I.J. Intelligent Systems and Applications, **2012, 10, 103-112**
Published Online September 2012 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijisa.2012.10.11

Petri Net: A Tool for Modeling and Analyze Multi-agent Oriented Systems

Shiladitya Pujari
Department of Information Technology, UIT, Burdwan University, Burdwan, West Bengal, India
shiladitya.pujari@gmail.com, shiladityapujari@uit.buruniv.ac.in

Sripati Mukhopadhyay
Department of Computer Science, Burdwan University, Burdwan, West Bengal, India
dr.sripatim@gmail.com

Abstract—Analysis and proper assessment of multi-agent system properties are very much important. In this paper, we discussed about methodologies for modeling, analysis and design of multi-agent oriented system with the help of Petri net. A Multi-agent system can be considered as a discrete-event dynamic system and Petri nets are used as a modeling tool to assess the structural properties of the multi-agent system. Petri net provides an assessment of the interaction properties of the multi-agent.

Index Terms—Multi-Agent System, Petri-Net, Transition

I. Introduction

Multi-agent systems can be regarded as the most emerging technology in the recent era. For the past several decades, multi-agent systems have been studied extensively. Several frameworks relevant to multi-agent systems have been defined to apply the multi-agent system concept to different applications in control as well as optimization of complex systems [1– 4]. An agent is a computer program or a computer system that consists of several complex characteristics and one of the most important characteristic of an agent is autonomy. An intelligent agent inhabits an environment and is capable of conducting autonomous actions in order to satisfy its design objective [3], [5 - 7].

An intelligent or autonomous agent presents some degree of autonomy by being reactive, pro-active and perhaps sociable [3]. The autonomy of an agent is considered as a building block in a multi-agent system. In a multi-agent system, several agents communicate and interact among them in order to solve a complex problem. In general, a multi-agent system is a system which is expected to work properly in a dynamic large-scale complex environment or open environment by having autonomy, robustness, adaptability and flexibility.

A Multi-agent system can be viewed as Discrete-Event Dynamic Systems (DEDS) [8] because a multi-agent system is a concurrent, asynchronous, stochastic and distributed computer system, and Petri nets are used as a modeling tool for assessment of properties of the system. The complexity and capabilities of a multi-agent system are greater than any distributed software system. The study of system properties is becoming more important because both cases deal with large complex dynamic systems.

Petri nets have a well defined mathematical structure that can do formal analysis on discrete-event systems. Undoubtedly, Petri Nets have been successfully used in several areas for the modeling and analysis of distributed systems [10], concurrent and parallel programs. From the discrete event dynamic system (DEDS) point of view, multi-agent systems lack analysis and design methodologies. Petri net methods can be used to develop analytical methodologies for such systems [8]. If a multi-agent system is considered as a discrete-event system and Petri nets are used to model this system, then properties are important in the discrete event systems and Petri net domains can be used to study multi-agent systems.

Again, if considering that a multi-agent system is a discrete-event system and Petri nets can be used as a modeling tool, then this requires some methodologies for mapping multi-agent systems into Petri net models. These methodologies require the right level of detail/abstraction in order to map all the important behaviors of the communication and interaction framework into the resulting Petri net models. Petri net models of multi-agent systems allow using the existing analysis methodologies for Petri nets. Some important properties of discrete event systems such as the reachability graph and the analysis of the network invariants can be obtained with Petri net analysis methods.

1.1 Previous Works

Previous works show that Petri nets have been used to model, assessment and analyze multi-agent systems.

Murata et al. [14] presented in their research work an algorithm to construct predicate/transition models of robotic operations. Basically, robot actions were considered as firing transitions and the model was used for the planning of concurrent activities of multiple robots (agents). Xu et al. [18] proposed a methodology based on predicate/transition nets for multiple agents under static planning of activities. In addition, they proposed a validation algorithm for plans with parallel activities. The verification is done based on reachability graphs due to the fact that agents actions are modeled as transitions. Petri nets also have been used to model specific multi-agent system frameworks but the resulting models have not been used to provide a study of the properties of the multi-agent system. Ahn et al [19] proposed multi-agent system architecture for distributed and collaborative supply-chain management. A Petri net model is presented but no structural analysis of the model and no verification of the coordination activities were performed. The work of Leitao et al. [20], proposed a Petri net model approach to formal specification of holonic control systems for manufacturing. They developed a Petri net sub-model for each of the four types of holons (agents) suggested in the Adaptive Holonic Control Architecture for Distributed Manufacturing Systems architecture. There was no attempt to study the structural properties of the Petri net model in order to assess some sort of dependability in the proposed architecture.

The paper has been arranged in some sections. Section I mentioned a brief introduction and a discussion on the previous works in the relevant field. Section II defined about the characteristics of intelligent and autonomous agents. The next section described what a multi-agent oriented system is. Section Iv described about Petri nets and its characteristics. Next section of the paper is on the behavioral properties of Petri nets and the following section is on the structural analysis of the net. Section VII portrayed how to model a multi-agent system using Petri-net and section VIII described the generic Petri-net model for an individual agent within a multi-agent system. The last section concluded on the research paper.

II. Characteristics of Intelligent Agent

In large and complex system, an agent is regarded as software component situated within some complex environments, which acts autonomously, cooperates, communicates and negotiate with similar entities to achieve a whole of predefined goals. An agent may also associate with its mental state. That states can be composed of components like belief, desire, intension, knowledge, capabilities, choices and commitments [22]. The characteristics of an autonomous agent are as follows:

- Autonomy: An agent is said to be autonomous if it is able to take any decisions based on some pre determined states the agent is composed of, without any direct intervention of actors of the environment.
- Proactive and Reactive*:* Agents are not only acts in response to the events of the environment where it is situated but they may also act autonomously. Besides this proactive nature, agent may act dynamically to understand the environment, apprehend any changes in this environment and respond timely to the changes that may occur.
- Capabilities: Agents are capable of performing certain activities to achieve their predefined goal. These activities are often characterized by set of well defined services which may be provided by the agent.
- Goal oriented: Agents are always goal-oriented, that means, agents always act to achieve their predefined target or goal. If a system is composed of multiple agents then they can achieve the goal through their cooperative activities.
- Knowledge Driven: An agent can have the ability to acquire new knowledge about the environment in which it is situated and can update dynamically.
- Condition/Constraint: The environmental constraints may affect the activities of agent. Moreover such constraint may be imposed by actors of the environment and which can be adopted dynamically by the agent.

Besides these characteristics of agent, the agent oriented system or multi-agent system has some important characteristics that can be summarized as follows,

- Agents are Social: An agent-oriented system or multi-agent system is comprised of multiple agents which are supposed to operate together in an open operational environment. Hence they can interact with each other, share their resources and knowledge, and also can collaborate with each others to achieve the preset goals.
- Agents are Resource Driven: Agent acts on environmental resources. Any agent of multi-agent system can hold, use and release resources of the environment where it is situated. The activities can change the state of the resources to fulfill predetermined goal or objective.
- Agents are Event Driven: Agents of multi-agent system response on events occurred in the environment. Events may occur due to some state changes or achieving certain condition or achieving certain goal or certain interaction of actors. Even more events may occur due to certain changes in environment. An multi-agent system achieves any goals using a series of events occurrences and the ongoing events may determine the system behavior.

- Agents are Dynamic: Due to event driven nature of multi-agent system and with the feature like autonomous and reactive nature of agent, such systems are truly dynamic. Moreover, the knowledge of any agent can be dynamic in nature. Further, the series of events and its corresponding responses may occur dynamically from such system. Designer simply set the initial state, knowledge and goals, on next, multi-agent system manages the things dynamically to achieve the goal.
- Agents are Heterogeneous: Several agents of a multi-agent system may be heterogeneous in nature in terms of their features. They may initially belong to different environment. Any agent may be reused to cooperate with other set of agents to achieve some goals in new environment. These facts require migrating of some specific agent from one environment to another in pre determined fashion. This also characterizes the *mobility* of agent.

While modeling a multi-agent system, designer must also ensure about the followings:

a. The system may achieve the goal with finite number of events or interactions,

b. The system may able to handle the situation if goal will not be achieved after certain set of events,

c. The system may operate in deadlock free fashion, as the system may handle the resources from the environment,

d. The system and environment should be transformed in some acceptable states with the occurrences of events and

e. The knowledge and the state of the resources are dynamically manageable.

In view of these critical features of multi-agent system, Petri Net [14] is obvious tool choice for modeling the dynamic behavior of this kind of complex system. Petri Net based tools will be useful to complement the dynamic part of such system and analyze the states and behavior of agents in the environment.

III. Multi-Agent System (MAS)

Two main aspects in a multi-agent system framework are the architecture of the agents within a multi-agent system and the interaction among them. The architecture of an agent defines that the agent can be reactive, deliberative or it may be a hybrid of both. The interactions between agents can be either a direct agent-to-agent interaction or an indirect interaction. Indirect interactions depend on the environment. In the indirect interaction, an agent may modify the environment of another agent by initiating a reaction. Usually, when two or more agents share a subset of the environment, the indirect interaction occurs. Every agent working within a multi-agent system has its own environment that is somehow related to the Meta level environment of the multi-agent system. This environment is called an open environment. An open environment provided by a complex problem is dynamic, it has components that are unknown in advance; its structure changes over time and might be heterogeneous in nature [8][9].

3.1 Abstruct Architecture for Inteligent Agent

The abstract architecture represents the behavior of an agent with respect to changes in its environment. Here, an agent has its own environment and this environment is defined by the nature of the agent. The environment is defined by the goals, objectives and the general purpose of the agent. The environment of an agent only considers things that are concerned to that specific agent. Consider two agents controlling different elements of a manufacturing cell. For example, one agent is controlling a conveyor belt and the other agent controlling an assembly operation. The environment of each of the agents will be different. This does not mean that these environments are independent from each other. In fact, actions taken by one agent could result eventually in a change in the environment of the other agent.

An agent with perception and internal state capabilities has more computational power than an agent without them and its computational power is now comparable with that of the Belief-Desired-Intention architecture as described by Wooldridge in [7].

IV. Introduction to Petri-nets

Petri nets were first introduced by Carl Adam Petri in 1962 in Germany. Petri nets are a graphical and mathematical modeling tool used to describe and analyze different kinds of real systems and suitable tool for the study of systems that are concurrent, asynchronous, distributed, parallel and/or stochastic. A multi-agent system is a kind of discrete system, particularly the competitive, parallel and non-determinist ones that is concurrent, asynchronous, stochastic and distributed. Petri net methods are suitable to model and analyze multi-agent systems.

4.1 Petri nets definition:

The following is the formal definition of a Petri net.

A Petri net is a five-tuple PN = (P; T; F; W; M_0) where:

P: { p1, p2, ..., pn} is a finite set of places,

T :{ t1, t2, ..., ts} is a finite set of transitions,

$F \in (P \times T) \cup (T \times P)$ is a set of arcs or flow relation,

$W: F \rightarrow \{1, 2, 3,...\}$ is a weight function,

$M_0: P \rightarrow Z_+$ is the initial marking,

$P \cap T = \emptyset$ and $P \cup T \neq \emptyset$

A Petri Net structure without any specific initial marking is denoted by N = (P, T, F, W) and a Petri Net with the given initial marking is denoted as (N, M_0) [11].

More specifically, a Petri net is a directed graph, along with an initial state called as initial marking M_0. The underlying graph N = (P, T, F, W) as stated above, of a Petri net is a directed, weighted and bipartite graph which consists of two types of nodes. These nodes are called places and transitions, where arcs go either from a place to a transition or vice versa. Places and transitions are graphically represented by a circle and a bar or box respectively. Every arc is labeled with their weights which is a positive integer number. A k-weighted arc can be viewed as a set of k number of parallel arcs. A marking represents the state. A marking assigns a non-negative integer to each place. If a marking assigns a non-negative integer k to a place p, then it is said that p is marked with k tokens. A token is represented by a black dot within a place. A marking is denoted by M, an m-vector, where m is the total number of places. The p-th component of M, denoted by M (p) is the number of tokens in place p.

The meanings of places and transitions in Petri nets depend directly on the modeling approach. When modeling, several interpretations can be assigned to places and transitions. For a discrete event distributed system, a transition is regarded as an event and the places are interpreted as a condition for an event to occur. The places contain tokens that travel through the net depending on the firing of a transition. When an arc is directed from p to t, the place p is said to be an input place to a transition t. Similarly an output place of t is any place in the net with an incoming arc from transition t to place p. A transition or an event has a certain number of input and output places, which represents as the pre-conditions and post-conditions of the event respectively. The presence of a token in a place means holding the truth of the condition associated with the place. In another way, k tokens can be put in a place to indicate that there are k data items or resources available.

4.2 Transition Firing:

The behavior of a system can be described by the state of the system and their changes. To model the dynamic behavior of a system, a state or marking in a Petri nets is changed as per the transition rules stated below:

- A transition t is considered enabled if each input place p of t is marked with at least w (p, t) tokens, where w (p, t) is the weight of the arc from p to t.
- A transition which is enabled may or may not fire (depending on the event taking place).
- A firing of an enabled transition t removes w (p, t) tokens from each input place p of t, and adds w (p, t) tokens to each output place p of t, where w (p, t) is the weight of the arc from p to t.

Some terminologies need to be mentioned and defined here to get a clearer picture of Petri nets and transitions. A transition without any input place is called as *source transition*, whereas transition without any output place is called a *sink transition*. Eventually a source transition is being enabled unconditionally. On the other hand, the firing of a sink transition only consumes tokens, but produces none [11].

A pair of a place p and a transition t is called as self-loop, if p is both input and output place of t. A Petri net is called pure Petri net if it has no self-loops. A Petri net is called as ordinary if weight of all arcs are 1.

A transition can fire only if it is enabled. For a transition t to be enabled, all the input places of t must contain at least one token. When a transition is fired, a token is removed from each input place, and one token is added to each output place. In this way the tokens travel through the net depending on the transitions fired.

4.3 Basic Construct of Petri-nets:

Basic constructs of Petri nets can be classified depending of the transitions occur and in which fashion the firing rules are applicable. Basic constructs of Petri nets are sequential actions, dependency, conflict (decision, choice), concurrency, cycles and synchronization (mutually exclusive actions, resource sharing, communication, and queues). Following figures showing the basic constructs of Petri nets.

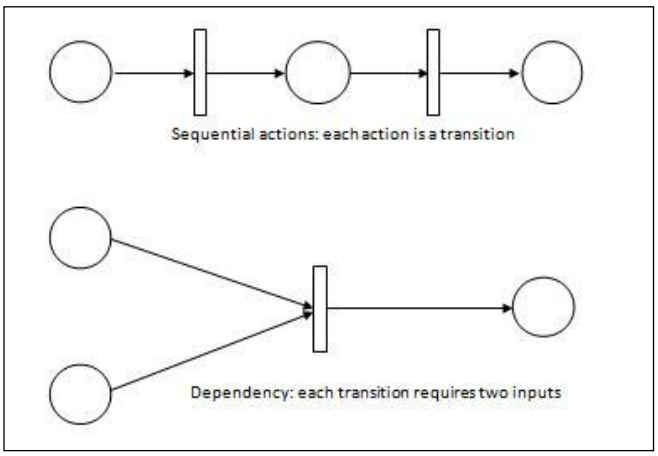

Fig.1: Sequential action and dependency

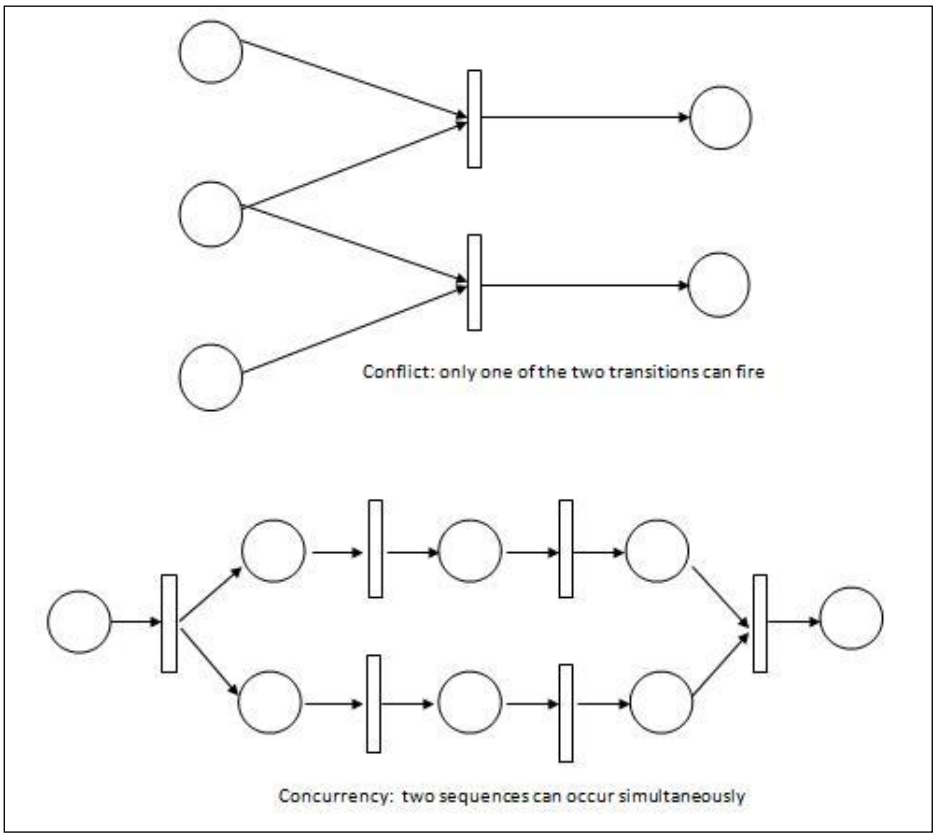

Fig.2: Conflict and concurrency

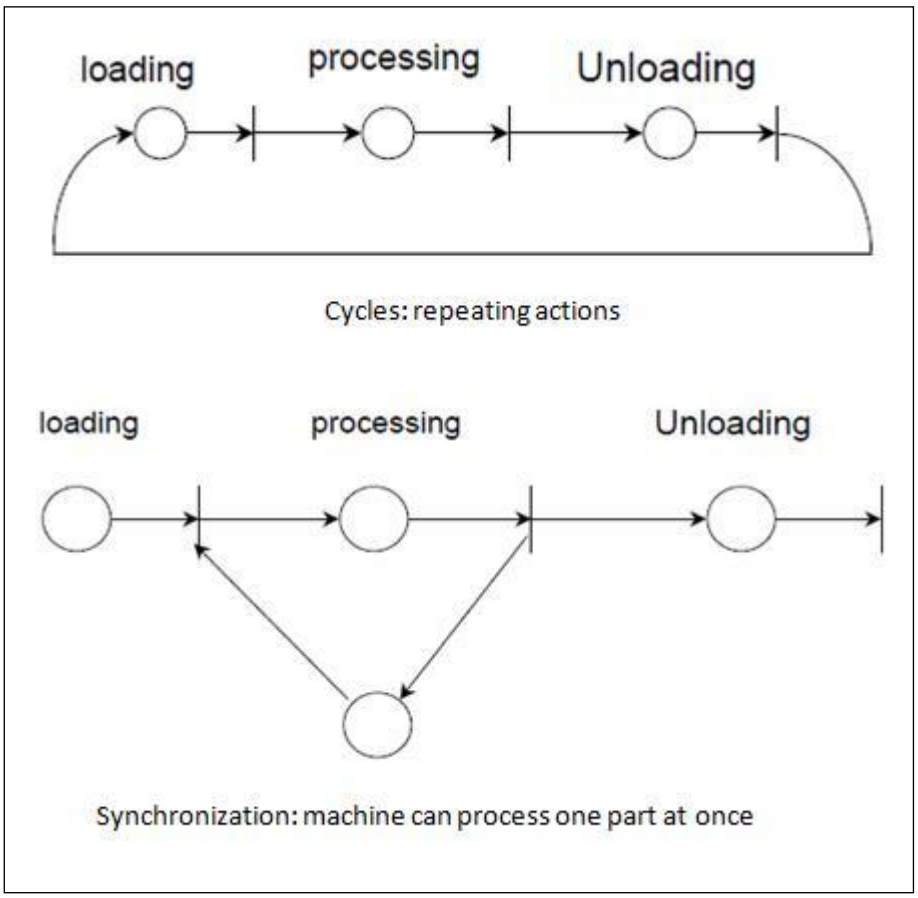

Fig.3: Cycles and synchronization

4.4 Marking of Petri-nets:

The marking m_i of a place $p_i \in P$ is a non-negative quantity which represents the number of tokens in that place at a given state of the Petri net. The marking of the Petri net is also defined as the function M: $P \rightarrow Z_+$ that means the function maps the set of places to the set of non-negative integers. It is also defined as a vector $M_j = (m_1, m_2, \ldots m_P)$ where $m_i = M(p_i)$, which represents the j^{th} state of the net. M_j contains the marking of all the places and the initial marking is denoted by M_0.

The marking represents the state of the net. As described above, the transitions change the state of the Petri net in the same way an event changes the state of a Discrete Events Dynamic System (DEDS).

4.5 Reachability Graph:

The reachability graph expresses the reachable places of a Petri-net. Reachability graph is a directed graph. Every node of the graph is identified as marking of the net R (N, M_0), where M_0 is called the initial marking and arcs are represented by the transition of the net N. Reachability graph is used to define a Petri-net N and marking M, where $M \in R$ (N). Every M_0 has an associated reachability set which consists of all marking reachable from M0 through the firing of transitions.

V. Behavioral Properties of Petri-net

Two types of properties can be studied with a Petri net model. Properties which are dependent on the initial marking of the net are called marking-dependent or behavioral properties. The properties which are independent of the initial marking are called structural properties. This section covers some of the most important behavioral properties of Petri nets such as *Reachability*, *Liveness*, *Boundedness* and *Reversibility*. Among those, some properties can be applied to multi-agent systems models. Examples of these properties are *Boundedness* and *Liveness* since they are related to deadlock avoidance in DEDS. Other properties are going to be relevant to multi-agent systems particularly to the communication, interaction, and single agent architectures.

5.1 Reachability:

A marking Mj is said to be reachable from marking Mi if there exist a sequence of transitions that changes the state of the Petri net from Mi to Mj. The set of all possible markings that are reachable from M0 is called the reachability set and is defined by R (M_0).

The concept of reachability is essential for the study of the dynamic properties of a Petri net [13], [14].

5.2 Liveliness:

A Petri net is said to be live for a marking M0 if for any marking in R (M_0) it is possible to fire a transition. The liveness property guaranties the absence of deadlock in a Petri net. This property can also be observed from the reachability graph. If the reachability graph contains an absorbent state, then the Petri net is not live at that state and it is said to have a deadlock [13][14].

5.3 Boundedness:

A Petri net is said to be bounded or k-bounded if the number of tokens in each place does not exceed a finite number k for any marking in R (M_0). Furthermore, a Petri net is structurally bounded if it is bounded for any finite initial marking M_0. A Petri net is said to be safe if it is 1-bounded, means that each place have a maximum number of token count 1 or 0 [13].

5.4 Reversibility:

A Petri net is reversible, if for any marking in R (M_0), M_0 is reachable. This means that the Petri net can always return to the initial marking M_0 [13][14].

VI. Structural Analysis of Petri-net

The Liveness and Boundedness of the net will be assessed by using P-invariants and T-invariants. These invariants are obtained from the incidence matrix of the net and they are used to assess the overall Liveness and Boundedness of the net.

6.1 Incident matrix:

Let $a^+_{ij} = w(i, j)$ be the weight of the arc that goes from transition t_i to place p_j and $a^-_{ij} = w(j, i)$ be the weight of the arc from place p_j to transition t_i. The incidence matrix A of a Petri net has |T| number of rows and |P| number of columns. It is defined as $A = [a_{ij}]$ where $a_{ij} = a^+_{ij} - a^-_{ij}$.

6.2 Net-invarients:

Let A be the incidence matrix. A P-invariant is a vector that satisfies the equation $A_x = 0$ and a T-invariant is a vector that satisfies the equation $A^T{}_y = 0$.

- Boundedness assessment: A Petri net model is covered by P-invariants if and only if, for each place s in the net, there exists a positive P-invariant x such that $x(s) > 0$. Furthermore, a Petri net is structurally bounded if it is covered by P-invariants and the initial marking M0 is finite.
- Liveness assessment: A Petri net model is covered by T-invariants if and only if, for each transition t in the net, there exists a positive T-invariant y such that $y(t) > 0$. Furthermore, a Petri net that is finite is live and bounded if it is covered by T-invariants. This is a necessary condition but not sufficient.

VII. Modeling Multi-agent System with Petri-net

Modeling approach based on interaction among agents. The interactions between agents can be either a direct agent-to-agent interaction or an indirect interaction. It shows how the agents interact among each other and how they operate over a meta-level (multi-agent level) environment. Arrows define direct agent interactions from agent to agent; the indirect interactions are based on the environment. In the indirect interaction, an agent modifies another agent's environment triggering a reaction. The indirect interaction occurs in the cases when two or more agents share a subset of the environment. It should be noted that the overall multi-agent system acts over a macro level environment. An agent that is part of the multi-agent system has its own environment that is somehow related to the macro level environment of the multi-agent system. This macro level environment of the multi-agent system is referred to in the literature as being an open environment [8][9]. A complex problem will provide an open environment, which is dynamic, has components that are unknown in advance, its structure changes over time and might be heterogeneous in its implementation [9]. By focusing on the interactions among agents as described above, it is natural to regard a multi-agent system as a discrete-event system.

7.1 Petri net models from the abstract architecture

The artificial intelligence research considers three different paradigms for intelligent agents: a) reactive and b) deliberative; and c) hybrids between them. The abstract architecture models how an agent behaves with respect to changes in its environment. Here, an agent has its own environment and this environment is defined by the nature of the agent. The goals, objectives and the general purpose of the agent define its environment. This abstract architecture is based on the reactive paradigm of perception and action. A purely reactive agent has a perception of the environment and it is used in the decision mechanism that provides an action in the agent.

A reactive agent can also have an internal state as a decision mechanism for the actions to be undertaken. An agent with perception and internal state capabilities has more computational power than an agent without them and its computational power is now comparable with that of the Belief-Desired-Intention architecture as described in [7].

7.2 Abstract model for purely reactive agents:

In case of a purely reactive agent, the perception part records the changes in the state of the environment. The action part computes the actions to be taken in order to react to changes in the environment. The environment of an agent changes based on the actions applied by the agent, as well as actions by other agents, and it may be dynamic in nature, i.e. it may change by itself. The environment consists of a set of states S = {s1, s2, …}, the agent can undertake a set of actions A = {a1, a2, …} and perceive a set of percepts P = {p1, p2, …}. For a purely reactive agent, the behavior of the agent can be represented as the function action: $P \rightarrow A$ and perception: $S \rightarrow P$. The deterministic behavior of an environment can be represented by the function environment: $S \times A \rightarrow S$.

7.3 Petri net modeling of multi-agent systems:

A Petri net is defined as a five-tuple (P; T; A;W; M_0) where P is a finite set of places and T is a finite set of transitions, $A \in (P \times T) \cup (T \times P)$ is a set of arcs, $W : A \rightarrow \{1; 2; 3; \ldots.\}$ is a weight function, and $M0 : P \rightarrow Z_+^{|P|}$ is the initial marking.

7.4 Obtaining Petri net models from the abstract architecture:

Places represent the environmental states of the agent. Having a token in a place means that the agent is currently in that state. Transitions represent the actions of an agent. The environmental state is changed by actions and for the Petri net model having tokens move from one place to another by firing transitions, this agrees with the execution process of the abstract architecture.

7.5 Algorithm to develop Petrinet sub model for agent i

The algorithm to develop Petri net sub-model for an agent is described here. Let Si be the set of environmental states of agent i, and $s_{ij} \in S_i$ be the j^{th} environmental state of agent i. Similarly, let Ai be the set of actions of agent i, and $a_{ik} \in A_i$ be the k^{th} action of agent i.

(a). Add a place for each element of the environment Si and label each place using notation P_{ij} for s_{ij} .

(b). Add a transition for each action in Ai and label each transition using notation T_{ik} for a_{ik}.

(c). For each instance of the function environment : Si $\times$ Ai $\rightarrow$ Si say sij $\times$ aik $\rightarrow$ sil: a) add an arc leaving from place Pij and ending in transition Tik;b) add an arc leaving from transition Tik and endingin place Pil; c) add a weight of 1 to each arc. If an arc from transition Tik to place Pil already exists, add a new transition and label it T'_{ik}; perform this step using T0 ik instead of Tik.

(d). Add a token in the place representing the initial state of the environment.

7.6 Petri net model of the multi-agent system

The Petri net sub-models of each of the individual agents in the system should be joined based on their indirect interactions. In general, this indirect interaction

will be in such manner that an action of agent i will change an environment state of agent j. This communication act can be regarded as a regular action in the construction of the complete model. There will be arcs added from the places representing the environmental states of agent j to the transition modeling the communication in agent i.

7.7 Analysis of the Petri net model:

Inspection of the reachability graph of the Petri net model can indicate if the model is live and bounded. On the other hand, liveness and boundedness properties can also be assessed using invariant analysis [15].

VIII. Generic Petri-net Model for Agent

A generic Petri-net model has been proposed by Varakantham, Gangwani and Karlapelam in their research work. The proposed generic petri-net model defines the states and transitions of individual agents within a multi-agent system. To accomplish the goal-oriented tasks, agents must proceed through a set of predefined states [21]. The following figure describes the generic Petri-net model clearly.

According to the generic Petri-net model, initial state of every individual agent is called Waiting State denoted as P_0. In this state, agent is waiting for some message either form environment, or other agents. When the agent receives a message, it changes its state or place and goes to state P1 through transition T10. An assumption is made here as the communications between the agents are accomplished by transferring messages between them. Every message consists of some fields such as Sending agent (SA), Receiving agent (RA), Message ID (MID), Response ID (RID), Action of the message (ACT) and Content of the Message (CNT). After going to the state P1, it checks for the response ID (RID). A non-zero value of response ID indicates that the message is a response message and the agent goes to the action place or state P5 directly through transition T2. A zero-valued response ID states that it is a new or request message for the agent to perform any task. In that case, the agent goes to the state P1 through transition T1. In this state, agent checks for its capability of performing the task or action received in the message. Checking of capability involves if the beliefs of the agent allow it to take up and performing the task or not. If capability checking fails, the agent sends a negative response message to the sending agent and transit to the end state P7 through transition T11. Contrary, the agent moves to state P3 through transition T3. P3 state is the commitment making state. In this state, the agent sends the acceptance of commitment to the agent that sent message to it and the agent changes the state by moving to P4 through the transition T4. As it is necessary to perform the task as it is committed, the agent goes to the state P5 through transition T5 which is the action place for the agent. In this generic model, action place for the agent is considered as a single place to avoid complexities. A variable named ActionNum is associated with the action place P5 which indicates the number of actions each of which is about to be executed in the sub-task if more than one actions can be performed in the action place. Initial value of this variable is set to 0, and the value is incremented each time the agent enters into the action place P5. After accomplishing the task, the agent goes to state P6 through transition T6 from where it transmits the message to the receiving agent. After completion of the task, it moves to the end place P7 through the transition T9. Figure 4 is the graphical representation of this generic Petri-net model described above. This generic Petri-net model shows that an agent within a multi-agent system can be modeled and analyze with the help of Petri-net.

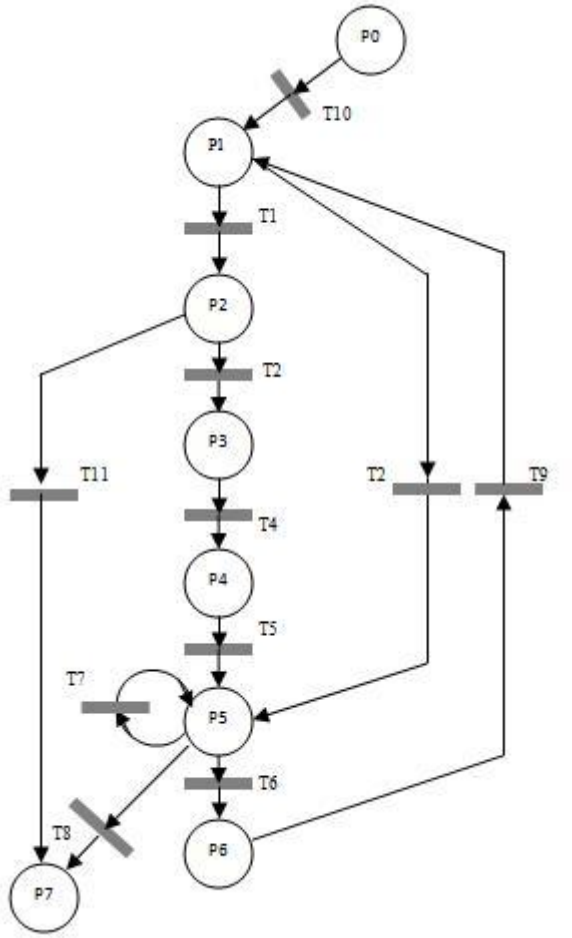

Fig. 4: Generic Petri-net Model

Places description of this generic Petri-net is given below. The table below showing the places or states of an agent.

TABLE I. places of generic Petri-net

Places	Description of Places /states
P0	Waiting state of an agent
P1	Message Received
P2	Received message is a request for a task
P3	Agent has capability of doing the task
P4	Commitment accepted
P5	Action place or task sub-Petri-net
P6	Message handling
P7	End place

TABLE II. Transitions of generic Petri-net

Transitions	Description of Places /states
T1	If the message is a request for doing task
T2	Checking of message whether it is a response
T3	Checking of the capability of agent for doing task
T4	Agent sends commitment accepted message
T5	Starting the task
T6	Agent want to send message
T7, T8	Execution of actions
T9	Sending of message
T10	Receiving the message
T11	Agent does not have capability of doing task

From the above two tables (Table I and Table II) we can see the places or states of an agent and the transitions for the generic Petri-net model.

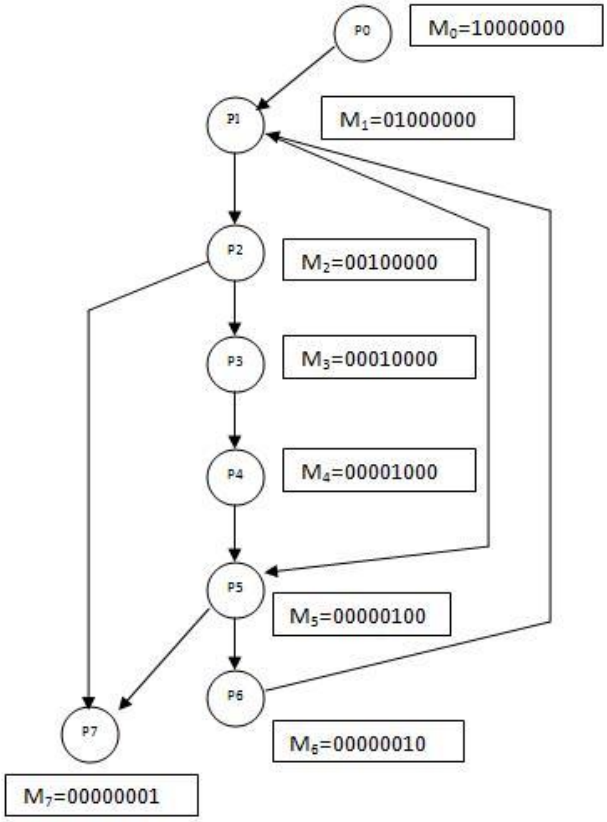

Fig. 5: Marking and reachability graph of generic PN model

Marking and reachability graph has been depicted in the above figure. From the above figure, we can see that the each of the initial marking M_0 has an associated reachability set which is consists of all the markings that can be reached form M_0 through the firing of one or more transitions. In case of the generic Petri net model described above, the reachability graph begins with the initial marking M_0 = [10000000] and reached the state with marking M_7 = [00000001] finally, which is the end place of a agent. The place of a reachability graph is said safe, if the number of tokens is either 0 or 1 at that place. In case of this generic Petri net model, any of the places between P_0 to P_7 represent a combination of 0 and 1, where 0 means no token and 1 means token is there, which implies that after the occurrence of firing, there will be a token at the position bit, else not. This property shows that each place have a maximum number of token count 1 or 0. That says that the Petri net is *safe*. The generic Petri net is *bonded*, because there is no overflow at any place as well as no deadlock situation is there at any stage between P_0 and P_7. As a marking M_p is reachable form some marking M_0 after firing an existing sequence of transitions that transform M_0 to M_p. hence the reachability graph is *reachable*. Again as stated earlier, there is no deadlock situation within the net, which clearly showing the *liveliness* of the net. That means the generic Petri net shows all the properties such as safeness, Boundedness, reachability and liveness.

IX. Conclusions

In this paper, we discussed about multi-agent system and how this kind of system can be analyzed and designed by using Petri net. After finding out aspects of Petri net and discussing about the properties, it can be concluded that Petri net is the best tool for modeling and analyzing multi-agent system which may be defined as a discrete events dynamic system.

References

[1] M. Greaves, V. Stavridou-Coleman, and R. Laddaga, "Guest editors' introduction: Dependable agent systems," IEEE Intelligent Systems, vol. 19, no. 5, pp. 20–23, 2004.J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[2] R. Khosla and T. Dillon, Engineering intelligent hybrid multi-agent systems. Kluwer Academic Publishers, 1998.K. Elissa, "Title of paper if known," unpublished.

[3] G. Weiss, Ed., Multiagent systems: a modern approach to distributed artificial intelligence. Cambridge, MA, USA: MIT Press, 1999.

[4] S. S. Heragu, R. J. Graves, B.-I. Kim, and A. St Onge, "Intelligent agent based framework for manufacturing systems control," IEEE.

[5] N. J. Nilsson, Artificial intelligence: a new synthesis. Morgan Kaufmann Publishers Inc., 1998.

[6] S. J. Russell and P. Norvig, Artificial Intelligence: A Modern Approach. Pearson Education, 2003.

[7] M. J. Wooldridge, Introduction to Multiagent Systems. John Wiley & Sons, Inc., 2001.

[8] J. R. Celaya, A. A. Desrochers and R. J. Graves, Modeling and Analysis of Multi-agent Systems using Petri Nets, Journal of Computers, October 2009.

[9] K. P. Sycara, "Multiagent systems," AI Magazine, pp. 79–92, 1998.

[10] W. Reisig, Elements of distributed algorithms: modeling and analysis with Petri nets. New York, NY, USA: Springer-Verlag New York, 1998.

[11] A. Desrochers and R. Y. Al-Jaar, Applications of Petri Nets in Manufacturing Systems: Modeling, Control, and Performance Analysis. IEEE, 1995

[12] T. Murata, "Petri nets: Properties, analysis and applications," Proceedings of the IEEE, vol. 77, no. 4, pp. 541–580, April 1989.

[13] M. Zhou, F. DiCesare, and A. A. Desrochers, "A hybrid methodology for synthesis of petri net models for manufacturing systems," IEEE Transactions on Robotics and Automation, vol. 8, no. 3, pp. 350–361, 1992.

[14] T. Murata, "Petri nets: Properties, analysis and applications," Proceedings of the IEEE, vol. 77, no. 4, pp. 541–580, April 1989.

[15] A. A. Desrochers, "Performance analysis using petri nets," Journal of Intelligent and Robotic Systems, vol. 6, no. 1, pp. 65–79, August 1992.

[16] W. Reisig, Petri nets, An Introduction, ser. EATCS: Monographs on Theoretical Computer Science. Springer-Verlag, 1985, vol. 4.

[17] J. L. Peterson, Petri net theory an the modeling of systems. Prentice Hall, 1981.

[18] D. Xu, R. Volz, T. Ioerger, and J. Yen, "Modeling and verifying multi-agent behaviors using predicate/transition nets," in SEKE '02: Proceedings of the 14th international conference on Software engineering and knowledge engineering. New York, NY, USA: ACM Press, 2002, pp. 193–200.

[19] H. J. Ahn and S. J. Park, "Modeling of a multi-agent system for coordination of supply chains with complexity and uncertainty,"in Intelligent Agents and Multi-Agent Systems, ser. Lecture Notes in Computer Science, J. Lee and M. Barley, Eds., vol. 2891, 6th Pacific Rim International Workshop on Multi-Agents, PRIMA 2003 Seoul, Korea. Springer-Verlag Berlin Heidelberg, November 2003, pp. 13–24.

[20] P. Leit˜ao, A. W. Colombo, and F. Restivo, "An approach to the formal specification of holonic control systems," in Holonic and Multi-Agent Systems for Manufacturing, ser. Lecture Notes in Computer Science, V. Mar´ık, D. McFarlane, and P. Valckenaers, Eds., vol. 2744, First International Conference on Industrial Applications of Holonic and Multi-Agent Systems, HoloMAS 2003 Prague, Czech Republic, September 1-3, 2003. Springer Berlin / Heidelberg, 2004, pp. 59–70.

[21] P. R. Varakantham, S. K. Gangwani and K. Karlapelam, On Handling Component and Transaction Failure in Multi-agent System, ACM.

[22] P. K. Biswas, Towards an agent-oriented approach to conceptualization, Journal of Applied Soft Computing, Vol 8, No 1, pp 127-139, January 2008.

Pujari Shiladitya, ME, PhD (Cont.) is an assistant professor in the department of Information Technology of University Institute of Technology, Burdwan University, India. He is working in the field of Multi-agent System, Artificial Intelligence as well as Cryptography and Steganography.

Mukhopadhyay Sripati, PhD, is working as a senior professor and head, department of Computer Science, Burdwan University, India. He is also acting as the registrar of the university for time being. He is working in the field of artificial intelligence for more than 15 years.

