P_{init} \in P$ and represent the initial protocols, which their "required elements" are provided by external events*

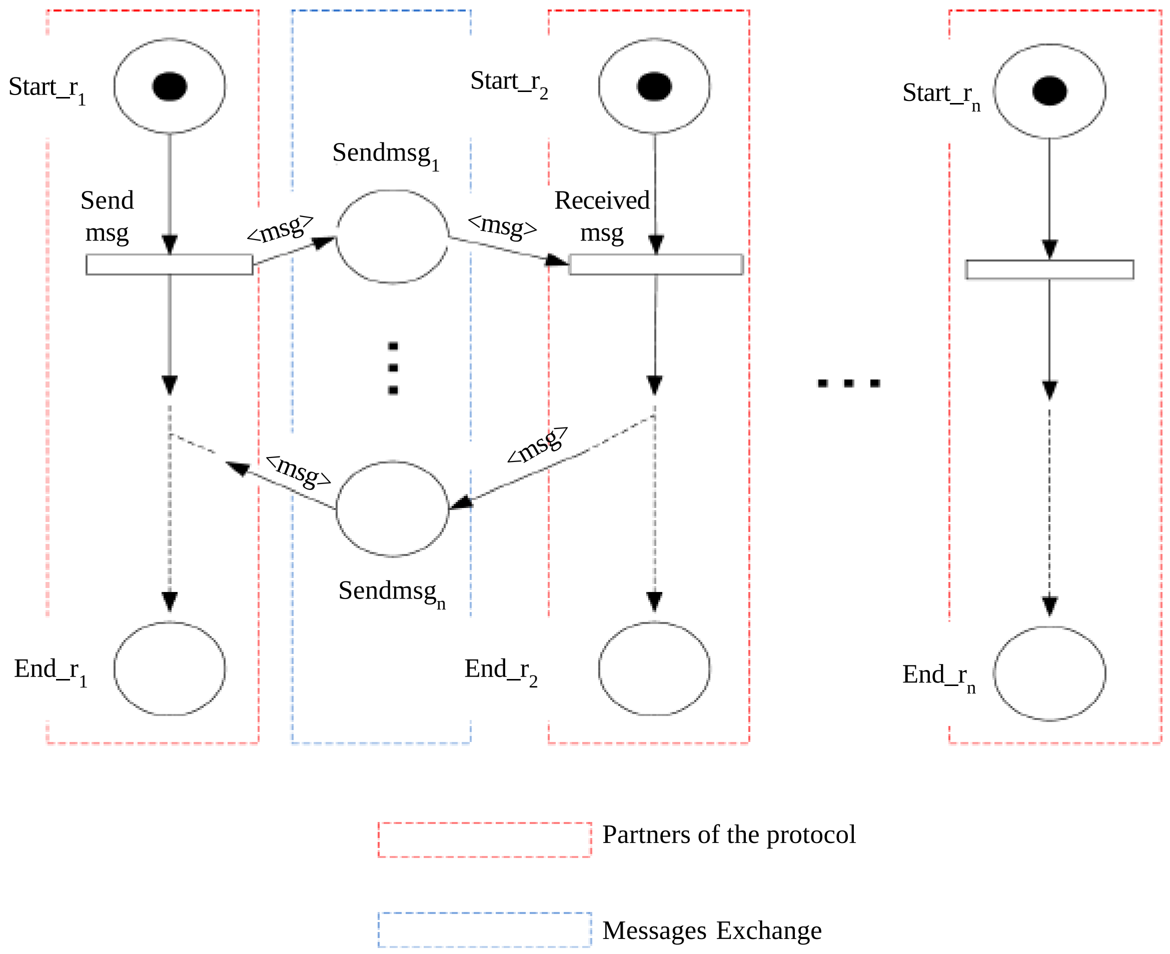

Figure 1. An Example of Interaction Protocol

Where:

• $P_1 \rightarrow P_2$ represents a composite protocol that performs the protocol P_1 followed by the protocol P_1, i.e., $\rightarrow$ is an operator of Sequence.

This is typically the case when a protocol depends on the output of the previous protocol. For example, the protocol Payment is executed after the completion of the protocol Delivery.

Formally, $P_1 \rightarrow P_2$ is represented by the Petri net shown in Figure 2-a. In this case, the two protocols are connected by a transition and n arcs, where n is a number of the participants in the protocols P_1 and P_2.

In fact, the Petri net representation of the composite protocol $P_1 \rightarrow P_2$ is CPN as defined in Definition 2, where:

– $Plc = Plc_{p1} \cup Plc_{p2}$,

– $Tr = Tr_{p1} \cup Tr_{p2} \cup \{t\}$

– $Ar = Ar_{p1} \cup Ar_{p2} \cup \{(FinalPlc_{p1}, t), (t, StartPlc_{p2})\}$, where $FinalPlc_{p1}$ is a set of a final places of the protocol$_{p1}$ and $StartPlc_{p2}$ is a set of a start places of the protocol$_{p2}$.

• $P_1 \diamond P_2$ represents a composite protocol that behaves as either protocol P_1 or protocol P_2. i.e., $\diamond$ is an Alternative (or a Choice)

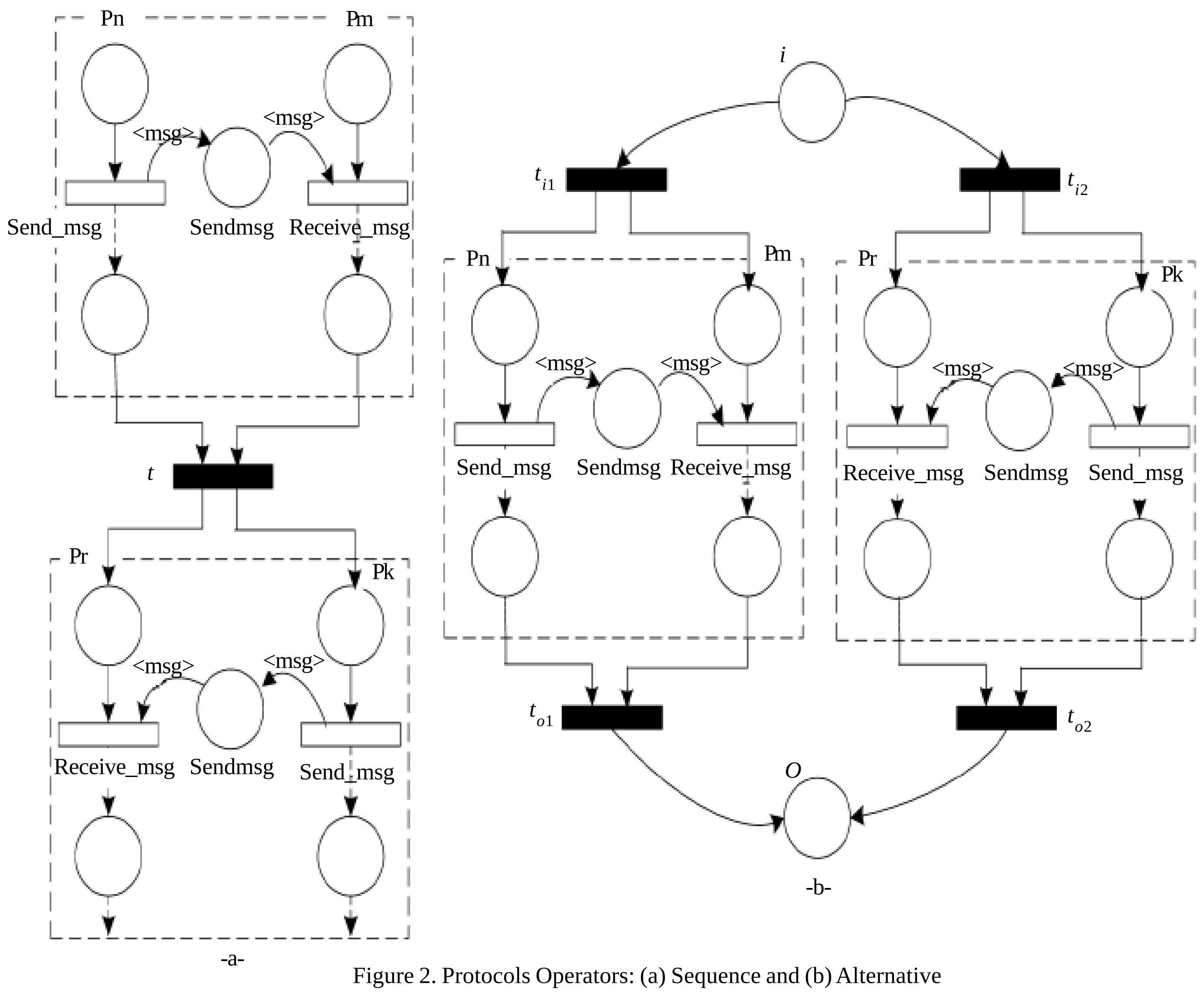

Figure 2. Protocols Operators: (a) Sequence and (b) Alternative

operator.

Figure 2-b shows a Petri net representation of the alternative operator, so that only one protocol can be executed. In this case, each basic protocol is associated to a transition.

The Petri net representation of the composite protocol $P_1 \diamond P_2$ is CPN, where:

– $Plc = Plc_{p1} \cup Plc_{p2} \cup \{i, o\}$,

– $Tr = Tr_{p1} \cup Tr_{p2} \cup \{t_{i1}, t_{i2}, t_{o1}, t_{o2}\}$

– $Ar = Ar_{p1} \cup Ar_{p2} \cup \{(i, t_{i1}), (i, t_{i2}), (t_{i1}, StartPlc_{p1})\}$

$\cup \{(t_{i2}, StartPlc_{p2}), (t_{o1}, \mathrm{o}), (t_{o2}, \mathrm{o}), (FinalPlc_{p1}, t_{o1})\}$
$\cup \{(FinalPlc_{p2}, t_{o2})\}$.

• $P_1 \blacklozenge P_2$ represents a composite protocol that performs the protocol P_1 and P_2 independently from each other. i.e., $\blacklozenge$ is a Parallel operator.

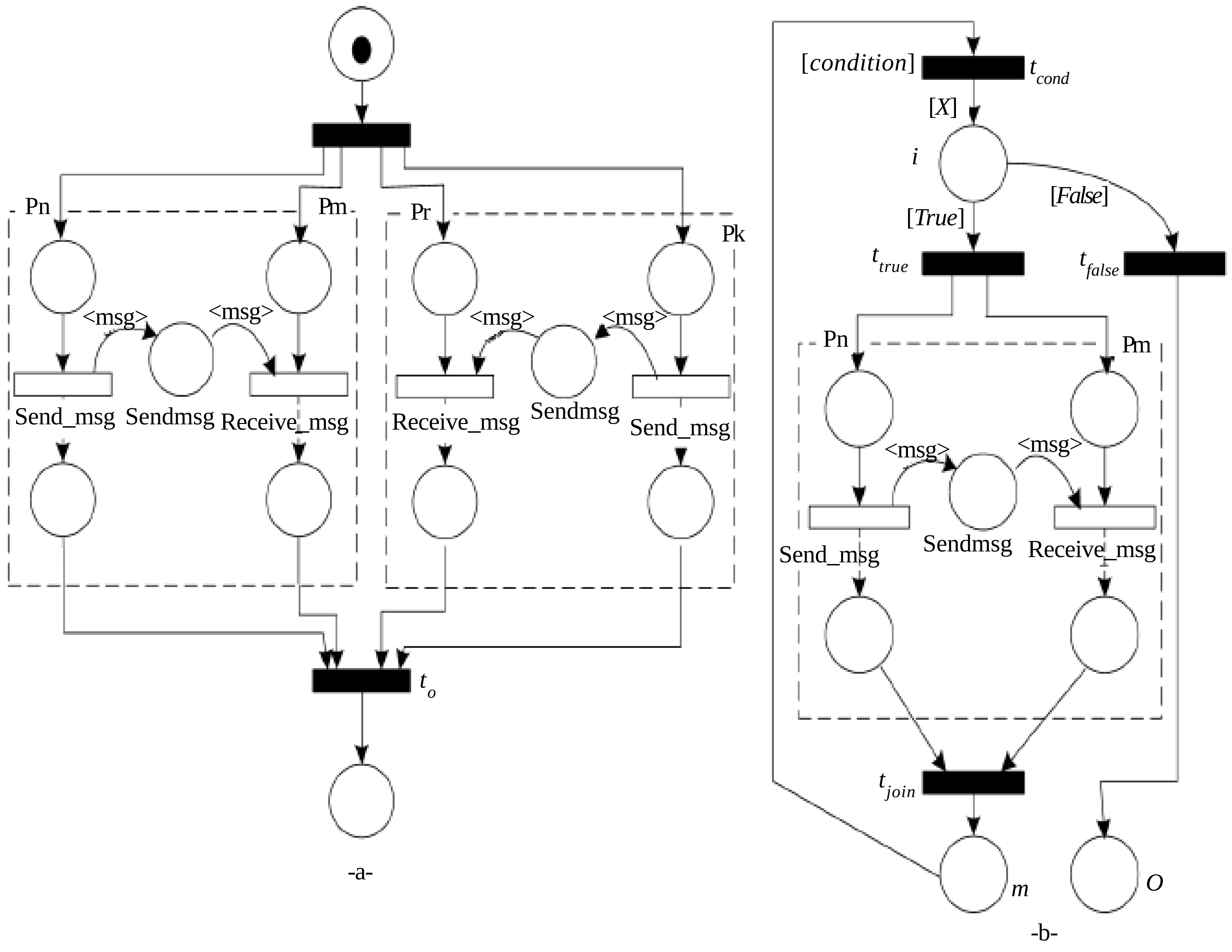

Figure 3. Protocols Operators: (a) Parallelism and (b) Iteration

This operator which models concurrency protocols execution is represented by means of parallel case or multi-threading in Petri net (see figure 3-a). In fact, we add one transition and *n* arcs, where *n* is the number of the participants in the protocols P_1 and P_2.

The Petri net representation of the composite protocol $P_1 \blacklozenge P_2$, is CPN, where:

– $Plc = Plc_{p1} \cup Plc_{p2} \cup \{i, o\}$,

– $Tr = Tr_{p1} \cup Tr_{p2} \cup \{t_i, t_o\}$

– $Ar = Ar_{p1} \cup Ar_{p2} \cup \{(i, t_i), (t_i, StartPlc_{p1})\} \cup \{(t_i, StartPlc_{p2}), (t_o, \text{o}), FinalPlc_{p1}\} \cup$
$\{(t_o), (FinalPlc_{p2}, t_o)\}$.

• $P^{\Lsh}$ represents the execution of a protocol followed a certain number of times by itself. i.e., $\Lsh$ is an Iteration operator.

In Petri net (see figure 3-b), an iteration has a transitions that is connected to all the participants places in the protocol *P*. So we add a transition and an arc from the end place to this transition and *n* arcs from this transition to the beginning participants places in the protocol *P*.

Consequently, the Petri net representation of the composite protocol $P^{\Lsh}$ is CPN as defined in Definition 2, where:

– $Plc = Plc_p \cup \{i, m, o\}$,

– $Tr = Tr_p \cup \{t_{cond}, t_{true}, t_{false}, t_{join}\}$

– $Ar = Ar_p \cup \{(t_{cond}, i), (i, t_{true}), (i, t_{false}), (t_{true}\} \cup \{StartPlc_p), (FinalPlc_p, t_{join})\} \cup \{(t_{join}, m), (m, t_{true}), (t_{false}, o)\}$.

• $P_1 \xrightarrow{e} P_2$ represents a composite protocol that behaves as P_1 except for an event e in P_1, P_2 can be executed, then P_1 can resumes its execution. i.e., $\xrightarrow{e}$ is a Refinement operator.

The refinement operator, in which protocols are replaced by more detailed non empty protocols. Refinement is the transformation of a design from a high level abstract form to a lower level more concrete form hence allowing hierarchical modeling.

4. Verification of Composite Protocol Correctness

Business protocols interact with each other for conducting a specific task although they are created. Since IP which contain errors may cause inconsistance in the system, it is thus important to analyze the composite protocols before they are put into operation.

The transformation into Petri net is performed using our tool IP2CPN. The formal verification is performed using the CPN-AMI Petri net modeling and model checking environment [14].

CPN allow us to validate and evaluate the usability of a system by performing automatic and/or guided executions. These simulation techniques can also carry out performance analysis by calculating transaction throughputs, etc. Moreover, by applying the CPNAMI tool it is possible to verify static and dynamic properties in order to provide the complement to the simulation. Some of these properties are that:

• There are no activities in the system that cannot be realized (dead transitions). If initially dead transitions exist, then the system was bad designed. - The IP specification exhibits the liveness property (e.g., the output CPN guarantees the existence of an initial state such that for any accessible state, at least one operation is executed).

• It is always possible to return to a previous state (home properties). For instance, to compare the results of applying different decisions from the same state. (the case of XOR and OR decision)

• The system may stop before completion (deadlock). Thus, a work might never be finished, or it might be necessary to allocate more resources to perform it.

• Certain tokens are never destroyed (conservation). Hence, resources are maintained in the system.

5. Related Work

With the growing trend of service oriented computing, composition of Web services has received much interest to support CIS. As stated in [20] and [25], current efforts in Web services composition can be generally grouped into three categories: manual, automatic, and semi-automatic composition.

By manual composition, we mean that the composite service is designed by a human designer (i.e., service provider) and the whole service composition takes place during the design time. This approach works fine as long as the service environment, business partners, and component services do not or rarely change. On the other hand, automatic service composition approaches typically exploit the Semantic Web and artificial intelligence planning techniques. By giving a set of component services and a specified requirement (e.g., user's request), a composite service specification can be generated automatically [9].

However, realizing a fully automatic service composition is still very difficult and presents several open issues [20], [9], [12]. The basic weakness of most research efforts proposed so far is that Web services do not share a full understanding of their semantics, which largely affects the automatic selection of services.

There exist some research efforts that encourage manual and automatic compositions. Instead of coupling component services tightly in the service model, such approaches feature a high-level abstraction (e.g., UML activity model, protocol specifications, and interface) of the process models at the design time, while the concrete composite services are either generated automatically using tools or decided dynamically at run time (e.g., BPEL4WS [18]).

Our proposition is similar to these approaches in the sense that we also adopt a semi-automatic approach for application integration. The collaboration scenario is specified as interaction protocols not high-level goals and the component applications are selected, at run time, based on the protocol specification specified at the design time.

Also, the Web services related standards for services composition and interoperability, such as the BPEL4WS [18] are lower level abstractions than ours since they specify flows in terms of message sequences. Also, they mix interaction activities and business logic making them unsuitable for reuse. In contrast to our approach, the BPEL4WS elements are only used to specify messages exchanges between the different business partners. Afterwards, this specification is used by agents to enact the integration of business processes at run time. Agents have the capability to dynamically form social structures through which they share commitments to the common goal. The individual agents, through their coordinated interactions achieve globally coherent behavior; they act as a collective entity known as a multiagent system. In our previous work [6] [5], we have explored the relationship between Web services, multiagent systems and enterprise application integration.

6. Conclusion and Future Work

In this paper, we presented a formal framework that allows for the verification of the conformance between interaction protocols. The presented framework is based on our previous work [7][8] that provides a formal model for the composition of local participants implementations.

The key feature of this framework is the ability to model and formally verify composition of business interaction protocols, where particular protocols may then be selected and composed to support a new business task. The proposed framework is based on Petri net for composing protocols. The formal semantics of the composition operators is expressed in terms of Petri nets by providing a direct mapping from each operator to a Petri net construction. In addition, the use of a formal model allows the verification of properties and the detection of inconsistencies both within and between protocols.

There are other issues in B2B e-commerce which, we believe, could be successfully addressed by extending the framework presented in this paper, e.g., to include management of time and resources. We expect that these problems can be dealt with using a suitable high-level Petri net, such as timed Petri nets.

References

[1] Vasilios Andrikopoulos, Salima Benbernou, Michael P. Papazoglou. (2012). On the evolution of services. *IEEE Trans. Software Eng.*, 38 (3) 609–628.

[2] Bauer, B., Bergenti, F., Massonet, P., Odell, J. (2001). Agents and the UML: A Unified Notation for Agents and Multiagent

Systems. *In*: Agent-Oriented Software Engineering II, Second International Workshop, AOSE'01, V. 2222 of LNCS, p. 148–150. Springer.

[3] Joseph Barjis, Ashish Gupta, Ramesh Sharda. (2011). Knowledge work and communication challenges in networked enterprises. *Information Systems Frontiers*, 13 (5) 615–619.

[4] Bernhard Bauer, James Odell. (2005). UML 2.0 and agents: how to build agent-based systems with the new UML standard. *International Journal of Eng. Appl. of AI*, 18 (2) 141 – 157.

[5] Djamel Benmerzoug. (2013). Agent approach in support of enterprise application integration. *International Journal of Computer Science and Telecommunications*, 4 (1) 47–53.

[6] Djamel Benmerzoug, Mahmoud Boufaida, Fabrice Kordon. (2007). A Specification and Validation Approach for Business Process Integration based onWeb Services and Agents. *In*: Proceedings of the 5th International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems, MSVVEIS-2007, In conjunction with ICEIS, p. 163–168. NSTIIC press.

[7] Djamel Benmerzoug, Fabrice Kordon, Mahmoud Boufaida. (2008). A Petri-Net based Formalisation of Interaction Protocols applied to Business Process Integration. *In*: Advances in Enterprise Engineering I, 4th International Workshop on Enterprise & Organizational Modeling and Simulation (EOMAS'08), V. 10 of LNBIP, p. 78–92, Montpellier, France, June. Springer.

[8] Djamel Benmerzoug, Fabrice Kordon, Mahmoud Boufaida. (2008). Formalisation and Verification of Interaction Protocols for Business Process Integration: a Petri net Approach. *International Journal of Simulation and Process Modelling*, 4 (3–4) 195–204.

[9] Daniela Berardi, Giuseppe De Giacomo, Massimo Mecella, Diego Calvanese. (2005). Automatic composition of process-based web services: a challenge. *In*: Proc. 14th Int. World Wide Web Conf. (WWW '05).

[10] Mohamed Ali Bouanaka, Naceredine Zarour. (2013). An approach for an optimized web service selection based on skyline. *International Journal of Computer Science Issues*, 10 (1) 412–418.

[11] Jürgen Branke, Moez Mnif, Christian Müller-Schloer, Holger Prothmann, Urban Richter, Fabian Rochner, Hartmut Schmeck . (2006). Organic computing - addressing complexity by controlled self-organization. *In*: ISoLA, p. 185–191.

[12] Jeppe Bronsted, Klaus Marius Hansen, Mads Ingstrup. (2010). Service composition issues in pervasive computing. *IEEE Pervasive Computing*, 9 (1) 62–70.

[13] Denada Cfarku, Yehia Taher, Rafiqul Haque, Willem-Jan van den Heuvel, Michael P. Papazoglou. (2012). Paean - a risk-mitigation framework for business transaction at run-time. *In*: EC-Web, p. 25–37.

[14] CPN-AMI:. http://move.lip6.fr/software/cpnami/.

[15] Nirmit Desai, Munindar P. Singh. (2007). A modular action description language for protocol composition. *In*: Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, p. 962–967. AAAI Press.

[16] Leymann K. Pfitzner M. Weske G. Decker, Kopp, O. (2008). Modelling service choreographies using BPMN and BPEL4Chor. In Z. Bellahsene, Leonard, M, editors, CAiSE, 5074, p. 79–93. LNCS, Springer, Heidelberg.

[17] Girault, C., Valk, R. Petri Nets for Systems Engineering, A Guide to Modeling, Verification, and Applications. Springer Verlag, July.

[18] SAP Siebel Systems IBM, Microsoft. (2003). Business process execution language for web services version 1.1. Technical report.

[19] Venkatraman, M., Munindar, P. S. (1999). Verifying compliance with commitment protocols. *Int. Journal of Autonomous Agents and Multi-Agent Systems*, 2 (3) 217 – 236.

[20] Nikola Milanovic, Miroslaw Malek. (2004). Current solutions for web service composition. *IEEE Internet Computing*, 8 (6) 51–59.

[21] Bouanaka mohamed Ali, Nacereddine Zarour. (2012). Web service-based emergence control in cooperative information system. *In*: International Conference on Information Technology and e-Services (ICITeS), p. 24–29, Sousse-Tunisia. IEEE.

[22] Papazoglou, M. P., Kratz, B. (2007). Web services technology in support of business transactions. *Int. Journal of Service Oriented Computing*, 1 (1) 51 – 63.

[23] Urban Richter, Moez Mnif, Jürgen Branke, Christian Müller-Schloer, Hartmut Schmeck. (2006). Towards a generic observer/controller architecture for organic computing. *In*: GI Jahrestagung, (1), 112–119.

[24] Thorsten Schöler, Christian Müller-Schloer. (2005). An observer/controller architecture for adaptive reconfigurable stacks. *In*: ARCS, p. 139–153.

[25] Quan Z. Sheng, Boualem Benatallah, Zakaria Maamar, Anne H. H. Ngu. (2009). Configurable composition and adaptive provisioning of web services. *IEEE Transactions on Services Computing*, 2 (1) 34–49.

Authors Bibliography

Mohamed Ali Bouanaka is a PhD student at the department of TLSI (Software Technologies and Information Systems), University Constantine 2, Algeria. He is a member of the LIRE laboratory, and his research interests include, Information systems and Web services.

Djamel Benmerzoug received his Ph.D. in computer science from Pierre and Marie Curie University (Paris - France) and UMC University (Constantine - Algeria), respectively. Dr. Djamel Benmerzoug is an Associate Professor in the department of TLSI (Software Technologies and Information Systems) at University Constantine 2, Algeria. His current research interests include Cloud computing, multiagent systems, service oriented computing, and business processes modelling and verification.

Nacereddine Zarour is a Full Professor at the department of TLSI (Software Technologies and Information Systems) of the University of Constantine 2, Algeria. His current research activities are about advanced information systems, requirements engineering, multiagent systems, and semantic Web.