DOI 10.12694/scpe.v14i2.843
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 14, Number 2, pp. 95–109. http://www.scpe.org 

# AN AGENT-BASED APPROACH FOR HYBRID MULTI-CLOUD APPLICATIONS

DJAMEL BENMERZOUG*

**Abstract.** Cloud service offerings provide a competitive advantages to enterprises through flexible and scalable access to computing resources. With the recent advances in Cloud computing, the need is emerging for interoperability between Cloud services so that a complex and developed business applications on Clouds are interoperable. In fact, combining different independent Cloud services necessitates a uniform description format that facilitates the design, customization, and composition. In this context, Agent Interaction Protocols (IP) are a useful way for structuring communicative interaction among business partners, by organizing messages into relevant contexts and providing a common guide to the all parts. The challenge here is twofold. First, we must propose 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 work presented in this paper is considered as a first step toward Agent Interaction Protocols as a Service. In fact, 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. Also, it proposes a set of operators that allows the creation of new value-added protocols using existing ones as building blocks.

**Key words:** Agent Interaction Protocols, Cloud Computing, Protocols Composition, Petri Net, Verification.

**1. Introduction.** Cloud computing is a model for enabling convenient, on demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with a minimal management effort or service provider interaction [34]. According to the intended access methods and availability of Cloud computing environments, four major types of Cloud deployments are known: public Clouds, private Clouds, community Clouds, and hybrid Clouds [31].

The increasing adoption rate of Cloud computing is currently driving developers, integrators and hosting enterprises to take Cloud computing into account. As a result, enterprises need to revise their current assets as Cloud computing is becoming a strategic asset.

However, enterprises are driven by different reasons to maintain their own data center, such as legislation of storing data in-house, investments in the current infrastructure, or the extra latency and performance requirements. This drive is supported by the fact that enterprises have already invested heavily in their own private server equipment and software [21].

Consequently, we believe that a hybrid approach makes more sense for enterprises. This approach allows one to use the internal infrastructure combined with public Cloud resources to build a hybrid Cloud. For example, on one hand, critical applications can run on the infrastructure/platform of the private data center, and, on the other hand, the public Cloud can be used as a solution to manage peak demands or for disaster recovery.

While the different Cloud solutions offer support for applications development and deploying, there are different issues that require special attention through specialized developments, like resource management, service integration, or service orchestration [16].

Combining different independent Cloud services is the ability to integrate multiple services into higher-level applications. This integration necessitates a uniform description format that facilitates the design, customization, and composition. In this context, Agent Interaction Protocols (IP) are a useful way for structuring communicative interaction among business partners, by organizing messages into relevant contexts and providing a common guide to the all parts.

In previous work [7] [8], 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 business protocols, which conducive to reuse, refinement and aggregation of our business protocols. 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.

*Department of Software Technologies and Information Systems, Faculty of New Technologies of Information and Communication, University Constantine 2, Algeria. (benmerzoug@gmail.com).

Cloud services composition and orchestration have seen increased attention in published works, and researchers have introduced numerous approaches to automate it. As such, current techniques suggest that enterprises will acquire related tools and perform integration activities locally. The knowledge and expertise to perform composition activities is hard to attain and equally difficult to maintain. Thus, the knowledge obtained during Cloud services composition is not stored, reused, or shared. To be competitive, enterprises must be able to transfer and reuse knowledge attained after each composition scenario.

Driven by the motivation of reuse, we 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 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 provides a set of operators that allows the creation of new value-added protocols using existing ones as building blocks. Sequence, alternative, iteration, and parallel are typical constructs specified in the control flow. We also give a formal semantics to the proposed operators in terms of Petri nets.

The remainder of the paper is organized as follows: Section 2 overviews our previous work. Based on that, in Section 3, we define the concept of the Cloud business protocols. Section 4 gives details about protocols composition including protocols contract and protocols operators. Section 5 focuses on the verification of composite protocol correctness. Finally, Section 6 overviews some related work and Section 7 provides some concluding remarks.

**2. Interaction Protocols in support of Service-Based Enterprise Application.** 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 service-based enterprise application is in the number of autonomic entities involved and the complexity of the interactions within them [37]. 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 service-based enterprise application modelling. This is also why new methods and formal techniques become necessary.

To tackle these challenges, our approach is based on the notion of 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.

Figure 2.1 shows our conceptual model for the treatment of service-based enterprise application. Business partners participating in this scenario publish and implement a public process. So, a public business process is the aggregation of the private processes and/or Web services participating in it.

Let us notice that private processes are considered as the set of processes of the enterprise itself and they are managed in an autonomous way to serve local needs. The public processes span organizational boundaries. They belong to the enterprises involved in a B2B relationship and have to be agreed and jointly managed by the partners.

Interaction protocols have been used in the area of multi-agent systems to represent interactions among agents. In the context of B2B relationships, a business protocol is a specification of the allowed interactions between two or more participant business partners. These interactions represent public business processes that enterprises agreed on the collaboration.

In our case the public process is specified by interaction protocols, which are used by the integrator agent to enact the public process at run time. Interaction protocols should be published, shared and reused among the business partners.

**2.1. Specification of Interaction Protocols.** The B2B integration scenarios typically involve distributed business processes that are autonomous to some degree, hence the importance of Interaction protocols

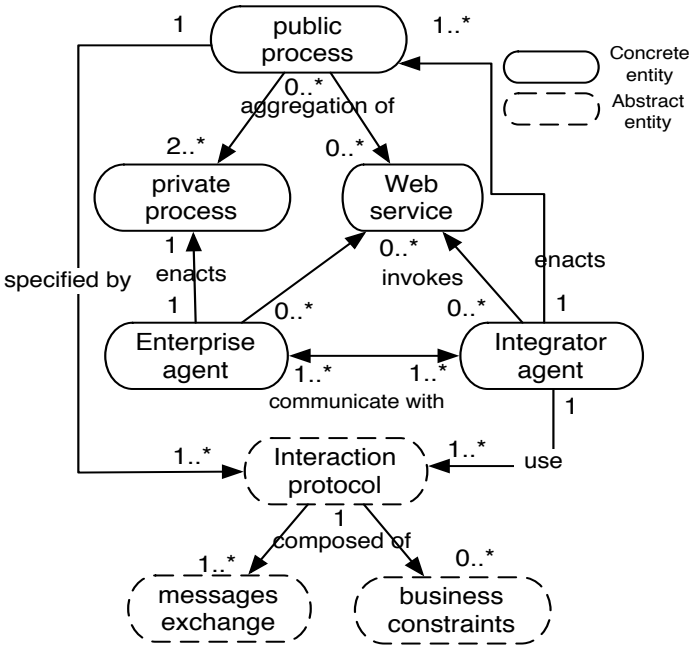


FIG. 2.1. *Conceptual Model: Enterprise Application Based on Interaction Protocols*

based modelling. They are a useful way for structuring communicative interaction among business partners, by organizing messages into relevant contexts and providing a common guide to the all parts. Formally an interaction protocol is defined as follow:

***Definition 1***. An Interaction Protocol is a quadruplet:

IP = <ID, R, M, $f_M$ >, where:

- $ID$ is the identifier of the interaction protocol
- $R = r_1, r_2, \ldots, r_n$ $(n > 1)$ is a set of Roles (private business process or Web Services)
- $M$ is a set of non-empty primitive (or/and) complex messages, where:
  - A Primitive Message ($PM$) corresponds to the simple message, it is defined as follow:
    $PM$ = <Sender, Receiver, CA, Option>, where:
    - Sender, Receiver ∈ R
    - CA ∈ FIPA ACL Communicative Act[1] (such as: cfp, inform, ...)
    - Option: contain additional information (Synchronous / Asynchronous message, constraints on message,...)
  - A Complex Message ($CM$) is built from simpler (primitive) ones by means of operators:
    $CM = PM_1 op PM_2 \ldots op PM_m$, where:
    - $m > 1$, op ∈ {XOR, OR, AND}, and
    - ∀ i ∈ [1,m-1], $PM_i.Sender = PM_{i+1}.Sender$, $PM_i.Sender \in R$.
- $f_M$: a flow relation defined as : $f_M \subseteq$ (RxR), where (RxR) is a Cartesian product $(r_1, r_2) \in$ (RxR), for $r_1, r_2 \in$ R

Developing effective protocols to be executed by autonomous partners is challenging. Similar to protocols in traditional systems, IP in open and Web-based settings need to be specified rigorously so that business partners can interact successfully.

For this reason, we have developed a method for IP design and verification. The proposed method (see figure 2.2) uses different models and languages.

[1]FIPA: Foundation for Intelligent Physical Agents [36].

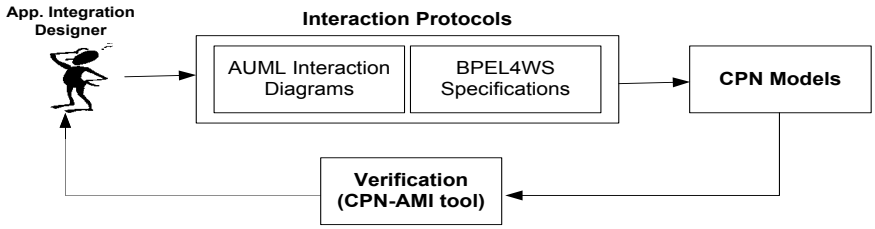


FIG. 2.2. *The proposed method*

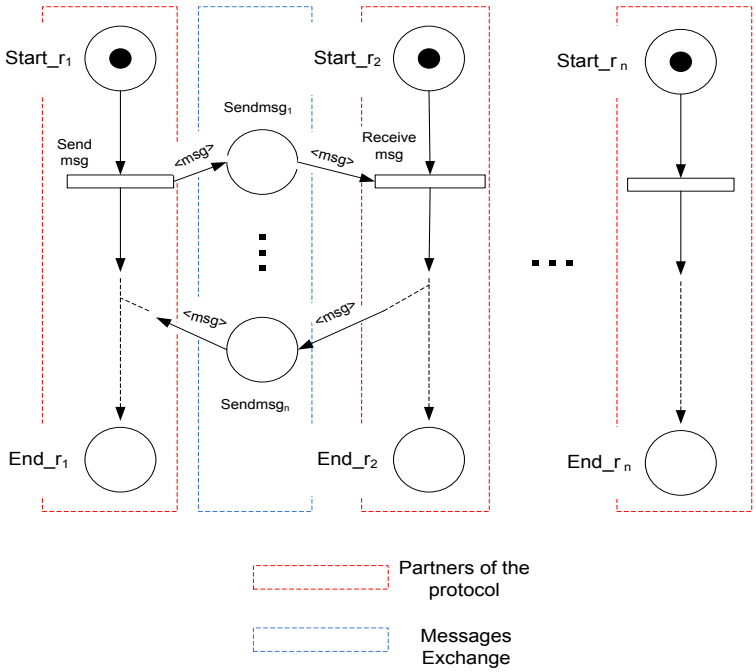


FIG. 2.3. *An Example of Interaction Protocol*

Our method motivates the use of IP based on AUML/BPEL4WS, where pre- and post-conditions, rules, guards are specified in OCL[2].

AUML (Agent UML) notation [1] [2] is a UML profile dedicated to agents trying to simplify the transition from software engineering to multi-agent system engineering. In other hand, BPEL4WS [22] (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 [6].

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 [17] 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 [20]).

[2]OCL: Object Constraint Language [28].

**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;

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 set $Role$, 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 = $\{\bullet\}$.

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 2.3, 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 place 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.

**3. Cloud Computing and Enterprises Application Integration.** Cloud computing infrastructures can allow enterprises to achieve more efficient use of their IT hardware and software investments. They do this by breaking down the physical barriers inherent in isolated systems, and automating the management of the group of systems as a single entity. Cloud computing is an example of an ultimately virtualized system, and a natural evolution for data centers that employ automated systems management, workload balancing, and virtualization technologies.

**3.1. Cloud Computing Models.** According to the intended access methods and availability of Cloud computing environments, there are different models of deployment [31]. They include private Cloud, public Cloud, community Cloud, and hybrid Cloud, which are briefly analyzed below (see figure 3.1).

**private Cloud:** In this model, the Cloud infrastructure is exclusively used by a specific organization. The Cloud may be local or remote, and managed by the enterprise itself or by a third party. There are policies for accessing Cloud services. The techniques employed to enforce such private model may be implemented by means of network management, service provider configuration, authorization and authentication technologies or a combination of these.

**public Cloud:** Infrastructure is made available to the public at large and can be accessed by any user that knows the service location.

**community Cloud:** Several organizations may share the Cloud services. These services are supported by a specific community with similar interests such as mission, security requirements and policies, or considerations about flexibility. A Cloud environment operating according to this model may exist locally or remotely and is normally managed by a commission that represents the community or by a third party.

**hybrid Cloud:** Involves the composition of two or more Clouds. These can be private, community or public Clouds which are linked by a proprietary technology that provides portability of data and applications among the composing Clouds.

According to the Forrester Research market [23], many businesses want interoperability between their internal infrastructure combined with public Cloud resources. They might want to use private application to process data in the Cloud, they might want to use a Cloud-based application to process private data, or they

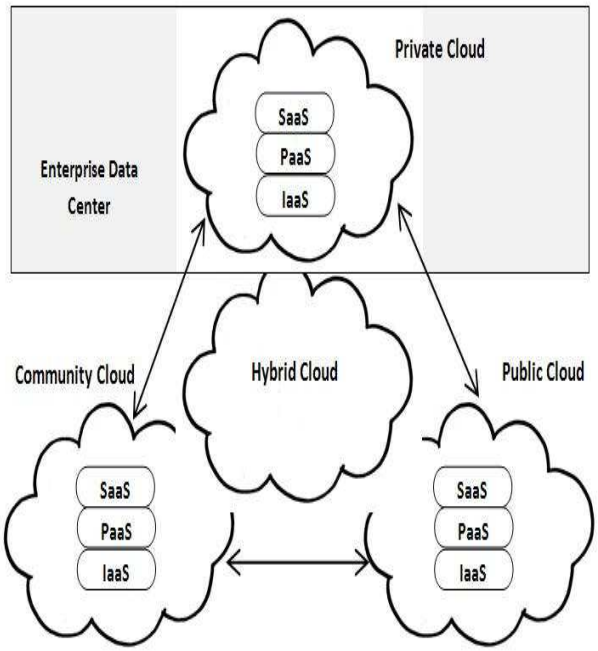


FIG. 3.1. *Cloud Computing Models*

might want to use applications or tools that will run both private and on the public Cloud. Consequently, we believe that a hybrid approach makes more sense for enterprises. In such approach, there is a need for complex developed business applications on the Clouds to be interoperable. Cloud adoption will be hampered if there is not a good way of integrating data and applications across Clouds.

In the next section, we introduce the Cloud Business Protocols, which are a useful way for structuring interaction among Cloud resources, by organising activities into relevant contexts and providing a common guide to the all parts.

**3.2. The Cloud Business Protocol.** As stated in [27], to ensure a meaningful composition of two or more Cloud services, there is a clear need for placing emphasis on how to develop enhanced composite service offerings at the application-level and assign or reassign different virtual and physical resources dynamically and elastically. In fact, combining different independent Cloud services necessitates a uniform description format that facilitates the design, customization, and composition.

Business protocol is a specification of the allowed interactions between two or more participant business partners. Applied to Cloud computing, we propose the following variation: *A Cloud Business Protocol is two or more business parties linked by the provision of Cloud services and related information.*

In fact, business parties in the Cloud computing area are interconnected by the Cloud business protocol. These parties are involved in the end-to-end provision of products and services from the Cloud service provider for end Cloud customers. Because protocols address different business goals, they often need to be composed to be put to good use. For example, a process for purchasing goods may involve protocols for ordering, shipping, and paying for goods.

Driven by the motivation of reuse, we 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.

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.

**4. 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 [19], 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, an enterprise 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 business protocols 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 $CP = (P, Op, Input, Output, P_{init}, P_{fin})$ where:

- $P$ is a non empty set of *basic protocols*,
- $Op$ is a non empty set of *operators*, $Op \subseteq (P \times P) \cup (P \times CP) \cup (CP \times P) \cup (CP \times CP)$,
- $Input, Output$ 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 business protocols, 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 value-added 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 business protocols in the form of a shared contract.

**4.1. Protocol Contract.** 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)***. A contract C, is a collection of elements that are common across two 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 $\theta$, called a *contract-mapping*, that maps a set of $P_k^{out}$ elements (produced by the protocol $P_k$) and a set of $P_r^{in}$ (consumed by the protocol $P_r$), $\theta = \{\exists\, y_i \in P_k^{out} \wedge \exists\, x_j \in P_r^{in} \mid (x_i, y_j)\}$, which means that the protocol $P_r$ consumes the element $y_j$ provided by the protocol $P_k$, and $C = (P_k^{out})_\theta(C) \cup (P_r^{in})_\theta(C)$.

**4.2. 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. Figure 4.1 shows an example of business protocols composition where the composite protocol has six sub-protocols (called $P_1, P_2, \ldots, P_6$) and ⧫ is a parallel operator.

We can classify the nature of protocols composition on the dependance degree between them. We may thus distinguish between two kinds of Compositability relationship:

***Definition 4. (Partial Compositability)***. Two protocols $P_k$ and $P_r$ meet the Partial Compositability relationship *iff* $\exists\, x_i \in (P_r^{in})_\theta(C), \exists\, y_j \in (P_k^{out})_\theta(C) \mid (x_i, y_j)$, 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 others protocols.

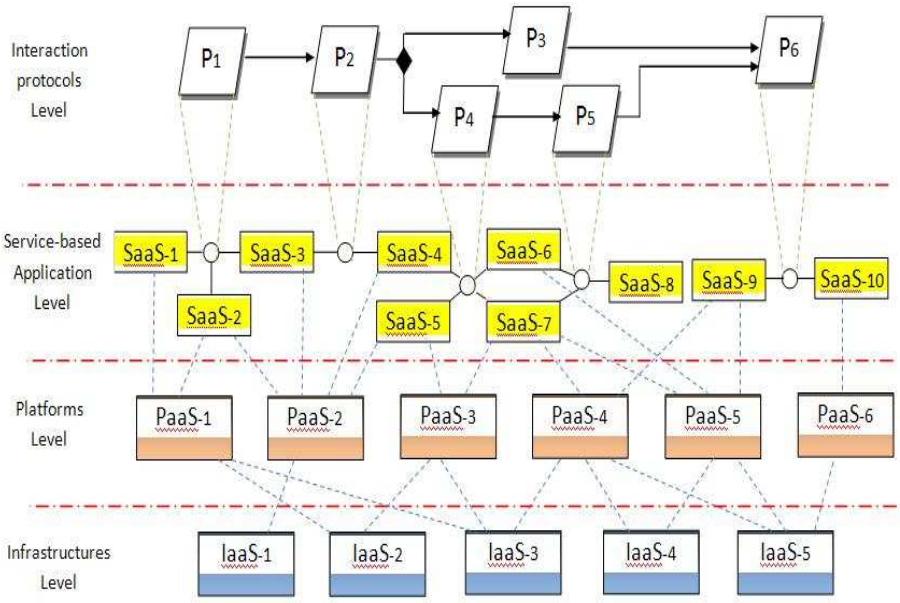


FIG. 4.1. *An Example of Business Protocols Composition*

In the example presented in figure 4.1, we have two partial Compositability relationships: $(P_3, P_6)$ and $(P_5, P_6)$.

***Definition 5. (Full Compositability)***. Two protocols $P_k$ and $P_r$ are called Full Compositability *iff* $\forall x_i \in (P_r^{in})_\theta(C), \exists\ y_j \in (P_k^{out})_\theta(C) \mid (x_i, y_j)$, 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 $P_k$.

In our example, we have four full Compositability relationships: $(P_1, P_2)$, $(P_2, P_3)$, $(P_2, P_4)$ and $(P_4, P_5)$.

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 does not means that $P_r$ has a partial (full) compositability relationship with $P_k$ .

***Proposition***. Let P $=\{P_1, P_2, \ldots, P_m\}$ a set of business protocols. The set $P$ constitutes a meaningful composition if:

$$\forall P_k \in (P - P_{init}{}^3), \forall x_i \in (P_k^{in})_\theta(C), \exists\ y_j \in (P_r^{out})_\theta(C) \mid (x_i, y_j), \text{ where } r, k \in [1, m] \text{ and } r \neq k.$$

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

In the next section we present a set of operators that allows the creation of new value-added protocols using existing ones as building blocks. Sequence, alternative, iteration, and parallel are typical constructs specified in the control flow.

**4.3. 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:

$$P ::= \quad \varepsilon \quad | \quad P \rightarrow P \quad | \quad P \diamondsuit P \quad | \quad P \blacklozenge P \quad | \quad P \curvearrowleft \quad | \quad P \xrightarrow{e} P$$

Where:

**Empty Protocol.** The empty protocol $\varepsilon$ is a protocol that performs no operation. It is used for technical and theoretical reasons.

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

**Sequence Operator.** $P_1 \rightarrow P_2$ 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 4.2-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 $P_2$.

**Choice Operator.** $P_1 \diamondsuit P_2$ represents a composite protocol that behaves as either protocol $P_1$ or protocol $P_2$. i.e., $\diamondsuit$ is an Alternative (or a Choice) operator.

Figure 4.2-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.

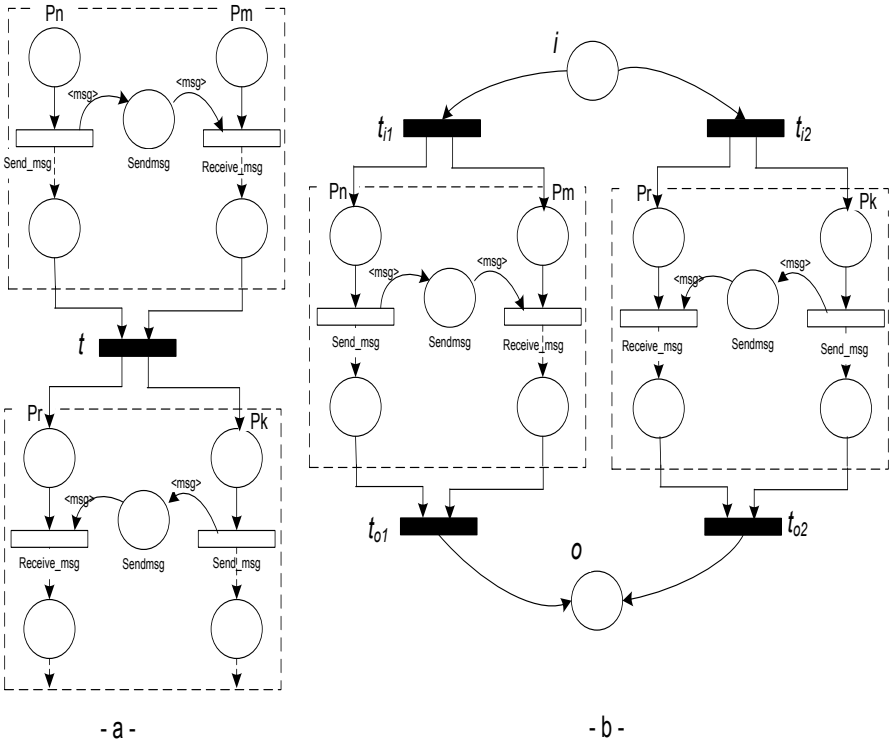


FIG. 4.2. *Protocols Operators: (a) Sequence and (b) Alternative*

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

This operator which models concurrency protocols execution is represented by means of parallel case or multi-threading in Petri net (see figure 4.3-a). In fact, we add one transition and $n$ arcs, where $n$ is the number of the participants in the protocols $P_1$ and $P_2$.

**Iteration Operator.** $P\,\circlearrowleft$ represents the execution of a protocol followed a certain number of times by itself. i.e., $\circlearrowleft$ is an Iteration operator.

In Petri net (see figure 4.3-b), an iteration has a transition 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 $P$.

**Refinement Operator.** $P_1 \xrightarrow{e} P_2$ 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.

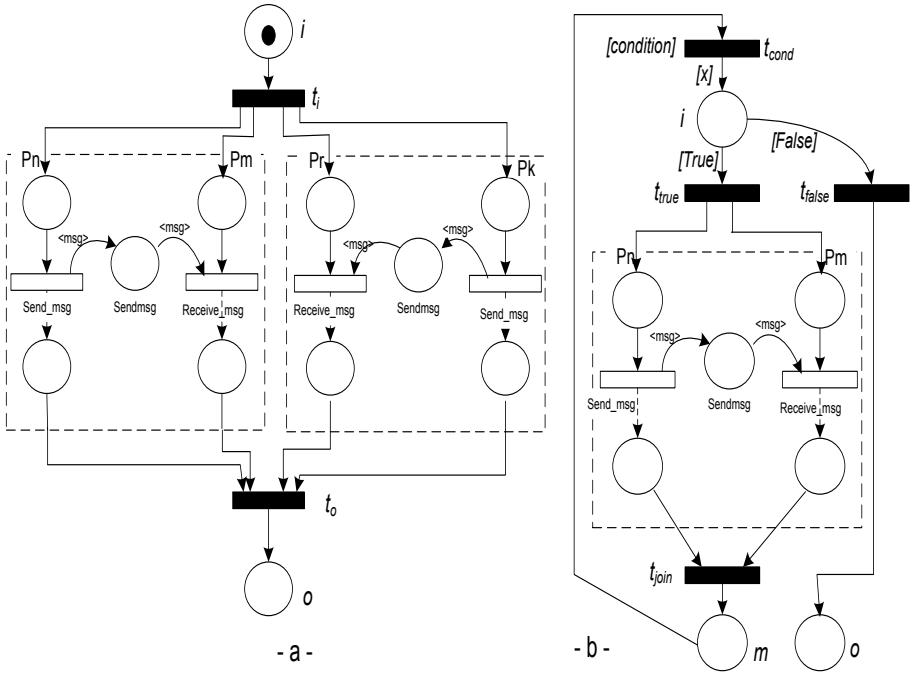


FIG. 4.3. *Protocols Operators: (a) Parallelism and (b) Iteration*

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.

Figure 4.4 shows an example of a refined protocol $P_1 \xrightarrow{e} P_2$ where the event $e = Receive\_msg\_1$. The refined protocol behaves as $P_1$ except for the event $e$ in $P_1$, $P_2$ can be executed, then $P_1$ can resumes its execution.

**5. Verification of Composite Protocol Correctness .** A composite business protocol is a system that consists of several conceptually autonomous but cooperating units. It is difficult to specify how this system should behave and ensure that it behaves as required by the specification. The reason is that, given the same input and initial state, the system may produce several different outputs.

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.

However, analyzing an interaction protocol is a significantly phase. Petri nets allow us to validate and evaluate the usability of a system by performing automatic and/or guided executions. Moreover, by applying other analysis techniques it is possible to verify general properties, such as absence of deadlocks and livelocks, and application-specific properties.

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 [17]. From the Petri net model, we can explore its state space and look for deadlock (state from which no progression is possible), look for inconsistent state or test for more complex logical formulas.

General properties deal with application-independent concerns. These properties include boundness, liveness and deadlock-freedom. The first two properties were investigated to validate and debug the model and to provide insight into CPN behaviour.

We are interested to define the minimal conditions for the composite protocol correctness. The conditions we impose are:

1. The net is deadlock free.

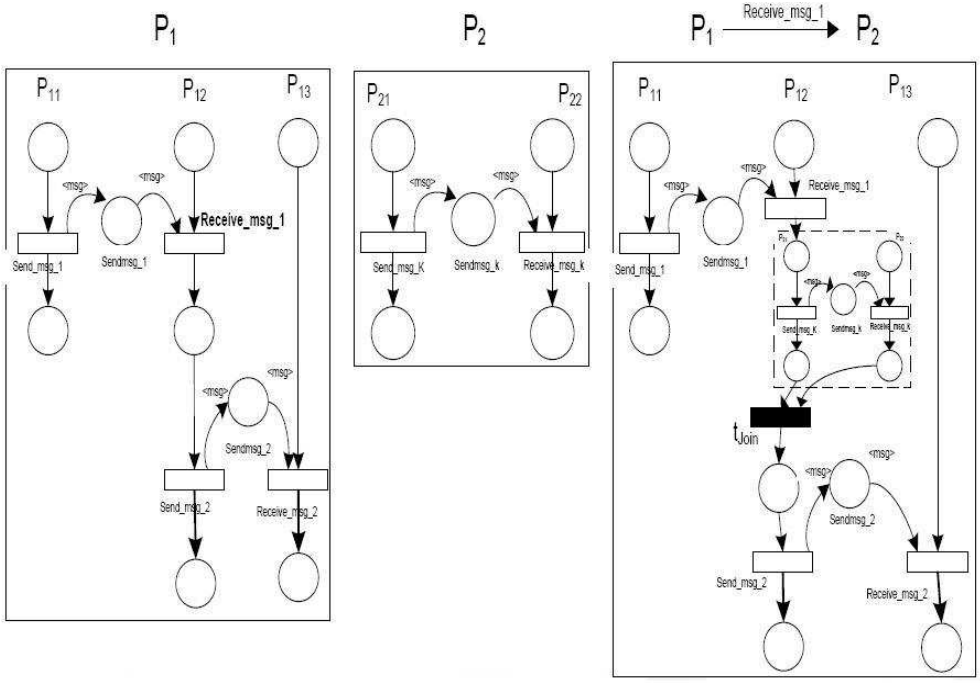


FIG. 4.4. *An Example of Protocol Refinement*

2. For any reachable marking $M$ in the Petri net there is an execution sequence from $M$ such that all the component protocols will still be able to terminate correctly their execution.
3. The communication places is bounded.

From the state space, we can look for deadlocks. The Petri net is deadlock-free if and only if all the role processes of the *final* protocols are terminated. More precisely, let $term_r$ be the termination place for each role process $r$ acting during the final interaction protocols ($r \in P_{fin}$) and let $endsta$ be the terminal states of the state space.

$$\forall r \in P_{fin}, term_r \in endsta \Leftrightarrow the\,net\,is\,deadlock\,free$$

This formula states that it is eventually true that each role $r$ will be at its termination point labeled with $term_r$. However, a role process that ends in the state $r$ where, $r \notin \{term_r\}$ represents an active role: termination in a state where roles are active indicates a violation.

Also, we require that the participant protocols should not be allowed to send an infinite number of messages to the $Sendmsg_i$ (places that correspond to the exchanged messages in the Petri nets). So, each communication place $Sendmsg_i$ may has at most one token. This corresponds to the maximum capacity of the message buffer (ie one).

**6. Related Work.** Over the last few years, the prosperous research on collaborative enterprises applications has led to a multitude of results. In this section, we overview major techniques that are most closely related to our approach.

**6.1. Cloud Computing Based Approaches.** Today, large technology vendors as well as open-source software projects both address the hybrid Cloud market and are developing virtual infrastructure management tools to set-up and manage hybrid Clouds [21].

The Reservoir architecture [32] aims to satisfy the vision of service oriented computing by distinguishing and addressing the needs of service providers and infrastructure providers. Service providers interact with the

end-users, understand and address their needs. They do not own the computational resources; instead, they lease them from infrastructure providers which interoperate with each other creating a seamlessly infinitive pool of IT resources.

The VMware workstation [12] offers live migration of virtual appliances and machines between data centers and allows service providers to offer IaaS while maintaining compatibility with internal VMware deployments.

HP [15] provides three offerings for hybrid Cloud computing: HP Operations Orchestration for provisioning, HP Cloud Assure for cost control, and HP Communications as a Service for service providers to offer small businesses on-demand solutions. The Cloud-based HP Aggregation Platform for SaaS streamlines operations for both the service provider and the businesses customer by automating processes such as provisioning, activation, mediation charging, revenue settlement and service assurance.

Model-driven approaches are also employed for the purpose of automating the deployment of complex IaaS services on Cloud infrastructure. For instance, in [26], authors propose the mOSAIC Ontology and the MetaMORP(h)OSY methodology for enabling model driven engineering of Cloud services. The methodology uses model driven engineering and model transformation techniques to analyse services. Due to the complexity of the systems to analyse, the mOSAIC Ontology is used in order to build modelling profiles in MetaMORP(h)OSY able to address Cloud domain-related properties.

When examining the Cloud based approaches we observe a recurrent theme. They do not allow for easy extensibility or customization options. Better ways are necessary for Cloud service consumers to orchestrate a cohesive Cloud computing solution and provision Cloud stack services that range across networking, computing, storage resources, and applications from diverse Cloud providers.

The work presented in this paper is considered as a first step toward AIPaaS (Agent Interaction Protocols as a Service). The AIPaaS approach is based on the idea of reducing the complexity involved when developing a composite application. In our work, the knowledge obtained during Cloud services composition is stored, reused, and shared. In fact, we proposed 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 provides a set of operators that allows the creation of new value-added protocols using existing ones as building blocks.

**6.2. Web Services Based Approaches.** With the growing trends of service oriented computing, composition of Web services has received much interest to support flexible inter-enterprise application integration. As stated in [24] and [33], 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 [24], [9], [11]. 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 [29] [25]. 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 [22]).

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 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] [3], we have explored the relationship between Web services, multiagent systems and enterprise application integration.

**6.3. Agent Based Approaches.** Multiagent systems are a very active area of research and development. In fact, several researchers are working at the intersection of agents and enterprise systems.

For example, Buhler et al. [13] summarize the relationship between agents and Web services with the aphorism Adaptive Workflow Engines = Web Services + Agents: namely, Web services provide the computational resources and agents provide the coordination framework. They propose the use of the BPEL4WS language as a specification language for expressing the initial social order of the multi-agent system. [13] does not provide any design issues to ensure the correctness of their interaction protocols.

Driven by the motivation for reuse of interaction protocols, [35] and [19] consider protocols as a modular abstractions that capture patterns of interaction among agents. In these approaches, composite protocols can be specified with a Protocol Description Language (such as: CPDL[4] or MAD-P[5]). Although formal, [35] and [19] do not not provide any step for the verification of the composite protocols.

Agent-oriented software methodologies aim to apply software engineering principles in the agent context e.g. Tropos, AMCIS, Amoeba, and Gaia. Tropos [10] [30] and AMCIS [5] [4] differ from these in that they include an early requirements stage in the process. Amoeba [18] is a methodology for business processes that is based on business protocols. Protocols capture the business meaning of interactions among autonomous parties via commitments. Gaia [14] differs from others in that it describes roles in the software system being developed and identifies processes in which they are involved as well as safety and liveness conditions for the processes. It incorporates protocols under the interaction model and can be used with commitment protocols.

Our methodology differs from these in that it is aimed at achieving protocol re-usability by separation of protocols and business rules. It advocates and enables reuse of protocols as building blocks of business processes. Protocols can be composed and refined to yield more robust protocols.

**7. Conclusion and Future Work.** In this paper, we presented a formal framework that allows 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 participant implementations.

The key feature of this framework is the ability to model and formally verify composition of business 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 interaction 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.

An agent-based system that supports the work is currently being developed. The proposed system has been designed to enable interoperability and cross-Cloud application management. It solves the interoperability issues between heterogeneous Cloud services environments by offering a harmonized API. Also, it enables the deployment of applications at public, private or hybrid multi-Cloud environments.

**Acknowledgements.** The author would like to thank the anonymous reviewers for their valuable comments and suggestions, which were helpful in improving the paper.

REFERENCES

[4]CPDL: Communication Protocol Description Language
[5]MAD-P: Modular Action Description for Protocols

[1] B. Bauer, J. P. Müller, and J. Odell, *Agents and the UML: A Unified Notation for Agents and Multi-agent Systems*, in Agent-Oriented Software Engineering II, Second International Workshop, AOSE'01, vol. 2222 of LNCS, Springer, 2001, pp. 148–150.

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

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

[4] D. Benmerzoug, M. Boufaida, and Z. Boufaida, *Developing Cooperative Information Agent-Based Systems with the AMCIS Methodology*, in IEEE International Conference on Advances in Intelligent Systems: Theories and Application, Luxembourg, November 2004, IEEE press.

[5] D. Benmerzoug, M. Boufaida, and Z. Boufaida, *From the Analysis of Cooperation Within Organizational Environments to the Design of Cooperative Information Systems: An Agent-Based Approach*, in OTM Workshops, vol. 3292 of LNCS, Larnaca, Chypre, October 2004, Springer, pp. 495–506.

[6] D. Benmerzoug, M. Boufaida, and F. Kordon, *A Specification and Validation Approach for Business Process Integration based on Web 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 2007, NSTIIC press, 2007, pp. 163–168.

[7] D. Benmerzoug, F. Kordon, and M. Boufaida, *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), vol. 10 of LNBIP, Montpellier, France, June 2008, Springer, pp. 78–92.

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

[9] D. Berardi, G. D. Giacomo, M. Mecella, and D. Calvanese, *Automatic composition of process-based web services: a challenge*, in Proc. 14th Int. World Wide Web Conf. (WWW'05), 2005.

[10] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos, *Tropos: An Agent-Oriented Software Development Methodology*, International journal of Autonomous Agents and Multi-Agent Systems, 8 (2004), pp. 203–236.

[11] J. Bronsted, K. M. Hansen, and M. Ingstrup, *Service composition issues in pervasive computing*, IEEE Pervasive Computing, 9 (2010), pp. 62–70.

[12] E. Bugnion, S. Devine, M. Rosenblum, J. Sugerman, and E. Y. Wang, *Bringing virtualization to the x86 architecture with the original vmware workstation*, ACM Trans. Comput. Syst., 30 (2012), pp. 12:1–12:51.

[13] P. A. Buhler and J. M. Vidal, *Towards adaptive workflow enactment using multiagent systems*, International Journal On Information Technology and Management, 6 (2005), pp. 61–87.

[14] L. Cernuzzi, A. Molesini, A. Omicini, and F. Zambonelli, *Adaptable multi-agent systems: the case of the gaia methodology*, International Journal of Software Engineering and Knowledge Engineering, 21 (2011), pp. 491–521.

[15] D. Collins, *Communications as a service for midsize businesses*, 2009.

[16] A. Copie, T.-F. Fortis, V. I. Munteanu, and V. Negru, *Datastores supporting services lifecycle in the framework of cloud governance*, Scalable Computing: Practice and Experience, 13 (2012), pp. 251-7.

[17] CPN-AMI:, *http://move.lip6.fr/software/cpnami/*.

[18] N. Desai, A. K. Chopra, and M. P. Singh, *Amoeba: A Methodology for Fodeling and Evolving Cross-Organizational Business Processes*, Journal of ACM Trans. Softw. Eng. Methodol., 19 (2009).

[19] N. Desai and M. P. Singh, *A modular action description language for protocol composition*, in Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, AAAI Press, 2007, pp. 962–967.

[20] C. Girault and R. Valk, *Petri Nets for Systems Engineering, A Guide to Modeling, Verification, and Applications*, Springer Verlag, July 2002.

[21] S. V. Hoecke, T. Waterbley, J. Devos, T. Deneut, and J. D. Gelas, *Efficient management of hybrid Clouds*, in The Second International Conference on Cloud Computing, GRIDs, and Virtualization, Rome, Italy, September 2011, pp. 167–172.

[22] SAP, Siebel Systems, IBM, Microsoft, *Business process execution language for web services version 1.1*, tech. rep., 2003.

[23] T. F. R. market:, *http://www.forrester.com/*.

[24] N. Milanovic and M. Malek, *Current solutions for web service composition*, IEEE Internet Computing, 8 (2004), pp. 51–59.

[25] M. Montali, M. Pesic, W. M. P. van der Aalst, F. Chesani, P. Mello, and S. Storari, *Declarative specification and verification of service choreographiess*, International journal of ACM Transactions on the Web, 4 (2010).

[26] F. Moscato, B. D. Martino, and R. Aversa, *Enabling Model Driven Engineering of Cloud Services by using mOSAIC Ontology*, Scalable Computing: Practice and Experience, 13 (2012). pp. 29–44.

[27] D. K. Nguyen, F. Lelli, M. P. Papazoglou, and W.-J. van den Heuvel, *Blueprinting Approach in Support of Cloud Computing*, International Journal of Future Internet, 4 (2012), pp. 322–346.

[28] OCL:, *Object constraint language.* (www.omg.org/cgi-bin).

[29] M. P. Papazoglou, K. Pohl, M. Parkin, and A. Metzger, eds., *Service Research Challenges and Solutions for the Future Internet - S-Cube - Towards Engineering, Managing and Adapting Service-Based Systems*, vol. 6500 of Lecture Notes in Computer Science, Springer, 2010.

[30] L. Penserini, T. Kuflik, P. Busetta, and P. Bresciani, *Agent-based organizational structures for ambient intelligence scenarios*, Ambient Intelligence and Smart Environments, 2 (2010), pp. 409–433.
[31] P. Mell, and T. Grance, *The NIST Definition of Cloud Computing*, 2009.
[32] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I. M. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Cáceres, M. Ben-Yehuda, W. Emmerich, and F. Galán, *The reservoir model and architecture for open federated cloud computing*, IBM Journal of Research and Development, 53 (2009), pp. 535–545.
[33] Q. Z. Sheng, B. Benatallah, Z. Maamar, and A. H. H. Ngu, *Configurable composition and adaptive provisioning of web services*, IEEE T. Services Computing, 2 (2009), pp. 34–49.
[34] S. Subashini and V. Kavitha, *A survey on security issues in service delivery models of cloud computing*, J. Network and Computer Applications, 34 (2011), pp. 1–11.
[35] B. Vitteau and M.-P. Huget, *Modularity in interaction protocols*, in Advances in Agent Communication, vol. 2922 of LNCS, Springer, 2004, pp. 291-9.
[36] FIPA - Foundation for Intelligent Physical Agents:, *FIPA Communicative Act Library Specification.* (http://www.fipa.org/specs/XC00037/).
[37] J. Abreu, L. Bocchi, J.L. Fiadeiro, and A. Lopes, *Specifying and Composing Interaction Protocols for Service-Oriented System Modelling*, in International Conference on Formal Techniques for Networked and Distributed Systems - FORTE'07, vol. 4574 of LNCS, Springer, 2007, pp. 358-373.

*Edited by:* Viorel Negru and Daniela Zaharie
*Received:* May 25, 2013
*Accepted:* Jul 9, 2013