AN APPLICATION OF AN EXPRESSIVE COLOURED PETRI NETS MODELING METHODOLOGY TO A BUSINESS TO BUSINESS ENVIRONMENT

JUAN FRAUSTO-SOLIS[1], FRANCISCO CAMARGO-SANTACRUZ[2] and FERNANDO RAMOS-QUINTANA[1]

Instituto Tecnológico y de Estudios Superiores de Monterrey, [1]Campus Cuernavaca, [2]Campus Estado de México, Km. 3.5, Carretera al Lago de Guadalupe, Atizapán, 52926, Estado de México, México, phone ++52-5864 5560, fax ++52-5864 5557, http://www.itesm.mx.

E-mail: [1]{framos, jfrausto} @campus.mor.itesm.mx, [2]fcamargo@campus.cem.itesm.mx

The dynamic nature of cooperative agent environment makes considerably more difficult the task of modeling permanent interactions among agents. This problem becomes rather hard if more than two agents are involved. So far, traditional approaches deal with the problem of modeling interactions in static conditions and commonly with only two agents participating concurrently in cooperative tasks. The complex nature of such dynamic environments, such as e-business, demands to build adequate tools to manage multiple interactions with efficient expressiveness. This problem remains one of the most important challenges in cooperative multi-agents system research. In this paper, it is proposed an application of an efficient methodology based on Coloured Petri Nets to model multiple interactions, which takes into account the expressiveness as its most important property. The model was tested within a business to business (B2B) environment where concurrent interactions among buyers, suppliers and the marketplace constitute a dynamic process that need to be permanently monitored and controlled. This methodology provides great advantages in the representation and reasoning for the interaction mechanism modeled in cooperative information systems. The methodology integrates mainly: a) the action basic loop in order to represent the system interactions and to model organization conversations, b) the use of CPN for the interaction modeling and system simulation, c) the communicative acts of FIPA (Foundation for Intelligent Physical Agents), included in the Agent Communication Language Specification.

Keywords: Agent-Oriented Software Engineering, Cooperative Information Systems, Coloured Petri Nets, Multi-Agent Systems, Business to Business.

1. Interaction and Cooperative Information Systems

A Cooperative Information System (CIS) supports daily activities in the organization. It is a cooperative multi-agent system [24] integrated by a set of agents, data, and procedures, working in a cooperative way. They have a common goal, exchange information, and work together in order to achieve the objective [22]. The business to business systems are considered CIS and they present a set of particular characteristics, by their dynamic and interactive nature that required to be modeled in an expressive style.

To model a CIS in dynamic environments is a complex task and so the system engineer needs adequate tools in order to appropriately manage the process of modeling, in particular in this paper, the interactions among agents during the whole of information exchange. Static modeling approaches are insufficient for representing system behavior over time. The reason is that they provide no way to represent how the system's state will change as time passes. Lacking such provisions, static models can not handle dynamic interactions properly [18]; on the contrary, a dynamic modeling is one that can represent both the structure and the behavior of a system at any time of the process. In a CIS, a relevant topic is the one related with the cooperation among agents, working collectively with a common goal, but this is not an easy modeling task, because the need to represent several collaborative layers to get a global view of the cooperative agents behavior expressively and, in particular, the interaction model of the system. In order to explain the interaction role, it is proposed to satisfy it within a more general framework, which is illustrated by the model shown in figure 1. In the model we consider that the *Cooperation* problem involves several layers: a *Communication* layer, an *Interaction* layer, and the *Coordination* layer. At the low level we have a Communication Protocol, which enables the information exchange among the agents of the system and produces a change of the system state [1], [2], [20]. The Interaction mechanism is a set of behavior rules that defines the information exchange among agents [9], [4]. The Coordination mechanism establishes the action sequence and execution according to the agents' individual goals and the common goal of the CIS [19], [8], [7]. Finally, at the high level we have a Cooperation, which is a result of the mechanisms that support interactions among the agents [17]. In figure 1, it is modeled a general cooperation situation between two agents and the relationship with the proposed framework. Here, we represent the communication layer like a link relation between agents, the interaction with a set of behavior rules, one set for each agent, and the coordination with an ordered set of actions for the different agents. It is important to point out that the order set of actions, in the coordination layer, is an order message set according to the interaction layer rules and the messages of the communication layer. By Cooperation, it is understood the agent's behavior to coordinate interaction and information exchange in order to achieve a common goal. We aim to deal more specifically with the Interaction layer in this work.

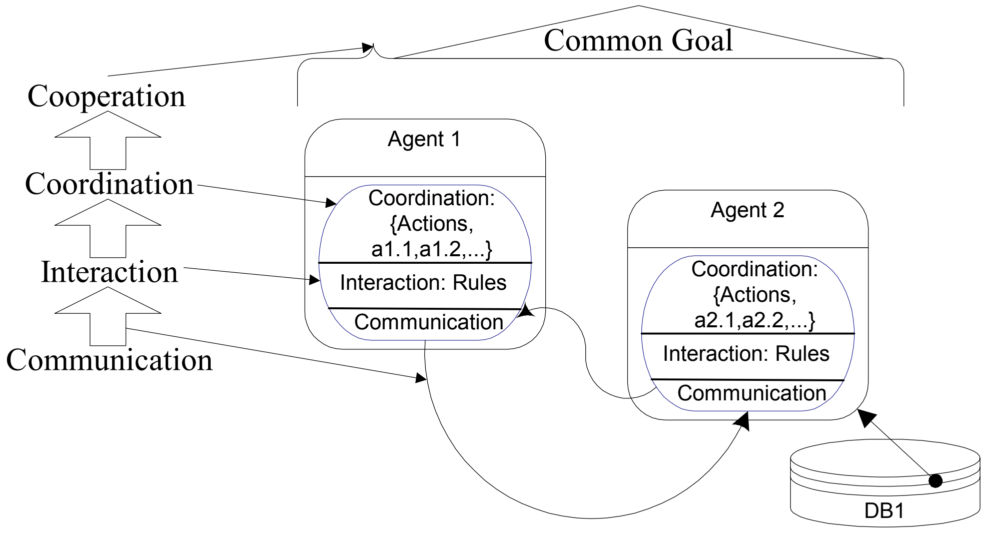

Figure 1: A Cooperation Reference Framework.

The interaction mechanism is central for the cooperation in CIS, because it is the bridge between the communication protocol and the coordination mechanism for the agents in the system. The interaction problem for CIS is immersed in a natural dynamic world, consequently, the interaction modeling and control are hard to manipulate and normally they have ambiguity and control problems. It is believed that the use of a formal method is a feasible approach to properly deal with the associated complexity in the modeling of a CIS. This will ease us to reduce ambiguity in the interaction model and allow to simulate the dynamic of the system, which is related to multiple simultaneous interactions.

The paper is organized as follow, in section 2, it is reviewed other works in modeling the interaction using different formal methods, and therefore clarifies the most important identified problems. In section 3, an explanation is provided in how to reduce the complexity in the interaction modeling among agents using CPN, compared to other works using C/E Petri Nets. In section 4, the action basic loop is introduced to build the interaction diagrams, with the comparison between Flores's communicative acts and the FIPA communicative acts. Section 5 illustrates in detail the IMCIS (Interaction Methodology for CIS) for the interaction modeling. The methodology is explained, using an example, step by step, based on the knowledge of the action basic loop and the CPN modeling approach. Finally, the conclusion and further works are presented.

2. Interaction and Formal Methods

The uses of different formal methods in order to model the interaction mechanism are present in related literature such as: The First Order Logic [16], State Transition Diagrams [1], [14], Condition/Event (C/E) Petri Nets [12]. In the different applications of these methods, the interaction usually shows isolatedly, that is, just between two agents; however, agents are sometimes involved in several interactions simultaneously and they have to manage these multiple interactions, which involve a complex problem to be solved. Some limitations of these methods are: 1) they are practical to specify the structure of the interaction when they appear in isolated communication situations, but they are not adapted to model complex protocols with several interactions simultaneously in CIS. 2) They are very complex to manage and modify in order to respond to changes in the system specification. 3) The methods are good for representing static systems, but we need to model dynamic interactions situations, and the methods pose limitations for this. 4) The combinatory explosion of the methods when the need to model complex simultaneous interactions in CIS arise.

The most important problems to be coped with in order to improve the performance of a model oriented to these applications aforementioned are the following:

- The state of simultaneous interaction among more than two agents.
- The behavior of the agents in the interaction according to a precise state.
- The representation of different messages for different agents in different states.

The classic formal methods like the first order logic, state transitions diagrams, among others, are very difficult to manipulate, and the use of other methods such as the Petri Nets become functional, but some of them lack the expressiveness to model this kind of problem.

3. Coloured Petri Nets for Modeling Interaction

A Petri net is a formal and graphic appealing language, which is appropriate for modeling complex systems with concurrency [18]. The Coloured Petri net are high level Petri net where each token of a different color represents an arbitrary data values [21]. This extension increases the descriptive power for modeling. The firing of transitions is then made dependent on the availability of an appropriately coloured token [18]. The coloured Petri nets are a good formalism for describing concurrency, synchronization and causality [3], and are suitable for modeling, analyzing and prototyping dynamic systems with parallel activities [8], [10] as CIS in our approach. In this work, it is proposed the use of CPN, because they have relevant characteristics for modeling interaction in CIS, such as: 1) the graphical representation, 2) the well defined semantics, 3) the formal analysis of the models and 4) the capacity for modeling the system hierarchically. The properties of CPN should allow to model the state of the interaction simultaneously among more than two agents and the behavior of the interaction according to its state, and representing different message for different agents in different states. The use of CPN computer tools, such as Design/CPN [18], helps us make a dynamic simulation of the system interaction, and to find problems before the system implementation.

Other works that model interactions using CPN are El Fallah et al. [9], [10], [11] and Cost et al. [4], [5], [6]. El Fallah et al., focuses on the study of multi-agent systems design, combining two aspects: 1) Distributed observation to capture the interactions between agents and 2) CPN as a formalism to identify interaction-oriented designs. Cost et al., focuses in the construction of a language for conversation specification, named Protolingua within the framework of the Jackal agent development environment, and they proposed use CPN as a model underlying a language for conversation specification. They do not deal with the problem of complexity and expressivities.

A central point in this work is how to reduce the associated complexity in the modeling of the interaction among agents in a CIS. In figure 2 we can observe a model used in Demazeau et al. [12] with C/E Petri Nets, where three agents or entities are represented: Entity i-1, Entity i and Entity i+1, and *m* messages, where i=1..n represent the total agents in the system, and j=1..m represent the total message number in the system. Their approach uses a "message line concept", which is modeled with C/E PN, where the virtual medium linking two agents, capable of exchanging *m* different sorts of messages, consists of *k* lines like the ones shown in figure 2, and where the dotted line represents the virtual medium. The agents are modeled with C/E PN. The C/E PN details, such as place and transitions, are evident to the system engineer, in the virtual medium, and the complexity of the model is associated with the number of links among agents and message lines. In this model the incorporation of new simultaneous interactions among agents is harder to represent, because we need to model and include new message lines, and the associated complexity is increased in $m^2 * s$ links, where *s* is the number of additional agents which are participating in the interaction and *m* is the total number of messages.

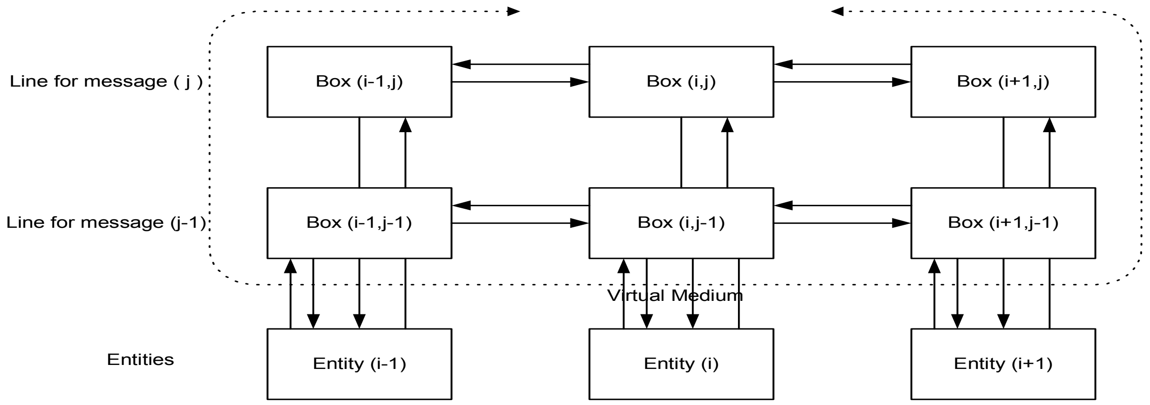

Figure 2: Demazeau Interaction Model.

It can be compared with the same problem represented in the proposed approach in figure 3, where we can see a CPN inside each agent and modeling each message that is part of the interaction relationship of the system, represented by a link between agents. CPN is not used in order to model the virtual medium, and when it is needed to incorporate new simultaneous interactions or agents, the agent and its message representation are only modified using the CPN color declarations.

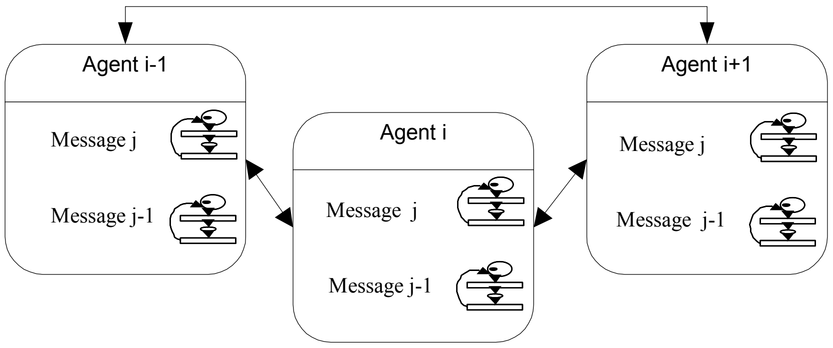

Figure 3: The Proposed Interaction Model.

If more than two agents are interacting, the use of a virtual medium to model the interaction is very difficult. Instead, by using this approach based on CPN, this difficulty is reduced considerably. We can see in [12] the associated complexity in terms of the number of links between agents interactions, according to the message numbers *(m)*, and the systems agents *(n)*. In the model presented in figure 2, the complexity is $O(m^2*n)$. While in the proposed model of figure 3, using CPN for modeling agent interaction, the associated complexity is $O(n^2)$, but, there is message independency because we just take the interactions among agents. The differences between our approach and the Demazeau approach are: 1) in the modeling of the virtual medium, such as in our approach, we use distributed systems techniques and we do not need to model this explicitly, 2) our model is independent of the number of messages among agents and 3) when modeling a new simultaneous interaction, since the associated complexity of [12] model is $O(m^2*s)$, and in our model's is $O(2n*s)$, where s is the number of additional agents which are participating in the interaction, n is the total number of agents and m is the total number of messages. Clearly, our modeling technique is suitable for a large scale agents applications, like a CIS, in contrast with Demazeau's technique, because of the latter's combinatory explosion of the method when modeling complex simultaneous interactions.

4. Interaction Modeling

Build the interaction diagrams is a central activity. Our analysis is centered on the system interactions, where we determine who talks to whom and in which way. We propose the use of the action basic loop [15] for modeling the interaction between agents. The action basic loop proposes an ontology of communicative acts: Request, Promise, Inform and Declare. In figure 4 we can see the communicative acts and their loop position, and in figure 5 we present the loop processes.

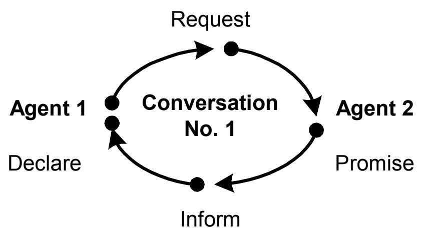

Figure 4: Communicative Acts in the Action Basic Loop.

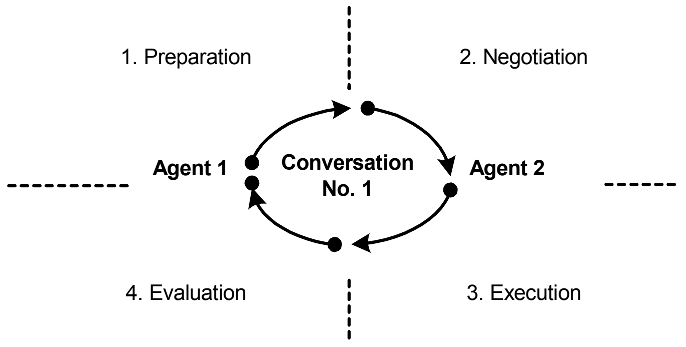

Figure 5: The Action Basic Loop.

In the first step, agent 1 *prepares* the *request* for agent 2 in a conversation. After that, agent 1 and agent 2 make a *negotiation* about the request, and agent 2 issues a *promise*. In the next stage, agent 2 *executes* the *promise* and when they finish the task, they give an *inform* to agent 1. In the last step, agent 2 makes a *declaration* for the *evaluation* of agent 1 work.

Many conversations are part of an interaction, and we need to build an interaction diagram, as the one shown in figure 6, for each system interaction. The methodology proposes a documentation set and a notation for the interaction model, but they are not shown in this paper.

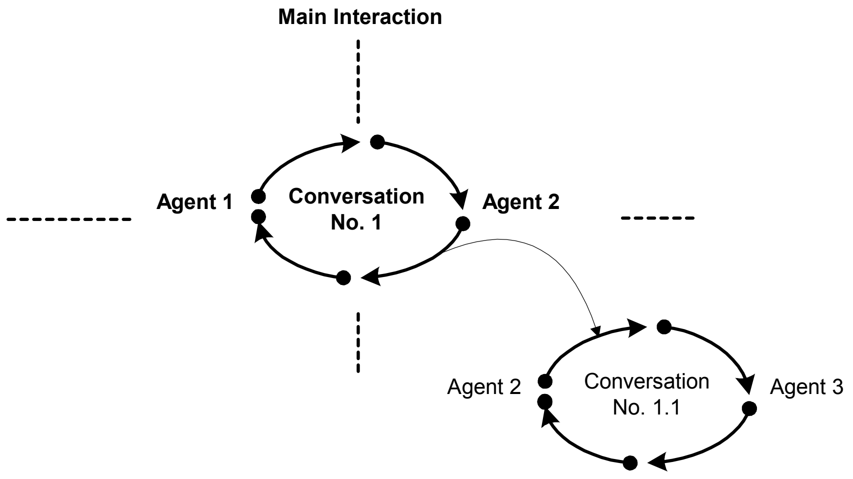

Figure 6: Interaction Diagram.

The FIPA [13] communicative acts are more expressive than the Flores's communicative acts, for instance, the FIPA ontology sets different types of *request* acts whereas Flores sets a single *request* act. It is proposed a categorization of the FIPA communicative acts according to Flores communicative acts, in table 1. Our proposal is to use the Flores's communicative acts, for the action basic loop, but exploring the FIPA approach for improving model expressiveness because, if we only use the four Flores basic acts, our communication language will lose expressivity.

The basic action loop helps us to have a coordinate conversation between agents, and to join more that two agents simultaneously in the coordinate conversation, building an Interaction. The works of El Fallah et al. [9] and Cost et al. [4], have different interaction protocols and they increase the complexity, because the agents need to recognize the interaction being used and its different states, in order to have a set of behavior rules that defines the information exchange.

FIPA	Communication for action (Flores)			
Communicative act	**Request**	**Promise**	**Inform**	**Declare**
accept-proposal				X
Agree				X
Cancel			X	
Cfp (call for proposals)	X			
Confirm			X	
Disconfirm			X	
Failure			X	
Inform			X	
Inform-if (macro act)			X	
Inform-ref (macro act)			X	
not-understood			X	
propose		X		
query-if	X			
query-ref	X			
Refuse			X	
reject-proposal				X
request	X			
request-when	X			
Request-whenever	X			
subscribe	X			

Table 1: FIPA and Flores Communicative Acts.

5. *IMCIS*: The Methodology for Modeling Interaction in CIS [23]

CIS are complex due to their dynamic nature and the management of many simultaneous interactions, and the software specification is hard to be implemented due to the reasons stated above. A proper structured way of building software for the aforementioned needs is relevant to ease the specification of complex systems. In addition to this, the problem of building powerful software tools to specify dynamic complex systems, such as CIS, has not been dealt with enough. We center our work in two areas: analysis and design of interactions in CIS. We actually hold different views about the system: 1) the explicit analysis and specification for the static view and 2) the implicit analysis and specification for the dynamic view. Current works by different authors give us a static model, but the CIS are very dynamic [22]. It is proposed to build a behavior model using the individual and the structural model. In order to make a model of a system, we need a set of abstractions that will allow us to capture the essence of the behavior of the system we wish to model. The CIS frequently perform complex tasks that are distributed over space and time, and that involve discrete flows of objects and/or information. It is proposed the use of CPN in order to represent the agent behavior and its intentions, and to simulate the resulting model. Figure 7 shows the integration of the models. At the low layer we have the *individual model*, in which we describe the agents separately. At the medium layer, we have the *structural model*, in which we describe the interactions among agents, and finally, at the upper layer, we have the *dynamic model*, in which we observe and control the simulation of the system interaction.

The explicit model is built from the system specification, but the dynamic model is built from the explicit model and simulated in the tool. The CPN model captures both the static and the dynamic behavior of the specification.

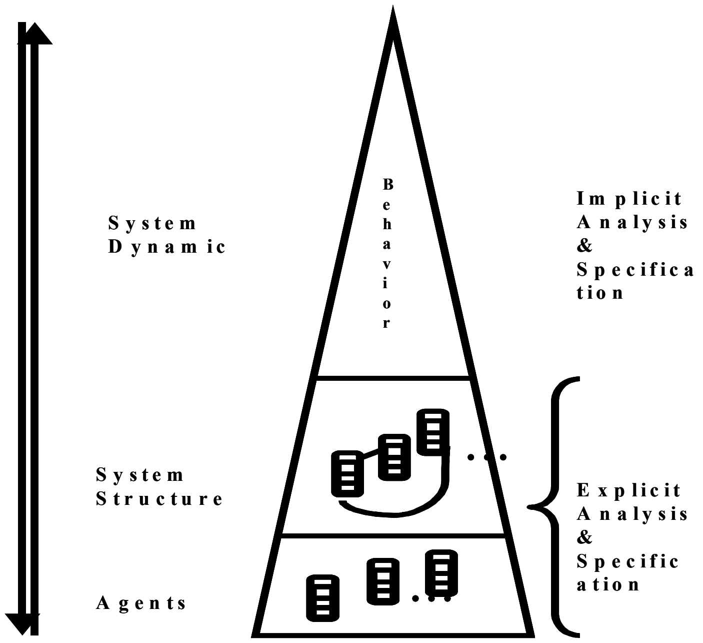
System Dynamic
Behavior
Implicit Analysis & Specification
System Structure
Explicit Analysis & Specification
Agents

Figure 7: Methodology Views.

The IMCIS [23], the methodology for modeling interaction in CIS has the following steps:

IMCIS *(Interaction Methodology for CIS)* .

Explicit Analysis and Specification:

1. Identify agents and their intentions *(individual model).*
2. Build the agents diagram *(structural model).*
3. Build the interaction diagrams *(structural model).*
4. Design the agent ports *(structural model).*
5. Design the messages *(structural model).*
6. Specify the systems messages using Coloured Petri Nets *(structural model).*

Implicit Analysis and Specification:

7. Simulate the system interactions *(dynamic model).*

5.1. Identify agents and their intentions and build the agents diagram

The methodology can be exemplified with the following case: a business to business (B2B) environment involves simultaneous interactions among buyers, suppliers and the marketplace that need to be controlled. A detail description of the methodology is shown in [23]. The first step is to identify the agents and their intentions. Here we have three agents: *Buyer*, *Marketplace* and *Supplier*. We build the agents set A, with agents and his intentions: A={(Buyer, to buy a product with the best option in the market), (Marketplace, to make effective relationships among business partners easier), (Supplier, to offer the best quality - price - volume products)}. In figure 8 we present the agent diagram, spawned from the second steep.

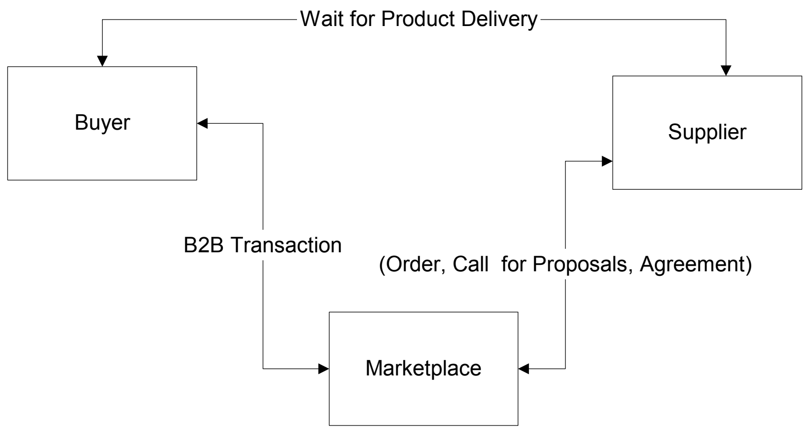
Wait for Product Delivery
Buyer
Supplier
B2B Transaction
(Order, Call for Proposals, Agreement)
Marketplace

Figure 8: Agent Diagram.

The equivalent graph for the agent diagram is: AD={(Buyer, Marketplace, Supplier), (*B2B Transaction*, *Call for Proposals*, *Order*, *Wait for Product Delivery, Agreement*)}.

5.2. Build the interaction diagrams

To build the interaction diagram, the main system interaction is identified as: B2B Transaction, which is composed of four conversations: Buyer *B2B Transaction* Marketplace, Marketplace *Call for Proposals* Supplier, Marketplace *Order* Supplier, Buyer *Wait for Product Delivery* Supplier. The interaction Marketplace *Agreement* Supplier is a Context Interaction, and just gives us prerequisite information or a reference information. Figure 9 shows the interaction diagram.

Interaction: B2B Transaction

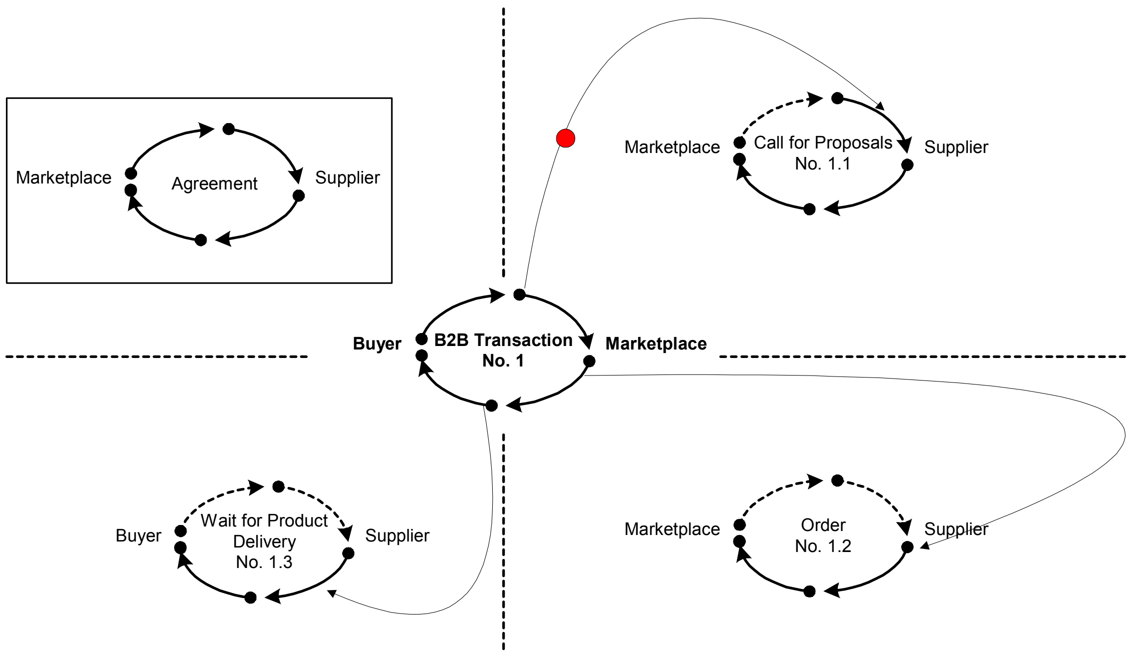

Figure 9: Interaction Diagram B2B Transaction.

So a CIS specification is represented by the following expression: CIS={A, Cg, Gk, I}, where:

- Agent Set, A={(Buyer, to buy a product with the best option in the market), (Marketplace, to make effective relationships among business partners easier), (Supplier, to offer the best quality - price - volume products)}.
- Common Goal, Cg= To satisfy the buyer need with the best offer, and the best agreements with suppliers.
- Global Knowledge, Gk={Buyer needs, products, agreements}.
- Interaction Set, I={B2B Transaction}.

In the next step, the design of the interaction in the CIS. In tables 2 and 3, the different steps and cases in a conversation are modeled, where the client/provider communicative acts [15] take place according to the steps in the basic action loop and its role in the conversation.

	Interaction			
Basic Communicative Acts	Preparation	Negotiation	Execution	Evaluation
Client				
Request	Start			
Declare satisfaction				
No agr				Start
No rep				Start
Void				Use
Cancel				
No agr		Use		
Preparation	Start			
Void				Start
Cancel make new request		Start		
Decline to accept				
No agr		Start		
Void			Start	
Close				
Revoked				Start
Declined		Use		Use
Ask recons				
Revoked		Start		
Declined		Start		
Counter		Use		
Decline counteroffer		Use		
Agree to counteroffer			Start	

Notation:
Start: Conversation Firing.
Use: Conversation Use.

Table 2: Client Communicative Acts.

	Interaction			
Basic Communicative Acts	Preparation	Negotiation	Execution	Evaluation
Provider				
Agree			Start	
Decline		Start		
Counteroffer		Start		
Revoke and counteroffer		Start		
Revoke				
No agr		Use		
Void			Use	
Report completion				
No agr				Start
Void				Start
Ask recons				
Cancel		Start		
Close				
Canceled				Use
Satisfied				Use

Notation:
Start: Conversation Firing.
Use: Conversation Use.

Table 3: Provider Communicative Acts.

5.3. Design the agents ports and the messages

The design of agents ports helps us to reduce the modeling complexity, in particular the links number among agents. The design of the agents ports is shown:

Interaction: B2B Transaction (Buyer, Marketplace, Supplier).

Conversations: 1) Buyer *B2B Transaction* Marketplace, 2) Marketplace *Call for Proposals* Supplier, 3) Marketplace *Order* Supplier, 4) Buyer *Wait for Product Delivery* Supplier.

Ports: The communicative acts are used, according to the basic interaction loop. The Buyer, Marketplace, and Supplier play two different roles in the conversation: Client and Provider. In the Pin (Port In) and Pout (Port Out) ports, the *provider* and *client* conversation messages are represent. The notation for port representation in Agent 1 Conversation Agent 2 is: Agent1.P(in or out) $_{\text{Conversation}}$, Agent2 (*m1, m2, ..., mx*), where *mi* is the *i* message in the port in the conversation between Agent 1 and Agent 2, and *x* is the total message number. The graphic representation is shown in figure 10.

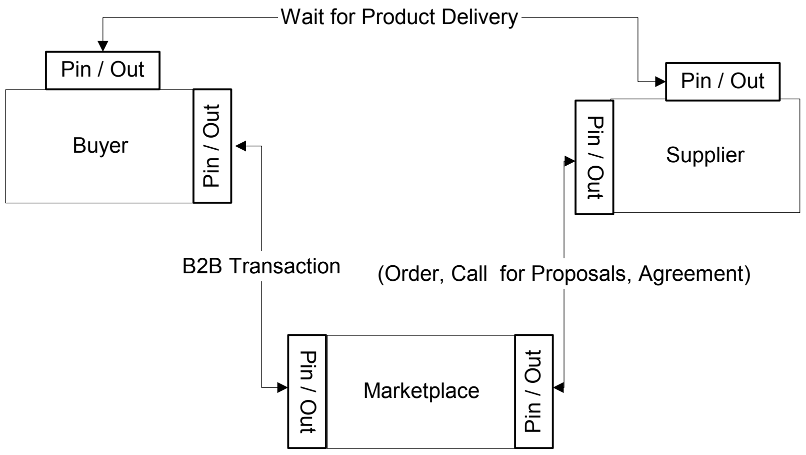

Figure 10: Agents Ports Diagram.

The *Buyer* ports are:

- Buyer.Pin $_{\text{B2B Transaction}}$, Marketplace (Agree, Decline, Counteroffer, Report Completion(Type), Revoke(Type), Revoke and Counteroffer, Close(Type), Ask Recons(Type)).
- Buyer.Pout $_{\text{B2B Transaction}}$, Marketplace (Request, Declare Satisfaction(Type), Cancel(Type), Cancel Make New Request, Decline to Accept(Type), Close(Type), Ask Recons(Type), Counter, Decline Counteroffer, Agree to Counteroffer).
- Buyer.Pin $_{\text{Wait for Product Delivery}}$, Supplier (Agree, Decline, Counteroffer, Report Completion(Type), Revoke(Type), Revoke and Counteroffer, Close(Type), Ask Recons(Type)).
- Buyer.Pout $_{\text{Wait for Product Delivery}}$, Supplier (Request, Declare Satisfaction(Type), Cancel(Type), Cancel Make New Request, Decline to Accept(Type), Close(Type), Ask Recons(Type), Counter, Decline Counteroffer, Agree to Counteroffer).

The *Marketplace* ports are:

- Marketplace.Pin$_{\text{B2B Transaction}}$, Buyer = Buyer.Pout$_{\text{B2B Transaction}}$, Marketplace.
- Marketplace.Pout$_{\text{B2B Transaction}}$, Buyer = Buyer.Pin$_{\text{B2B Transaction}}$, Marketplace.
- Marketplace.Pin$_{\text{Call for Proposals/Order/Agreement}}$, Supplier (Agree, Decline, Counteroffer, Report Completion(Type), Revoke(Type), Revoke and Counteroffer, Close(Type), Ask Recons(Type)).
- Marketplace.Pout$_{\text{Call for Proposals/Order/Agreement}}$, Supplier (Request, Declare Satisfaction(Type), Cancel(Type), Cancel Make New Request, Decline to Accept(Type), Close(Type), Ask Recons(Type), Counter, Decline Counteroffer, Agree to Counteroffer).

The *Supplier* ports are:

- Supplier.Pin$_{\text{Call for Proposals/Order/Agreement}}$, Marketplace = Marketplace.Pout$_{\text{Call for Proposals/Order/Agreement}}$, Supplier.
- Supplier.Pout$_{\text{Call for Proposals/Order/Agreement}}$, Marketplace = Marketplace.Pin$_{\text{Call for Proposals/Order/Agreement}}$, Supplier.
- Supplier.Pin$_{\text{Wait for Product Delivery}}$, Buyer = Buyer.Pout$_{\text{Wait for Product Delivery}}$, Supplier.
- Supplier.Pout$_{\text{Wait for Product Delivery}}$, Buyer = Buyer.Pin$_{\text{Wait for Product Delivery}}$, Supplier.

The Conversation Specification form shown in figures 11 and 12 is part of the documentation form set. Here the basic information about the interaction and its conversations are described. In the example, the conversation *Call for Proposals* between Marketplace and Supplier is shown. The Conversation specification form has two parts, the conversation description shown in figure 11, and the interaction diagram shown in figure 12.

Conversation Specification:

Interaction Name: *B2B Transaction*	Interaction ID: *1*	Date: *July 2001*	Version: *1.0*
Conversation Name: *Call for Proposals*		Conversation ID: *1.1*	
Main Conversation Goal: *To receive the best Supplier offers to the Marketplace according to the Buyer's request.*			
Client: *Marketplace*	Provider: *Supplier*		

Figure 11: The Conversation Description.

In the conversation diagram, the exchange of communicative acts between the Marketplace (Client) and Supplier (Provider) in each step of the action basic loop are described. All conversations in the interaction diagram must have a conversation diagram like the one shown in figure 12. The messages are represented by the communicative acts and they are present in each basic loop step, in tables 2 and 3.

In our example, the conversation *Call for Proposals* is simple, because the marketplace request product information and economic proposals, inside a previous agreement, and the negotiation space is closed to this. The suppliers can or can not send proposals, but the marketplace has the decision to accept or reject this, according to the agreement. When an agents are involved in an interaction, the conversation flow is controlled by the basic action loop. For example, in figure 12, at the *evaluation* step, if the marketplace returns the message **Declare Satisfaction**, the conversation comes to an end, but if the message is **Decline to accept**, the conversation moves forward to the *execution* step.

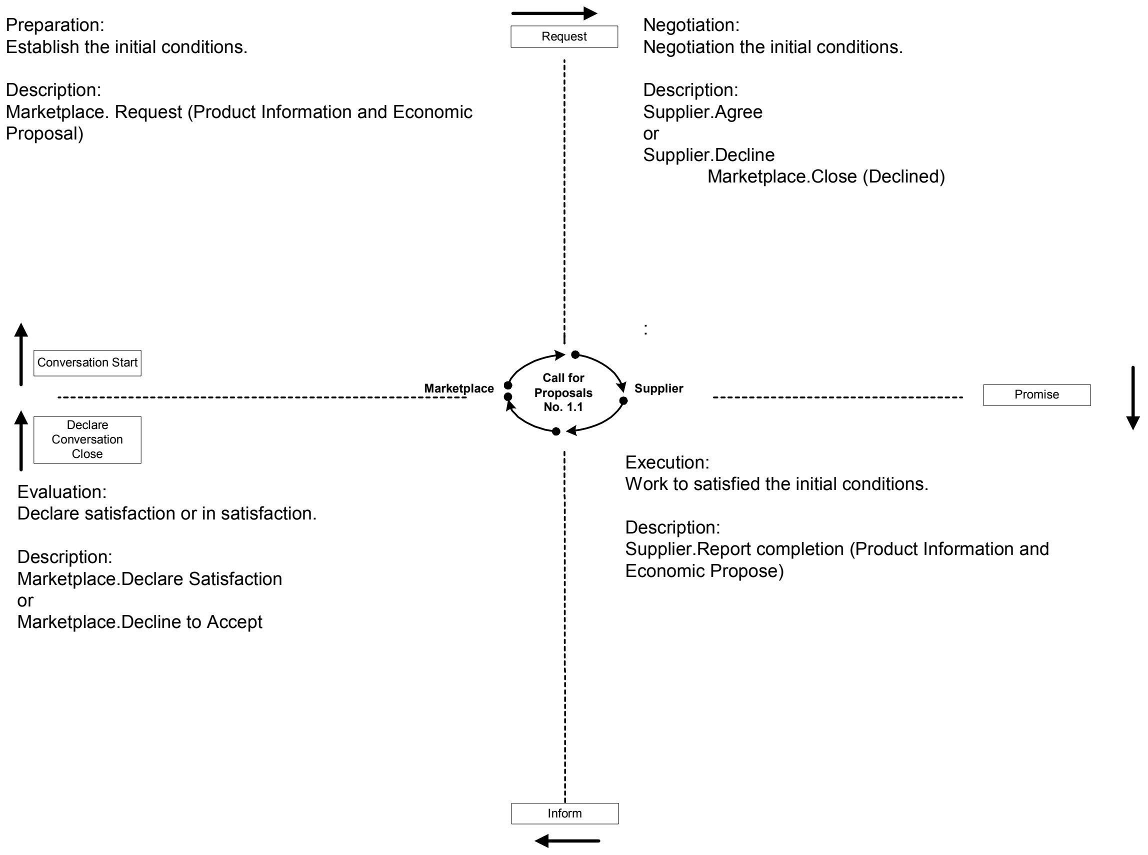
Request
Preparation:
Establish the initial conditions.
Description:
Marketplace. Request (Product Information and Economic Proposal)
Negotiation:
Negotiation the initial conditions.
Description:
Supplier.Agree
or
Supplier.Decline
Marketplace.Close (Declined)
Conversation Start
Declare Conversation Close
Marketplace
Call for Proposals No. 1.1
Supplier
Promise
Evaluation:
Declare satisfaction or in satisfaction.
Description:
Marketplace.Declare Satisfaction
or
Marketplace.Decline to Accept
Execution:
Work to satisfied the initial conditions.
Description:
Supplier.Report completion (Product Information and Economic Propose)
Inform

Figure 12: The Detail Interaction Diagram

5.4. Specify the System Messages Using CPN and Simulate the System Interactions

The general interaction mechanism used by the Buyer, Marketplace and Supplier is modeled. Figure 13 shows the hierarchy page *BasicActionLoopForInteraction*, build in Design/CPN, where the two agents, Client (Marketplace) and Provider (Supplier), are represented and the different pages and their relationships are shown. The Client is represented by the *MainClient* CPN, and the Provider by the *MainProvider* CPN. The *fusion place* mechanism is used for interconnecting net structure on different pages. A *fusion place* is a place that has been equated with one or more other places, so that the fused places act as a single place with a single marking [18]. The defined fusion set is *CommunicationMedium* and it represent the common places in the interaction. The Color declaration of the *Tinteraction* type is a record with a list of buyers, a marketplace, a list of suppliers, a product and a communicative act in the interaction. The buyer, supplier, marketplace and product have its own color definition. The *Tinteraction*, represent a system interaction, for example, an instance like ((Buyer 1), Marketplace, (Supplier 1, Supplier 2, … , Supplier n), Product x, Marketplace.Counteroffer)). In the following figures, *interaction* and *interactiontemp are* a *Tinteraction* declared variables.

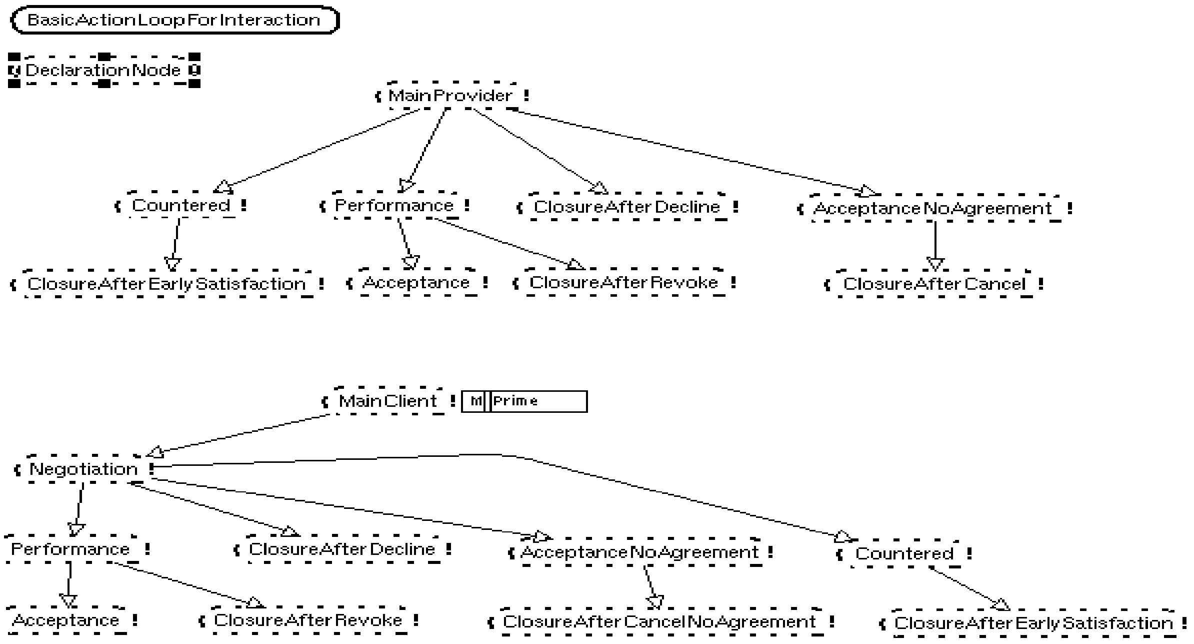

Figure 13: The General Interaction Mechanism CPN Hierarchy Page.

The first step in a general conversation shown in figure 14 is the *preparation* step in the *MainClient* CPN, and the conversation begins with a client message **request**, and the provider can respond with different kinds of messages such as **agree**, **decline**, **report completion with no agreement**, and **counteroffer**, inside the *negotiation* step.

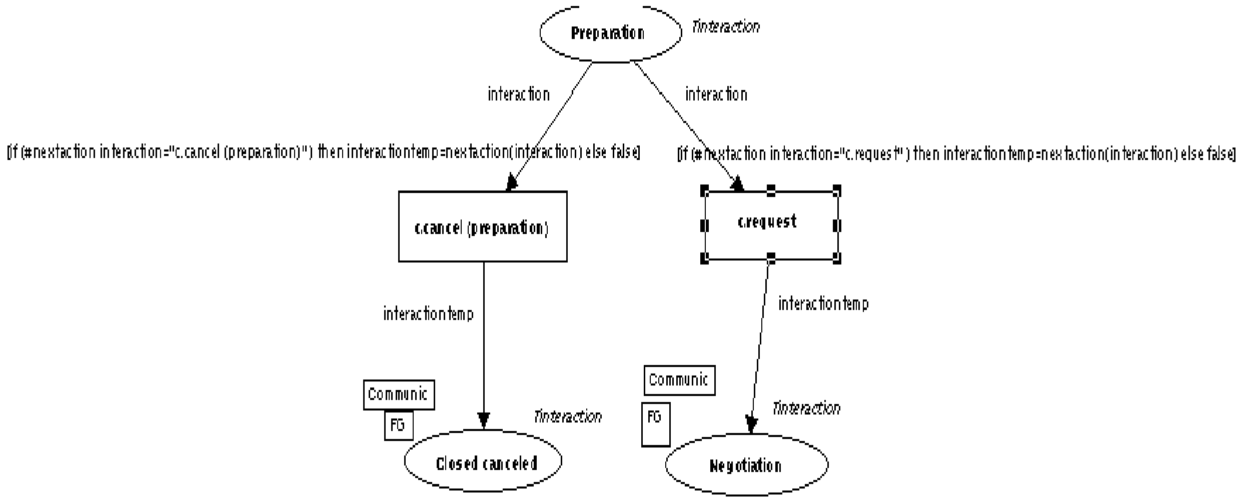
Preparation
Tinteraction
interaction
interaction
[if (#nextaction interaction="c.cancel (preparation)") then interactiontemp=nextaction(interaction) else false]
[if (#nextaction interaction="c.request") then interactiontemp=nextaction(interaction) else false]
c.cancel (preparation)
c.request
interactiontemp
interactiontemp
Communic
FG
Communic
FG
Tinteraction
Tinteraction
Closed canceled
Negotiation

Figure 14: The Preparation Step CPN.

In a common interaction, the coordination communicative flow is:

- client **request**
- provider **agree**
- provider **report completion**
- client **declare satisfaction**

but, if the agents face an interaction conflict, a possible coordination communicative flow is: client **request** and the provider has different options:

- **Agree** to accept the requirement
- **Decline** the request
- **Report completion** with no agreement
- **Counteroffer** to provide a new option

In our example the client **counteroffer** the provider with a different product, the client has different options:

- **Counter** to ask for a different option
- **Decline** counteroffer
- **Cancel and make new request**
- **Cancel** no agreement
- **Declare** satisfaction no agreement
- **Agree** to counteroffer

If the option is **agree to counteroffer**, the interaction flow is similar to the common interaction flow. This interaction situation is part of the *negotiation* step in the *MainProvider* CPN, shown in figure 15, where the client and provider have a message exchange before advancing to the next step, as seen in tables 2 and 3. In our example, the *Call for Proposals* conversation shown in figure 12 has the following communicative coordination flow: Marketplace.Request, Supplier.Agree or Supplier.Decline, Supplier.Report Completion, Marketplace.Declare Satisfaction or Marketplace.Decline to Accept.

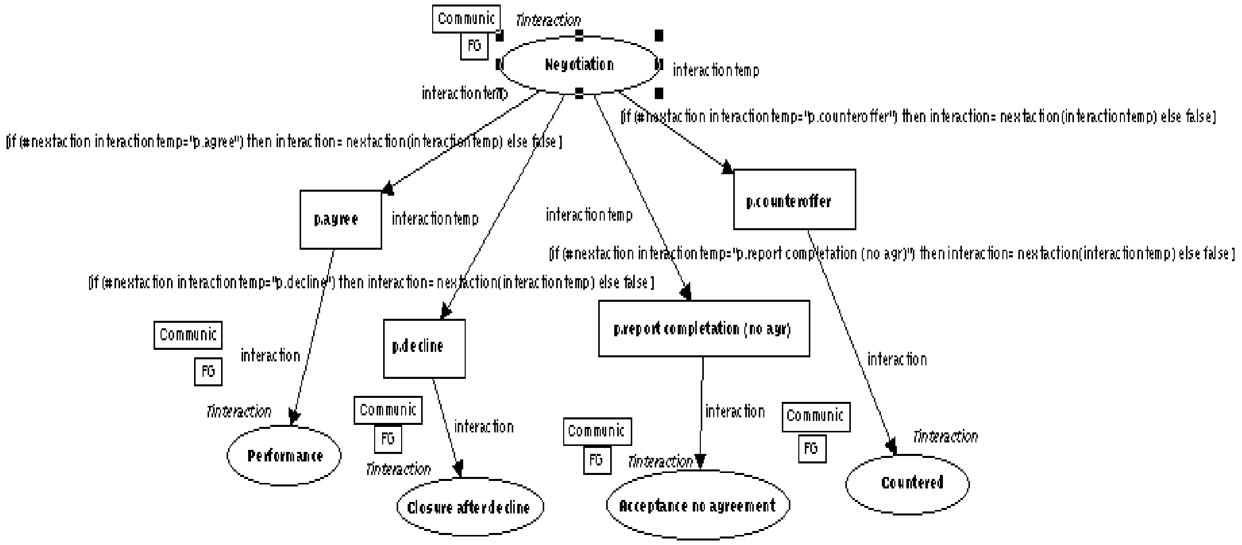
Communic
FG
Tinteraction
Negotiation
interactiontemp
interactiontemp
[if (#nextaction interactiontemp="p.counteroffer") then interaction= nextaction(interactiontemp) else false]
[if (#nextaction interactiontemp="p.agree") then interaction= nextaction(interactiontemp) else false]
p.agree
interactiontemp
interactiontemp
p.counteroffer
[if (#nextaction interactiontemp="p.report completation (no agr)") then interaction= nextaction(interactiontemp) else false]
[if (#nextaction interactiontemp="p.decline") then interaction= nextaction(interactiontemp) else false]
p.decline
p.report completation (no agr)
Communic
FG
interaction
interaction
Tinteraction
Performance
Communic
FG
interaction
Tinteraction
Closure after decline
Communic
FG
interaction
Tinteraction
Acceptance no agreement
Communic
FG
Tinteraction
Countered

Figure 15: The Negotiation Step CPN.

Figure 16 shows part of the *negotiation step*: the c*ountered* situation, where the client or provider can **accept**, **reject** or **counteroffer** the request. The *countered* step is central in the interaction mechanism in order to resolve conflicts and make agreements.

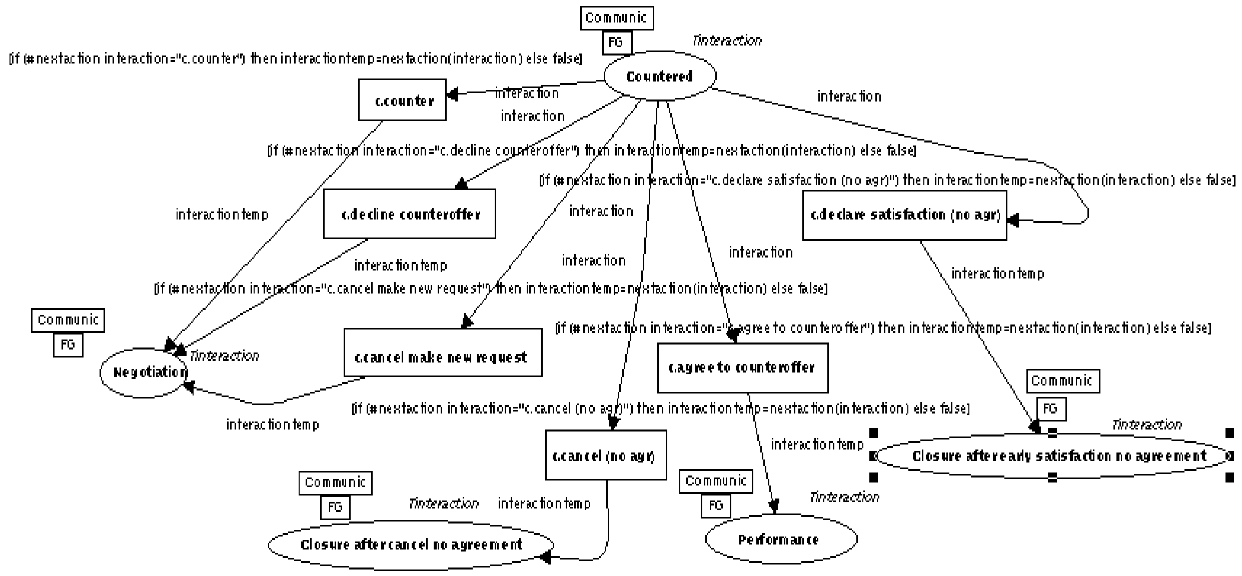
Communic
FG
Tinteraction
Countered
c.counter
c.decline counteroffer
c.declare satisfaction (no agr)
c.cancel make new request
c.agree to counteroffer
c.cancel (no agr)
Negotiation
Closure after early satisfaction no agreement
Closure after cancel no agreement
Performance
interaction
interactiontemp
[if (#nextaction interaction="c.counter") then interactiontemp=nextaction(interaction) else false]
[if (#nextaction interaction="c.decline counteroffer") then interactiontemp=nextaction(interaction) else false]
[if (#nextaction interaction="c.declare satisfaction (no agr)") then interactiontemp=nextaction(interaction) else false]
[if (#nextaction interaction="c.cancel make new request") then interactiontemp=nextaction(interaction) else false]
[if (#nextaction interaction="c.agree to counteroffer") then interactiontemp=nextaction(interaction) else false]
[if (#nextaction interaction="c.cancel (no agr)") then interactiontemp=nextaction(interaction) else false]

Figure 16: The Countered Step CPN.

Figure 17 shows the *execution* step, where the provider works in order to satisfy the client request. The provider can use the following messages: **report completion**, **revoke**, **revoke and counteroffer**. In the case of the **revoke** message, the conversation probably returns to the *negotiation* step, and with the **report completion** message the conversation moves forward to the *evaluation* step.

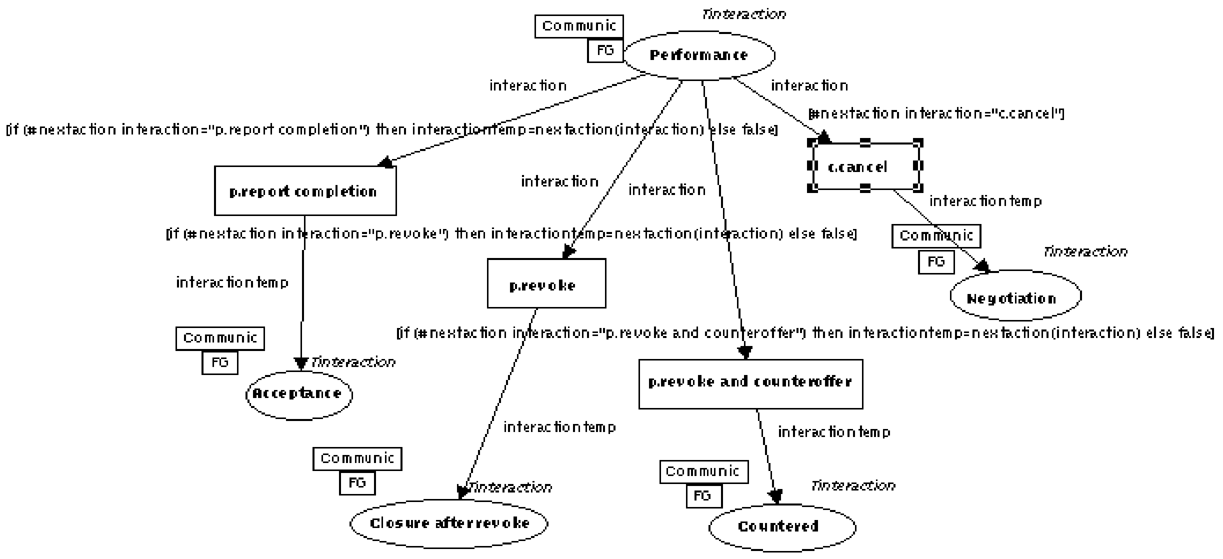
Communic
FG
Tinteraction
Performance
p.report completion
p.revoke
p.revoke and counteroffer
c.cancel
Acceptance
Closure after revoke
Countered
Negotiation
interaction
interactiontemp
[if (#nextaction interaction="p.report completion") then interactiontemp=nextaction(interaction) else false]
[#nextaction interaction="c.cancel"]
[if (#nextaction interaction="p.revoke") then interactiontemp=nextaction(interaction) else false]
[if (#nextaction interaction="p.revoke and counteroffer") then interactiontemp=nextaction(interaction) else false]

Figure 17: The Execution Step CPN.

When the conversation is at the *evaluation* step, presented in figure 18, the client must evaluate the provider's work result. The messages are: **declare satisfaction**, **decline to accept**, **cancel** or **cancel and make new request**. If the messages are **declare satisfaction** or **cancel**, the conversation comes to an end, but if the message is **decline to accept**, the conversation returns to the *execution* step or with **cancel and make new request** message, this return to the *negotiation* step.

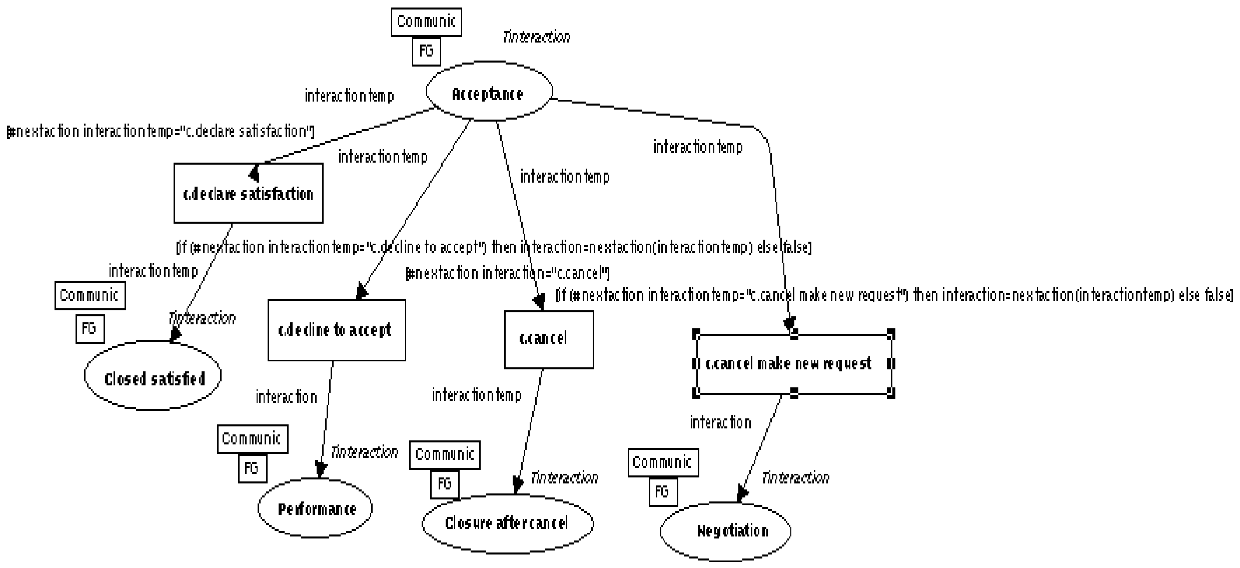
Communic
FG
Tinteraction
Acceptance
interactiontemp
[#nextaction interactiontemp="c.declare satisfaction"]
interactiontemp
c.declare satisfaction
interactiontemp
interactiontemp
[if (#nextaction interactiontemp="c.decline to accept") then interaction=nextaction(interactiontemp) else false]
[#nextaction interaction="c.cancel"]
[if (#nextaction interactiontemp="c.cancel make new request") then interaction=nextaction(interactiontemp) else false]
interactiontemp
Communic
FG
Tinteraction
Closed satisfied
c.decline to accept
c.cancel
c.cancel make new request
interaction
interactiontemp
interaction
Communic
FG
Tinteraction
Performance
Communic
FG
Tinteraction
Closure after cancel
Communic
FG
Tinteraction
Negotiation

Figure 18: The Evaluation Step CPN.

An important aspect is how we model the interaction. There are two different instances of the color token interaction in the marketplace – supplier call for proposal conversation:

interaction 1 = ((Buyer 1), Marketplace, (Supplier 1, Supplier 2, … , Supplier n), Product 1, Marketplace.Declare Satisfaction))

interaction 2 = ((Buyer 2), Marketplace, (Supplier 1, Supplier 2, … , Supplier m), Product 2, Marketplace.Request))

In the instance 1, the marketplace is in an *evaluation* step within interaction 1, and in the instance 2, the marketplace is in the *preparation* step within interaction 2. Here, in the same CPN, multiple and simultaneous conversations are expressively modeled and controlled according to the token instances. Each instance has different buyers, suppliers, and products with given communicative acts within each particular conversation state.

Conclusion and Further Works

This paper provides a guide for modeling interaction in cooperative information systems by means of the use of coloured Petri nets to deal with the associated complexity for modeling the dynamic of the system. The interaction relations among agents are represented via an agent diagram, and formally specified using CPN. The use of CPN formalism offers the main advantages for modeling interaction in CIS: 1) It allows to easily model the state of the simultaneous interaction among more than two agents, 2) It allows to easily model the behavior of the interactions according to the state of the agents, 3) It allows an easy representation of different messages for different agents in different states, 4) It allows to simulate the system interaction dynamics.

The use of the basic action loop in order to model the organization interaction in the CIS helps to understand and represent different situations with a common action and coordination language, like the B2B system, but need to be tested in more application domains. The explicit and implicit analysis and specification where the system engineer may be managing the structure complexity and the system dynamics with the use of CPN. The use of *fusion place*, for modeling the system hierarchically, is a viable solution for representing the communication medium. The colors declarations helps us model complex data types and allow us to model and control multiple simultaneous interactions among agents. An attractive area is business to consumer (B2C) systems, which are high dependent of the user decisions in the interactive model, and may required to consider temporal and fuzzy aspects that need to be modeled with the IMCIS.

References

1. P.R Cohen and H.J. Levesque, "Communicative actions for artificial agents", Proceedings of the International Conference on Multi-Agent Systems, AAAI Press, San Francisco, June, 1995.
2. P.R. Cohen and H.J. Levesque, "Rational Interaction as a Basis for Communication", Cohen P. R., Morgan, J., and Pollack, M. E. (eds.), in Intentions in Communication, SDF Benchmark Series, MIT Press, pp. 221-255, 1990.
3. J.M. Cordero, and M. Toro, "A Components Model based on Interaction-Nets", ISAS/SCI 1999, Orlando, Florida, 1999.
4. R.S. Cost, Y. Chen, T. Finin, Y. Labrou and Y. Peng, "Modeling Agent Conversations with Coloured Petri Nets", To appear in Working Notes of the Workshop on Specifying and Implementing Conversation Policies, Autonomous Agents ´99, Seattle, WA, May 1999.
5. R.S. Cost, Y. Chen, T. Finin, Y. Labrou and Y. Peng, "Using Coloured Petri Nets for Conversation Modeling ", IJCAI ´99, 1999.
6. R.S. Cost, Y. Chen, T. Finin, Y. Labrou and Y. Peng, "A Negotiation-based multi-agent system for supply chain management ", in Working Notes of the Agents ´99 Workshop on Agents for Electronic Commerce and Managing the Internet-Enable Supply Chain, Seattle, WA, April 1999.
7. K. Decker, "Environment centered analysis and design of coordination mechanisms", PhD Thesis, University of Massachusetts Amherst, 1995.
8. A. El Fallah, and S. Haddad, "A Recursive Model for Distributed Planning", ICMAS-96, p.307-314, 1996.
9. A.El Fallah, S. Haddad and H. Mazouzi, "Observation répartie et analyse des interaction dans un système multi-agents", JFIADSMA-98, Eds Hermès, Nancy 1998.
10. A. El Fallah, S. Haddad and H. Mazouzi, "Une demarche méthodologique pour l´ingénierie des protocols d´interaction", JFIADSMA-99, Eds Hermès, Nancy 1999.
11. A. El Fallah, S. Haddad and H. Mazouzi, "Protocol Engineering for Multi-agent Interaction", 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW´99, Springer, 1999.
12. Y. Demazeau, J.L. Koning, and G. Françoise, "Formalization and pre- validation for interaction protocols on multi-agent systems", Distributed AI and Multi-agent Systems, p- 298-302, 1998.
13. FIPA, Foundation for Intelligent Physical Agents, "Agent Communication Language Specification", http://www.fipa.org, 2000.
14. F. Flores, and T. Winograd, "Understanding computer and cognition, a new foundation for design", Addison Wesley, 1986.
15. F. Flores, "Introducción al Ciclo Básico de la Acción ("Loop")", Business Design Associates, Inc., 1996.
16. A. Haddadi, "Towards a Pragmatic Theory of Interactions", Morgan Kaufmann Publishers, San Francisco California, United States of America, 1998.
17. M. Huhns, and M.P. Singh, "Readings in Agents", Morgan Kaufmann Publishers, San Francisco California, United States of America, 1998.
18. K. Jensen, "Coloured Petri Nets, Basic Concepts, Analysis Methods and Practical Use", volume 1, 2, 3, second edition, Springer, Germany, 1997.
19. V. Lesser, "Reflections on the nature of multi-agent coordination and its implications for the agent architecture", Autonomous agents and multi-agent systems, Kluwer academic publishers, 1, 89-111, July 1998.
20. H. Levesque and P. Cohen, "Teamwork", Nous 25(4), Special Issue on Cognitive Science and Artificial Intelligence, pp. 487-512. To appear in: Handbook of MultiAgent Systems, 1991.
21. D. Moldt and F. Wienberg, "Multi-agent systems based on coloured Petri nets", Proceedings of the 18th International Conference on Application and Theory of Petri Nets (ICATPN´97), number 1248 in Lecture Notes in Computer Science, p-82-101, Toulouse, France, June 1997.
22. M. Papazoglou and G. Schlageter, "Cooperative Information Systems, Trends and Directions", Academic Press, San Diego, 1998.
23. F. Ramos-Quintana, J. Frausto-Solis and F. Camargo-Santacruz, "A Methodology for Modeling Interactions in Cooperative Information Systems Using Coloured Petri Nets", to appear in the International Journal of Software Engineering and Knowledge Engineering, World Scientific, http://www.ksi.edu/ijsk.html, 2001.
24. H.J. Wooldidge and N.R. Jennings, "Intelligent Agents: Theory and Practice", The Knowledge Engineering Review, 10 (2), p. 115-152, 1995.