

Conducting The Agile Negotiation Processes Involved In The BPEL4WS Model On a Multi-agent Platform

Li Guo¹, David Robertson¹, Yun-Heh Chen-Burger¹ and Jianquan Wang²
CISA, Informatics, The university of
Edinburgh, United Kingdom
L.Guo@sms.ed.ac.uk, {Jessicac,dr}@inf.ed.ac.uk

Abstract

A key requirement for current e-Business is to build inter-operable e-Business processes for the emerging business models within different enterprise boundaries. Negotiation processes are at the core of the inter-operable e-Business. While research on negotiation is not new, the vast majority of the studies to date have been based on the multi-agent platform. However, it is very common that negotiation processes are interleaved with other business processes that are automated normally by a workflow system. Therefore, the inter-operability with other internal- and external- systems is critical in such case, which may require extra efforts on the integration issues.

In this paper, we propose an approach for building a workflow system with negotiation processes incorporating the BPEL4WS[8], a standard for building and managing web service based business processes, on a multi-agent platform in a pure distributed manner. By adapting our approach, negotiation process can be involved in the BPEL4WS process and deployed seamlessly. The existing work on negotiation based on multi-agent system also can be adapted directly with our approach.

Keyword: *Business Process Model, BPEL4WS, Web Services, Multi-agent System, Interaction Protocol, Automatic Negotiation*

1. Introduction

With the increases of customer-driven business marketplace in the open environment (internet), a key requirement for this circumstance is building inter-operable e-Business processes for the emerging business models within different enterprise boundaries. Composition of web services has received much interest as a mean of supporting Business-To-Business or enterprise application integration.

Currently, a main approach for the web services composition is a static workflow technology based approach, for example, BPEL4WS, which is the de facto standard for distributed workflow system using web services composition. Using such method, web services are described as activities/atomic activities in a business process model. A workflow engine is used to run the whole business process model, web services thus can be invoked as the business process executes.

Negotiation processes are at the core of the inter-operable e-Business. It is very common that negotiation processes are interleaved with other business processes that are automated normally by a workflow system. But as addressed in [4], the current web services standards like BPEL4WS do not allow for all the possible business negotiation processes. The vast majority of the researches to date have been based on the multi-agent platform and this is completely natural because negotiations often involve many parties, multiple issues and decision-making process is always required. Therefore, when executing a business process that involves the negotiation processes, the inter-operability with other internal- and external- systems (agent-based systems) is critical. Based on our previous work[1], an BPEL4WS specification can be used directly in a multi-agent system. Thus, the inter-operability problem that is addressed above can be eliminated since we don't have mixed system problem at all.

In this paper, we propose an approach for building a workflow system with negotiation processes based on a multi-agent system in a pure distributed manner. The process model that is to be used in the intended system is depicted by BPEL4WS. The negotiation process is changeable (negotiation strategy can be changed at any time when users want to/requirements change) with our approach because it is actually controlled by the LCC[3] negotiation protocols rather than by BPEL4WS model directly. LCC as a multi-agent interaction protocol language is more expressive and

is more natural for negotiation purpose.

The paper is organised as follows. In the next section, some of the necessary background knowledge for understanding our work is introduced. Then, in section 3, we describe our framework in detail for implementing one-to-one/one-to-many negotiations involved a BPEL4WS model including: the basic architecture of our system; the necessary extension of BPEL4WS for negotiation purpose. The implementation of some of the underlying LCC[3] negotiation protocols for different negotiation strategies is addressed in section 4. We state the major conclusions of our work and outline future research in section 5.

2. Background

2.1. Lightweight Coordination Calculus (LCC)

The Lightweight Coordination Calculus(LCC)[3] is a language for representing coordination between distributed agents. In a multi-agent system the speech acts conveying information between agents are performed only by sending and receiving messages. For example, suppose a dialogue allows an agent $a(r1,a1)$ to send a message $m1$ to agent $a(r2,a2)$ and agent $a(r2,a2)$ is expected to reply with message $m2$. Assuming each agent operates sequentially, the sets of possible dialogue sequences we wish to allow for the two agents in the example are as given below, where $M1 \Rightarrow A1$ denotes a message, $M1$, send to $A1$, and $M2 \Leftarrow A2$ denotes a message, $M2$, received from $A2$.

$$\begin{aligned} a(r1, a1) &:: (m1 \Rightarrow a(r2, a2) \text{ then } m2 \Leftarrow a(r2, a2)) \\ a(r2, a2) &:: (m1 \Leftarrow a(r1, a1) \text{ then } m2 \Rightarrow a(r1, a1)) \end{aligned}$$

Any agent can change its role according to the definition of the dialogue:

$$\begin{aligned} a(r1, a1) &:: m1 \Rightarrow a(r2, a2) \text{ then } a(r3, a1) \\ a(r3, ID) &:: m2 \Rightarrow a(r4, a3) \text{ then } m3 \Leftarrow a(r4, a3) \end{aligned}$$

The above clause means that agent $a1$ takes the role of $r1$ initially and after sending a message $m1$ to agent $a(r2,a2)$, it changes its role to $r3$ and then takes the appropriate behaviours that are defined for $a(r3,ID)$. This capability of LCC is very important for the our work described in this paper.

We refer to this definition of the message passing behavior of the dialogue as the *dialogue framework*. Its complete syntax can be found in [3]. A dialogue framework defines a space of possible dialogues determined by message passing, so the protocols allow constraints to be specified on the circumstances under which messages are sent or received. Two forms of constraints are permitted:

- Constraints under which message, M , is allowed to be sent to agent A . We write $M \Rightarrow A \leftarrow C$ to attach a constraint C to an output message.

- Constraints under which message, M , is allowed to be received to agent A . We write $M \Leftarrow A \leftarrow C$ to attach a constraint C to an input message.

For the earlier example above, to constrain agent $a(r1,a1)$ to send message $m1$ to agent $a(r2,a2)$ when condition $c1$ holds in $a(r1,a1)$ we could write: $m1 \Rightarrow a(r2,a2) \leftarrow c1$.

Agent dialogue may also assume *common knowledge*, either as an inherent part of the dialogue or generated by agents in the course of a dialogue. This knowledge could be expressed in any form, as long as it can be understood by appropriate agents. We recognise the importance of preserving a shared understanding of knowledge between agents but cannot cover this issue in the current paper. As a dialogue protocol is shared among a group of agents it is essential that each agent when presented with a message from that protocol can retrieve the *state* of the dialogue relevant to it and to that message [3].

Pulling all the above elements together, we describe a LCC dialogue protocol as the term:

$$protocol(S, F, K)$$

Where S is the dialogue state; F is the dialogue framework(sets of dialogue clauses); and K is a set of axioms defining common knowledge assumed among the agents.

2.2. A Multi-agent Platform For Distributed Business Workflow Based on BPEL4WS

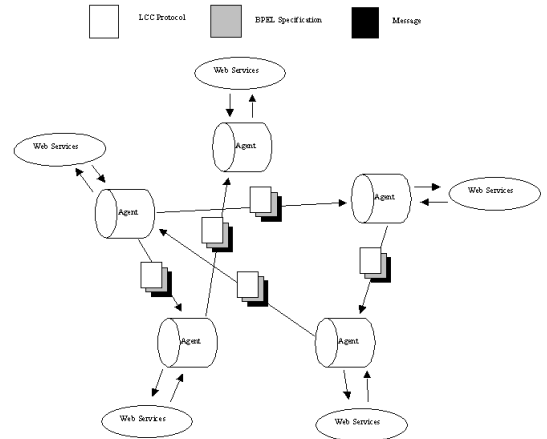


Figure 1: The infrastructure of our generic MAS platform

In the infrastructure, the LCC protocol acts as a BPEL4WS interpreter. The BPEL4WS specification and the LCC protocol (BPEL4WS interpreter) are then passed together between the agents to enable their coordination. The LCC protocol can be understood as an interpreter that is able to interpret all the BPEL4WS syntaxes, and thus can be used to process the BPEL4WS

specification. Based on this idea, a BPEL4WS specification that is defined in any fashion can be interpreted neatly by the LCC protocol when they are passed together in the multi-agent system. The infrastructure of the system based on this approach is given in figure 1. In this system, the multi-agent interaction protocol, the BPEL4WS specification and the messages are packed and passed together between the agents. Once an agent receives the package, it processes: the incoming message (takes appropriate behaviors), interaction protocol and BPEL4WS (resolves the next action it needs to take) that are contained in the package, then it sends out a new package to next agent to make the coordination continue. Further details of our system can be found in [1].

3. The MAS Based Architecture For Implementing the BPEL4WS Depicted Negotiation Process Model

A BPEL4WS model that involves negotiation behaviours can be used on our existing multi-agent platform directly without interacting with any inter- or external- non-multi-agent paradigm based system. The architecture of it is illustrated below in figure 2: In this architecture, the tasks that

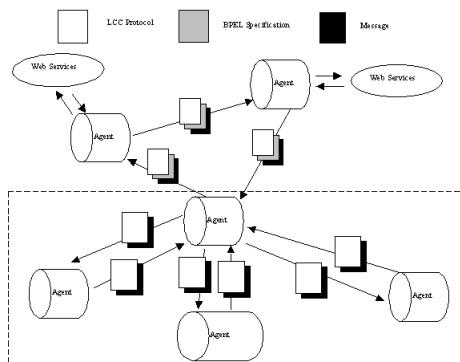


Figure 2: The MAS Based Architecture For Implementing the Negotiation Process Model

are defined in the BPEL4WS specification are performed by a group of agent using required web services. The agents that are in the area of dashed square represent the interaction that takes place for the negotiation activity defined in the BPEL4WS specification.

Although BPEL4WS provides a rich set of primitives to specify Web Service compositions, it does not support multiple instantiation. There is a clear need to invent a such activity that are executed multiple times within the same process instance without knowing the number of parallel executions a priori. This is especially the case for inter-organisational negotiation processes that often include 1 : n interactions.

3.1. Extending BPEL4WS for Negotiation

Typically, a negotiation process can be divided into two parts (see e.g. [7]) as the example of an online auction process illustrates:

- **One-to-many phase:** A set of potential partners is created . In an auction process each bidder can be regarded as a potential business partner. The bidder with the best offer is chosen as the partner for further interaction.
- **Bilateral phase:** The offerer and the auction winner continue the process in a bilateral way. The winner receives a bill, and the offerer initiates the shipment.

In essence, BPEL4WS defines conversational relationships between two parties via a so called *partnerLink* that links one internal party to one corresponding external party. In order to allow for negotiation process, the following minimum issues have to be declared in a negotiation activity:

- **Array of External Parties:** In a negotiation activity different partners may act in the same role. The respective *partnerLink* should include an attribute to indicate such capabilities.
- **Sets of Negotiation Issues:** The input variable of a negotiation activity should be sets of issues that needed to be negotiated.
- **Negotiation Strategy:** The negotiation strategy decides how the negotiation process can be carried, for example, in some cases time constraint is highly emphasised.

Based on the above issues, the extended negotiation activity for BPEL4WS has the basic syntax as follows:

```
< negotiation partnerLink = "ncname" portType = "qname"
  NegotiationStrategy = "ncname" inputVariable = "ncname"
  outputVariable = "ncname"
  standard-attributes >
  standard-elements
< /negotiation >
```

The definition for *partnerLink* needs be revised gently in the following form:

```
< partnerLink name = "ncname" partnerLinkType = "qname"
  myRole = "ncname"? partnerRole = "ncname"?
  multiple = "yes/no" > +
< /partnerLink >
```

If the attribute *multiple* is set to yes, the *partnerLink* represents a one-to-many relationship.

The negotiation activity invented is a pure abstract activity, which means it doesn't define any semantic of the negotiating behaviours. All the concrete negotiation process, like how the partners interact with each other, is carried out by the underlying LCC protocol. The reason for

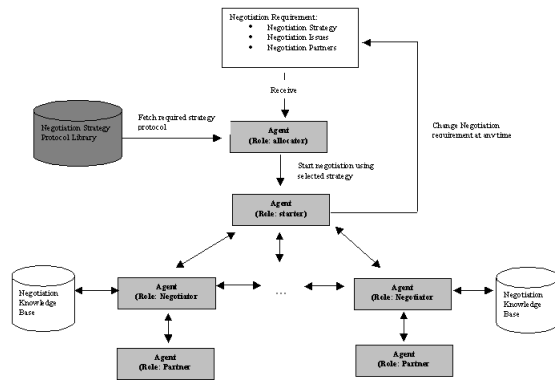


Figure 3: The Agile Negotiating Framework

us to choose LCC protocol for representing the negotiation process rather than using BPEL4WS provided activities directly is because: 1>the limited expressing power of the current version BPEL4WS[4] and the relative stable nature of LCC (first-order logic based); 2>when representing a negotiation process, it is easier to model from the view of individual agent rather than from the view of process.

4. The Agile Negotiation Framework

Figure 3 shows a framework that illustrates the basic negotiation architecture of our system, which is based on the one-to-many negotiation structure proposed by Iyad Rahwan[6]. With this framework, during the process of one-to-many negotiation, an agent can negotiate with many other agents by creating a number of one-to-one negotiating agents negotiate on its behalf.

The components in the framework are:

- **Negotiation Requirements:** stores all the negotiation related information and can be updated by internal process model of an organisation at a time. It is composed of three parts:
 - **Negotiation Issues:** defines a set of intended issues that are going to be negotiated and the numbers and contents of the issues can be changed during the negotiation process.
 - **Negotiation Partners:** defines a set of partners that we are going to negotiate with.
 - **Negotiation Strategy:** defines the negotiation strategy that is going to be used in for controlling the negotiation process.
- **Negotiation Strategy Protocol Library:** According to different negotiation strategies, we need different negotiation protocols for controlling the varied sequences of conversions between agents since there is no centralised message control server in our system. Furthermore, because the negotiation strategies can be changed by end users at any time (before/during/after

the process of negotiation), the negotiation protocol has to be as agile as possible to fit this feature. We developed an extendable LCC protocol library, which contains sets of agile LCC negotiation protocols based on different negotiation strategies. There are two levels of negotiation strategies, namely strategies exercised by individual negotiating agent and their partners in their one-to-one encounter, and strategies exercised by the initial agent in organising and issuing commands to their negotiators[6].

- **Agents:** Four types of agents are defined:

- **allocator:** is used to decide which LCC negotiation protocol can be used according to the given negotiation strategy; fetch and initiate the appropriate *starter* according to the potential strategy and the number of the partners.
- **starter/starter....:** starts the negotiation using appropriate negotiation strategy and also responsible for collecting the negotiation result/terminating the negotiation process.
- **negotiator:** is the real agent that negotiates with the partner on certain negotiation issues.
- **partner:** is the business partner that we negotiate with.

- **Negotiation Knowledge Base:** stores the business related information and is used for evaluating the negotiation issues.

4.1. Different Negotiation Strategies

Negotiation strategies of individual negotiators in our protocol library is assumed to be requestor-driven satisfactory deal strategies. With this strategy, the negotiator keeps negotiating with its partners until it receives an satisfactory offer. Six roles are defined in the for the negotiation protocol:

- **starter:** is a coordinating agent that control the coordination between different negotiators based on different negotiation strategies. For one-to-one negotiation, since there is only one negotiator required, the behaviours defined for starter are simply used to: inform a negotiator to start a negotiation; terminate the running negotiation at any time when the user wants to change their negotiation preferences (strategies, negotiation issues etc.).
- **collector:** is used to collect the final offer from the negotiator.
- **negotiator:** is the actual agent that negotiate with the business partners on certain negotiation issues. Once it receives a start message (including negotiation issues,

negotiation partner of it) from *starter*, it starts the negotiation, gets the negotiation results and sends it to *starter*.

- **negotiator_it**: is defined for representing the iterative negotiation process.
- **terminator**: is used to inform all the negotiator to terminate the negotiation with its partner (in SB_1).
- **Partner**: represents the real business partners that are extracted from $\langle partnerLink \rangle$ defined in BPEL4WS specification.

All the LCC protocols defined for these roles are used as the basic negotiation protocol components in the protocol library except for *starter* and *collector*, which vary for different negotiation strategies. For simplicity, we will ignore them in the following parts.

A few simple coordination strategies that can be exercised by the initial agent for controlling negotiators are outlined in the following subsections.

4.1.1. Desperate Strategy This is a very simple strategy in which the time constraints may be important and the agent wants to close a deal fast. In this strategy, as soon as a negotiator finds an acceptable offer from a partner, it accepts it and sends messages to all the other partners to terminate their negotiation.

4.1.2. Patient Strategy: In this strategy, even if an acceptable deal is offered by one or more partners, those agents are asked to wait while all other agents are asked to resume their negotiations. Once all partners complete their negotiation process (whether with success or failure), the best offer is chosen. This strategy guarantees that the best possible deal can be reached, but does not give regard to time constraints. This might be a significant limitation in a marketplace with too many potential suppliers to negotiate with. One variation of the patient strategy is one in which a time limit is set by the user, within which if no better deal was found, the negotiation terminates and the best deal so far wins.

5. Conclusion and Future Work

In this paper, we proposed a web service and multi-agent system based negotiation system, which supports agile but complex negotiation processes involved in a BPEL4WS process. By adapting our approach, an agile negotiation process can be conducted during the execution of a BPEL4WS model without any inter-operation problem.

Currently, the negotiation protocol has to be produced manually. We are going to automate this protocol production process (generate LCC protocol automatically) based on the business rational of participants of the negotiation.

References

- [1] L.Guo, D. Robertson, Y. Chen-Burger *Enacting the Distributed Business Workflows Using BPEL4WS on the Multi-Agent Platform*. The proceedings of the MATES 2005 conference.
- [2] J. Shen, Y. Yang, J. Yan. *Adapting P2P based Decentralised Workflow System SwinDeW-S with Web Service Profile Support*, the 9th International Conference on Computer Supported Cooperative Work in Design (CSCWD 2005), Coventry, UK, May 2005.
- [3] D. Roberston, *A Lightweight Method for Coordination of Agent Oriented Web Services*, Proceedings of AAAI Spring Symposium on Semantic Web Services, 2004.
- [4] J Kim, A Segev, *Framework for Dynamic eBusiness Negotiation Processes*. The proceedings of IEEE Conference on E-Commerce, 2003
- [5] J.Yan, Y. Yang and G. K. Raikundalia, *Enacting Business Processes in a Decentralised Environment with p2p-Based Workflow Support* Lecture Notes in Computer Science Publisher: Springer-Verlag GmbH ISSN: 0302-9743 Subject: Computer Science Volume 2762 / 2003
- [6] I Rahwan, R Kowalczyk, HH Pham, *Intelligent agents for automated one-to-many e-commerce negotiation*. The proceedings of Twenty-Fifth Australian Computer Science Conference, 2002
- [7] Y. Bakos, *The Emerging Role of Electronic Marketplaces on the Internet*. Communications of the ACM. 41(8):35.42. 1998.
- [8] *Business Process Execution Language For Web Services specification*, <http://www-128.ibm.com/developerworks/library/ws-bpel/>.
- [9] *OWL-S 1.0 Release*. <http://www.daml.org/services/owl-s/1.0/>