Autonomic Protocol-based Coordination in Dynamic Inter-Organizational Workflow

Wassim BOUAZIZ
SOGETI High Tech
AEROPARK, 3 Chemin de Laporte
31300 Toulouse, France
wbouaziz@gmail.com

Eric ANDONOFF
Laboratoire IRIT / Université Toulouse 1
2 rue doyen Gabriel Marty
31042 Toulouse Cedex, France
andonoff@univ-tlse1.fr

***Abstract*— Given the maturity of Internet standards, inter-organizational workflow is expected to be deployed in environments that are more dynamic and open than before. This paper addresses Inter Organizational Workflow (IOW) coordination in such a context, mainly investigating autonomic coordination managed at run-time. It is based on the idea that an agent-based approach is suitable to deal with this issue. More precisely, this paper introduces a framework for dynamic IOW in which involved processes are encapsulated into agents, called Process Agents (PA) in order to give them the capability to autonomously decide with whom, when and how to cooperate, and in which involved processes can access protocol components for their coordination needs. Our approach is based on the capability of PAs in playing different coordination protocols in order to take part in new business opportunities. This solution has numerous advantages. First, it provides extendable and reusable coordination components. Then, it supports run-time protocol integration. Finally, it eases openness since it imposes very few constraints.**

Keywords*— *Dynamic Inter-Organizational Workflow, Autonomic Coordination, Agent Protocol

I. INTRODUCTION

Nowadays, competition in which organizations are involved leads them to cooperate and share business opportunities: they coordinate their business processes in order to reach a common goal corresponding to a value added service [1]. Moreover, these coordinated processes constantly evolve due to both organizational (e.g., business objectives, process improvement) and technological changes (e.g., new resources, new technologies) that occur in the involved organizations. Thus, these organizations need a flexible framework to support their cooperation, i.e. a framework that supports dynamic inter-organizational workflow applications.

Inter-Organizational Workflow (IOW) is a key concept to support the cooperation between distributed and heterogeneous processes running in different autonomous organizations [2]. It thus requires specific features such as autonomy, decentralization, definition of the universe of discourse through a repository (to solve the various semantic conflicts that are bound to occur between the involved processes) and capability of interaction among processes. Dynamic IOW refers to applications where partners (i.e. organizations) are not necessarily known before IOW execution; these partners can also be unable to provide anymore the required service, they can provide it with a bad quality of service or they can be temporarily unavailable. Dynamic IOW also takes into account specific additional features: openness and flexibility. Openness refers to the freedom the partners have to join or leave the cooperation while flexibility corresponds to the ability of the IOW to face changes that occur in its environment [3].

Coordination of business processes involved in an IOW is defined as the set of activities concretizing the cooperation between the involved organizations in order to reach their common goal. In a dynamic context, it also includes additional coordination services (patterns) such as:

- Finding partners, which consists in, for a requester organization, selecting one or several provider organizations able to execute a requested process.
- Negotiation of a process between the requester organization and the previous selected provider organizations. Negotiation criteria may be as varied as due time, quality of the process, visibility of its evolution and way of executing it. The result of this step is the identification of the provider organization in charge of the requested process.
- Contracting between the requester and the selected provider aiming at formalizing their cooperation.
- Synchronization of the distributed and concurrent execution of these processes.

Autonomic coordination refers to the ability of processes involved in a dynamic IOW to automatically and autonomously coordinate their activities. It means that the involved processes decide by themselves how to maintain the cooperation.

The issue being addressed in this paper is how to provide a framework for autonomic coordination in dynamic IOW, i.e. a framework making involved processes able (i) to support the different additional coordination services listed before (finding partners, negotiation, contracting between partners…), and (ii) to decide by themselves when, with whom, and how to cooperate.

Coordination of processes involved in IOW is addressed in different works following two types of approaches. The first

one advocates to use a service-oriented approach [4–9]. This approach, supported by different industrials companies, gave rise to several languages for inter-organizational processes interconnection such as ebXML or WSCL, and also to the development of service oriented architectures for the execution of these processes. For instance, [7] and [9] define a framework for both process specification including their resources, and process selection, interconnection and execution. But works following this approach do not address the issue of this article that is autonomic coordination in dynamic IOW: they lack the dynamic dimension as they do not propose coordination mechanisms deployed during the IOW execution.

The second type of approach advocates an agent-based approach. Indeed, as different authors [10–18] suggest, agent technology is being used in an increasingly wide variety of applications, ranging from small systems for personal assistance to open complex systems for industrial applications. The Internet has also pushed the use of agent technologies in the business process field and electronic commerce. Multi-agent systems bring technical solutions and abstractions to deal with distribution, autonomy and openness, which are inherent to the automation of dynamic IOW. For instance, [10, 12, 13, 15, 17, 18] use the agent notion to enhance the capabilities of workflow management systems for autonomous cooperation including finding or subcontracting workflow services. But these works do not consider all the coordination services identified before to address the problem of autonomic coordination in dynamic IOW in a comprehensive and unified framework.

As a consequence, even if IOW coordination has been heavily investigated, this issue remains open in a dynamic context: it is still a topical issue, which needs to be tackled and solved.

In this paper, we advocate an agent-based approach to deal with this issue. On the one hand, agent coordination protocols serve as a basis for dynamic IOW business process coordination. Coordination protocols, which are widely recognized as an essential mechanism for coordination within multi-agent systems, are also well suited for process coordination. Indeed, whatever the coordination service (finding partners, negotiation or contracting between partners) identified before, it follows a recurrent schema. After an informal interaction, the participating processes commit to follow a strict coordination protocol. This protocol rules the conversation by a set of laws which constraints the behavior of the participants, assigns a role to each of them and organizes their cooperation. Therefore, protocols constitute well identifiable and reusable coordination patterns in dynamic IOW. For this reason, we decided to isolate them in order to better study and implement them as first class entities and reuse them at run-time; we entrusted their management to a Protocol Management System (PMS), which can be viewed as a server of coordination protocols for processes involved in dynamic IOW [19]. Doing so, we apply the principle of separation of concerns, recognized as a good design practice from a software engineering point of view [20], in order to separate the functional capability of each IOW participant from its interaction capabilities.

On the other hand, agent technology is also used to make processes involved in dynamic IOW more autonomous than before, providing them with decision-making abilities to decide by themselves when, with whom, and how to coordinate.

Thus, the features of our approach are the following:

- Coordination of processes involved in dynamic IOW is protocol-based.
- Processes are encapsulated into agents, called Process Agents (PA), in order to give them the ability to autonomously decide with whom, when and how to cooperate.

The paper contributions are (i) an ontology for specifying coordination protocols as separated services managed by a PMS, (ii) a model enabling dynamic extraction, instantiation and execution of roles that the different protocol participants may hold in a protocol, and (iii) an illustration of how processes involved in a dynamic IOW coordinate together using coordination services offered by the PMS.

The remainder of the paper is organized as follows. Section 2 gives an overview of our approach to deal with autonomic coordination in dynamic IOW. It also introduces an example, which will be used throughout the paper. Section 3 is dedicated to the PMS presentation. It first presents the Protocol ontology for protocol specification, while section 4 deals with dynamic role behavior extraction and execution from protocol specification. Section 5 first gives a brief description of the implementation and then illustrates coordination of processes involved in the dynamic IOW application introduced in section 2. Finally, section 6 stands our contribution according to related works and concludes the paper.

II. APPROACH OVERVIEW

This section presents the two main features of our approach to deal with autonomic coordination in dynamic IOW and introduces the running example.

A. *Using Agent Technology and the Semantic Web*

As said before, dynamic IOW is a specific case of IOW where inter-organizational business process is defined at run-time: partners are not necessarily known before the IOW execution either because they are unknown at design-time or no more available or defective at run-time. Consequently it is necessary that the different business process involved in a dynamic IOW integrates specific abilities to face this dynamic context.

On the one hand, agent technology [21] provides natural abstractions to design and model dynamic IOW taking into account autonomic abilities, flexibility and openness [2].

Regarding autonomic abilities, agent technology permits to model each participating process as an autonomous agent, called Process Agent (PA) representing an organization that it is able to cooperate with the other organizations involved in the IOW. In addition to their internal behavior implementing the process they represent, these PAs may be completed by supervising, reasoning and decision-making abilities in order to

be able to decide by themselves when, with whom and how to coordinate. Modeling IOW processes as agents is thus natural from the autonomic point of view.

Openness is a specific feature of dynamic IOW since process partners are not necessarily known at design-time, but also may change during IOW execution. Openness is also a property of multi-agent applications where agents may freely appear or disappear during execution. Modeling IOW processes as agents is thus natural from the openness point of view.

Flexibility, is a consequence of autonomy and openness. Since dynamic IOW is an open structure, that process partners may leave or enter freely, and since process partners are implemented as agents (PAs) having autonomy, reasoning and decision-making abilities, these agents have to be flexible to adapt their interactions. These PAs must be able to execute coordination services listed before (finding partners, negotiation between partners…) and eventually modify their internal behavior redefining, reordering or subcontracting their activities. Modeling IOW processes as agents is thus natural from the flexibility point of view.

On the other hand, the semantic Web is a complementary enabling technology. It first helps to represent a shared business view, through a common terminology or an ontology, without which it would not be possible to solve the various semantic conflicts that are bound to occur between the heterogeneous, distributed and autonomous processes involved in an IOW. Moreover, in open and dynamic environments, where process partners are numerous and not necessarily known a priori, the semantic Web also provides means to describe, publish and discover processes, called process services, offered by involved partners. Finally, the semantic Web permits to describe coordination means as explicit, machine readable and sharable specifications: it facilitates communication and semantic inter-operability between processes involved in an IOW, and makes reasoning about coordination means (i.e. protocols) possible. Consequently, the semantic Web contributes to make automated coordination possible [22, 23].

B. *Protocols for Dynamic IOW Processes Coordination*

This second feature is divided into the two following principles.

1) Protocols as Coordination Services. As previously said, another feature of our approach is that agent coordination protocols serve as a basis for dynamic IOW processes coordination. Indeed, whatever the coordination service (finding partners, negotiation or contracting between partners) identified before, it may be implemented as an agent coordination protocol. Several protocols in multi-agent applications such brocker, matchmaker, argumentation, heuristic, delegation, or call for proposal may be used to implement these coordination services (e.g. matchmaker or broker protocols for the finding partners coordination service). To sum up, the idea is to consider multi-agent protocols as coordination patterns to support the entire coordination life cycle in dynamic IOW.

2) PMS as a Coordination Middleware. We also advocate the separation of coordination protocols from IOW process. The consequence of pushing coordination protocols out of dynamic IOW is that coordination protocols are managed by an external component called a Protocol Management System (PMS), whose architecture has been presented in [19]. A PMS proposes the three following services: (i) specification of coordination protocol, (ii) selection of a protocol according to the IOW coordination needs, and finaly (iii) for each process agent involved in a coordination protocol, the dynamic integration and execution of the role it holds in the protocol. This paper only focuses on the specification and execution services; the selection service is out of the scope of the paper.

As illustrated in Fig. 1 and as suggested in [24], a specific agent, called Moderator, rules the conversation between PAs involved in an IOW. Instead of being duplicated and encapsulated into each PA, these shared rules are centralized in the Moderator agent whose aim is to ensure that each interaction in the conversation is compliant with the underlying protocol rules. Fig. 1 below illustrates this idea.

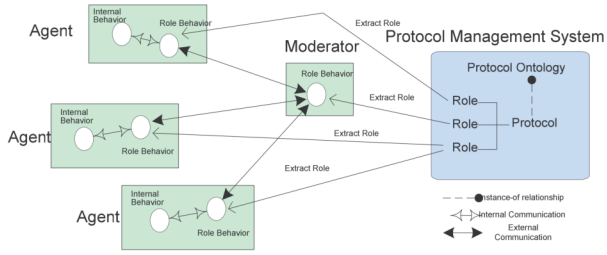

Fig. 1. Moderator Agent for Ruling Conversations

Each moderator manages a single conversation which is consistent with a coordination protocol, and it has the same lifetime as the conversation it manages. It also grants roles to participating agents and stores in a database all the communication acts issued by participating agents. A moderator also exploits a Domain Ontology to ensure that these participating agents use an adequate vocabulary. Finally, the moderator is an agent that runs inside the PMS.

C. *Running Example*

The proposed running example represents a dynamic IOW application for repairing electronic equipments. This example involves several organizations, including the organization responsible for the reparation. We call this organization the Pilot Organization (PO). The other organizations are seen as contractors, helping the PO to reach the repairing objective. According to the type of repair to carry out, the PO dynamically finds a partner able to help it. Protocols intervene at this stage. In the example, we use the Iterative Contract Net protocol [25] to support PO's partner selection. Each organization involved in the process is represented by a process agent and the Iterative Contract Net (ICN) protocol is used to rule the interaction between them. Of course, process agents do not necessarily implement the ICN protocol: they just need to dynamically integrate it, according to the role they hold in the protocol.

Below, we detail the PO's process along with the Interactive Contract Net protocol used for partner selection. In

both cases, we use the Petri Net (PN) formalism for cooperation description.

The PO process. is defined as follows. A client initiates the process by submitting to the PO a request for repairing an electronic equipment. After analyzing the request, an estimation is done and a quote is sent to the client. If the quote is accepted by the client, a diagnosis determines if the repair is outsourced or not.

Fig. 2 gives a PN representation of this process and illustrates the interaction between the different involved organizations.

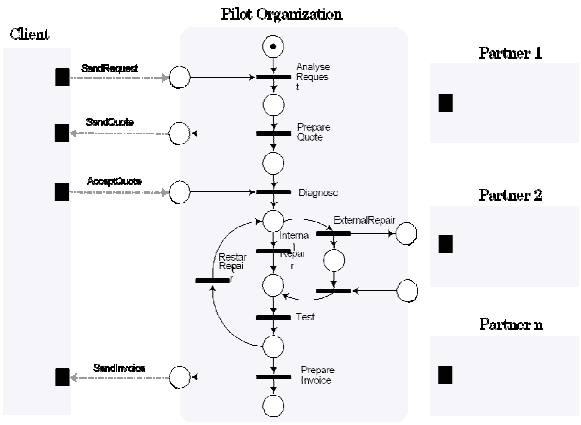

Fig. 2. Inter-Organizational Repairing Process of Electronic Equipment.

In the Iterative Contract Net (ICN) protocol, an agent assumes the initiator role looking for a service performed by one or several other agents, called contractors. To this end, the initiator sends a call for proposal to the different possible contractors. After evaluation, contractors can accept or refuse the proposal. Then, the initiator evaluates the received offers and can decide to accept one, reject all, or send a modified call.

Fig. 3 below illustrates this protocol. We have simulated and validated it using Renew, a Petri net-based environment [15]. This figure shows a simplified version of the ICN protocol which includes three roles held by several agents: the Moderator role, the Manager role and the Contractor role. The Manager role (M) corresponds to the role held by the PO while the Contractor role (C) corresponds to the role held by the different contractors to which the call for proposal is sent. The Moderator role (Mo) supports the interaction between the two previous roles: interaction places are places of the Petri-net which are shared between two roles (e.g. Confirm, Accept…). Suppose that five agents are involved in a conversation based on this protocol: the manager, the moderator and three partners playing the Contractor role. The manager advertises the moderator for a requested service, e.g. repairing a specific component of an electronic equipment task. This latter sends a call for proposal to three contractors. Each one analyzes this proposal and can decide to accept or refuse. The moderator manages the reception of answers in parallel and forwards to the manager only the accepted bids.

If all the contractors decide to not answer to the proposal, the moderator informs the manager that there is no bid. When the manager selects a bid, the moderator sends an Accept to the corresponding contractor and a Reject to others, and then waits for an acknowledgement from the selected contractor before ending the protocol.

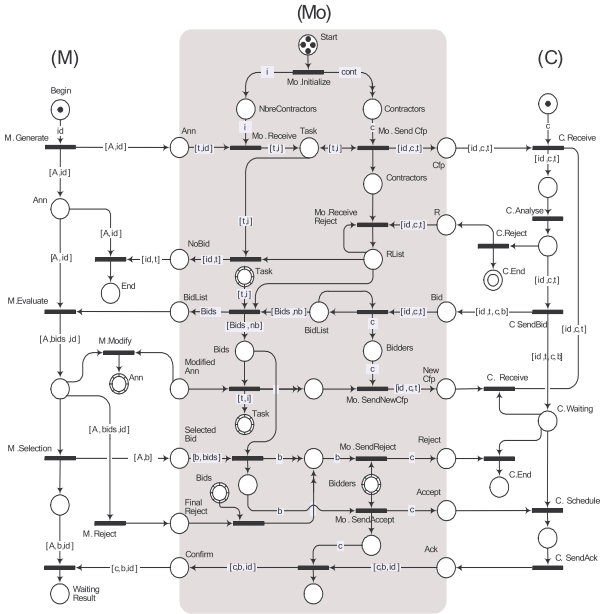

Fig. 3. Iterative Contract Net Protocol

III. COORDINATION PROTOCOL ONTOLOGY

The first step for engineering coordination protocols is to deal with protocol specification issue. As said before, the semantic Web is useful to deal with this issue since it permits to describe coordination protocols as an explicit, machine readable and sharable ontology [23].

This section gives a precise and non-ambiguous definition of what is a coordination protocol. It first introduces the three abstraction levels for protocols. It then presents the protocol ontology and a protocol classification model taking into account only dynamic IOW coordination protocols. These models have been specified in OWL using Protege-2000 software, but they are presented with equivalent UML models for readability reasons.

A. Three Levels for Coordination Protocols

As illustrated in Fig., we distinguish three abstraction levels for coordination protocols.

The first and more abstract level corresponds to the Protocol meta-model (protocol ontology), defining the invariant structure shared by all the protocols considering both static (profile) and dynamic (behavior) aspects of protocol in order to facilitate their dynamic selection and execution to process agents. The second abstraction level is the specification level. Concrete coordination protocols, such as for example the iterative Contract Net protocol [25], are defined as instances of the previous ontology.

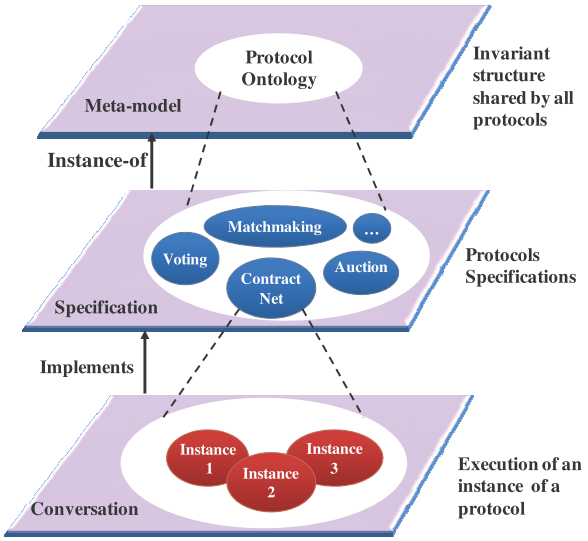

Fig. 4. Abstraction Levels of Protocols

The third abstraction level is the execution level. At this level, we find conversations (occurrence or instance of a protocol) between the different IOW Process Agents (PA), each one playing a role in the conversation. For instance, considering the previous running example section, we can have different instances of the Iterative Contract Net (ICN) protocol ruling the execution of different requests submitted by clients to the PO for repairing electronic equipments.

B. *Protocol Ontology*

The Protocol ontology defines coordination protocols as services considering both their profile and behavior. The profile defines the purpose of a protocol while the behavior describes actions achievable by roles involved in a protocol. Fig. 5 and 8 give an UML representation of the concepts of these two notions (profile and behavior).

1) Protocols Profile. The protocol profile defines a structured description of protocol purpose.

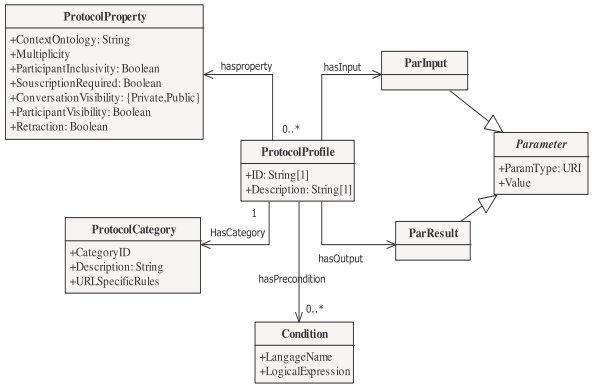

Fig. 5. Protocol Profile

As illustrated in Fig. 5, the protocol profile is described through a set of properties (ProtocolProperty class), parameters (Parameter class), and a category characterizing the type of protocol (ProtocolCategory). ProtocolProperty describes static properties of the protocol, as pairs <name, value>. Among these properties, we note the ContextOntology property, which refers to the domain ontology of the protocol, and the ProtocolCategory property, which refers to a protocol taxonomy clustering coordination protocols according to the purpose. This taxonomy is visualized in Fig. 6.

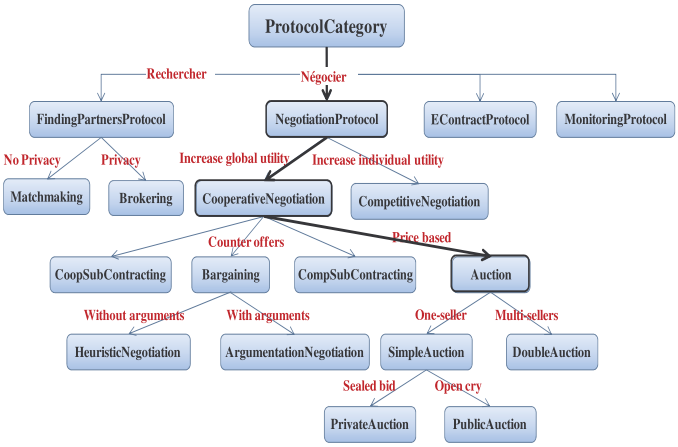

Fig. 6. Protocol Taxonomy

This taxonomy includes multi-agent coordination protocols useful in dynamic IOW. More particularly, these protocols support coordination services listed before, i.e. finding partners, negotiation between partners and contracting services. This taxonomy is thus specific to the IOW field.

Fig. 7 below illustrates the ICN protocol profile. Its category is SubContracting; it includes several parameters (input and result parameters) such as the call for proposal, the agreement, the chosen contractor, and several properties including a reference to the domain ontology underlying the conversation.

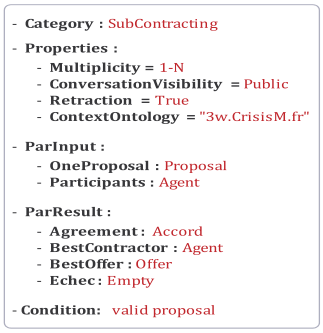

Fig. 7. Profile of the Iterative Contract-Net Protocol

2) Protocols Behavior. The protocol behavior describes the control structure of the protocol. It is defined through the concepts of role, action, conversation acts, data, which can be local variables, input or output parameters, and objectFlow linking actions and conversation acts. Three types of roles are distinguished in a protocol: i) the initiator, responsible for initializing the conversation, ii) the moderator, responsible for monitoring and ruling the conversation, and iii) the participant, representing other stakeholders involved in the conversation.

Roles exchange conversations acts. These latter are described through a type and a content. AbstractAction are operations performed by a role. Every action has a name, and a content type, i.e. an action of sending or receiving a message. An AbstractAction is activated after receiving a message from another role, or after a deadline. Temporal events are represented by data variables including conditions. The description of these actions is abstract; it means that only the signature of the action is specified. The two relationships hasInitialAction and hasFinalAction define respectively the initial action of a protocol and one or more final actions. ObjectFlow, DataFlow and MessageFlow classes are introduced to express the possible transmission of data variables between actions. More precisely, the class ObjectFlow expresses the transition between two actions. Two types of transitions are defined: MessageFlow for the exchange of conversation acts, and DataFlow to represent data streams.

Finally, the Parameter class describes the necessary data structures for protocol execution. Two types of parameters are considered: input and output parameters. Input parameters enable the initialization of a protocol while output parameters are the result of protocol execution. Conditions may be associated to these parameters.

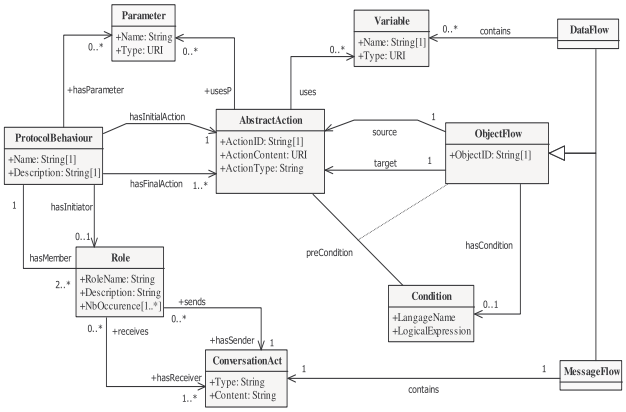

Fig. 8. Protocol Behavior

We do not give additional details about this protocol ontology. Interested readers can refer to [3] for a more detailed presentation of these concepts.

IV. Dynamic Protocol Execution

The second step for engineering coordination protocols is to deal with protocol execution. The section first defines protocol life cycle along with the adopted approach for role extraction and execution according this life cycle. The section then presents role behavior extraction from protocol specification, and finally focuses on dynamic extracted role execution.

A. *Protocol Life Cycle*

Protocol life cycle includes several states which are presented in Fig. 9. We distinguish two levels for these states: a specification level and a deployment level. Regarding the specification level, three states are defined. The first state (state designed) corresponds to specified protocols, i.e. protocols specified in OWL as instances of the Protocol ontology. The second state (state validated) corresponds to a state of a validated protocol, i.e. a protocol whose behavior is validated by simulation and that checks specific properties (e.g. accessibility, ending...). The third state (state implemented) is a complex state since it includes role extraction, role generation and role concretization. Role extraction (state extracted) is the activity which identifies actions performed by PAs holding the role, while role generation (state generated) is the activity which translates extracted roles into XML specifications which are readable and executable by a role engine (cf. section IV.C). Finally, each role is concretized in order to specify how each process agent interprets the actions it integrates (state concretized).

Regarding the deployment level, two states are defined: selected and deployed. The first one is the process agent state when it has chosen the role it is going to hold in a protocol, while the second one is the process agent state when it is holding the chosen role.

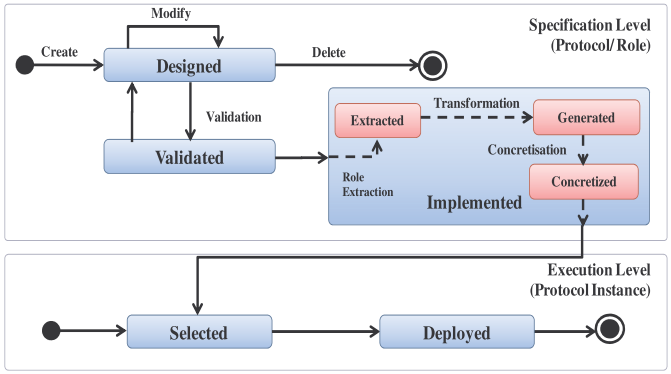

Fig. 9. Protocol Life Cycle in UML.

B. *Role Extraction, Generation and Implementation*

Role implementation is based on the OWL protocol specification obtained after instantiation of the protocol ontology. However, this protocol specification is not executable. To make it executable, we adopt the principle illustrated in Fig. 10 following the protocol life cycle.

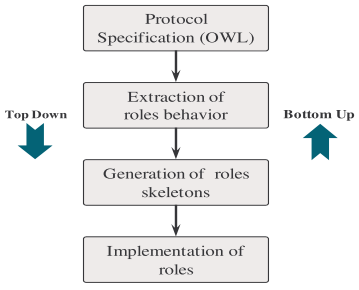

Fig. 10. Top-Down Approach for Role Implementation.

In this paper, we will only focus on the top-down approach to generate role behavior. The bottom-up approach is out of the scope of the paper. Extraction of role behavior from OWL protocol specification consists in performing a projection operation on roles. This operation, well known in relational database algebra, is to only keep control structure related to a

particular role (actions, messages, data, etc.). It is based on the analysis of conversation acts sent or received by a role, and variables shared and actions it performs. The result is a set of role skeleton specifications, described in XML and consistent with a model of roles.

We define a role in the same way as a protocol, i.e. as a composition of a profile and a behavior. The profile describes the properties, parameters and data variables of a role while the behavior defines authorized actions performed by a participant holding this role in a conversation. However, because of space limitation, we neither present the proposed algorithm to carry out the projection nor present the model of roles. The interested reader can consult [3] for more information about this algorithm and the role model.

To illustrate role extraction, we use the example of Iterative Contract-Net protocol presented in section 2. We applied the projection algorithm to extract manager, moderator and contractor roles. A partial view of the manager role is presented in Fig. 11 below. This role skeleton specification includes two parts. The first one describes the role profile through properties such as multiplicity, protocol name, referenced protocol ontology, list of parameters necessary for enacting the role and list of results it produces. The second part describes the role behavior in terms of the sequence of actions to be performed.

Manager role

```
<?xml version="1.0" encoding="UTF-8"?>
<xrdl:Role Id="GestionnaireCni" Name="GestionnaireCni"
                xmlns:xrdl="http://w3.univ-tlse1.fr/~/MonSchema.xsd"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:schemaLocation="http://w3.univ-tlse1.fr/~/MonSchema.xsd">
```

Profile

```
<xrdl:RoleProfile MultiParticipant="Yes" OntologyDomain="ProtocolOntolgy"
                  ProtocolName="IteratifContractNet"
                  ProtocolOntology="http://www.univ-tlse1.fr/~/ProOnto.owl">
        <xrdl:Description>Le r"le du Gestionnaire </xrdl:Description>
        <xrdl:ParticipantType Type="Initiateur"/>
    <xrdl:Parameters>
        <xrdl:Parameter Id="uneAnnonce" Mode="INPUT">
            <xrdl:Type Type="CallForProposal" />
        </xrdl:Parameter>
        ...
    </xrdl:Parameters>
</xrdl:RoleProfile>
```

Behavior

```
<xrdl:RoleBehavior>
  <xrdl:Actions>
    <xrdl:Action Final="No" Initial="Yes" Name="EnvoyerAnnonce"
          IdAction="ActionEnvoyerAnnonce" typeAction="JavaCode">
    </xrdl:Action>
....
  </xrdl:Actions>

  <xrdl:Transitions>
    <xrdl:Transition From="ActionEnvoyerAnnonce" To="ActionAttendreReponseAnnonce"
          IdTransition="Transition1">
          <xrdl:Description>Annoncé envoyée au modérateur</xrdl:Description>
    </xrdl:Transition>
...
  </xrdl:Transitions>
<xrdl:RoleVariables>
    <xrdl:Variable Name="TimeOutOrReponse">
      <xrdl:VariableType>boolean</xrdl:VariableType>
      <xrdl:DefaultValue>false</xrdl:DefaultValue>
      <xrdl:Description>Condition sur la réception d'une réponse</xrdl:Description>
    </xrdl:Variable>
...
  </xrdl:RoleVariables>
 </xrdl:RoleBehavior>
</xrdl:Role>
```

Fig. 11. Extract of manager role in ICN protocol.

The role generation step uses an XML role skeleton specification to generate the corresponding executable role skeleton specification. This specification depends on the chosen target platform. Several platforms are eligible. In our work, we used the WADE platform [25] and generated skeletons roles in Java (since Java is the language of WADE). Thus, the transformation process is an algorithm for mapping XML specifications (describing roles) to Java skeletons (implementing roles). XSLT is used to describe the mapping rules. The use of this language makes the transformation algorithm independent of any programming language.

More specifically, each role gives rise to the production of a Java class, each action of a role is implemented by a method of the class and each variable of a role is represented by a class attribute. Once generated, roles are then refined and concretized by developers of process agents in order to indicate the functional code of these actions (e.g. SendBid, MakeAdvertisement…). Refined roles are stored in the PMS library to be exploited by agents participating in conversations. A partial view of the Java class that defines the role Manager is presented below.

```
public class ImpGestionnaire extends RoleBehaviour
{
ANALYSEEVENEMENT_ACTIVITY = "AnalyseEvenement";
EVALUEROFFRES_ACTIVITY = "EvaluerOffres";
                                                    ...
@FormalParameter
private AID idModerateur;
private boolean timeOutOrReception;
                                                    ...
  private void defineActivities()
  { CodeExecutionBehaviour envoieAnnonceActivity =
new CodeExecutionBehaviour("SendsAdvertisement", this);
                                                    ...}
  private void defineTransitions()
  { registerTransition(new Transition(), " SendsAdvertisement ", "WaitResponse");
    registerTransition(new Transition(), "SendsSelectedOffers", "WaitConfirmation");
                                                    ...}
  protected void executeSendsAdvertisement () throws Exception {}
  protected void executeWaitResponse ()throws Exception {...}
  protected void executeSendsSelectedOffers () throws Exception {...}
}
```

Fig. 12. Extract of the Java class implementing the role Manager

C. *Role Execution*

This section introduces our strategy for role integration and execution at run time. It presents the Micro-Role (MR) engine component we defined for loading and executing role behavior. It also explains how a process agent holding a role drive the execution of the MR engine.

1) Micro Role Engine. To hold a role, a process agent needs to load and execute the extracted behavior at run time. Load means instantiate the role behavior according to the internal state of the agent (with it specific settings or parameters), and execute means enact it on the fly.

In order to support this, we extend the traditional agent behavior adding a specific component, called Micro-Role (MR) engine, responsible for loading and executing extracted and generated roles. When created, an agent is equipped with this specific component. It runs its MR engine only when participating in a conversation ruled by a specific protocol. Thus, we distinguish the internal behavior of the agent from its external behavior, which corresponds to the actions the agent has to execute when it holds a role in a conversation. The agent is itself responsible for the execution of its internal behavior (i.e. what it has to do), while the MR engine it integrates is responsible for the execution of its external behavior (i.e. the actions it has to execute within the role it holds). The connection between the internal behavior and the role behavior monitored by the MR engine is supported through communication between agent local behavior and the MR engine it integrates. This communication permits to give values to the different variables defined in the role profile. Fig. 13 below illustrates the architecture of an agent integrating a MR engine.

To go further into details, the MR engine is a small component which is able to execute generated role behavior. More precisely, for each role behavior, we have generated a corresponding Java class that exactly implements the behavior defined in the role. Thus, the MR engine is able to load, using an ad hoc class loader, a compiled Java class and to execute it.

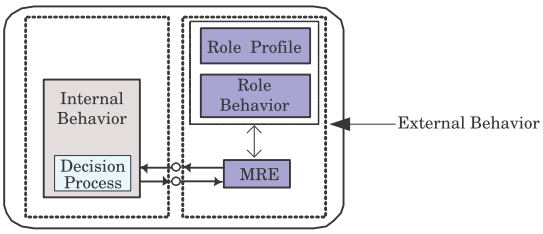

Fig. 13. Agent with Micro Role Engine.

So, when a process agent integrates a new role, the MR engine initializes the different variables of the role (role profile), reads the role specification, triggers the actions to be done and waits for incoming messages from others agents involved in the protocol. The actions triggered by the MR engine correspond to messages sent to the other participants of the conversation. When receiving a message, the MR engine reads the role specification and triggers the actions to be done according to its current state. Then, either it waits for new messages or it ends its participation in the protocol.

2) Driving Role Execution. It is also important to discuss about driving role execution, highlighting the communication between the internal behavior of the agent and its MR engine. We distinguish three objectives for driving role execution: (i) binding values to variables corresponding to those defined in the role profile, (ii) decision-making, which corresponds to the agent strategy in the way it holds the role, and (iii) supervision of the MR engine execution. The first objective is mandatory for a minimal execution of a role, while the others are required for a more advanced execution of a role.

As indicated before, the interface between the internal behavior of the agent and the MR engine ensures the communication between the agent and its MR engine providing services. This interface is the support for driving role execution. We now discuss in details how to use this interface to reach the three driving role execution objectives.

First, two services are provided to support the binding of values introduced before: the RequestVariables(parameters) service and the ProvideValues() service, where parameters correspond to the values defined in the role profile. RequestVariables will be used by the agent to receive a request from the MR engine while ProvideValues provides the MR engine with values given by the agent to the parameters. For instance, an agent holding the role Manager in the Iterative Contract Net protocol defines the minimal number of available contractor agent's to start a conversation.

Second, we propose two other services to support both decision making and role execution supervision: RequestDecision() and ProvideDecision(). Regarding decision-making, Request-Decision() corresponds to a request from the MR engine when alternatives occur in role execution, while ProvideDecision() is used to answer to this request. Of course, to be able to answer to a request, we consider that agents have a set of basic abilities. For instance, an agent holding the role Manager in the Iterative Contract Net protocol is able to evaluate a bid, to compare bids... The service ProvideDecision may be used when the agent has to make decides. This corresponds to strategic aspects in the way of holding a role for an agent. In order to help agents in their decisions, we have introduced a Process Decision ontology. Such an ontology defines a set of strategies for each role embedded in coordination protocols. Strategies are represented as sets of rules. An agent that has previously defined as being able to interpret a strategy described in the ontology is then able to load and execute any coordination protocol that is defined in concordance with this ontology. The presentation of this ontology is out of the scope of the paper.

Regarding the supervision of the MR engine execution, the ProvideDecision() service is used by agents to suspend, follow-up or stop the execution of the role.

To sum up, we do not impose any constraints on the type of agents holding roles but just assume they ensure the listed services to be able to drive role execution. The type of these agents can differ according the application in which they are involved. For instance, in the context of a dynamic IOW, process agents must integrate, in addition to the MR engine, a workflow engine in order to execute their local processes, which are a part of the inter-organizational process.

V. Implementing Dynamic IOW Coordination Using Protocols

This section first addresses the PMS implementation before focusing on the implementation of protocol-based coordination in dynamic IOW.

A. PMS Implementation

We first discuss of the technical choices we did for the PMS implementation and then give its technical architecture.

1) Technical choices. To implement the PMS server, we have chosen JADE (Java Agent DEvelopment Framework) [27], integrated in a JEE architecture. This platform eases the implementation of multi-agent applications in compliance with FIPA specifications (http://www.fipa.org). Morever, process-agent are implemented using WADE (Workflow Agent Development Environment), built on top of JADE. Indeed, WADE, unlike JADE, allows the definition of agents able to execute workflow processes on the fly. It also offers a set of mechanisms to handle the complexity of administration and fault tolerance operations in a decentralized and distributed environment. In WADE, each workflow agent is equipped with a set of workflow abilities and the main duty of an agent is to enact its proper workflow depending on the dynamic situations it faces.

2) PMS technical architecture. According to the technology discussed above, we propose the technical architecture of PMS given in Fig. 14 below.

In this architecture; data layer includes following data sources: conversation database, protocol ontology, domain (context) ontology and roles skeletons models, which are used

by process agents. The conversation database plays an important role in the system. It saves and maintains not only the runtime data of different conversations, but also the data of every connected process agent involved in the IOW.

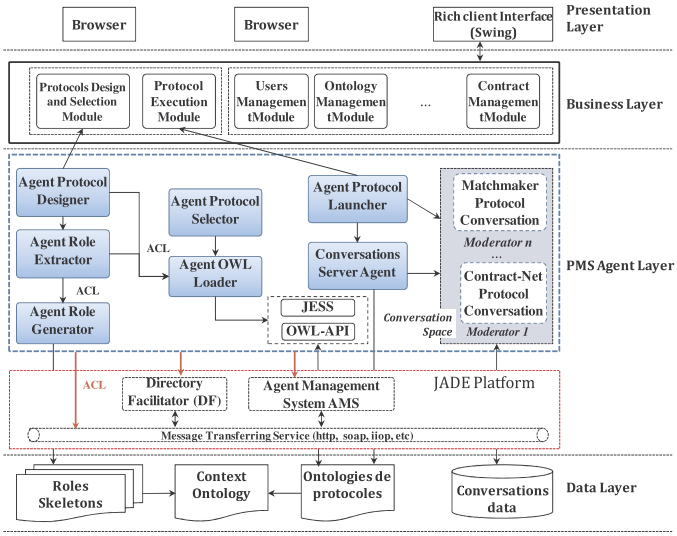

Fig. 14. Agent with Micro Role Engine.

PMS agent layer consists of seven agents implementing different services. These services are:

- The Designer, Extractor and Generator agents are used at the design level. The Designer agent helps to define new protocols from the protocol ontology. The Extractor agent is used to produce the XML role specifications of the defined protocols, while the Generator agent generates Java executable role specifications. This transformation is carried out using XSLT (eXtensible Stylesheet Language Transformations).
- The Launcher, Selector, OWLLoader and Conversations server agents, are used at the execution level to select and execute protocols, or to subscribe to new conversations. The Launcher helps process agents to instantiate a new conversation while the Selector agent handles process agent requests for protocol selection. The Conversations server agent makes information about conversations (current or past) accessible to agents connected to the PMS. Finally, the OWLLoader agent loads ontology protocol.

The communication between PMS agents is based on FIPA ACL messages. ACL is a special kind of messages transport format, which clearly expresses the intention of interactions between agents, but also describes in detail the content of interactions. At the same time, these agents can interact with remote process agents via HTTP, IIOP and SOAP protocols.

The Agent Management System (AMS) and the Directory Facilitator (DF) are two JADE system agents. The former one (AMS), assigns an ID to each PMS agent and performs basic operations such as creating and deleting an agent, modifying the agent's description, and monitoring agents migrating among different platforms. The latter (DF), provides yellow pages query service for all PMS agents. Every agent on the PMS has to publish information via DF, such as its name, its address, the services it provides...

The business layer is divided into two modules, the protocol management module and the administration support module. The protocol management module is responsible for providing interfaces for protocol design and selection. It is also responsible for supporting protocol execution and monitoring. The administration support module handles the management of users, of connected clients (process agent), and of data sources. The whole business layer is deployed in the EJB (Enterprise Java Bean) container; it provides data persistence and transaction management for the system, and provides security and stability.

The presentation layer uses JSP, Java Servlet and HTML technologies to dynamically produce web pages on the browser, to ease the interaction between the system and the users. We also developed a rich client interface for protocol design and protocol enactment simulation.

B. Protocol-based Coordination Implementation

In order to illustrate autonomic coordination in dynamic IOW implementation using the PMS, we discuss about the implementation of the autonomic dimension of process agents, and their dynamic coordination dimension following a coordination service process. We also report about the implementation of the dynamic IOW application presented in section 2.

1) Autonomic Coordination Dimension. Autonomic coordination means that the process agents are able to identify by themselves when they need dynamic coordination. This ientification occurs during process agent execution; it cannot be defined at design-time. There are two main situations that lead to this dynamic coordination:

- Presence of subcontracting activities in the process (e.g. Repair activity in the running example of section 2), scheduled to be executed by external partners whose identity is unknown at design-time.
- Unavailability or defection of partners at run-time. These failures can affect both human actors and physical resources, and can be, for instance, due to a decrease in the quality of service, a workload of an existing partner, or the discovery of a new partner offering a better quality of service.

The supervision of the execution process is needed to identify the failure and/or the presence of unknown subcontracting activities realized at runtime.

2) Dynamic Coordination Dimension. When failure and/or presence of unknown subcontracting activities are identified, protocols are used as follows. The running process agent is first suspended, and an instance of what we call the coordination service process is launched.

This process, introduced in Fig. 15, and detailed in Fig. 16, executes the different coordination services supporting finding partners, negotiation, contracting and execution of requested process (see introduction). For each service of this process, a

protocol is selected from the set of protocols provided by the PMS and an instance of the selected protocol is launched to give rise to a new conversation. The conversation result will be used for the next service. Example of used protocols are *matchmaking* protocol for finding partners, *iterative contract net* protocol for negotiation between partners, *template-contract* protocol for contracting between partners. The result of the finding partners step (i.e. a set of potential partners able to provide the requested process) is the starting point of the negotiation step. In the same way, the result of the negotiation step (the chosen partner) is the starting point of the contracting step. The last step is the execution of the subcontracted business process (see Fig. 15, BPMN middle pool).

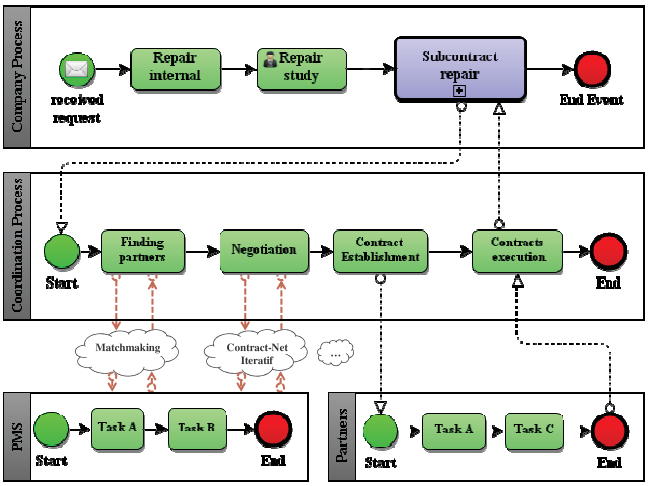

Fig. 15. Using Protocols for Coordination.

Fig. 15 below shows how to use protocols for the *Subcontract repair* activity, which is an unknown subcontracting activity of the process agent representing the organization responsible for the reparation of the equipment. Thus, this process agent must find a partner able to execute the repair activity. The execution of the process agent is suspended and the coordination service process (see Fig. 15, BPMN middle pool) is launched according to the following principle. For each coordination step (finding partners, negotiation, specification or contract and execution), there is an interaction with the PMS (see flow messages exchanged between the coordination service process and the PMS) and new conversations are opened for each of these steps. As indicated before, the result of the first three steps of the coordination service process is the selected partner with which partnership will be established. The final step is the execution of the requested service, i.e. the reparation of the damaged equipment (see messages flow between the activity execution of contracts coordination process and initial and final events of the repair process of the chosen partner).

The coordination service process presented above is implemented using the platform WADE. Fig. 16 gives a detailed view of this process showing how its activities are gradually refined according to the step in which it is located.

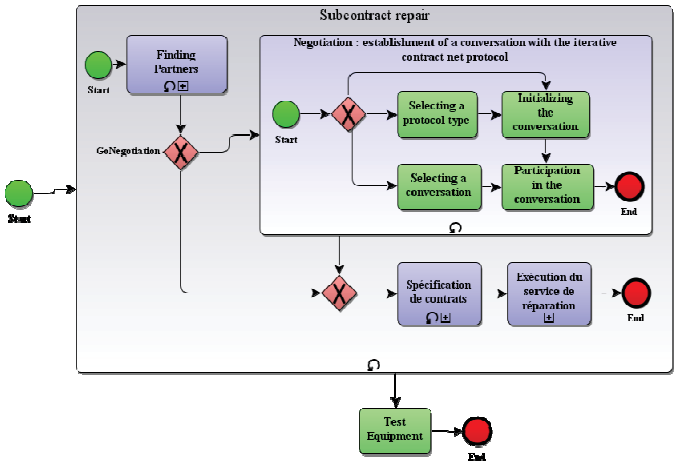

Fig. 16. Implementation of the Coordination Service Process

To illustrate how the coordination service process runs, we take an example that only focuses on the negotiation step. This step is described as a sub process that expresses three different scenarios for the implementation of the negotiation protocol: i) whether the protocol to be deployed is known, ii) whether the protocol is unknown and the process agent aiming at repairing the equipment looks for a specific protocol (activity Selecting protocol type, in the BPMN schema of Fig. 16) and then initiates a conversation, or, iii) the repairing company process agent is looking for an existing conversation. Depending on the scenario from previous alternatives, the repairing company process agent will therefore directly open a conversation with for example the Iterative Contract Net protocol, or choose a protocol by sending a selection request to the PMS as explained in [3].

3) Implementation of the Example. We illustrate here the implementation of the dynamic IOW application presented in section 2.

To implement this inter-organizational process, we have created WADE agents for each local process involved in the IOW. More precisely, we defined the following WADE agents: *ClientAgent* that implements the process of a client, *CompanyAgent* that implements the process of the company and as many *SubContractingAgents* as partners involved in equipments reparation. The CompanyAgent uses other internal agents which are not detailed in the paper. In addition, to be able to execute process, the CompanyAgent and the different SubContractingAgents are equipped with the Micro Role engine in order to be able to execute roles; they also integrate primitives to communicate with the PMS. Finally, the CompanyAgent also includes supervision to support autonomic coordination in order to identify situations of dynamic coordination (e.g. for dealing with failure situations).

As shown in Fig. 17, these agents run in distributed mode on several real machines. They are identified by their name and IP address (Internet Protocol). Note that the Inter-Organizational Workflow itself is distributed: each agent encapsulates a part of this process.

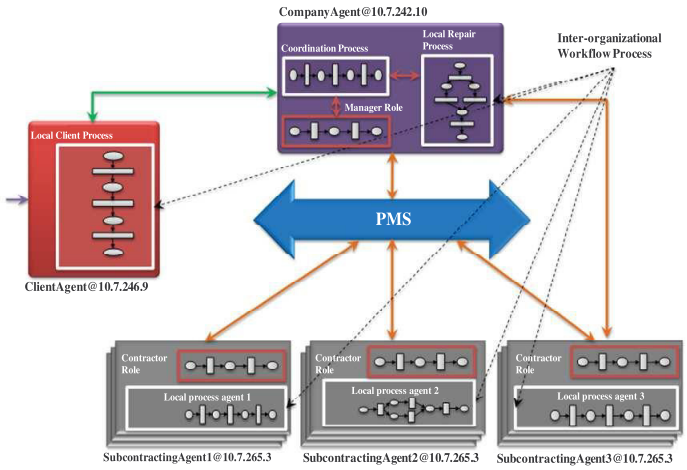

Fig. 17. Agents Implementing the Dynamic IOW Example.

VI. DISCUSSION AND CONCLUSION

This paper has addressed the issue of autonomic coordination in open and flexible business processes environments such as dynamic IOW. We adopt an agent-based approach to deal with this issue: (i) coordination of processes involved in dynamic IOW is protocol-based, and (ii)•dynamic IOW processes are encapsulated into agents, called Process Agents (PA). Our work is based on the assumption that multi-agent coordination protocol support coordination services which are inherent to dynamic IOW, i.e. finding partners, negotiation between partners, and contracting services. It is also based on the assumption that encapsulating processes into process agents gives them the ability to autonomously decide with whom, when and how to cooperate.

In our approach coordination protocols are isolated from processes involved in dynamic IOW and their management is entrusted to a Protocol Management System (PMS), which can be seen as a server of protocols. In an engineering perspective, we defined a life cycle for protocols and specified services ensuring their specification, selection and execution. We also illustrated how process agents involved in a dynamic IOW could coordinate using these coordination protocols.

Related works may be considered according to two complementary points of view: the IOW coordination and the protocol points of view.

Regarding the IOW coordination point of view, the main works are [10,12,13,15,16,28,29]. All these works provide middleware-based solutions to deal with one of the following coordination services: finding partners or negotiation between partners. These works also exploit the agent approach as an enabling technology to both model coordination and implement flexible and adaptive processes. Regarding finding partners, [10,12] define a matchmaker to find and coordinate agents implementing processes. [13,29] both mix Web services and agents to implement flexible processes running on the web. While [13] implements a matchmaker for finding partners, [29] implements a broker. [15] is the only work dealing with negotiation of processes. It defines an agent-based architecture including a moderator implementing a coordination protocol, and a conversation server recording information about open negotiations. Unfortunately, none of these works adopt a comprehensive approach to deal with all the coordination services in a coherent and uniform framework. They also lack an engineering perspective to deal with this coordination issue.

Regarding the protocol point of view, we highlight four main works [22,24,30,31]. First of all, [22] defines an ontology to support negotiation in E-Commerce thanks to a conceptual model describing the general concepts of E-commerce negotiation protocols, and suggests to use the Protocol Specification Language (PSL) to specify the behavior of these protocols. Second, [24] defines a conceptual model for protocols using a declarative approach and shows how to transform modeled protocols onto corresponding Petri Nets, thus obtaining an executable specification. Unfortunately, these two works neither identify and address IOW protocols, nor provide solutions to the dynamic selection of protocols. Moreover, [22] does not show how to specify the behavior of a protocol using PSL while [24] does not address the classification and selection of protocol issues. Third, [30] defines an ontology of protocols devoted to business processes, and address their coordination through their composition. It also explains how to compile them into executable rules. However, it does not address the selection issue and does not provide a protocol classification useful to help process partners in selecting the appropriate protocol according to the execution context. Finally, [31] is a complementary work to ours. It deals with protocol engineering issues focusing particularly on the notion of protocol compatibility, equivalence and replaceability. Actually, this works aims at defining a protocol algebra which can be very useful to our PMS. At design time, it could be used to establish links between protocols, while at run-time, these links could be used by the PMS selection service to propose set of equivalent protocols.

The originality of our proposal is based on two elements. The first one is the proposed approach that provides a coherent and unified protocol-based framework to deal with dynamic IOW coordination. This approach takes into account the integrity of the coordination process. Protocols are used as reusable components to handle each step of the coordination process. They are managed by a dedicated system called protocols management system, which provides engineering services covering the entire protocol life cycle. Another strong element of our work is the definition and implementation of a dynamic execution model of protocols for process agents to integrate at real-time the roles they are playing, without being stopped or reprogrammed. We believe that this contribution is important since supporting dynamic execution of roles permit agents to face new open and distributed modern MAS applications such as e-commerce, e-government, crisis management and web services conversations. It makes the participation to several conversations at the same time possible: agents can switch their role behavior at run-time without being shutdown, retooled and restarted.

Regarding future works, we plan to complete the implementation of the PMS and we will focus on the specification and the development of a set of connectors to allow Workflow Management Systems (SGWf) such as YAWL [32] or Bonita [33] to connect to our PMS. We also have to consider specific issues about loaded roles by process

agents. Are these loaded roles consistent with the internal behavior of the agent ? Are they consistent with each others (for instance, a process agent may load different roles if it is involved in different conversations). The issue has begun to be addressed in [34]; it needs to be revisited in our context.

Finally, protocols are essential components having an important place in any area where coordination or collaboration may be considered as a first class citizen. Therefore, even if our PMS is dynamic IOW-oriented, we believe that protocols are useful in other application domains. For example, they could be used to manage web services conversations.

REFERENCES

[1] W. van der Aalst, "Inter-Organizational Workflows: An Approach Based on Message Sequence Charts and Petri Nets". *Systems Analysis, Modeling and Simulation*, vol. 34, no. 3, 1999, pp. 335–367.

[2] M. Divitini, C. Hanachi and C. Sibertin-Blanc, "Inter Organizational Workflows for Enterprise Coordination". A. Omicini, F. Zambonelli, M. Klusch, and R. Tolksdorf (eds): *Coordination of Internet Agents*, 2001, pp. 46–77.

[3] W. Bouaziz, "Coordination à base de protocoles dans les systèmes multi-agents : application au Workflow Inter Organisationnel". PhD Dissertation, IRIT, Toulouse, September 2010.

[4] K. Baïna, K. Benali, and C. Godart, "Dynamic Interconnection of Heterogeneous Workflow Processes through Services". *International Conference On the Move to Meaningful Internet Systems*, CoopIS 2003, R. Meersman, Z. Tari, D. Schmidt (Eds.), Catania, Sicily, Italy, November 2003 pp. 444–461.

[5] B. Benatallah, F. Casati and F. Toumani, "Web Service Conversation Modeling : A Cornerstone for E-Business Automation". *IEEE Internet Computing*, vol. 8, no. 1, 2004, pp. 46–54.

[6] X. Zhao, C. Liu and Y. Yang, "Web Service Based Architecture for Workflow Management Systems". *International Conference on Database and Expert Systems Applications*, DEXA 2004, Zaragoza, Spain, September 2004, p. 34–43.

[7] I. Chebbi., S. Dustdar and S. Tata, "The view-based approach to dynamic inter-organizational workflow cooperation". *Data Knowledge Engineering*, vol. 56, no. 2, 2006, pp. 139–173.

[8] J. Meng, S. Su, H. Lam, H. Abdelsalam, X. Jingqi, L. Xiaoli and Y. Seok-Won, "DynaFlow: a Dynamic Inter-Organizational Workflow Management System". *Business Process Integration and Management*, vol. 1, n°2, 2006, pp. 101–115.

[9] S. Tata, K. Klai, and N. M'Bareck, "CoopFlow: a Bottom-Up Approach to Workflow Cooperation for Short-Term Virtual Enterprises". *IEEE Transactions on Services Computing,* vol. 1, no. 4, 2008, pp. 214–228.

[10] L. Zeng, A. Ngu, B. Benatallah and M. O'Dell, "An Agent-Based Approach for Supporting Cross-Enterprise Workflows". *Australian Database Conference*, ADC 2003, Bond, Australia, February 2001, pp. 123–130,.

[11] A. Ricci, A. Omicini, and E. Denti E., "Virtual Enterprises and Workflow Management as Agent Coordination Issues". *Cooperative Information Systems*, vol. 11, no. 3-4, 2002, p. 355–379.

[12] E. Andonoff, L. Bouzguenda, C. Hanachi and C. Sibertin-Blanc, "Finding Partners in the Coordination of Loose Inter-Organizational Workflow". *International Conference on the Design of Cooperative Systems*, COOP'04, Hyères, France, May 2004, pp. 147–162.

[13] C. Aberg, C. Lambrix and N. Shahmehri, "An Agent-Based Framework for Integrating Workflows and Web Services", *International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, WETICE'05, Linköping, Sweden, June 2005, pp. 27–32.

[14] T. Savarimuthu, M. Purvis, MK. Purvis and S. Cranefield, "Integrating Web Services with Agent Based Workflow Management System", *International Conference on Web Intelligence*, WI'05, Compiègne, France, September 2005, pp. 471–474.

[15] E. Andonoff and L. Bouzguenda, "Agent-Based Negotiation between Partners in Loose Inter-Organizational Workflow". *International Conference on Intelligent Agent Technology*, IAT'05, Compiègne, France, September 2005, pp. 619–625.

[16] P. Buhler and J. Vidal, "Towards Adaptive Workflow Enactment Using Multi Agent Systems". *Information Technology and Management*, vol. 6, no. 1, 2005, pp. 61–87.

[17] B. Blake and H. Gomaa, "Agent-Oriented Compositional Approaches to Services-based Cross Organizational Workflow". *Decision Support Systems*, vol. 40, no. 1, 2005, pp. 31–50.

[18] E. Andonoff, W. Bouaziz, C. Hanachi and L. Bouzguenda., "An Agent-based Model for Autonomic Coordination of Inter-Organizational Business Processes". *Informatica*, vol. 20, no. 3, September 2009, pp. 323–342.

[19] E. Andonoff., W. Bouaziz. and C. Hanachi, "Protocol Management Systems as a Middleware for Inter-Organizational Workflow Coordination". *IEEE International Conference on Research Challenge in Information Science*, RCIS 07, Ouarzazate, Morocco, April 2007, pp. 85–96.

[20] C. Ghezzi, M. Jazayeri and D. Mandrioli, "Fundamentals of Software Engineering". Prentice-Hall International, 1991.

[21] M. Wooldridge, "An Introduction to MultiAgent Systems". Whiley Editions, 2009.

[22] V. Tamma, S. Phelps, I. Dickinson and M. Wooldridge, "Ontologies for Supporting Negotiation in E-Commerce". *Engineering Applications of Artificial Intelligence*, vol. 18, no 2, 2005, pp. 223–236.

[23] B. Lithgow Smith, V. Tamma and M. Wooldridge, "An Ontology for Coordination". *Applied Artificial Intelligence*, vol. 25, no 3, 2011, pp. 235–265.

[24] C. Hanachi and C. Sibertin-Blanc, "Protocol Moderators as active Middle-Agents in Multi-Agent Systems". *Autonomous Agents and Multi-Agent Systems*, vol. 8, no. 3, March 2004, pp. 131–164.

[25] Foundation for Intelligent Physical Agents, FIPA ACL Message Structure Specification. December 2002, Available at http://www.fipa.org/specs/fipa00061/

[26] G. Caire, D. Gotta and M. Banzi, "WADE : a software platform to develop mission critical applications exploiting agents and workflows". *International Conference on Autonomous Agents and Multiagent Systems*, AAMAS 08, Industrial Track, Estoril, Portugal, May 2008, pp. 29–36.

[27] F. Bellifemine, G. Caire and D. Greenwood, "Developing Multi-Agent Systems With Jade". John Wiley & Sons Ltd, 2007.

[28] A. Negri, A. Poggi, M. Tamaiuolo and P., Turci, "Agents for e-Business Applications". *International. Conference on Autonomous Agents and Multi-Agent Systems*, AAMAS 04, Hokodate, Japan, May 2004, pp. 907–914.

[29] B. Blake, "Agent-Based Communication for Distributed Workflow Management using JINI Technologies". *Artificial Intelligence Tools*, vol. 12, no. 1, 2003, pp. 81–99.

[30] N. Desai., A. Mallya, A. Chopra and M. Singh, "Interaction Protocol as Design Abstractions for Business Processes". *Transactions on Software Engineering,* vol. 31, no. 12, 2005, pp. 1015–1027.

[31] B. Benatallah, F. Casati and F. Toumani, "Representing, Analyzing and Managing Web Service Protocols". *Data and Knowledge Engineering*, vol. 58, no. 3, 2006, pp. 327–357.

[32] W. Van der Aalst and A ter Hofstede, "YAWL : yet another workflow language". *Information Systems*, vol. 30, no. 4, 2005, pp. 245–275.

[33] D. Grigori, F. Charoy and C. Godart, "Coo-Flow: A Process Technology to Support Cooperative Processes". *Software Engineering and Knowledge Engineering*, vol. 14, no.1, 2004, pp. 61–78.

[34] A. Chopra, F. Dalpiaz, P. Giorgini, J. Mylopoulos. "Modeling and Reasoning about Service-Oriented Applications via Goals and Commitments". International Conference on Advanced Information Systems Engineering, CAiSE'10, Hammamet, Tunisia, June 2010, pp. 113–128.: