Modeling and Analysis of Agent Oriented System: Petri Net Based Approach

Rajib Kr. Chatterjee[1], Anirban Sarkar[2], Swapan Bhattacharya[3]
[1]Department of Computer Centre, National Institute of Technology, Durgapur, India
[2]Department of Computer Applications, National Institute of Technology, Durgapur, India
[3]Department of Computer Science & Engineering, Jadavpur University, Kolkata, India
{chatterjee.rajib@gmail.com , sarkar.anirban@gmail.com , bswapan2000@yahoo.co.in}

Abstract - *Large and complex system are now a days conceptualized using Agent Oriented Paradigm. Agent oriented systems are dynamic in nature. In this paper, we have proposed a conceptual framework for agent to conceptualize the artifacts of such system. The paper also has proposed a Petri Net based model and analysis methodology based on that conceptual framework to analyze the crucial behavioral feature of such system which is also dynamic in nature.*

Keywords: Agent, Agent Oriented System, Petri Net, Dynamic Modeling, Behavior Analysis.

1 Introduction

Over the last decades, software engineers have derived a progressively easy and better perception of the characteristics of large and complex software. Among others, such software is characterized by dynamically interacting components to provide wide range of services. In this context, Agent Oriented System (AgOS) recently emerged as powerful technology to handle the dynamism of component level interactions for large and complex information system. AgOS based computing promotes, designing and developing applications in terms of autonomous software entities (agents), situated in an environment, and that can flexibly achieve their goals by interacting with one another dynamically in terms of high-level protocols or languages [3].

In large information system, agent refers to a software component that situates within some environments, operates autonomously and cooperates with similar entities to achieve a set of preset goals. An agent also may associate with its mental state that can be composed of components like belief, knowledge, capabilities, choices and commitments [6]. The critical features of agent are as follows,

- *Autonomous:* Agent is composed of some predetermined states and is able to take any decisions based on those states without any direct intervention of actors of the environment.
- *Goal oriented:* Agent always acts or works to achieve or reach the preset target or goal. If an AgOS composed of multiple agents then they together can achieve the goal through the cooperative activities.
- *Capabilities:* Each agent is capable to perform certain activities towards achieving the goal. These activities are often characterized by set of well defined services which may be provided by the agent. The agent capabilities can be defined using these set of services.
- *Situatedness:* An agent performs its activities while situated in a particular environment and it is able to sense and affect such environment.
- *Proactive/Reactive:* Agent not only acts in response to the events of the environment where it is situated but they may also become active 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.
- *Knowledge Driven:* An agent can have the ability to acquire new knowledge about the environment in which it is deployed 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.

Several of these features have been summarized in recent literatures [1, 3, 6]. Along with these agent level characteristics, the crucial features for AgOS can be summarized as follows,

- *Agent Social:* An AgOS may be 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.
- *Resource Driven:* Agent acts on environmental resources. Any agent of AgOS 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.
- *Event Driven:* Agents of AgOS 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 AgOS achieve any goals using a series of events occurrences and the ongoing events may determine the system behavior.

- *Dynamic:* Due to event driven nature of AgOS and with the feature like autonomous and reactiveness 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, AgOS manage the things dynamically to achieve the goal.
- *Heterogeneous:* Several agents of AgOS 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 of AgOS, designer must also ensure that, (i) system will achieve the goal with finite number of events or interactions, (ii) system will be able to handle the situation where goal will not be achieved after certain set of events, (iii) system will operate in deadlock free way, as the system will be handling the resources from the environment, (iv) system and environment should transform in acceptable states with the occurrences of events and (v) the knowledge and the state of the resources are dynamically manageable. These all are very crucial features of the dynamic behavior of AgOS. In view of these features, Petri Net [2] is obvious tool choice for modeling the dynamic behavior of AgOS. *Conceptual modeling* of AgOS in this respect, defines the components and their inter relationship to conceptualize the environment, agent, related events and interactions. This specifies the static part of the AgOS. 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.

Several researches in last decade have been done to devise conceptual model for AgOS [4, 5, 6, 7, 8, 9, 11]. Among those proposed approaches, [4, 6, 9, 11] have extended the Unified Modeling (UML) notations to conceptualize the AgOS using object oriented paradigm. In [10], a detail study has been done on these proposed approaches and raised the demand of new paradigm beyond the object oriented paradigm to conceptualize the AgOS. It also states that agent architecture is far more complex than the object architecture, especially because of the dynamic aspects of AgOS. But majority of those proposed approaches have not been dealt with the dynamic behavior of AgOS. In [5], a formal frame work for AgOS has been devised using ObjectZ notation and the semantic of behavior has been represented using state chart diagram. But it lacks the methodology for dynamic behavior analysis of such model, which is crucial for the successful deployment of AgOS.

In [12], a high level Petri Net has been defined to formalize the AgOS. It is efficient to model the external behavior (interaction with the environment) of the AgOS comprises with homogeneous set of multiple agents. But it lacks to exhibit the dynamism of internal behavior (within the agents) of the system which comprises of heterogeneous set of agents. Moreover, several crucial properties related to dynamic behavior have not been analyzed using the proposed high level Petri Net.

The focus of this paper is two folds. *Firstly*, it proposes a conceptual framework for Agent and AgOS to conceptualize its artifacts. *Secondly*, based on the proposed conceptual framework, the dynamic behavior analysis of AgOS has been done using classical Petri Net. For the purpose, we have proposed a generic Petri Net representation for the conceptual framework of agent. Moreover, Petri Net based analysis has been used to ensure the correctness of the crucial characteristics of dynamic behavior of the agent.

2 Proposed Conceptual Framework for Agent

In this section a conceptual framework for AgOS has been proposed. A conceptual model of AgOS deals with high level representation of the candidate environment in order to capture the user ideas using rich set of semantic constructs and interrelationship thereof. Such conceptual model will separate the intention of designer from the implementation and also will provide a better insight about the effective design of AgOS. The framework has been drawn from the system features discussed in the last section.

An environment *Env* where the agents will work can be realized using four tuples. It can be defined as *Env = [Res, Actor, Agent, Relation]*, where, in the given environment *Res* is the set of resources, *Actors* are the users, *Agent* is the set of autonomous entities with pre specified goal and *Relation* is set of semantic association among them.

In the context of *Env*, an agent will apprehend the occurrences of events automatically and response towards the environment with a set of activities or services, those are within its capability. An agent will also be able to verify the environmental conditions / constraints associated with the services or occurrence of any events. Moreover, any agent acts on the environmental resources *Res* and is able to create / maintain the knowledge base for the states of resources. The states of agent can be realized using a set of attribute associated with that agent.

Formally an agent *Ag* in the environment *Env* can be defined as, *Ag, = [E, C, R, PR, K, S]* where,

- *E* is the set of events $\{e_1, e_2, e_3, \ldots, e_p\}$ on which the agent will response. The events may occur from the *Actor* or changes in states of *Res* or on achieving some condition.

- *C* is a set of environmental conditions or constraints $\{c_1, c_2, \ldots, c_q\}$ to be checked in order to response on some event.
- *R* is a set of environmental resources those are available and necessary for fulfillment of the goal of the agents. Also $R \subset Res$.
- *PR* is the set of agent properties which will hold the state of the agent and also will maintain the state of the resources *R* on which the agent is acting.
- *K* is the set of information that forms the main knowledge base. Initially it comprises of the states of available resources that the agent will use to response on some event. The *K* can be updated dynamically.
- *S* is the set of services that the agent can provide. The set *S* is used to conceptualize the capability of the agent. The agent may provide the service $s_i \in S$ to the environment on the occurrences of some set of events $E' \in E$ to achieve the pre specified goal.

A *Multi Agent System* in this context can be defined as *MAS = [A, I]*, where *A* is the set of agents and *I* is the set of interactions among those agents. The set *I* determines cooperation and collaboration among the agents and also it can operate either in two modes namely, asynchronous or synchronous.

From the above definitions we can represent the conceptual framework of an agent graphically as shown in Figure 1.

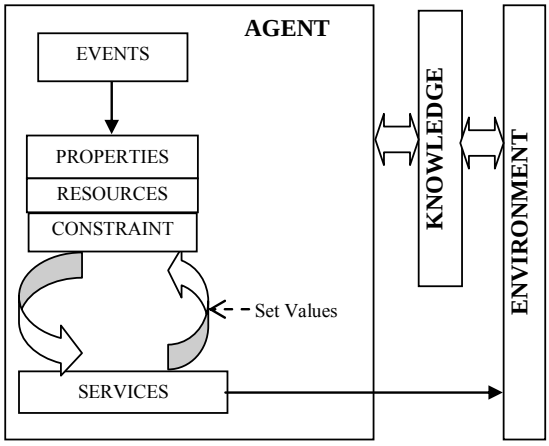

Figure 1: Conceptual Framework of AGENT

2.1 Illustration of Conceptual Framework of Agent with Example

In this sub section we have illustrated the conceptual framework of agent using a real world example. Let consider an environment comprised of set *Computers* with different types of *Operating System* (OS) connected with network. The environment also contains an autonomous *Agent* to act on the *Computers*. After initiation, the goal of the agent is to search the *Computers* with *MS Windows OS* and to shutdown it automatically. The agent will also maintain a time delay of 15 seconds while performing the shutdown activities on the next target *Computer*. Users are allowed to interrupt the agent responses at any point of time. The number of *Computers*, their *OS, Network Addresses (IP addresses)* and the *Port Numbers* (through which agent can communicate with Computer) are well known in the environment.

For the given environment, an agent can be designed using the proposed conceptual framework are given below. For the purpose one need to define all the components of the agent definitions of that said framework to achieve the given goal.

i) The set of events E will be, e_1 = Initiate, e_2 = User Interrupt, e_3 = User Resume, e_4 = User Cancel Job, e_5 = Search, e_6 = Target Computer Found, e_7 = Service Initiate, e_8 = Service Resume, e_9 = Service Completed, e_{10} = Service Interrupt, e_{11} = Service Revoked, e_{12} = Timer start, e_{13} = Terminate, e_{14} = No Action.

Those events may occur after satisfying some environmental constraints C. The agent will response due to some events based on some specific constraint.

ii) The set of constraints C are, c_1 = Time Delay of 15 second, c_2 = Operating System is MS WINDOWS, c_3 = Next IP address to be processed.

After the verification of the constraints, the agent may acts on a set of resources from the environment. For the purpose, it performs some activities on those resources to achieve the goal.

iii) The set of resources R are, r_1 = The network services, r_2 = Computers, r_3 = The OS port where the agent will interact, r_4 = The timer to keep track of the time delay of 15 seconds

Now the agent will use several properties to hold the state of the resources and the states of the agent itself. An agent may changes its state based some events and the state of resources may change based on the activities performed by that agent.

iv) The set of properties PR are, pr_1 = Computers identity with MS Windows OS, pr_2 = OS type of the current Computer, pr_3 = Time Elapsed. Its initial value will be 0, pr_4 = Status type of the agent and it can be of the following types, a) "INIT", b) "INTERRUPT", c) "COMPLETED", d) "CANCELLED" and e) "RESUME".

Agent starts working with the minimal set of knowledge of the environment to render the services. The knowledge base accumulates the initial facts of the resource states which are static in nature. The knowledge base can be updated dynamically once the agent starts working.

v) The set of knowledge K are, k_1 = IP Addresses list of the Computers, k_2 = OS of the Computers, k_3 = Selected Port Numbers of the Computers.

To achieve the pre specified goal, agent acts on environmental resources with certain activities. These activities are realized using a set of services.

vi) The set of services S are, s_1 = Seek OS type from the computer, s_2 = Action Shutdown, s_3 = Action Paused (Act on agent itself), s_4 = Action Revoked (Act on agent itself), s_5 = Seek Port Number, s_6 = Set Port Number, s_7 = Action Terminate (Act on agent itself), s_8 = Action resume (Act on agent itself).

As a result of the triggered events, some set of services will be performed by the agent based on the certain values of the properties and constraints. It will use the resources which are all accessible from the knowledge base. Thus with all these components a generic and autonomous agent will be able to perform the pre specified goal to shutdown all the *Computers* with *MS Windows OS* in the given environment.

3 Petri Net based Modeling of Conceptual Agent

AgOS behavior is dynamic in nature. Also several crucial features are required to ensure while designing such system. Since, agent response on series of events and provide services to the environment in autonomous way. For the purpose it holds the resources and makes changes in its states. Petri Net (PN) is a suitable tool to model the behavior of such system. Moreover, several features of AgOS like, occurrence of finite number of events, deadlock free operations, achievement of goals through firing of events etc. can be analyzed through the analysis of PN properties like, safeness, boundedness, liveness, reachability etc. Further, the PN based analysis will give detail insight about the internal behavior of the agent.

In this context, a PN is a particular kind of bipartite directed graph, populated by three kinds of objects namely, places, transitions and directed arcs connecting places to transitions and transitions to places. An enabled transition removes one token from each of the input places, and adds one token to each of its output places. This is called the firing rule. The PN graph also has an initial state called the initial marking M_0. Formally, a PN is a 5 tuple, $PN = \{P, T, F, W, M_0\}$ where, $P = \{p_1, p_2, \ldots, p_m\}$is a finite set of places, $T = \{t_1, t_2, \ldots, t_n\}$ is a finite set of transitions, F is a set of arcs such that $F \in (P \times T) \cup (T \times P)$, W is a weight function $W: F \rightarrow \{1, 2, \ldots\}$, M_0 is the initial marking $M_0: P \rightarrow \{1,2,3,\ldots\}$ is the initial marking. As stated earlier, the behavioral properties of the target system can be analyzed using the properties like reachability, boundedness, liveness, coverability, persistence, reversibility, fairness etc.

3.1 Components Wise Mapping from Conceptual Framework to Petri Net

In the proposed conceptual framework, agent definition has various components namely events, constraints, resources, properties, knowledge and services. All these are the individual items which together make AgOS successful to achieve the pre specified goal. In this sub section we have mapped the different components of agent definition in PN components called places and transitions.

A place P in PN comprises of set of tokens T_k belongs to constraints, resources, properties, knowledge and services of any agent. Formally, $P \rightarrow T_k$ where, $T_k \in C \cup K \cup PR \cup K \cup S$. All the events of any agent will be mapped as transitions T of a PN. Formally, $T \rightarrow E$.

The graphical notation of place and transition are represented as usual notation of PN and those are *Circle* and *Bar* respectively.

Also it is important to note that, in a PN of AgOS, due to firing of any transition *T,* all the sub components of the output place *P* will be affected simultaneously. Hence for any place in resulted PN one can set the mark as *0* or *1*.

3.2 Generic PN Representation of Conceptual Framework

As discussed earlier, an AgOS is event driven system, where agent is an autonomous entity. Irrespective of any environment, any agent in AgOS will have certain generic set of services which will be used in response to a generic set of events. Further, we have also proposed the mapping rules for the components of any agent to resultant PN for the system. These facts will result the formation of *Generic PN* for the analysis of agent's dynamic behavior.

A generic set of events associated with any agent can be *Initiate* (e_1), *Search for knowledge* (e_2), *service provided* (e_3), *Interrupt from the environment or actors* (e_4), *Activity resume* (e_5), *Activity cancel* (e_6). Further these events can be mapped into the transitions of Generic PN will be t_1, t_2, t_3, t_4, t_5 and t_6 respectively. Similarly a generic set of services can be

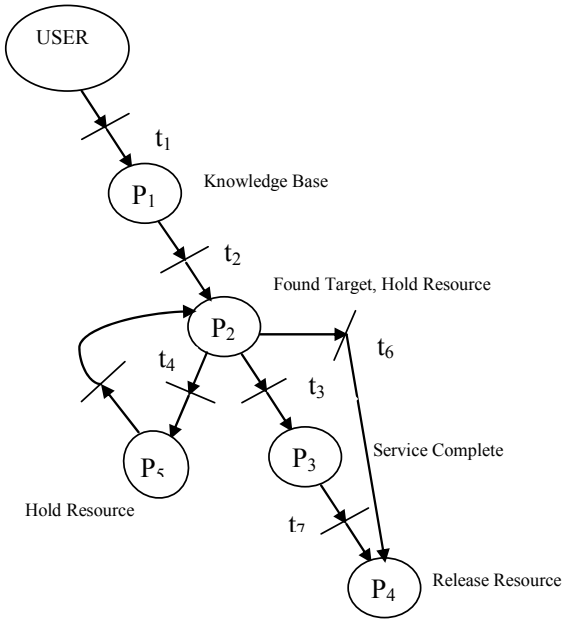

Figure 2: Generic PN Representation of Conceptual Framework

performed by any generic agent and those are *Initiate* (s_1), *Handle resources* (s_2), *Handle knowledge* (s_3), *Handle constraints* (s_4), *Handle properties* (s_5) and *Goal completed* (s_6). Those can be mapped into p_1, p_2, p_3, p_4, p_5 and p_6 respectively. The Generic PN with the specified generic set of transitions and places has been shown in Figure 2.

4 Behavioral Analysis of Conceptual Agent

The behavioral aspect of an AgOS can be studied once the resulted PN can be developed for that system. For the purpose first we need to map the components of the conceptual agents of such system into the well defined places and transitions. The mapping rules for that have been discussed in section 3. On next the several properties of the resulted PN can be studied to analyze the crucial behavioral features of the AgOS in respect to the target environment. The analysis can be performed using the incidence matrix and reachability graph of the resulted PN.

For the example described in the section 2.1, the places and transitions have been summarized in Table 1 and Table 2 respectively. The proposed mapping rules have been used to devise the said places and transitions.

Using the Table 1 and Table 2, we can draw the required PN for the system as shown in Figure 3. The process starts from a place p_0 which is the user initiate and after a transition t_1 will reach a place $\boldsymbol{p_1}$. The process continues further on and we finally arrive at the place p_{12}**.** Serially as the transitions occur the process moves on to each of the places as explained in the tables. If the process is interrupted then from place p_3 it will follow the path of places p_7, p_8, p_9 for the transition t_7, t_8, t_9, t_{10} respectively. If the process is cancelled then the path of places p_{10}, p_{11}, p_{12} will be followed for the transitions t_{11}, t_{12}, t_{13} respectively. If during the process no *IP address* is left to be processed then because of the transition t_{13} the place p_{12} is reached. If unprocessed *IP address* is still found then the process follows the path of place p_6, p_1 via transition t_{10}.

Table 1: Places for PN

Places	Components Of The Place
p_0	It consists of the Agent User.
p_1	Knowledge k_1
p_2	Service s_1; Property pr_2; resource r_2 on hold
p_3	Service s_5 s_6 s_8; Knowledge k_3 and resource r_3 updated
p_4	Service s_2; Property pr_1 and pr_4 updated
p_5	Resource r_2 ,r_3 released and r_4 restarted; Property pr_4 updated; check constraint c_3 from k_1
p_6	Check c_1 from r_4; release r_4; set property pr_3
p_7	Hold property r_1, r_2, r_3
p_8	Service s_3
p_9	Update property pr_4
p_{10}	Release r_1 r_2 r_3 r_4
p_{11}	Service s_4; Property pr_4 updated
p_{12}	Service s_7

Table 2: Transition for PN

Transitions	Details Of The Events / Transitions
t_1 for e_1	Initiate.
t_2 for e_2	User Interrupt.
t_3 for e_3	User Resume.
t_4 for e_4	User Cancel Job.
t_5 for e_5	Search.
t_6 for e_6	Target Computer Found.
t_7 for e_7	Service Init.
t_8 for e_8	Service Resume.
t_9 for e_9	Service Completed.
t_{10} for e_{10}	Service Interrupt.
t_{11} for e_{11}	Service Revoked.
t_{12} for e_{12}	Timer start.
t_{13} for e_{13}	Terminate.
t_{14} for e_{14}	No Action.

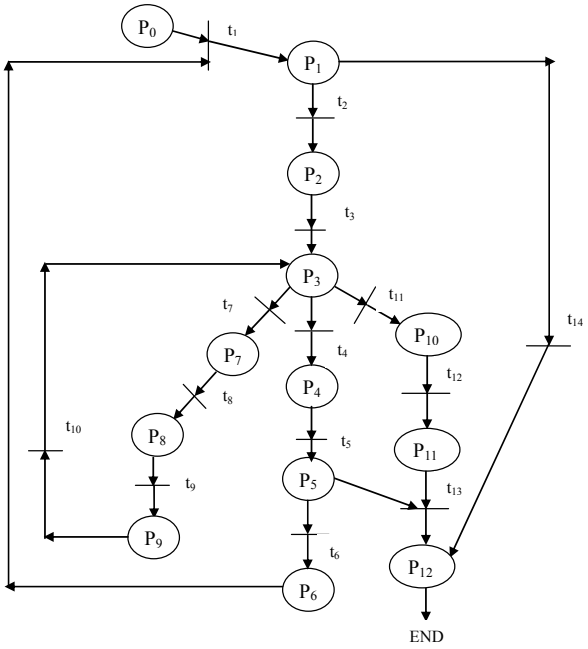

Figure 3: PN for the Agent Based Example

4.1 Incidence Matrices of PN Model

There is a pre-incidence matrix (Table 3) representing the initial state, Post-incidence matrix (Table 4) representing operational state after firing of the set of events of the agent and the combined matrix (Table 5) representing the token status at any instance after initiating the process of some agent. Each of these matrix has been formed using the row constituents p_0, p_1, …, p_{12} and the column constituents t_1, t_2, …., t_{14}.

In the resulted PN model, among the places p_0, p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}, p_{12} none of them are covered and hence the net is not covered by *P* invariants. The same is the case for the transitions and the net is not covered by *T* invariants.

Table 3: Pre-incidence matrix for PN Model

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}
p_0	1	1	0	0	0	0	0	0	0	0	0	0	0	1
p_1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
p_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
p_3	0	0	0	1	0	0	1	0	0	0	1	0	0	0
p_4	0	0	0	0	1	0	0	0	0	0	0	0	0	0
p_5	0	0	0	0	0	1	0	0	0	0	0	0	0	0
p_6	1	0	0	0	0	0	0	0	0	0	0	0	0	0
p_7	0	0	0	0	0	0	0	1	0	0	0	0	0	0
p_8	0	0	0	0	0	0	0	0	1	0	0	0	0	0
p_9	0	0	0	0	0	0	0	0	0	1	0	0	0	0
p_{10}	0	0	0	0	0	0	0	0	0	0	0	1	0	0
p_{11}	0	0	0	0	0	0	0	0	0	0	0	0	1	0
p_{12}	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 4: Post-incidence Matrix for PN Model

	t_1	T_2	T_3	t_4	T_5	t_6	t_7	t_8	t_9	t_{10}	T_{11}	t_{12}	t_{13}	t_{14}
p_0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
p_1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
p_2	0	1	0	0	0	0	0	0	0	0	0	0	0	0
p_3	0	0	1	0	0	0	0	0	0	1	0	0	0	0
p_4	0	0	0	1	0	0	0	0	0	0	0	0	0	0
p_5	0	0	0	0	1	0	0	0	0	0	0	0	0	0
p_6	0	0	0	0	0	1	0	0	0	0	0	0	0	0
p_7	0	0	0	0	0	0	1	0	0	0	0	0	0	0
p_8	0	0	0	0	0	0	0	1	0	0	0	0	0	0
p_9	0	0	0	0	0	0	0	0	1	0	0	0	0	0
p_{10}	0	0	0	0	0	0	0	0	0	0	1	0	0	0
p_{11}	0	0	0	0	0	0	0	0	0	0	0	1	0	0
p_{12}	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Table 5: Combined Matrix for PN Model

	t_1	T_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}
p_0	-1	0	0	0	0	0	0	0	0	0	0	0	0	-1
p_1	1	-1	0	0	0	0	0	0	0	0	0	0	0	0
p_2	0	1	0	0	0	0	0	0	0	0	0	0	0	0
p_3	0	0	1	-1	0	0	-1	0	0	0	-1	0	0	0
p_4	0	0	0	1	-1	0	0	0	0	0	0	0	0	0
p_5	0	0	0	0	1	-1	0	0	0	0	0	0	-1	0
p_6	-1	0	0	0	0	1	0	0	0	0	0	0	0	0
p_7	0	0	0	0	0	0	1	-1	0	0	0	0	0	0
p_8	0	0	0	0	0	0	0	1	-1	0	0	0	0	0
p_9	0	0	0	0	0	0	0	0	1	-1	0	0	0	0
p_{10}	0	0	0	0	0	0	0	0	0	0	1	-1	0	0
p_{11}	0	0	0	0	0	0	0	0	0	0	0	1	-1	0
p_{12}	0	0	0	0	0	0	0	0	0	0	0	0	1	1

4.2 Reachability Graph

The reachable place of a PN can be expressed by the reachability graph, which is a directed graph. The nodes of the graph are identified as markings of the net *R(N, M_0)*, where M_0 is the initial marking and the arcs are represented by the transitions of *N*. The graph is used to define a given PN *N* and marking *M*, where *M* belongs to *R(N)*. Each initial marking M_0 has an associated *Reachability* set. This set consists of all the markings that can be reached from M_0 through the firing of one or more transitions. In our PN model the reachability graph starts with initial marking $M_0 = [1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$ and finally reach to state $M_{12} = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1]^T$, where we conclude the agent session for the current process of shutdown. The reachability graph has been shown in Figure 4.

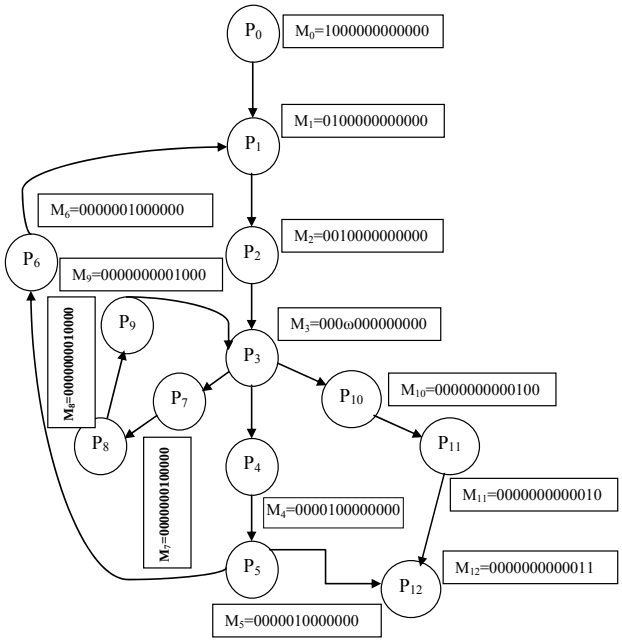

Figure 4: Reachability Graph for PN Model

As stated earlier, the mark of each place in *P* of PN model will be treated as 1 or 0 because all the agent related sub components of any output place $p_i \in P$ will be affected simultaneously.

4.3 Analysis of PN Model

In this sub section some of the crucial properties and behavior of the PN model of the example agent have been analyzed using the PN model and Reachability Graph presented in Figure 4 and Figure 5 respectively.

(a) Safeness: Any place of a Reachability graph is declared safe, if the number of tokens at that place is either *0* or *1*. In our PN model, the graph clearly shows that any of the places within p_0 to p_{12} represents a combination of *0* (no token) and *1* (token), which implies that if the firing occurs there will be a token at the position bit otherwise no token. Thus it shows each of the places has a maximum token count *1* or *0* and is declare safe. Also as all the places in the net are safe, the net as a whole can be declared safe.

(b) Boundedness: The boundedness is a generalized property of safeness. The limitation of token numbers in a place is restricted to *1* in case it is safe. It may enhance to some integer *i*, where *i* is known before hand for a place or we call it as a constraint to check the overflow condition at any stage calculated once the agent process start. When there is no overflow at any place, then the design guarantees the boundedness of the model. In our case there is no deadlock and at any stage within p_0 to p_{12} and hence it is bounded.

(c) Reachable: Reachability is a fundamental basis for studying the dynamic properties of any system. The firing of an enabled transition will change the token distribution

(markings) in a net according to the transition rules. A sequence of firings will result in a sequence of markings. A marking M_n is reachable from some marking M_0, if there exist a sequence of transitions that transforms M_0 to M_n. In our example all the markings are reachable starting from any marking in the net and hence reachability exists. This guarantees that the PN model for the AgOS will meet the pre-specified goal.

(d) Liveness: The liveness property of a PN is used to show continuous operation of the net model and ensure that the system will not get into a deadlock state as the process of interrupt or cancel or initiate needs to perform some transitions. If any marking exists in the graph such that no transitions are enabled from that marking, then that marking represents a deadlocked state, and the PN lacks the liveness property. Otherwise it is declared live. In our case there is no such deadlocked state or marking present in the net due to a series of events for the specified agent. Hence the net is live. Also in the example, if we have *k* computers to be shutdown then it means that transition *t* can be fired at least *k* number of times in some firing sequence. Hence the PN model is *L2* live.

(e) Conservativeness: Conservation property of a PN model checks the number of tokens remains constant before and after the execution. The process is to count the sum of all tokens at their initial markings and again after the execution. If all the markings in the reachability graph have the same sum of tokens then the Petri net is declared to be strictly conservative. The PN model of AgOS example is also strictly conservative.

5 Conclusion

In this paper, a methodology has been proposed to analyze the dynamic behavior of AgOS. Any such system comprises of autonomous entity called agent. They are highly dynamic in nature in terms of its interactions with the environments, handling of environmental resources, activities to achieve the pre specified goal and acquisition of knowledge. Petri net is best suitable tools to model and analyze such system behavior. To conceptualize the agent based system one need to follow entirely new paradigm than the object oriented paradigm. In this paper, firstly, we have proposed a conceptual level framework for agent as well as for such system to represent AgOS in simpler form and which can comply with the crucial features of such system. On next, to model the dynamic behavior of agent, we have used classical Petri Net as tool. A set of mapping rules also have been proposed for presenting the elements of conceptual framework for agent into the Petri net components. It also resulted a Generic Petri Net model for the agent oriented system. Finally we have used the Petri net model and its reachability graph to analyze the dynamic features of agent oriented system.

The model works fine for the system composed of simple agents. The proposed methodology also is useful for the analysis of external and internal behavior of agents. But the methodology will become less expressive for large system comprised of multiple agents and with complex agent level interactions. Future work includes, the development of a mechanism for more expressive behavioral analysis model of agent oriented system using High Level Petri Net by extending the proposed model.

6 References

[1] M. Wooldridge, P. Ciancarini, *"Agent-Oriented Software Engineering: The State of the Art"*, Book Title: Agent-Oriented Software Engineering, Springer – Verlag Lecture Notes in AI, Vol. 1957, pp 1-28, January 2001.

[2] Tadao Murata, *"Petri Nets: Properties, Analysis and Applications"*, Proceedings of the IEEE, Vol. 77, No. 4, pp 541 – 580, April 1989.

[3] F. Zambonelli, A. Omicini, *"Challenges and Research Directions in Agent-Oriented Software Engineering"*, Jnl. of Autonomous Agents and Multi-Agent Systems, Vol. 9, pp 253–283, 2004.

[4] B. Bauer, J. P. Müller, J. Odell, *"Agent UML: A Formalism for Specifying Multiagent Software Systems"*, International Journal of Software Engineering and Knowledge Engineering, Vol 11, No. 3, pp.1-24, 2001.

[5] P. Gruer, V. Hilaire, A. Koukam and K. Cetnarowicz, *"A Formal Framework for Multi-Agent Systems Analysis and Design"*, Journal of Expert Systems with Applications, Vol. 23, No. 4, pp. 349–355, 2002.

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

[7] F. Zambonelli, N. R. Jennings, M. Wooldridge, *"Developing Multiagent Systems: The Gaia Methodology"*, ACM Trans. on Software Engineering and Methodology, Vol. 12, No. 3, pp 317 – 370, 2003.

[8] S. A. Deloach, M. F. Wood, C. H. Sparkman, *"Multiagent Systems Engineering"*, International Journal of Software Engineering and Knowledge Engineering, Vol. 11, No. 3, pp 231 – 258, 2001.

[9] F. Giunchiglia, J. Mylopoulos, A. Perini, *"The Tropos Software Development Methodology: Processes, Models and Diagrams"*, 3rd Intl. Conference on Agent-Oriented Software Engineering (AOSE'02), pp 162-173, 2003.

[10] Jürgen Lind, *"Issues in Agent-Oriented Software Engineering"*, Transaction on Agent-Oriented Software Engineering, Springer-Verlag pp 45-58, June 2009 .

[11] B. Bauer, J. P. Muller, J. Odell, *"Agent Uml: A formalism for specifying Multiagent Software Systems"*, International Journal of Software Engineering and Knowledge Engineering, Vol. 11, No. 3, pp 1 – 24, 2001.

[12] B. Marzougui, K. Hassine, K. Barkaoui, *"A New Formalism for Modeling a Multi Agent Systems: Agent Petri Nets"*, Journal of Software Engineering and Applications, Vol. 3, No. 12, pp 1118-1124, 2010.