Models and Tools for MultiAgent Systems Analysis and Design

Nicandro Farias-Mendoza, Fernando Rodriguez-Haro

Electric and Mechanic Engineering Faculty of Colima University
Colima, Col. México Telephone 52+ (312) 31 611 65, fax 684 17 08
{Nmendoza, Ferharo}@Ucol.mx

Abstract. In this article we present a proposal for a methodology which includes a set of models and tools in order to develop software solutions based on agents. This methodology is based on Beliefs-Intentions agent architecture and uses the temporal logic alphabet and axioms for modelling system solutions. This methodology offers the next advantages: First, details explicitly the MultiAgent construction phases from the specification to the verification. Second, has the capacity to use the natural interaction process among agents to construct distributed applications. Third, the methodology guarantee the security through a language proposed and the coherence is supported by the interaction protocols used. This article propose an analysis and design methodology, which use interaction as support for develop distributed complex application and propose a formal Object language in order to specify in a natural and comprehensive way the MultiAgent characteristics.

1 Introduction

Most of complex applications are distributed and cooperative by nature and must consider the management of limited resources as the network, databases, processors, etc. Agents have shown to be useful for developing such kind of applications [1], however design problem is still complex mainly because still that today there is no methodology ensuring a reliable design of systems. Some work has been done in this direction but most them are adaptation of methodologies developed from other paradigms mainly objects [2], in the other hand Kinny and Georgeff [3] proposed a methodology for Multi-Agent Design, however this methodology lacks of mechanisms to define the agents interaction. Jennings and Wooldridge describe in [1] that agent oriented paradigm is genuine advance over the current methodologies for engineering complex systems. They also propose a life cycle for agent-based software construction, this life-cycle contains tree phases: specification, implementation and verification, however they don't define the complete process of the specification phase.

The methodology consists of tree phases: Specification, Implementation and Verification. The specification phase has the following principal purposes: Understanding the system to solve, determining the system characteristics, including the information that the system should to produce; also the operative characteristics to define the system to solve. The implementation phase takes as incoming the specification of the problem solve, in order to translate this specification into a concrete computational form. The verification phase define an algorithm to determine if we are constructing in a correct way the solving system.

This article is organized in the following way: section 2 describes the methodology, section 3 presents an application, and finally in section 4 we expose our conclusions.

2 The Methodology

The Methodology for MultiAgent Systems (MAS) is a systematic procedure, which takes the agent concept as its fundamental element. With this concept we create a set of models oriented to agent paradigm, with which it is possible: to consider, to abstract and to solve in a natural and manageable mode the currently complex real problems. Where this complex problems may arrive due to: its importance, its nature or for the industry demand to implementing they.

The importance of this methodology resides in its potential to cope the problems of interoperability between agents immersed in distributed and cooperative contexts. Figure 1 illustrate in general way the phases considered for proposed methodology. Next we delineate briefly these methodology phases.

The specification phase has the following principal purposes: Understanding the system to solve; determine the system characteristics,

J. Díaz de León, G. González, J. Figueroa (Eds.): Avances en: Ciencias de la Computación, pp. 64-106, 2003.

including the information that the system should to produce; and the operative characteristics to define the system to solve. This phase embraces the following steps:

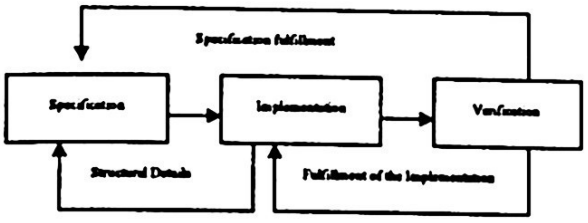

Fig. 1. Methodology phases

- Design perspective
- Develop an statement of the problem boundary
- Identify the agents functional and organizational roles of the application domain.
- Identify the responsibilities, time of life and services associated to each role identified previously
- Elaborate a hierarchy of agents' subsystems
- To build the agent model
- Develop the agents' formal model
- To define the agents' interaction model

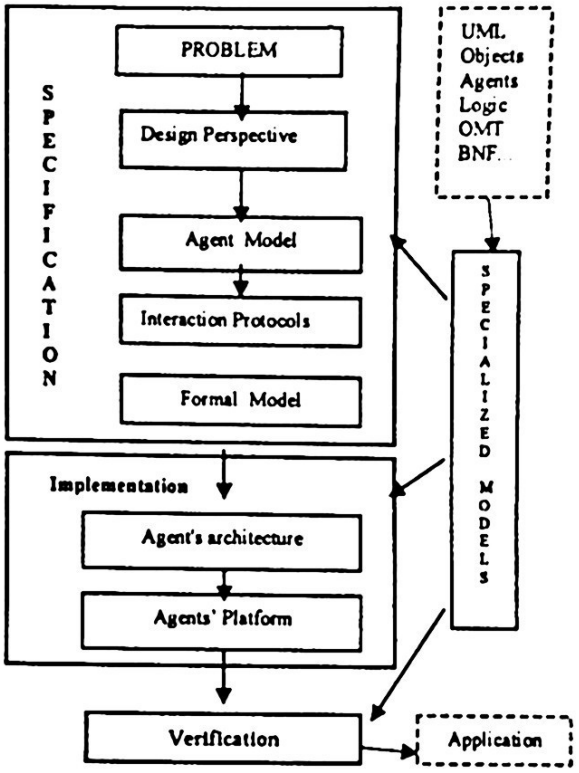

Fig. 2. Methodology phases in detail

The implementation phase take as incoming the specification of the problem to solve, in order to translate this specification into a concrete computational form. This phase enclose the next steps:

- To define the agent's architecture
- To build the platform of agents
- Develop the application

The verification phase consider the process to assure that an implemented system is correct with respect to its specification. In this phase we regard the following two kinds of verification:

- Life cycle verification
- Formal verification

A significant part of efforts in verification phase, are focused to life cycle verification, as we will discuss later in section 2.3.

In order to illustrate the methodology process, we propose as the problem to solve, an application

oriented to e-commerce activities, in particular we intend developing a computational system that makes viable the purchase and sale of products, using the facilities and services offered by the Internet and the web. Figure 3 exhibit a graphical description of the problem context to resolve (conceptual model), this description help us to visualize in a first instance the problem boundary and the agents involved in the problem solution.

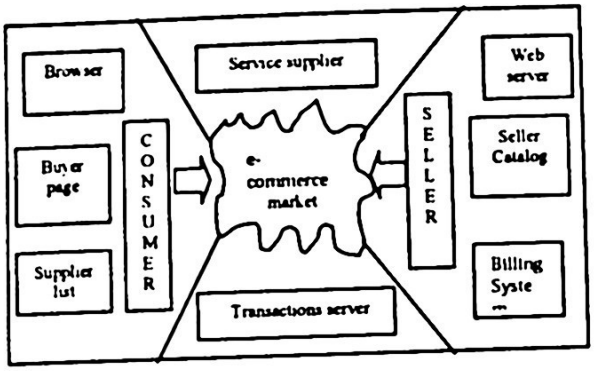

Fig. 3. Conceptual problem description

Already defined the problem, we next define shortly the methodology steps.

2.1 Specification Phase

The specification phase enclose the following aspects: the design perspective, the agent model, the formal model and interaction protocols.

2.1.1 Design perspective

In the design perspective we should know *what* to solve and *how* to solve it. To achieve it: we have to define the characteristic that the system should to exhibit; moreover, to identify an determine the involved agents in the resolving problem. Also, determine the agents' organization schema and the activities they carry out inside the system, in order to define in a explicit way the problem to solve.

An important element used as reference in this phase and through all the methodology is the agents' page. The agent's page is a table which contains information about the agents as: the agent identifier, agent purpose, agents' roles, agents' responsibilities, etc. The table 1 shows an schema for an agents' page.

Table 1. Agents' page. The agents' page contains all the necessary information about the agents involved into the system and the function they carry out.

Agent Attribute	At_1	At_2	At_3		At_{n-3}	At_{n-2}	At_n
Id-Agent$_1$				...			
Id-Agent$_2$				...			
Id-Agent$_3$							
....	...	...	...	...	...	...	...
Id-Agent$_{n-2}$				...			
Id-Agent$_{n-1}$				...			
Id-Agent$_n$				...			

The agents are characterized by an only one identifier and the agents' function is specified by attributes, where some of these attributes are the agents': roles, responsibilities, purposes, URL address, offered services, relationship with, etc.

The information contained in this table may be manipulated using the basic relational algebra operations see [4] for extensive reference, In order to satisfy the requirements of subsequent phases.

2.1.2 Agent model

After to obtain an schema representing the agents organization for problem to solve, we use a standard tool to modeling the obtained organizational structure, in order to obtain a system model easier to understand and to implement. We propose in this methodology the Unified Modeling Language (UML) as the standard tool used to obtain a model representing the system general structure. This model obtained is called the agent model.

The agent model captures the system static structure, showing the agents involved into the system, the relationship between they and the attributes characterizing them. The agent model is a conceptual model, used to make a hierarchical description of the relationship between the involved agents into the problem to solve. The Agent model for the e-commerce proposed application, will be illustrated later in section 3.

2.1.3 Formal model

The formal model is used to make a description of the functional and operational of the MultiAgent system, also is used to study the consistency that the system should to exhibit before to implanting it.

The formal model is constituted by two basic parts: first, the Agents Meta Language(AML), which is based on belief temporal logic; second, The object language LCIASA [5] which is associated to AML. LCIASA is used to give a syntactical structure to AML similar to a programming language, in order to obtain a system description easier to understand.

2.1.3.1 Agent Meta Language (AML)

AML is language based on logic, for this reason , its difficult the Multi Agent System specification and interpretation. To cope this problem, we create de Object Language LCIASA. This Object Language is used to give an syntactical structure an a semantic interpretation to AML. See [6] for extensive reference.

In this section the formal AML model and the object language LCIASA associated with AML are described. The Object language LCIASA is used to give a computational interpretation to AML. As we'll discuss later in the section 3.

The Agent Logic (AL) represents the base of AML, it gives a formal model to study the characteristics, properties and behavior of MultiAgent Systems, before implementing it as a computational system.

The temporal logic used for AML is a kind of non-classical logic [6,7] where the time model offers the basis to describe dynamical systems. In this paper linear discrete sequences in time are used as a basic temporal model. Using this temporal model it is possible to define an agent as the 4-tuple:

def

$A = < \beta^{\circ}, A, M, \eta >$

where:

$\beta^{\circ} \in BS$ is the agents' initial belief set..

A : $BS \rightarrow Ac$ is the agents' action set.

M : $BS \rightarrow \wp(Mess)$ is the agents' message generation function.

η : $BS \times Ac \times \wp(Mess) \rightarrow BS$ is de agents' next state function.

Using the previous agents' definition it's possible to specify agents' characteristics and observe their behavior and state changes through actions executed by the agent. Taking agents' initial beliefs an agent selects an action to execute using the function A and some messages to send using the function M. The function η then changes the agent from a state to another, on the foundation of received messages and the actions it has just achieved.

On the basis of the previous agents' definition, we give the Multi-Agent definition as an indexed set of such agents:

def

$MAS = \{ < \beta^{\circ}_i, A_i, M_i, \eta_i > : i \in Ag \}$

The linear time temporal belief logic is a branch of modal logic, it is associated with time situations development, that is, dynamic situations where the formulae interpretation is accomplished using frames with linear accessibility relations.

The AL [6] is a propositional logic and contains tree atomic operators: "Bel" to describe agents' Belief, "Send" to describe the messages that agents sends and "do" for describing the actions that agents execute

Likewise, AL contains a set of modal temporal operators, which allows the description of agents' dynamic properties

– *Syntactic rules for AL*

AL allows understanding the agents' Belief, actions and the messages they send. The language AL contains the next symbols:

- The symbols { True, Bel, Send, Do }
- A countable set of constant symbols Const made up of disjoint sets $Const_{Ag}$ (agent constant) y $Const_{Ac}$ (action constant).
- All the closed formulae of internal language L
- The unary propositional connective $\neg$ and the binary connector $\vee$
- The unary temporal connectives { O, $\otimes$ } and the binary temporal connectives { μ, ξ }
- The punctuation symbols: (), (}.

The set of well-formed (wff) formulae of AL based on internal language L are defined by the next rules:

1. if *i,j* are agent ids, ϕ is a closed formulae of L and α in an action constant then the next are atomic formulas of L:

   true (Bel i, ϕ) Send(i, j, ϕ) (Do i, α)

2. if ϕ and ψ are formulas of AL, then the following are formulas of AL:
   $\neg\phi$ $\phi \vee \psi$
3. if ϕ y ψ are formulas of AL, then the following are formulas of AL:
   $O\phi$ $\otimes\phi$ $\phi \mu \psi$ $\phi \omega \psi$

The first rules deal with atomic formulas of Al, the formula "true" is a logical constant for truth, it's always satisfied. The formula (Bel i, ϕ) is read: agent i believes ϕ. The formula (Do i, α) is read: agent i performs the action α. The formula Send(i, j, ϕ) describes the sending of messages and will be satisfied if agent i has sent j a message with content ϕ.

– *Semantic rules for AL*

The satisfiability relationship ($\models$) for the LA, is given among even in the way < M, u> and formulas of LA. Given a model M and a temporary formula ϕ, then we say that ϕ is satisfied in a position u≥ 0 of M and we denote it by : < M, u>$\models$ ϕ [7].

In order to give a representation and interpretation to AL expressions and also to verify that a Multi-Agent specification is accurate, are used the next semantic rules, see fig. 4.

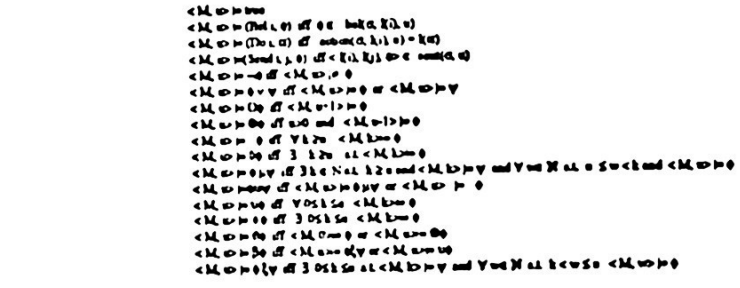

Fig. 4. Semantic rules for AL

2.1.3.2 LCIASA description

LCIASA corresponds to the Object Language. This language provides the syntactic structure and the semantic interpretation to the actions expressed in the AML. That is to say, with LCIASA a computational interpretation is given to the formal agents model proposed in AML. See [8] for extensive references.

LCIASA is formed for a set of messages of KQML [9] and a set of compound actions. This Compound actions are formed by grouping the KQML messages in order to satisfy a pre-defined function The fig. 5 describes LCIASA as a higher layer than KQML. LCIASA take messages from KQML to form its sentences.

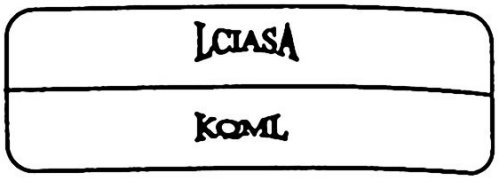

Fig. 5. Dependency of LCIASA respect to KQML

The LCIASA Compound actions let us to authenticate each one of actions involved into agents interaction processes in order to guarantee the security and integrity of specified systems. That is, using LCIASA we are able to develop robust applications immersed into open distributed contexts.

LCIASA syntax. The syntax is considered as the group of rules that generate the correct sentences of the language, including particulars like the placement of words and the punctuation signs. In this paper the BNF notation is used for the description of the language.

LCIASA corresponds to the Object Language. This language provides the syntactic structure and the semantic interpretation to the actions expressed in the AML. That is to say, with LCIASA a computational interpretation is given to the formal agents model proposed in AML.

The interaction mechanism among the agents is carried out by means of the exchange of messages, which are grouped to form a plan, this plan is perceived as a sequence of messages to realise a certain task.

The syntax of LCIASA is based on the plan concept. The general structure for the syntax of LCIASA is presented in the fig. 6.

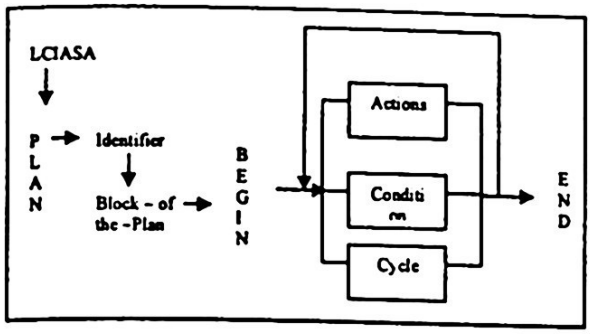

Fig. 6. General structure of LCIASA syntax

The fig. 7 sample in a general way the syntax of LCIASA, which is based on the message concepts and of plans. Each plan has an identifier that characterizes it, a block-of the-Plan that contains the attributes of the plan, its pre-conditions, the body of the plan and its post-conditions. Next, the notation BNF is used to express the syntax of LCIASA

<LCIASA> ::= <Plan>
<Plan> :: = <Heading - of the - Plan> Begin <Block - of the - Plan > End
<Heading - of the - Plan> ::= Plan <Identifier>
<Block - of the - Plan > ::= Begin - Plan [<Pre-conditions >] [<Declaration- of-Variables>] <Body - of the - plan > [<Post-conditions >] End - Plan
<Pre-conditions> ::= <Attributes - of the - Plan> <Conditions - of - Invocation >
<Attributes - of the - Plan> ::=Available | Reserved | Activated | Disabled
<Condition - of - Invocation> ::= <Condition - of - Activation> | <Event - of - activation>
<Condition - of - Activation ::= <Action > | <Expression >
<Event - of - activation ::= <Action> | <Expression >
<Declaration - of - Variables> ::=Variables < Lists - of - Identifiers > ; <Type >
<Lists - of - Identifiers> ::= <Identifier > {, <Identifier > }
<Type> ::= <Integer > | <Agent > | <Real > | <Boolean >
<Integer> ::= <Digit > {<Digit >}
<Agent> ::= <Id- Agent >
<Id- Agent ::= <Identifier>
<Real> ::= <Integer > , <I Integer>
<Boolean> ::=True | False
<Body - of the - plan ::={<Actions> }
<Actions> ::= <Action - AML > | <Action - Ordinary >
<Action - AML ::= Send (<Agent > <Agent > <Action - Simple >
| Bel (<Agent> , <Expression - Boolean >) | Request(<Action - Simple >)
| Do (<Agent> , <Action - Simple>) | Inform(<Action - Simple >)
<Action - Ordinary> ::= <Action - Simple > | <Action - Structured >
<Action - Simple> ::= <Action - Primitive - KQML> | <Action - Multiple >
<Action - Primitive - KQML> ::= <message >
<Action - Multiple> ::= <Actions - Compound>
<message> :: = <Informative> | <Base - of - Data > | <Responses> |<Notification> | <Connectivity <| <Definition - of - Capacities <| <Facilities> | <Query>
<Of - Effect < | <Multi-response> | <Generating>
<Informative> ::= Tell | Deny | Untell
<Base - of - Data> ::= Insert | Delete | Delete - One | Delete - All
<Answer > ::= Error | Sorry
<Query> ::= Evaluate | Reply | Ask-if | Ask-about | Ask-one | Ask-all
<Multi-response> ::= Stream-about | Stream-all | Eos
<Definition - of - Capacities> :: = Advertise
<Notification> :: = Subscribe | Monitor
<Connectivity> ::= Register | Unregister | Forward | Broadcast | Pipe | Break
| Transport-Address
<Facilities> ::= Broker-one | Broker-all | Recommend-one | Recommend-all | Recruit-one
| Recruit-all
<Of - Effect > ::= Achieve | Unachieve
<Generating> :: = Stand by | Ready | Next | Rest | Discard | Generator
<Actions-Compound>::=<Actions-of - Agent>| <Authorization> | <Encoded >
| <Authentication >| <Verification >|<Operations-in -the - VKB>
<Actions - of - Agent> ::= Finds - Solve - Message | Locations - of - Information
| Locate - all - Solutions | Respond - with - the - Solutions | Analyze - Plan
<Authorization> ::= Visualize - VKB | Confirm - Signs - Digital | Validate - Agent

<Encoded> ::= Dispose - Message | Protects - Message
<Authentication> ::= Authenticates - Agent | Authenticates - Originator | Authenticates - Message | authenticates - Receiver
<Verification> ::= Checks - Plan | Verifies - Order - of - Message
<Operations - in - the - VKB> ::= Admits - Belief | Eliminates - Plan | VKB Incorporates | Omits - Belief | Add - Plan
<Action - Structured> ::= <Action - Conditional >| <Action - Repetitive>
<Action - Conditional> ::=If <Expression > Then <Actions> Else <Actions>
<Action - Repetitive> ::= While <Expression> Do <Actions>
<Expression>::= <Identifier > <operator> <Identifier >
<Post-conditions> ::= <Conditions - of - abort >| <Action - of Failure>
| <Action - of - Approval >
<Condition - of - abort> ::= <Action - Simple>| <Expression >
<Action - of Failure>::= <Action - Simple > | <Expression >
<Action - of - Approval> ::= <Action - Simple >| <Expression>
<Identifier>::= <Letter> {<Letter>| <Digit >}
<Letter> ::=A | B |... | Y| Z | a | b |... | y | z
<Digit> ::=1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0
<Operator-arithmetic> ::= ± | - | • | /
<Operator - Boolean ::= ⊻ | △ | ⊐
<Operator - Relational ::= ≤ | ≡ | ≥ | ≧ | ≦
<Integer> ::= <Digit>{<Digit >}
<Expression - Boolean>::= <Action - Simple>[{<Operator- Boolean ><Action - simple> }] =<Boolean>
<Agent> ::= <Id- Agent>
<Id- Agent ::= <Identifier>

Fig. 7. LCIASA Syntax

In the specification of the LCIASA syntax, is perceived the way in which LCIASA is associated with AML. This association is perceived observing first: that the actions of AML correspond to the messages of LCIASA. Second that LCIASA provides a syntactic structure to AML.
With the purpose of providing an additional mechanism to facilitate the activity in the system and with the purpose of having an additional mechanism to guarantee the security and integrity of the Information exchanged among the agents. The messages that correspond to the primitive actions of KQML are grouped, to form Compound Actions. These Compound Actions have the purpose of facilitating the specification of the applications of MAS.
To facilitate the understanding of the compound actions, they are divided in six functional groups: Actions of Agents, Authorization, Encoded, Authentication, Verification and Operations in the VKB. As it is observed in the specification of the LCIASA syntax. As shown in fig. 7.

2.1.4 Interaction protocols

An interaction protocol is considered as the set of rules, followed by the agents to interchange a message sequence between them in order to achieve a specific objective. The interaction protocols govern the interchange of a sequence of message between the agents, that is, the interaction protocols control the dialog hold up by the agents to solve a task in a cooperative way [10].
The methodology deals with the next interaction protocols: Coordination, Cooperation and negotiation. The Coordination protocols let us to obtain the required coherence and actions control in the problems solution. In this paper we use the marketing mechanisms, see e.g. [11] for extensive references, which are used to obtain both: an agents organizational structure and the actions control mechanisms The cooperation protocols define the mechanisms to support that agents may work together in a task solution. In this paper we use the cooperation protocols based on contracts. The negotiation protocols define the elements to make possible that a set o agents arrive to a mutual agreement state.
To implement the MultiAgent system we propose an interaction protocol which take as basis the previous protocols and the LCIASA agent communication language discussed above. This protocol is called Contract Business Protocol (CB) We give a CB brief description in the following paragraph.

2.1.4.1 CB protocol description

The CB protocol includes: fist, the practices of business: market, sales, chain of supply, service to clients, information services, etc. And the necessary resources to maximize the commercial value of the www [11,12]; where these commercial practices are subject to standard of business. Second, coordination protocols. That is to define the CB properties and functionality we join together: Coordination protocols (contract net), coordination structures (centralized markets) and standards of business (OBI standard [13]). In order to obtain a protocol which embraces a coordination structure to define the agents' organization, a mechanism to govern the activities in the problem solution and the rules attached to de commercial practices to achieve a commercial transaction.

The main function of the business markets consists on presenting to a possible buyer the smallest price, which somebody is willing to sell, and to introduce to a possible salesperson the highest price, which somebody is willing to buy.

In the specification of the LCIASA syntax, is perceived the way in which LCIASA is associated with AML. This association is perceived observing first: that the actions of AML correspond to the messages of LCIASA. Second that LCIASA provides a syntactic structure to AML.

With the purpose of providing an additional mechanism to facilitate the activity in the system and with the purpose of having an additional mechanism to guarantee the security and integrity of the Information exchanged among the agents. The messages that correspond to the primitive actions of KQML are grouped, to form Compound Actions. These Compound Actions have the purpose of facilitating the specification of the applications of MAS.

To facilitate the understanding of the compound actions, they are divided in six functional groups: Actions of Agents, Authorization, Encoded, Authentication, Verification and Operations in the VKB. As it is observed in the specification of the LCIASA syntax. As shown in fig. 7.

2.2 Implementation phase

This phase study the change from the abstract specification to a concrete computational system. This phase embrace the agent architecture and the agents platform.

2.2.1 Agent's Architecture

The architecture captures the behavior of the individual agents. The agent architecture defined for the agents considers the mechanisms to make possible MultiAgent System construction so that, this systems satisfy the specifications given by the theoretical model.

The agent architecture used in this paper is a kind of classical architecture called deliberative architecture which contain an explicit symbolic model of the world, and decisions are made through a logical reasoning, see e.g. [14] for extensive references. For our proposes we develop an BI agent architecture which is a subset of BDI architectures [15].

This phase study the change from the abstract specification to a concrete computational system. This phase embrace the agent architecture and the agents platform.

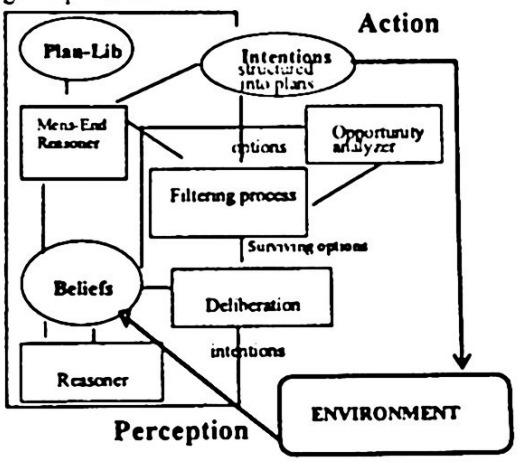

Fig. 8. Agent architecture

The figure 8 illustrated the agent BI architecture used for describing the system internal functionality in the proposed methodology. See [5] for extensive details.

2.2.2 Agent's Platform

The agent platform is a framework which lets us: create, execute, transfer, finish and group agents in order in order to implement a physical system. The agent platform is constructed in function of: the elements considered in the specification phase, specifically the interaction protocols and the agent architecture. We use as support to construct the agent platform the facilities an services offered by CORBA [16]. Aguirre P. Z.S. has developed an agent platform and the application for the problem that we have proposed in section 3. See e.g. [17] for extensive details.

2.3 Verification phase

The verification is the process to exhibit that an implemented system is correct whit respect to its specification. Formally a system verification involve a test in a mathematics sense to ensure that the system is correct. In this methodology we use: firstly; life cycle verification mechanisms. Secondly; formal verification tools, e.g. the Rao and Goergeff algorithm [18] to determine if a formula is valid into the model representing the MutiAgent System. Others formal verification instruments are discussed later in section 2.3.2.

2.3.1 Life cycle verification

Life cycle verification is the process to determine the satisfaction value of a MAS, for an established specification, during each one of developing phases. Fig. 1 illustrate the activities considered for life cycle verification process.

The assurance of liveness and security properties [18] of developing MAS, constitute the main activities considered during life cycle process. We in this paper take the following actions to guarantee these properties:

First; We define into LCIASA a set of actions to confirm that executed actions by the MAS been those we are expecting. As we discussed above LCIASA contains actions groups for actions authentication, authorization and security. *Second*; we impose restrictions into interaction protocols in order to check each one of the actions validity. *Third*; Assure that actions sequences corresponding to an accorded specification posses an appropriate syntactic structure and conclusion.

2.3.2 Formal verification

Formal verification In the same way that it's possible to use AL for Multi-Agent specification, also it's feasible to use it for Multi-Agent verification. Generally the verification is considered as process more complex than specification. Formally system verification involves proving in a mathematical sense that it's correct. The verification proofs used for our propose are: the application of modal semantic trees. The algorithm given by Rao and Georgeff [18] for determining whether a formula is valid in a model. The SAT algorithm, given by Manna and Pnueli or using the automatic theorems demonstrator STEP, developed by the Stanford University, USA [19].

3 Case of Study

In order to illustrate the methodology utility, we develop briefly an application oriented to electronic commerce, this problem was proposed above in section 2.

3.1 Perspective design

As we discussed above design perspective require initially the problem definition. The conceptual problem description was depicted above in fig. 3. Next we give a problem description in detail.

3.1.1 Problem description

The electronic commerce is the process dealing with the acquisition and buying of a product through the Internet [11,12]. For our purpose we'll use the CB protocol framework for Client-to-business internet commerce solutions. that is, a selling retail which is given between a final consumer and an enterprise.

In this section, we obtain a model which represent the problem solution, as discussed above in section 2.1. Fig. 9 represent a dialogue being held by agents. The purpose of this agents dialogue to acquire a product through a messages interchange, to obtain a concordant solution.

The interaction is presented by a Dooley graphic [20], modified by Parunak. In order to simplify the actions sequence only the principal actions are considered.

the graphic representation are listed two kinds of agents: an client agent (Purchaser) denoted by A, and three suppliers agents denoted by B, C, D (Sellers). A_1, B_1, C_1, D_1 are subsequent states of A, B, C and D respectively.

In our application we used the CB Protocol. In the functionality of CB protocol we identify the next four stages (discussed above): *First* find sellers, steps 1-9 in Fig. 9, In this phase the client explores the products' description of a list of suppliers and in function of the parameters like the price, cost, quality of the product, time of delivery and quality of service, selects to the provider most appropriate.

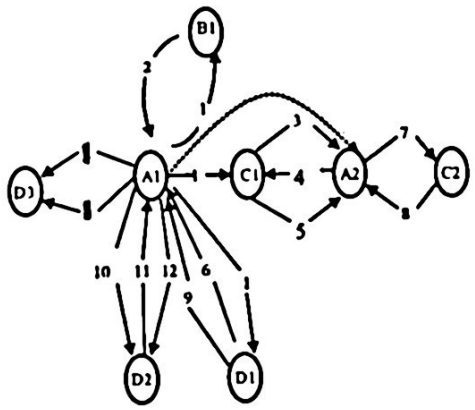

Fig. 9. Problem description model

Second term agreement, steps 10-11 in Fig. 9, In this step the buyer revises the salesperson's catalogue and he makes an order, he generates and fills the pre-purchase form, the salesperson calculates the taxes, shipping and handling and he sends the OBI petition to the buyer, the buyer approves the order, he carries out the payment and he sends the complete order to the salesperson.

Third supply the order, step 12 in Fig 9 In this stage the salesperson requests the credit authorization to the bank, he registers and notifies the order, if he has the authorization of the credit and the product is available the salesperson orders the product shipment to the client. *Fourth* support the client, steps 13, 14 in Fig 9. In this stage the order and the client personal data are registered to take it like reference in future operations

Table 2. Description for agents' actions

Seq	Actions
1	Send (A, B, Request(α) ∧ (Bel B friend(A))
	Send (A, C, Request(α) ∧ (Bel C friend(A))
	Send (A, D, Request(α) ∧ (Bel d friend(A))
2	Send (B, A, Abort(α))
3	Send (C, A, Request(tell(x))
4	Send (A, C, Request(replay(x))
5	Send (C, A, Inform(ϕ)) ∧ (Bel A, ϕ)
6	Send (D, A, Inform(ϕ))
7	Send (A, C, Request(subscribe(x))
8	Send (C, A, Abort(α)) ∧ (Bel A, ϕ)
9	Send (D, A, Request(tell(x))
10	Send (A, D, Request(replay(x))
11	Send (D, A, Request(tell(x)) ∧ (Bel a friend(D))
12	Send (D, A Request(tell(x)) ∧ (Bel a friend(D))
13	Send (D, A, Request(insert(x)) ∧ (Bel a friend(D))
14	Send (D, A, Request(insert(x)) ∧ (Bel a friend(D))

The interaction operations exhibited in Fig 9, are described in Table 2. This table [21] shows the actions performed, the participant agents, and a short operations description. Table 1 presents the sequence of actions, the activities performed by agents, which are specified using LCIASA [5].

3.1.2 The agent model

The *Agent Model* (AM) makes a hierarchical description of the relationships among the different agent class, be already these abstract ones or concrete (instances). Also it identifies the agent's instances, their multiplicity and when they end up. As it's depicted in figure 10.

The agent model has two components: first A model of agent's class that defines the abstract and concrete (instantiable) classes and it captures the inheritance and aggregation relationships among them. Second A model of the agent's instance that identifies the agent's instances and their properties.

The agent model has two components: first A model of agent's class that defines the abstract and concrete (instantiable) classes and it captures the inheritance and aggregation relationships

among them. Second A model of the agent's instance that identifies the agent's instances and their properties.
The diagram is represented using UML [22] notation, it contains abstract and concrete class, abstract class are distinguish by the presence of the O symbol in the upper section of the icon. The edges in the graph that denote the inheritance are distinguished by a triangle with the vertex pointing toward the superclass and the edges that denote aggregation for a diamond adjacent to the aggregate class.

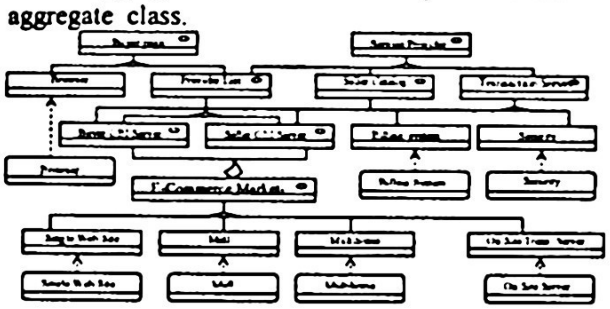

Fig. 10. Agent model for e-Commerce Agent application

For example, in fig. 10 Transaction server is an abstract agent which has the next functions: manages the payment process, handles credit and debit-card transactions, using secure electronic standard technology, handles the necessary authorization requests and recording of the transaction. Service provider is an abstract agent which provide internet access, Billing systems based on customer activity and advertising, site activity reports extraction and communication of orders. Buyer page is an agent which offers a suppliers list, captures buying options, it's responsible to locate any product. The browser agent provides access to the web, contacts web servers, moreover receives, interprets and exhibits HTML pages. Buyer OBI server agent, receives petitions from de seller, sends petitions to seller. Service provider agent, creates web sites, updates the catalogs, provide security also provides linking systems. Seller catalog agent, gives access to web page, gives an articles list also displays selling conditions. Seller OBI server agent, receives petitions from buyer and sends petitions to buyer

3.2 Application implementation

As mentioned above the problem implementation is done using de CB protocol and LCIASA language. In this paragraph we present the implementation of the CB protocol for the find sellers phase.
Next, we will expose the Find Suppliers phase, using, first: a graphic representation and second: using LCIASA to describe the actions associated these phases of the CB protocol.
The phase find suppliers: In which, the Client (Buyer) explores the description of products of a list of suppliers (Salespersons) and in function of parameters like the price, cost, quality of the product, cheat of delivery and quality of service, he selects the supplier is represented graphically in fig. 11, in the following way:

In the graph 11, the purchaser agent (P) announces the product that it wants to buy, using the agent mediator (F) (steps 1,2). The supplying agents with possibility of offering the requested product, offer the product (step 3), the suppliers offer their parameters of selling (step 4) and finally the purchaser agent selects to the most suitable supplier (step 5).

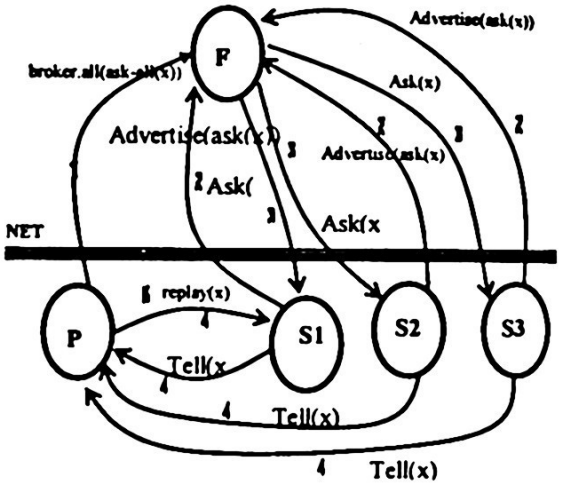

Fig. 11. Graphical Representation of the find suppliers phase

Next, we will use LCIASA [5] to represent the sequence of actions for the find suppliers phase implementation. As it's illustrated by the fig. 11:

Plan Find-suppliers
Begin
Variables Catalogue, P, S1, S2, S3, F : Agents
Preconditions: KQML protocol
Begin–Plan
While ((Catalogue = True) ∧ ¬Eos)
Send (P,F, recruit-all (ask-all(x))) ∧ (Bel F Authenticates-Originator (P))
Do (P, recruit(ask-all(x)))
Send (S1,F, Advertise (ask (x))) ∧ (Bel S1 Authenticates-Originator (S1))
Do (S1, Advertise(ask (x))
Send (S2,F, (Advertise (ask (x))) ∧ (Bel S1 Authenticates-Originator (S2))
Do (S2, Advertise(ask (x)))
Send (S3,F, Advertise (ask (x))) ∧ (Bel S1 Authenticates-Originator (S3))
Do (S3, Advertise(ask (x))
Send (F,S, broadcast(ask (x))) ∧ (Bel S2 Authenticates-Originator (F))
Do (F, ask(x))
Send (S1,P, Tell(x)) ∧ (Bel P Authenticates-Originator (S1))
Do (S1, Tell(P))
Send (S2,P, Tell(x)) ∧ (Bel P Authenticates-Originator (S2))
Do (S2, Tell(P))
Send (S3,P, Tell(x)) ∧ (Bel P Authenticates-Originator (S3))
Do (S3, Tell(P))
If (Val(S1) >((Val(S2) and Val(S2))) then
Send (P, S1, Tell(x)) ∧ ((Bel S1 Authenticates-Originator (P))
Do (P, Tell(x)
Do (Cheks-Plan)
End –Plan
End

In this description, we use the properties of liveness and safety [23, 24] to assure the consistency of the sequence of actions corresponding to this phase of the CB protocol.

4. Conclusion

In this article we present a formal methodology to develop correct complex applications based on agents. As shown in the previous sections, the proposed methodology allows an specification more expressive, trusted, legible and easy to use and understand, it facilitates the system's analysis and design through the application of a set of well-defined steeps. This methodology is suitable to develop cooperative applications as: selling-buying operations, bank transactions, electronic commerce and industrial applications.
The proposed methodology has the following advantages: first, has associated a formal model, which makes possible to specify and to validate the actions characteristic of the system. Second, the methodology uses the modeling techniques and the protocols accepted as standards for SMAs. Third, the implementation of our case of study shows that using CORBA [16] for communication facilitate the legacy, the interoperability and the implementation of new applications.
We are working on the verification process, using the STEP demonstrator [19] and Rao & Georgeff Algorithm [18]. We are also currently working on the problem to find the most suitable theoretical model and the interactions protocols for a specific problem.

References

1. Jennings N.R. and Wooldridge M. Agent-Oriented Software engineering, proceedings of the 9th European Workshop on modelling autonomous agents in Multi-Agent world: Multi-Agent System engineering (MAAMAV-99)..
2. Jennings N.R., On agent-based Software engineering, Journal Artificial intelligence, volume 177, number 2, pages 277-296, year 2000.
3. Kinny D. And Georgeff M. , *Modelling and Design Multi-Agent Systems*, Intelligent agents III, Muller, Wooldridge, Jennings (Eds), Springer pp.1-20, 1996.
4. Korth, Silberschatz, Database System Concepts, Mc Graw-Hill , pp 64-81, 1991
5. Farías M.N., LCIASA:Lenguaje de "Capacidad de Interacción" de Agentes en Sistemas Abiertos. Msc. Thesis CINVESTAV-IPN, Méxi., 2000.
6. Wooldridge,M., "Chapter 10- Temporal Belief logic for modelling Distributed Artificial Intelligence systems," Foundations od Distributed Artificial Intelligence, G.M.P. O'Hare and N.R. Jennings (esd), Jhon Wiley & Sons Inc, pp 269-285 1996.
7. Torres González, R.E., "Lógica Modal y lógica temporal: Apuntes del curso de LOGICA", *Cinvestav-IPN, Unidad Académica Guadalajara, Jal*, México 1998.
8. Farías M.N., Ramos C.F.F., LCIASA:Lenguaje de "Capacidad de Interacción" de Agentes en Sistemas

Abiertos: LCIASA, *ELECTRO 2000* technologic Institute of Chihuahua. Published in proceedings, pp. 337-343., Méx. 2000.
9. Tim Finn, Don Mc Kay, Rich Fritzon, AN OVERVIEW OF KQML: A_KNOWLWDGE QUERY AND MANIPULATION LANGUAJE, University of Maryland, Baltimore MD 21228
10. Barthes, J.P. MultiAgent Systems, International Symposium on Advanced Distributed Systems, Guadalajara, Jal. Mex. 2000.
11. Malone W.T., Modelling coordination in organisations and markets, Manage sci, 33 1317-1332.
12. Maldonado T.A. , TUTORIAL: Comercio Electrónico Interempresarial, ISADS 2000, Guadalajara, Jal. Méx. 2000.
13. Open Buying on the Internet (OBI), Technical specification release, v2.0 from OBI Consortium, 1999.
14. Jennigs, N.R., " Specification and Implementation of a Belief Desire Joint Intention Architecture for Collaborative Problem Solving," *Journal of Intelligent and Cooperative Information Systems* 2 (3), pp. 289-318, 1993.
15. Haddadi A., Sundermeyer K. " Chapter 5 –Belif-Desire-Intentions Agent Architectures , " *Foundations Of Distributed Artificial Intelligence*., G.M.P. O'Hare and N.R. Jennigs (eds) Jhon Wiley & SonsInc., pp 169-186 1996.
16. Pope Alan, The CORBA Reference Guide: Understanding the Common Object Request Broker Architecture, Addison Wesley 1998.
17. Aguirre P. Z.S. , Modelado dinámico de organizaciones para el trabajo cooperativo, Msc Thesis , CINVESTAV-IPN Unit Guadalajara, Méx., 2001.
18. Rao, A.S. & Georgeff, M.P., "A Model-theoretical approach to the verification of situated reasoning systems", Proc. of the 13° Int. Joint of artificial intelligence 318-328, Chambery, France..
19. STEP Automatic Theorem Demonstrator, Developed by the Stanford University USA, available from http://rodin.stanford.edu
20. Dooley, R.A., " Appendix B-Repartee as a Graph", an anatomy of speech Notions, pp 384-386, 1976.
21. Cohen, P.R. & Levesque, H.J., " Communicative Actions for Artificial Agents", ICMAS-95: *Proceedings, First International Conference* of MultiAgent Systems, pp 67-72, 1995.
22. Hans-Erick Erikson, Magnus Penker, UML Toolkit, JOHN WILEY & SONS, INC, 1997.
23. Foundation for intelligent Physical Agents. Specifications. 1997. Available from http://www.fipa.org.
24. Weiss G., Multiagent Systems A Modern Approach to Distributed Artificial Intelligence, The MIT Press Cambridge, Massachusetts London England, pp. 79-120, 2000.