Proceedings of the 4th WSEAS Int. Conf. on Information Security, Communications and Computers, Tenerife, Spain, December 16-18, 2005 (pp25-32)

Modeling Communications of Mobile Software Agents Using Petri Nets

Ali A. POUYAN, MEMBER IEEE
Department of computer engineering
University of Birjand
IRAN

STEVE REEVES
School of computing and mathematical sciences
University of Waikato
NEW ZEALAND

ALI HASSAN BEIGI
Department of computer engineering
University of Birjand
IRAN

Abstract: - This paper presents a Petri net approach to modeling communications of mobile software agents in multiagent systems. Mobile agents are conceptualized in Petri net semantics by formalizing in terms of basic agent template. The proposed software agent components constitute the elements and building blocks of the distributed mobile agent systems. A basic agent template is formalized as an entity consisting of a set of actions, a set of rules, which govern the agent communications. A theoretical formal model is presented for designing and describing the communications of mobile software agents in distributed asynchronous network systems. It supports formal reasoning based on Petri nets.

Key-Words: - mobile agents, Petri nets, modeling, communication, multiagent systems, formal methods

1 Introduction

Mobile software agents have emerged as a highly significant paradigm in distributed asynchronous computing, software engineering, robotics, artificial intelligence and industrial control applications. It is, especially, becoming increasingly important as the Web environment is undergoing a transformation into a platform for highly distributed applications such as web-based systems and electronic commerce. Specifically, mobile software agents are a generic network programming paradigm, where migrating software components (computer programs) carry out certain distributed tasks by roaming the heterogeneous network systems.

The concept of mobile software agents [1], [2], or simply mobile agents, evolved from autonomous agents [3] introduced a decade ago as a powerful abstraction for conceptualizing large-scale distributed asynchronous computer network systems [4]. It supports a wide range of different types of computer applications such as electronic commerce, network management, distributed information retrieval, workflow management, real-time conferencing; wireless/cellular based mobile computing and the implementation of telecommunication services. In general, the mobile agent paradigm is considered as a solution to reducing network congestion due to heavy traffic load in the network and managing its complexity.

Mobile agents are executing programs that migrate from machine to machine in a heterogeneous network [5], [6]. They run within agent server programs as logical places referred to as agencies. When a mobile agent migrates to a specific node in the network, its execution is suspended at the original agency. The program code, control information, data and execution status are transferred to the host agency. The mobile agent resumes execution after being re-instantiated at the destination environment. Mobile agents have the ability to prevent or solve problems encountered in the network during their journey, and they have the ability to communicate with other. Mobile agent based software systems have gained wide acceptance as a conceptual framework that provides, among others, the following benefits [7]: more efficient use of communication resources by using much less bandwidth than a conventional correspondent RPC-based client; dynamic load balancing by partitioning a task into components that are distributed across multiple processors; flexible management for software deployment and maintenance; adequate support for interactions with environment and flexible support for disconnected operations.

A fundamental issue in the development of software systems based on mobile agents is the support of formal reasoning and analysis of designed systems [3]. For most real-world applications with a large number of communicating agents, it is fundamental that system behavior exhibits certain desired logical properties such as absence of deadlocks and reversibility or cyclic behavior. Although mobile agent software systems have been investigated by many researchers from different points of view and diverse orientations [1], work in formal analysis of design and communication behavior in distributed systems implemented with mobile agents is still needed. The aim of this paper is to develop a theoretical framework and modeling approach for communication behavior of mobile agents in multiagent systems.

The rest of this paper is organized as follows: Section 2 gives a brief introduction to mobile agents and Petri nets and introduces a formal notion of basic agent template. Section 3 describes the proposed approach for modeling agent communications. In section 4 communications of agents are modeled based on the proposed approach. Finally, a conclusion is presented and a sketch is discussed for the future work.

2 Communications Modeling

In this section, we first introduce mobile agents briefly. Basic agent template will then be defined. To make the paper self-contained, Petri nets (PN) will be defined to be used throughout the paper. Then, the proposed modeling approach of mobile agent communications using Petri nets will be described.

2.1 Mobile Agents

In mobile agent software systems servers and agents are the most fundamental concepts [2]. A mobile agent system includes a number of servers, where various resources and services are provided and computation can take place. Mobile agent paradigm has evolved from two antecedents: client-server model and remote evaluation (REV) model [8]. In client-server model processes resided in the client and server communicate synchronously either through message passing or remote procedure call (RPC) mechanism. In RPC, data is transmitted in both directions between client and server. In REV model, client sends its own procedure code to a server rather than calling a remote procedure [9].

Mobile agents can autonomously visit several hosts without the need for continuously interacting with originating host. Agents can have multiple hops and can be detached from the client without being permanently connected to the originating host. This distinguished characteristic makes mobile agent-based software systems ideal for handling temporary network connections in mobile computing. This makes mobile agents different from applets and from the servlets according to the movement pattern. An agent can visit a number of hosts and it does not need to know the complete itinerary in advance. Furthermore, the routing table of a mobile agent can be changed based on information gathered at intermediate hops during its journey in the network. Two patterns of mobility can be defined based on the state from which a mobile agent resumes execution after migration: weak migration and strong migration [10]. By weak migration, the code and part of the execution state (code and data but no control state) are moved. After migration, the execution resumes from the beginning or from a specific procedure. Strong migration allows the migration of both the code and the whole execution state (code, data and state). Mobile agent resumes execution from the point where it was stopped before migration. Other aspects of mobile agents relating to agent migration can also be investigated based on the scope of the study.

2.2 Petri Nets

We use Petri Nets (PN) formalism to model communication behavior of software systems based on mobile agents. PNs; as a high level graphical specification language, have a sound and mature mathematical foundation. It allows a formal and direct investigation of factors such as resource conflicts, synchronization and concurrency in distributed systems. For quick reference, a brief overview of Petri nets is provided in this section, a more detailed coverage can be found in [11].

A Petri net consists of a structural part and a dynamic part. A PN structure, N, is a four-tuple, N = (P, T, V, F) where P = {p1,p2,..., pn} is a finite set of places, $n \geq 0$. T = {t1,t2,..., tm} is a finite set of transitions, $m \geq 0$ ($T \cup P$ form the nodes of N) $V \subseteq \{(P\times T) \cup (T\times P)\}$ is a set of directed arcs (or a flow relation). $F: V \rightarrow \aleph$ is a multiplicity (incidence) function, $\aleph = \{0,1,2,3\ldots\}$. $P \cap T = \varnothing$ and $P \cup T \neq \varnothing$ $(F\cap(T\times T) = (F\cap(P\times P) = \varnothing)$. A PN structure can be represented as a directed bipartite graph. In a Petri net graph, places are represented by circles and transitions by bars or boxes. Places and transitions are connected with directed arcs. Assignment of tokens to the places of a PN structure is called its marking and represents the state of the modeled system at each time instance. A marking μ of a Petri net N = (P, T, V, F) is a mapping $\mu : P \rightarrow \aleph$. Tokens in a Petri net graph are represented by dots or positive numbers in places. The number of tokens in place p of a Petri net is formally denoted by μ(p). A place $p\in P$ is marked if μ(p)>0, otherwise it is unmarked. A Petri net is marked if a marking function can be assigned to it. The state of a Petri net is defined by its marking. The dynamic part of a Petri net involves the change in markings over time. The initial state is denoted by μ0. The set of all possible markings (states) reachable from μ is called the reachability set. The reachability set of a PN determines the state space of the net system. A transition t is said to be enabled if $\mu(p) \geq F$ (pi, tj), $\forall$ pi $\in$ P. An enabled transition can fire. Firing transition, t removes F (pi, t) tokens from each pi belongs to the set of its input places and deposits F (t, pk) tokens in each pk belongs to the set of its output places. The firing of a transition changes the state of the Petri net.

A Petri net is bounded if $\mu(p) \leq k$, $\forall p\in P$, where k is some positive integer. Boundedness guarantees the stability of the system and lack of overflow. A Petri net, for μ0, is said to be reversible if for each marking $\mu \in$ Z(N,μ0), μ0 is reachable from μ. Reversibility guarantees the repeatability of discrete events (cyclic behavior) of the system. A Petri net is said to be live iff $\forall t\in T$, and $\forall \mu\in Z(N,\mu)$, there exists a firing sequence of transitions leading to a marking which enables transition t. The concept of liveness is related to deadlock situations in distributed concurrent systems. Liveness guarantees a deadlock free situation, which ensures that all the actions, associated with system specification become active. A Petri net that is live, bounded and reversible is called a well-behaved PN.

2.3 Basic Agent Template

In this section we give a definition of agent which is consistent with our approach. Before giving our formal definition, some fundamental concepts must be explained. These notions are “basic agent template”, “action” and “event. The environment of agents can be considered as a composite system made of agents of different kinds. Each agent is a flow of actions processing certain objects, is triggered by events, and changes the state of the system. Communicating agents have characteristics and behaviors that need to be taken into account to correctly model the behavior of the system.

An agent can be defined as an autonomous (having control over its own actions) software entity that is situated within an executing environment. It can also interact (communicate) with its environment and other agents while it is bond to certain predefined task on the user’s behalf. From an object-oriented point of view, an agent is conceptualized as an encapsulated software entity that can send messages to and receive messages from other objects. It has a number of methods to process the messages and change its state as an encapsulated entity. An autonomous agent, as an active object has its own tasks that may be composed of several kinds of sequential or concurrent subtasks to be accomplished. An agent with the property of mobility (migration) between different servers is a mobile agent. The following is a formal definition of a basic agent template:

Definition 1 A basic agent template is a tuple (P, T, V, F, Pc, Tc, μ0), where (P, T, V, F) is a PN structure, μ0 is the initial state of the PN, Pc $\subseteq$ P $\neq \varnothing$ and Tc $\subseteq$ T $\neq \varnothing$, Pc and Tc are called the interface sets of nodes such that (F(tj, pi) or F(pi, tj) $\neq \varnothing$ $\forall$ pi $\in$ Pc and tj $\in$ Tc).

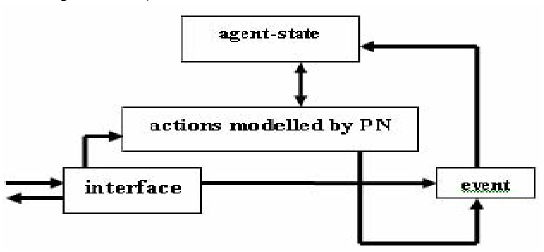

Fig.1. Abstract PN model of a basic agent template

A multi-agent system is a set of communicating agents; each agent is situated in some environment and is able to interact with its environment and with other agents. This definition appears to be adequate for capturing the characteristics of the systems we are dealing with. But we need a formal definition of the concept of agent which is also consistent with our proposed approach. The definition of agent needs to be the one that can be used to compare or to combine different approaches. Moreover, the definition of agent can be used to describe relevant entities uniformly, independently of their physical nature. If a single system model has to represent different kinds of entities, a unique concept of agent provided by the definition is used to uniformly describe conceptual interfaces among them. Using mathematical notions enables us to reach a common interpretation of the concept of agent.

Definition 2 An action is a function $\mathbb{F} : \mathbb{I} \rightarrow \mathbb{O}$, guarded (conditioned) by $\mathbb{P}$ (c1,c2, …,cn) where $\mathbb{I}$ is the input matrix consisting of input vectors, $\mathbb{O}$ is the output vector, $\mathbb{P}$ is a Boolean expression linking predicates c1,c2, …,cn.

$\mathbb{F}$ can imply any functionality; a mathematical function, a transformation rule, an algorithm, etc. Actions in general perform by transforming inputs into outputs. $\mathbb{P}$ indicates some pre-conditions which must be satisfied before $\mathbb{F}$ can be executed. These conditions may be internal or external to $\mathbb{F}$. The definition of action is critical in order to completely define an agent within a multi-agent system. Actions not only define the types of internal processing an agent must do, but also how interactions with other agents relate to those internal processes. This will be of vital importance in our modeling approach.

Definition 3 An event is an instantaneous or an atomic action without time duration which causes a change in the system state.

It is very important to distinguish between an event and an action in our discussion. According to our definitions actions are extended or time-consuming of which duration is bounded between a pair of (start, end) events. During the time interval of an action, other events may occur. Two actions can overlap in time if the start of one precedes the end of the other. Events can be expressed as logical propositions or predicates. It should be noted that events only signal the change in the system state and do not convey any information regarding how the change has been made.

Definition 4 An Agent A is a 5-tuple A ($\mathcal{A}, \mathcal{O}, \mathcal{G}, \mathcal{F}, \mathcal{C}$) where, $\mathcal{A}$ is a set of actions, $\mathcal{O}$ is a set of objects, $\mathcal{G}$ is a finite set of triggering conditions {c1,c2, …,cn} which must be satisfied to cause a state change, $\mathcal{F}$ is a function $\mathcal{F}$: $(\mathcal{O} \times \mathcal{A}) \cup (\mathcal{A} \times \mathcal{O}) \rightarrow \{0,1\}$; $\mathcal{A} \cap \mathcal{O} = \varnothing$, $\mathcal{A} \cup \mathcal{O} \neq \varnothing$; $\mathcal{F}$(oi,αj) = 1 if oi is an input of αj and 0 otherwise. $\mathcal{F}$(αj,oi) = 1 if oi is an output of αj and 0 otherwise, $\mathcal{C}$ is a finite set of communicative acts (rules) governing the agent communications (determine the type and content of messages).

In terms of definition 4 an agent is then described as a "temporal-logical" sequence of actions. That is, as a series of transitions from one (internal) state to another, triggered by events. This set of transitions comes to an end point when a pre-specified terminating state is reached or the state is considered to be final based on certain pro-active actions of an agent. This dimension of the definition captures system functionality. It is the fundamental idea for describing a dynamic system functioning. The other dimension focuses on the behavioral aspects. This captures the flow of information within the system based on proving correctness in transition from one state to the next, while certain conditions are related to each state.

We model agent behavior as consisting of several concurrent actions. Each of these actions can execute in parallel to define the behavior of the agent. Actions are used to specify actual functions carried out by the agent and are performed inside the agent states. Each action may have a set of invariants that must hold during the entire life of the action. Actions are defined in the form of functions. Each function may return a result and may have a number of input parameters.

While these actions execute concurrently and carry out high-level behavior, they can be coordinated using internal events. States encompass the processing that goes on internal to the agent. This processing is specified by a sequence of activities specified in a functional form. Transitions describe communications among agents. To communicate with other agents, external messages can be sent and received.

Semantics of concurrent actions are based on Petri nets. However, because a single agent is specified by a number of concurrent action models, the state of an agent is defined by the set of current states in each of the agent's active concurrent actions. Because activities occur in system states, agents are typically in a state for a finite amount of time. Furthermore, we assume that transitions between states occur instantaneously.

3 Inter-agent Communication

Multi-agent systems can be categorized as information-flow oriented, role oriented and control oriented [12]. In this paper, we focus on event-based information-flow oriented MAS architecture. Information-flow oriented architecture reflects the interactions and communication in multi-agent systems, and the inter-agent communications. It focuses on the capability of the system to deal with complex distributed real-world scenarios, and determines operation mechanisms.

In the previous section we have defined the concept of an event as an instantaneous action, whose occurrence may require simultaneous participation by certain actions as conditions or guards. In this section we shall consider a communication as a member of a special class of events. A multi-agent system is viewed as a concurrent environment. Formally speaking, a multi-agent system is considered as a pair (A, $\mathcal{C}$) in which A is a finite set of agents involved in a system.

$\mathcal{C}$ is the finite set of communicative acts maintaining the interactions between agents. At this stage we are able to produce a precise definition of a message whose type and content is defined by a communicative act.

Definition 5 A message $c \in \mathcal{C}$ is a point-to-point, one-way virtual entity which transfers 'information' from a source-agent $\mathsf{a}s$ to a target-agent $\mathsf{a}t$ such that $\mathcal{C} \subset \mathsf{A} \times \mathsf{A}$.

A communication is an event, which is described by its source and target agents and a signal that conveys information and flow of control. A message is then a virtual means on which a communication takes place, through a communicative act. Mapping this concept to the Petri net arena will be explained in the next section. Now, based on definition 3, a mapping F from $\mathcal{C}$ to $\mathsf{A} \times \mathsf{A}$ can be defined.

Definition 6 The set of rules governing inter-agent communication is a mapping from communicative acts to the set of $\mathsf{A} \times \mathsf{A}$, i.e., $\mathsf{F} : \mathcal{C} \rightarrow \mathsf{A} \times \mathsf{A}$.

Thus, $\mathsf{F}(c)$ = ($\mathsf{a}s$, $\mathsf{a}t$) implies that the rules governing communicative act depend on the relationship between the source and target agents. This implication is extremely important when we establish a 'functional' interdependency in Petri net-based agent models in our methodology. Communicative acts can also be classified based on their properties and impact on the target agents. This is beyond the scope of this paper and is a reason for future work.

4 PN Model of Agent Communications

In this section we provide a Petri net representation based on the definition of agent and other concepts provided in sections 2 and 3. In Petri net-based models places can be viewed as 'mailboxes' and instances of places, i.e. tokens, as messages, portraying a view of a Petri net model as a distributed model of concurrency with a form of asynchronous message passing. In the proposed method the interaction between agents, or message passing, takes place through Petri net structures rather than an arc between two nodes (place or transition) as suggested in other methods in the literature. In other words, inter-agent communications happen as events via certain communicative acts containing the type and information of a message. These rules may be defined according to the structural relationships between Petri net modules that specify the entire model as a set of inter-related components or modules which hide their internal details. The advantage of this method is that the resultant net model of the system has already been extended as a correct Petri net system and there is no need for any posterior analysis while it grows in complexity. This reduces the modeling effort by a major amount. It should be noted that in Petri net modeling when the systems become large, the state-space explosion problem happens, so net system analysis becomes computationally difficult and in some cases impractical. Theoretically, the augmented PN models are guaranteed to be well-behaved regardless of the application domain and the design level [13]. For instance, to host a mobile agent after migration, each host is supposed to provide the execution environment and the facilities for agent activation and deactivation. To accomplish its task, the mobile agent communicates with stationary environment, which consists of resources such as service agents. All these details can be modeled as PN structures, and the describing modules can then be composed and integrated to the PN model at system level. For a better understanding of the theoretical concepts developed in sections 2 and 3, we exemplify our method by constructing a multiagent system for the seller and buyer problem [14] in the domain of electronic commerce. All of the communications between seller and buyer are accomplished by the facilitator agent. This has been depicted in Figure 2.

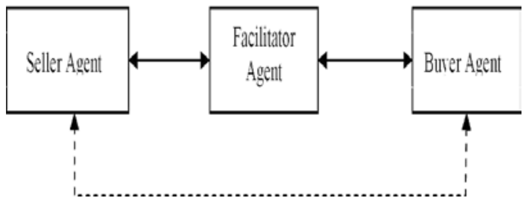

Fig.2. Communications between seller and buyer

The facilitator agent has the role of matchmaker between the sellers and buyers. It manages the marketplace, and includes all kinds of seller agents and buyer agents to interact in the market. The facilitator agent recommends a prospective seller to a buyer, while sellers have already advertised the desire to sell items (products and services) with the facilitator agent. The sellers and buyers communicate indirectly through the facilitator agent, after being introduced in the marketplace. Negotiations between seller and buyer through facilitator could proceed resulting in either a sale (accept) or rejection the offer. The following messages are being communicated through the facilitator during each session of sales negotiation: facilitator is asked by a buyer to recommend a seller for an item, the seller is introduced to the buyer, buyer asks seller if the item is available for sale, seller either makes an offer to the buyer with an initial price, or denies that the required tem is for sale, buyer accepts the offer (and echo the offer back to the seller) or make a counter offer to the seller, if the seller accepts the offer, the sales transaction is over (successful) and if the seller rejects the offer, the negotiation is over (unsuccessful)

Based the proposed method in this paper, we first define the seller, buyer and facilitator agents in terms of definitions given in sections 2 and 3. For each of this three agent type we need to define a PN set (A (**A, O, G, F, C**)).

SELLER AGENT:

Actions: presenting itself to facilitator agent, presenting products to facilitator agent, negotiating with buyer through facilitator agent, accepting buyer's offer or make a counter offer and rejection the transaction.

Objects: Products (items for sale)

Triggering conditions (appears as conditions): Presenting itself to Facilitator agent, seller and buyer start negotiation, acceptation of buyer price, Rejection of the transaction and making a counter offer.

F : (**O** $\times$ **A**) $\cup$ (**A** $\times$ **O**) $\rightarrow$ {1,0} **: input and output function**

Communicative rules

FACILITATOR AGENT:

Actions: listing the sellers and buyers, finding ordered items and its seller, presenting seller and buyer to another, passing offers and message between seller and buyer, presenting the final price to both sides and updating the lists.

Objects: Lists, orders, offers, messages and products (items)

Triggering conditions (appears as conditions): listing the sellers and buyers, finding ordered items and the seller, agreement of seller and buyer, offering a new price from one side, accepting the suggested price by one side and rejecting the transaction by one side.

F: (O × A) $\cup$ (A × O) $\rightarrow$ {1,0} : input and output function

Communicative rules

BUYER AGENT:

Actions: Presenting itself to Facilitator agent, ordering needed items one by one, negotiating with seller through facilitator agent, accepting of sellers offer or make a counter offer, and rejection the transaction

Objects: Orders

Triggering conditions (appear as conditions): Presenting itself to Facilitator agent, generation an order, accepting the seller price, rejecting the transaction and offering a new price. These conditions may appear in forms that can be evaluated as true or false conditions or pre-conditions.

F: (O × A) ∪ (A × O) → {1,0} : input and output function

Communicative rules: determine the connections of the actions modelled by Petri net modules

We know define the transitions in the Petri net model of the seller-buyer problem as follows:

t1: listing the products and orders, t2: ready notification, t3: Finding ordered items, t4: Presenting seller and buyer to each other, t5: ordering an item, t6: telling the price to buyer, t7: analyzing the price by buyer, t8: offering a new price by buyer, t9: accepting of seller's price by buyer, t10: rejecting the transaction, t11: telling the buyer's price to seller, t12: offering a new price, t13: accepting the buyer's price by seller, t14: rejecting the transaction, t15: presenting the final price to both sides, t16: updating the lists.

In this model places appears as triggering conditions and tokens in places of the corresponding Petri net model represents the satisfaction of certain conditions for an event to be triggered. The Petri net model of the seller and buyer problem is shown in Figure 3. Marked places P7,P1 and P8 represent the satisfaction of the condition "existence a list of items", which in turn enables transaction T1 to be fired. Places P2 and P5 represents "ready to receive orders" and "facilitator confirmation", respectively.

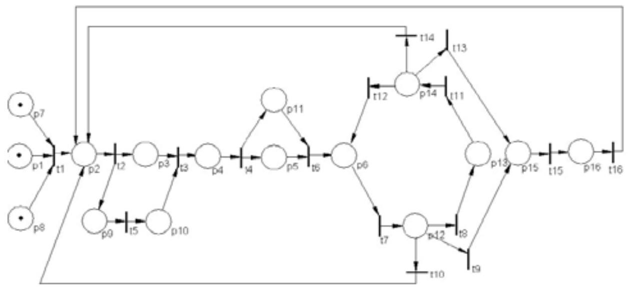

Fig.3. Petri net model of the seller and buyer problem

5 Conclusion

In this paper, we have presented a formal method based on Petri nets to model complex mobile agent communications in multiagent systems. We have formalized relationships between agents by defining a basic agent template. The agent template concept has been defined as an entity consisting of a set of rules, a set of internal actions and interface nodes for agent communications. Constructed Petri net structure has guarantee the well-behavedness of the Petri net based agent model after interacting with other agents. The constructed Petri net models of mobile agents can be expanded by adding details to the designed net systems. This enables us to construct Petri net models of mobile agent systems in an incremental and rule based fashion based on some architectural assumptions. Our work is ongoing, and currently still on theory. Additional methods need to be developed to relate each agent behavior with the dynamically changing system at the system level. This direction supported by application examples and tool supporting has been considered as a major trend of future work.

References:

[1] R. Guttman, A. Moukas, and P. Maes, Agent-mediated Electronic Commerce: *A Survey Knowledge Engineering Review*, June 1998.

[2] S. Green, L. Hurst, B. Nangle, P. Cunningham, F. Somers, and R. Evans, Software Agents: A Review*, Technical report TCD-CS-1997-06*, Trinity College Dublin, May 1997.

[3] T.J. Rogers, R. Ross, and V.S. Subramanian, IMPACT: A System for Building Agent Applications. *Journal of Intelligent Information Systems,* Vol. 14, 2000, pp. 95-113.

[4] F.M.T. Brazier, Dunin, B. Keplicz, N. Jennings, and J. Truer, DESIRE: Modeling Multi-Agent Systems in a Compositional Formal Framework, *Int'l Journal of Cooperative Information Systems,* Vol. 6, Special Issue on Formal Methods in Cooperative Information Systems: Multi-Agent Systems, 1997, pp. 67-94.

[5] M. Iglesias, Garrijo, and J. Centeno-González, A Survey of Agent-Oriented Methodologies, *Proceedings of the Fifth Int'l Workshop on Agent Theories, Architectures, and Language (ATAL-98),*1998, pp. 317-330.

[6] M. Wooldridge, and N.R. Jennings, Special Issue on Intelligent Agents and Multi-Agent Systems, *Applied Artificial Intelligence Journal*, 1996.

[7] M. Petit, P. Heymans, and P.Y. Schobbens, Agents as a key concept for Information Systems Engineering Requirements, *Position paper,*

Department of Computer Science, University of Namur, Belgium, 1999.

[8] M. Kolp, P. Giorgini, and J. Mylopoulos, Organizational multi-agent architectures: A mobile robot example, *Proceedings of AAMAS, 2002, Bologna, Italy*, 2002, pp. 94-95.

[9] G. Di. Marzo Seregeundo, *et al*, Survey of theories for mobile agents, *Technical report, No. 106, university of Geneva,* 1996.

[10] H. Xu, and S.M. Shatz, An Agent-Based Petri Net Model with Application to Seller/Buyer Design in Electronic Commerce, *Proceedings of the Fifth International Symposium on Autonomous Decentralized Systems (ISADS 2001),* March 26-28, Dallas, Texas, USA 2001, pp. 11-18.

[11] T. Murata, Petri Nets: Properties, Analysis and Applications, *Proceedings of the IEEE,* Vol.77, No.4, April 1989, pp. 541-580.

[12] A. Pouyan, Behavioral Modeling for Mobile Agent Systems Using Petri Nets, *Proceedings of SMC IEEE Conference, Hague, The Netherlands, October,* 2004.

[13] A. Pouyan, A Petri Net Based Approach to Design Well Behaved Discrete Event Systems, *Proceedings of SMC IEEE Conference,* Washington D.C., USA, October 2003.

[14] J. Bigus, and J. Bigus, Constructing Intelligent Agents with Java, John Wiley & Sons, USA, 1998.