Modeling and Analysis of Multi-agent Systems using Petri Nets

Conference Paper *in* Journal of Computers · November 2007

DOI: 10.1109/ICSMC.2007.4413960 · Source: IEEE Xplore

3 authors, including:

Jose Celaya
NASA
87 PUBLICATIONS **3,709** CITATIONS
SEE PROFILE

Alan A. Desrochers
Rensselaer Polytechnic Institute
112 PUBLICATIONS **2,955** CITATIONS
SEE PROFILE

Modeling and Analysis of Multi-agent Systems using Petri Nets

Jose R. Celaya, Alan A. Desrochers, and Robert J. Graves
Email: jcelaya@mail.arc.nasa.gov, aad@ecse.rpi.edu, and Robert.Graves@Dartmouth.edu

***Abstract*—The development of theoretical-based methods for the assessment of multi-agent systems properties is of critical importance. This work investigates methodologies for modeling, analysis and design of multi-agent systems. Multi-agent systems are regarded as discrete-event dynamic systems and Petri nets are used as a modeling tool to assess the structural properties of the multi-agent system. Our methodology consists of defining a simple multi-agent system based on the abstract architecture for intelligent agents. The abstract architecture is modeled using Petri nets and structural analysis of the net provides an assessment of the interaction properties of the multi-agent system. Deadlock avoidance in the multi-agent system is considered and it is evaluated using liveness and boundedness properties of the Petri net model.**

***Index Terms*—Petri nets, multi-agent systems, deadlock.**

I. Introduction

Multi-agent systems have been studied for the past few decades. Several multi-agent systems frameworks have been defined in order to apply the multi-agent system concept to different applications in control and optimization of complex systems [1]–[4]. An agent is a computer system or computer program that presents several complex characteristics. The most important characteristic of an agent is that of 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 agent or autonomous agent is an agent that presents some degree of autonomous flexibility by being reactive, pro-active and perhaps sociable [3]. This flexible autonomy is a building block in a multi-agent system. In a multi-agent system, several agents communicate and interact in order to solve a complex problem.

Several interaction frameworks have been defined and they range from collaboration among agents, through competition for resources requiring some level of negotiation in the multi-agent system [3], [7]. In general, the multi-agent system is expected to work properly in a dynamic large-scale complex environment (open environment) by having autonomy, adaptability, robustness, and flexibility.

This work presents analytical methodologies for modeling and analysis of multi-agent systems. Multi-agent systems are regarded as discrete-event dynamic systems and Petri nets are used as a modeling tool to assess properties of the system.

This document is organized as follows. The problem statement and related work are presented next. Section II presents a brief introduction to multi-agent systems which considers the abstract architecture of intelligent agents, as well as a description of communication and interaction in multi-agent system. Section III presents an introduction to Petri nets. Section IV shows a simple multi-agent system which is modeled with the abstract architecture for intelligent agents as well as Petri nets. Section V presents a methodology to model and analyze multi-agent systems with indirect communication. Finally, section VI discusses the results.

A. Motivation

The complexity and capabilities of a multi-agent system are greater than those presented in distributed software systems. In both cases the study of system properties is becoming more important due to the fact that we are faced more and more with dealing with large complex dynamic systems. Computer simulation is generally used to assess system properties and to verify that the system is achieving its design objectives. An important challenge in this field is the development of analytical methods to assess key properties of such systems [1], [8]. Such methods could be used to provide a preliminary analysis of the multi-agent system, providing design and operation feedback before the development of expensive simulation models.

B. Problem statement

A multi-agent system can be studied as a computer system that is concurrent, asynchronous, stochastic and distributed. These characteristics of multi-agent systems make them also a discrete-event dynamic system, and these have been studied under several analytical methodologies, particularly Petri nets. Petri nets have a well-defined mathematical structure that can be leveraged to provide formal analysis on discrete-event systems. In addition, Petri Nets have been successfully used in several

J. R. Celaya is with the Research Institute for Advanced Computer Science at NASA Ames Research Center, Moffett Field, CA 94035; previously with the Decision Sciences and Engineering Systems Department, Rensselaer Polytechnic Institute, Troy, New York 12180.

A. A. Desrochers is with the Electrical, Computer, and Systems Engineering Department, Rensselaer Polytechnic Institute, Troy, New York 12180 (corresponding author phone: 518-276-6718; fax: 518-276-8715; e-mail: aad@ecse.rpi.edu).

R. J. Graves is the Krehbiel Professor of Emerging Technologies, Thayer School of Engineering, Dartmouth College, Hanover, New Hampshire 03755.

areas for the modeling and analysis of distributed systems [9], flexible manufacturing systems [10]–[13], concurrent and parallel programs [14], [15]. From the discrete-event dynamic system (DEDS) point of view, multi-agent systems lack analysis and design methodologies. Petri net methods are used in this work to develop analytical methodologies for such systems.

As mentioned earlier, there are several frameworks and applications of multi-agent systems. High-level design is considered here as the actual architecture of individual agents, and the communication and interaction frameworks to use. This work focuses on the interaction level, and studies the interactions between different agents in the system. This will provide insight in how the interaction impacts the operation and performance of the multi-agent system.

C. Methodology

The problem addressed in this study can be considered in two parts: properties, and methodologies for modeling and analysis.

1) Properties: If a multi-agent system is regarded as a discrete-event system and modeled using Petri nets, then properties known to be important in the discrete-event systems and Petri net domains could be used to study multi-agent systems. There are properties we would like to analyze and demonstrate for multi-agent systems. Examples of these are boundedness and liveness as related to deadlock avoidance in the Petri net domain. These properties from the Petri net domain could be related to characteristics of the communication and interaction of the multi-agent system. If modeled properly, a deadlock found in a Petri net domain will mean that the interaction mechanism in use in the multi-agent system is prone to deadlocks in the interaction among individual agents.

2) Methodologies for modeling and analysis: Considering that a multi-agent system can be regarded as a discrete-event system, Petri nets can be used as a modeling tool. This will require methodologies for mapping multi-agent systems into Petri net models. These methodologies will 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. Having Petri net models of multi-agent systems will allow us to use the existing analysis methodologies for Petri nets. Important properties of discrete-event systems will be obtained with Petri net analysis methods such as the reachability graph and the analysis of the network invariants.

D. Related work

Petri nets and Petri net extension methodologies have been used to model systems with more than one agent. Murata et al. [16] presented 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). Even though the robotic system considered does not have direct interaction between the agents, the model used shows the ability of the Petri net-like models to capture interactions between the agents that are not evident in the design process. In a similar way, Xu et al. [17] 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. [18] proposed a 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. [19], proposed a Petri net model approach to formal specification of holonic control systems for manufacturing. They developed a Petri net submodel for each of the four types of holons (agents) suggested in the ADACOR (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.

II. Multi-agent Systems

The architecture of the agents in a multi-agent system and the interaction among them are the two main aspects in a multi-agent system framework. The architecture of an agent can be *reactive, deliberative* or a *hybrid* of both. The concepts from single agents as *"perception and action,"* and *"belief, desire and intention"* are being extrapolated to the concept of multi-agent systems [3]. Several interaction frameworks have been presented in the literature.

The interactions between agents can be either a direct agent-to-agent interaction or an indirect interaction. 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. An agent that is part of the multi-agent system has its own environment that is somehow related to the meta level environment of the multi-agent system. This meta level environment of the multi-agent system is described in the literature as being an *open* environment. 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 [8].

The abstract architecture of a single intelligent agent is presented by Wooldridge in [3] and [7] allows the representation of reactive, deliberative or hybrid agents

in a higher level of abstraction. It is based on the concept of perception and action where the agent is assumed to be living in an environment and reacting to changes on it [5]. The level of abstraction of agents modeled by the abstract architecture makes it a good candidate for a study of multi-agent systems as discrete-event dynamic systems. The concepts of the single agent abstract architecture can be extrapolated to multi-agent systems by assuming first a very simple communication and interaction scheme between the agents.

A. Intelligent agent architectures

Abstract architecture for intelligent agents: 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.

The environment of an agent only considers those things that matter to that specific agent. Consider two agents controlling different elements of a manufacturing cell. For example one agent 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.

The block diagram of figure 1 shows how the agent works. The Perception block records the changes in the state of the environment. The Action block computes the actions to be taken in order to change the environment. The agent environment changes based on the actions applied by the agent, as well as actions by other agents, and it may be dynamic in that it may change by itself.

An agent can also have an internal state as a decision mechanism for the actions to undertake. 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].

Definition 1: (Purely reactive agents with no internal state) The environment consists of a set of states $S = \{s_1, s_2, \cdots\}$. The agent can undertake a set of actions $A = \{a_1, a_2, \cdots\}$ and perceive a set of percepts $P = \{p_1, p_2, \cdots\}$. For a purely reactive agent, the behavior of the agent can be represented as the function

$$action : P \mapsto A \tag{1}$$

for the action block (refer to figure 1), and the function

$$perception : S \mapsto P \tag{2}$$

for the perception block. The deterministic behavior of an environment can be represented by the function

$$environment : S \times A \mapsto S. \tag{3}$$

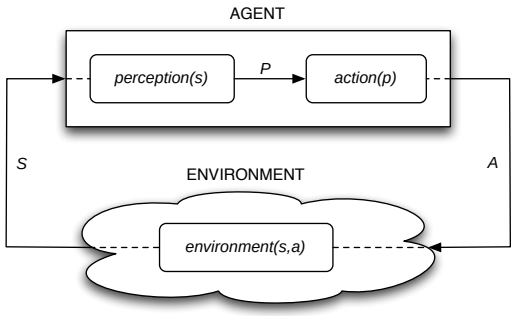

Fig. 1. Agent abstract architecture block diagram.

The perception part has to do with the way the agent senses the environment, which is called a percept. The perception function maps the environment state into a percept.

Having the perception mechanism provides a complete and more general characterization of the agent's abstract architecture by separating the sensory capabilities (perception) from the decision making capabilities (action). It is intended to represent the agent's capability of sensing the environment. Suppose that the agent will take the same actions for environmental states s_1 and s_2. This means that both states have the same percept: $perception(s_1) = perception(s_2) = p_1$, and then the action function will execute the same action as if it depends on the percepts instead of the environmental state.

Agents can also be defined with an internal state and the actions that the agents undertake depend on the internal state only. Figure 2 presents a block diagram of the complete abstract architecture. From this we can see that the capabilities of the abstract architecture previously presented are now enhanced in the complete definition, but the essential perception/action paradigm is still consistent. An intelligent agent with internal state will react to the changes in its internal state instead of reacting directly to the changes in the environment. The internal state will then change based on changes in the environment as processed by the perception mechanism.

Definition 2: (Purely reactive agents with internal state) Let a discrete set $I = \{i_1, i_2, \cdots\}$ be the set of all internal states, and S and P be the set of environmental states and percepts respectively as described in definition 1. Then the $action(\cdot)$ function will be defined as $action : I \mapsto A$ where A is the set of actions as described before. Since the agent's actions are now dictated by the evolution of the internal state and not that of the environment, an additional function is needed to describe that evolution as a function of the percepts and the current state; that function is defined as $new\_state : I \times P \mapsto I$. An execution loop is defined for the complete abstract architecture in [7].

B. Communication and interaction

The *communication* in multi-agent systems is the way agents exchange and interpret messages among themselves, it is concerned with the language to be used and its

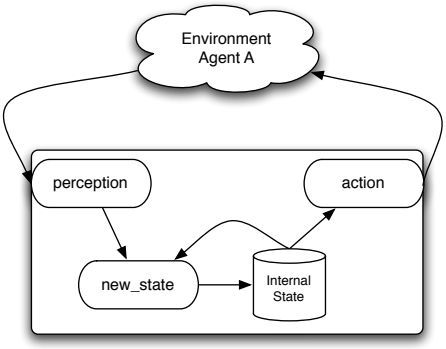

Fig. 2. Complete definition of the intelligent agent abstract architecture.

meaning in a certain context. A communication protocol defines a language for the exchange and understanding of messages between the agents [3], [5], [7]. *Interaction* in multi-agent systems is the way conversations take place among agents, which is provided by a structure for the exchange of messages. Such conversation is the passing of multiple messages among agents both ways. Interaction deals with a higher level of abstraction than communication. It defines if the agents will cooperate to execute tasks, or will compete for resources, or if they will engage in negotiations in order to fulfill their individual goals in the case of self-interested agents.

The *interaction* mechanism in multi-agent systems is the structure of the conversations that can take place among agents, and it is intended to provide social behavior which is essential to achieve system level goals [23]. Several issues related to interaction are present in the design of a multi-agent system. The interactions depend on the problem decomposition. As a result, there will be a need to try different types of interactions from the ones currently described in the literature or develop a new methodology for the specific problem [8], [23]. It is also not clear how to ensure coherence. The rule of thumb is to exploit positive interactions and avoid negative interactions but there is no clear methodology to identify conflicts nor direct how to solve them [8], [23]. It is also not clear how to coordinate the agents and how to make the agents reason about such coordination [8].

III. Introduction to Petri nets

Petri nets are a graphical and mathematical modeling tool used to describe and analyze different kinds of real systems. Petri nets were first introduced by Carl Adam Petri in 1962 in Germany [15], and evolved as a suitable tool for the study of systems that are concurrent, asynchronous, distributed, parallel and/or stochastic. A multi-agent system is a kind of DEDS that is concurrent, asynchronous, stochastic and distributed. From the DEDS point of view, multi-agent systems lack analysis and design methodologies. Petri net methods are used in this study to develop analytical methodologies for multi-agent systems.

A. Petri nets definition

Definition 3: The following is the formal definition of a *Petri net* [14], [15], [29], [30]. A Petri net is a five-tuple

$$(P, T, A, W, M_0) \tag{4}$$

where:
P is a finite set of *places*
T is a finite set of *transitions*
$A \subseteq (P \times T) \cup (T \times P)$ is a set of *arcs*
$W : A \mapsto \{1, 2, 3, \ldots\}$ is a *weight* function
$M_0 : P \mapsto Z_+$ is the initial *marking*

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 DEDS a transition is regarded as an event and the places are interpreted as a condition for an event to occur.

Places, transitions and arcs: Places are represented with circles and transitions are represented with bars. The arcs are directed from places to transitions or from transitions to places. The places contain *tokens* that travel through the net depending on the firing of a transition. A place p is said to be an input place to a transition t if an arc is directed from p to t. Similarly an output place of t is any place in the net with an incoming arc from transition t.

Transition firing: 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*[1]. 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.

Definition 4: (Marking) The *marking* m_i of a place $p_i \in P$ is a non-negative quantity representing the number of tokens in the place at a given state of the Petri net. The marking of the Petri net is defined as the function $M : P \mapsto Z_+$ that maps the set of places to the set of non-negative integers. It is also defined as a vector $M_j = (m_1, m_2, \cdots, m_{|P|})$ were $m_i = M(p_i)$, 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 of the Petri net 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 DEDS.

Definition 5: (Reachability graph) The *reachability graph* has the marking of the Petri net (or state of the Petri net) as a node. An arc of the graph joining M_i with M_j represents the transition when firing takes the Petri net from the marking (state) M_i to the marking M_j.

[1]Assuming the weights W of the Petri net are equal to one. When the weights are not indicated they are assumed to be one. The weight in an arc coming to a transition from one of the incoming places indicates the minimum number of tokens needed in the incoming place in order for that transition to be enabled. When the transition fires, it will remove from the incoming place the amount of tokens indicated by the weight of the arc.

B. Properties

This section covers some of the most important *properties* of Petri nets such as *Reachability, Liveness, Boundedness* and *Reversibility*. These are properties that could 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. The analysis methods developed in this research will focus on the following properties.

Definition 6: (Reachability) A marking M_j is said to be reachable from marking M_i if there exists a sequence of transitions that takes the Petri net from state M_i to M_j . The set of all possible markings that are reachable from M_0 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 [15], [28].

Definition 7: (Liveness) A Petri net is said to be live for a marking M_0 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 [15], [28].

Definition 8: (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* [15].

Definition 9: (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 [15], [28].

C. Structural analysis

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.

Definition 10: (Incidence 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 $a_{ij} = a_{ij}^{+} - a_{ij}^{-}$.

Definition 11: (Net-invariants) Let A be the incidence matrix. A P-invariant is a vector that satisfies the equation

$$Ax = 0 \tag{5}$$

and a T-invariant is a vector that satisfies the equation

$$A^T y = 0. \tag{6}$$

1) 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 M_0 is finite.

2) 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.

IV. Results in Modeling Multi-Agent Systems using Petri Nets

A simple multi-agent system is presented and studied in this section. The multi-agent system is modeled using the *abstract architecture* representation presented in section II-A. In addition, the multi-agent system is modeled using a Petri net with the intention of carrying out a structural analysis of the multi-agent system as a *discrete-event system*, that is, focusing on the *liveness* and *boundedness*, as they relate to the absence of deadlock in the system.

The multi-agent system presented here is loosely based on the behavior of ants as they move food from one place to another. The system consists of two ants, and each one will be modeled as an intelligent agent. They move objects from one place to another which will be the environment of the agents. The system under consideration can be viewed in two different ways: a) the system consists of two "physical" agents and the abstract architecture and Petri nets are used to model their behavior; b) the abstract architecture and Petri nets are used to model, analyze and design the reasoning/control of the two agents so they could perform their intended tasks in such an environment.

The objective of the work presented in this section is to present a proof of concept on how Petri nets can be used to model and analyze a simple multi-agent system. Furthermore, it also indicates how the abstract architecture of intelligent agents can be used to model the behavior of an intelligent agent. Three different scenarios based on the multi-agent system (cases) are modeled and analyzed. First, the system is considered to have an infinite number of objects to be moved (*case 1*), resulting in a system that is known in advance to be deadlock free. A Petri net model N_1 of case 1 is analyzed to corroborate that it is deadlock free. In addition, an abstract architecture model M_1 is presented for both agents. The system presented in *case 2* considers a finite number of objects to be moved and will fall into deadlock since the agents do not have the capability to handle such a situation (no objects to be moved). An updated Petri net model N_2 is analyzed to check that the deadlock can be detected by the assessment of Petri net properties. The multi-agent system is upgraded in *case 3* so it is able to handle a finite number of objects. A new abstract architecture model M_3 is presented including additional actions and

environmental states for each agent. A Petri net model N_3 is also presented to help demonstrate why the modeling approach (selection of places and transitions and detail level) is important to ensure that the key properties are captured by the Petri net model. It supports the theory that the abstract architecture can be used as an intermediate step to obtain a Petri net model.

A. Description of the multi-agent system

This section presents a description of a simple multi-agent system consisting of two physical agents that work together at a task. This system will be used throughout this section to illustrate how the Petri net and abstract architecture models can be applied to multi-agent systems.

This system consists of two agents referred to as *agent A* and *agent B*. They move objects from one end of a path to the other. Figure 3 shows 4 scenarios in the operation of the system. Both agents are capable of moving objects and agent A moves faster than agent B. The objective of agent A is to go to the end of the path, pick an object, and return to the start of the path (figure 3a). Since it is faster than agent B it reaches the objects first (figure 3b). At the time they intersect in the path, agent A gives its object to agent B and returns to the end of the path to pick another object (figures 3c and 3d). On the other hand, the objective of agent B is to go in the direction of the end of the path, intersect with agent A and get its object. Once the agent has an object, it returns to the start of the path, places the object down and starts all over again.

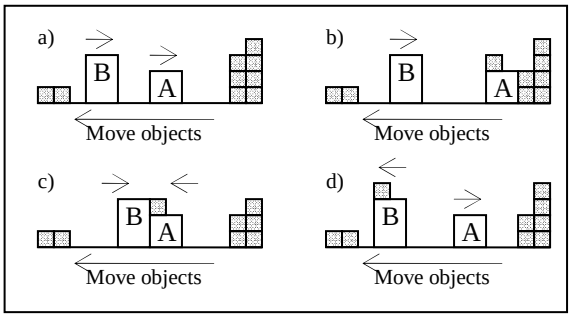

Fig. 3. Multi-agent system for modeling and analysis.

The *meta level environment* of the system consists of a single path with a set of objects on one side and no objects on the other side. The amount of objects at the end of the path can be considered finite or infinite and both cases will be considered in the following sections (see table I).

There is indirect interaction between the agents via changes in their environments. The exchange of the object between the agents should be regarded as a result of a change in the environment of both agents. The two agents are regarded as *purely reactive*, which implies they do not have a record of history and they do not have an internal state. The decisions they make are about which actions to undertake and those decisions are directly influenced by the location of the agent in the path and whether the agent has an object or not. It should be noted also, that the goal of the overall multi-agent system is a little different from the goal of each specific agent.

TABLE I
DESCRIPTION OF MODELS APPLIED TO THE MULTI-AGENT SYSTEM.

Case	Description	Petri net	Abstract architecture
1	Infinite number of objects to move.	N_1	M_1
2	Finite number of objects to move.	N_2	–
3	Finite number of objects to move. Added capabilities to agents to handle finite number of objects.	N_3	M_3

B. Case 1: Petri net and abstract architecture models

The multi-agent system described in the previous sections is studied here considering there is an infinite number of objects at the end of the path that are going to be moved to the start of the path by the agents. A Petri net model (N_1) of the system is presented and analyzed in order to check that the system is bounded and deadlock free. In addition, an abstract architecture model (M_1) is presented for both agents. The objective is to show how a multi-agent system can be modeled using the abstract architecture, as well as the Petri net methodologies.

1) Modeling as a discrete-event system using Petri nets: This section presents a Petri net model of the multi-agent system described in section IV-A. It assumes that the number of objects available at the end of the path is infinite. This implies that the system should operate without interruption based on the described agents' capabilities. The multi-agent system is considered as a discrete-event system and the liveness and boundedness properties are analyzed to check if the system is deadlock free. The purpose of this model is to show the ability of Petri net methodologies to model multi-agent systems including the agents interaction.

Several interpretations can be assigned to places and transitions. In the Petri net model presented in this section *places* model the activity that a particular agent is performing. Having a token in a place that represents a specific activity means that the agent is currently performing such activity. The *transitions* model the termination of an activity of a particular agent. Firing a transition indicates that the agent finished performing the activity indicated by the input place and will start performing the activity of the output place.

Let $N_1 = (P, T, A, W, M_0)$ be the Petri net presented in figure 4 that models the multi-agent system under consideration. Table II presents the description of the places $P = \{p_{A1}, p_{A2}, p_{A3}, p_x, p_{B1}, p_{B2}, p_{B3}\}$, and $W : A \mapsto 1$ therefore the weights for all the arcs are omitted from the graph. The transitions are $T = \{t_1, t_2, t_3, t_4, t_5, t_6\}$ and the initial marking $M_0 = [1, 0, 0, 0, 1, 0, 0]$.

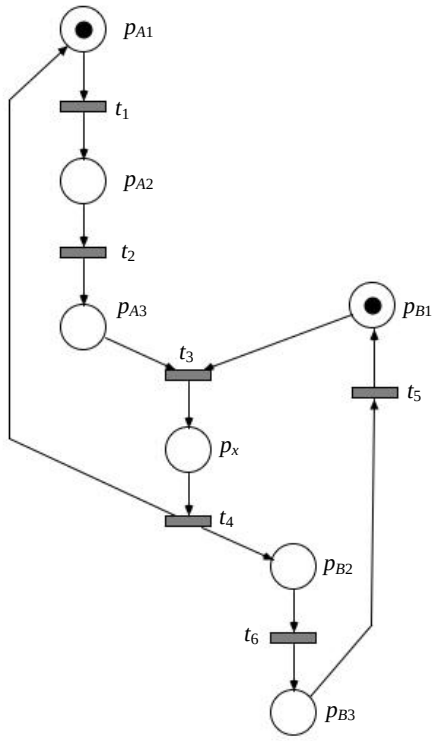

Fig. 4. Petri net model N_1 for system with infinite number of objects.

TABLE II
PLACES DESCRIPTION FOR N_1.

Places (P)	Description
p_{A1}	Agent A going to the end of the path to pick object
p_{A2}	Agent A is at the end of the path loading an object
p_{A3}	Agent A is going to start of the path to intersect with agent B
p_{B1}	Agent B going to the end of the path to intersect with object A
p_{B2}	Agent B going to the start of the path to leave object
p_{B3}	Agent B unloading object at the start of the path
p_x	Both agents are exchanging the object

2) Analysis of the reachability graph: At this point we are concerned with the net being *bounded* and *live*. An assessment of these properties based on the initial marking of the net can be performed by analyzing the *reachability tree* of N_1, using M_0 as the root node as shown in figure 5. A reachability tree is a graph representation of the markings of a net. Each node in the tree represents a marking and the edges represent a transition firing [15]. The resulting tree is small enough to be analyzed by inspection and it can be concluded that: a) the reachability set $R(M_0)$ is finite, b) the number of tokens of every place in all markings is bounded ($1 - bounded$), and c) there are no dead transitions (all transitions can fire). As a result, it can be concluded that the net is bounded.

By looking at the reachability graph of N_1 with initial marking M_0 presented in figure 6, it can be seen that there is always an active transition regardless of the state of the net, and all the transitions of T are included in the graph.

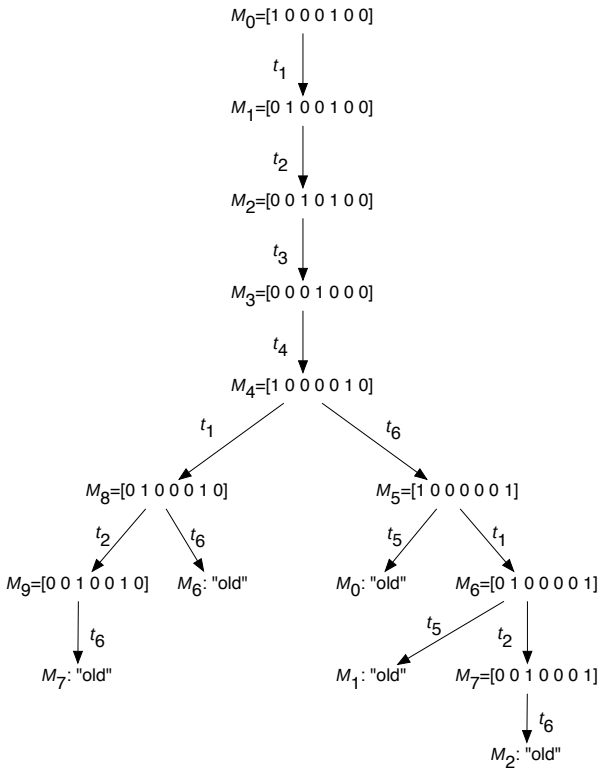

Fig. 5. Reachability tree of N_1.

From this it can be concluded that the Petri net model of the multi-agent system is bounded and live. This implies that the interaction between the agents does not fall into deadlocks.

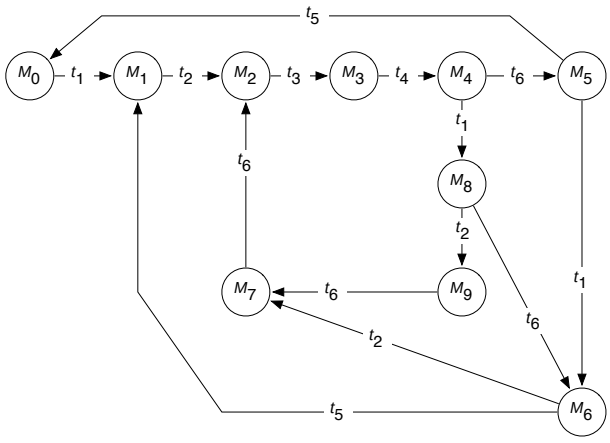

Fig. 6. Reachability graph for N_1.

For models with a large number of places and transitions, the reachability graph construction is not practical without the use of computer tools to automate the reachability graph generation and the assessment of properties. Under this scenario, it is preferable to use the algebraic network representation (incidence matrix) and invariance theorems to assess the structural net properties.

3) Structural analysis of N_1: The structural behavior of the net can be assessed using the algebraic analysis

of the incidence matrix (invariant analysis) as described in section III-C. The incidence matrix is defined as $A = [a_{ij}]$, where $a_{ij} = a_{ij}^{+} - a_{ij}^{-}$, $a_{ij}^{+} = w(i,j)$ is the weight of the arc from t_i to p_j, and $a_{ij}^{-} = w(j,i)$ is the weight of the arc from p_j to t_i. Let A_1 be the incidence matrix of N_1. The order of the places in the matrix is $P = \{p_{A1}, p_{A2}, p_{A3}, p_x, p_{B1}, p_{B2}, p_{B3}\}$ (columns), and the order of the transitions is $T = \{t_1, t_2, t_3, t_4, t_5, t_6\}$ (rows).

$$A = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & -1 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix} \quad (7)$$

A P-invariant is a vector that satisfies the equation $Ax = 0$, and a T-invariant is a vector that satisfies the equation $A^T y = 0$. The following invariants were obtained from the incidence matrix A_1. The T-invariant $y_1 = [1,1,1,1,1,1]^T$, the P-invariants $x_1 = [1,1,1,1,0,0,0]^T$, and $x_2 = [-1,-1,-1,0,1,1,1]^T$.

A net is said to be covered by P-invariants if and only if, for each place p in the net, there exists a positive P-invariant x such that $x(p) > 0$. The net N_1 is covered by P-invariants since there is a positive element either on x_1 or x_2 for every place, e.g., $x_1(p_{B1}) = 0$ but $x_2(p_{B1}) = 1$. In addition, a net 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 $y(t) > 0$. N_1 is also covered by T-invariants.

A Petri net is structurally bounded if it is covered by P-invariants and the initial marking M_0 is finite. Furthermore, a net is live and bounded if it is covered by T-invariants; this is a necessary condition only. The Petri net N_1 is covered by T-invariants and P-invariants. Since the initial marking is finite, we can conclude that the Petri net is bounded and that the necessary condition for liveness is met.

Discussion: A *marked graph* is a subclass of Petri nets where each place has exactly one incoming arc and exactly one outgoing arc [15] [30]. There is a theorem that states that every marked graph is live and also bounded if the initial marking is bounded [15]. The net N_1 is a marked graph and M_0 is bounded, therefore the net is live and bounded.

4) Abstract architecture model: The abstract architecture of intelligent agents presented in section II-A is used here to model the multi-agent system with an infinite number of objects present. Agents will be considered to be purely reactive where the environment consists of a set of states $S = \{s_1, s_2, \cdots\}$, the agent can undertake a set of actions $A = \{a_1, a_2, \cdots\}$, and perceive a set of percepts $P = \{p_1, p_2, \cdots\}$. For a purely reactive agent, the behavior of the agent can be represented as the function $action : P \mapsto A$ and $perception : S \mapsto P$. The deterministic behavior of an environment can be represented by the function $environment : S \times A \mapsto S$.

Let M_1 be the multi-agent system with agents A and B modeled with the abstract architecture. The model presented below assumes that $P = S$ for both agents, as a result $action : S \mapsto A$.

Abstract model for agent A: Agent A can undertake a set of actions A_A, its environment is defined by S_A and the function $action_A$ defines the actions to undertake based on the changes in the environment. The following tables describe the abstract model for this agent, the environmental states and actions descriptions are presented in tables III and IV respectively; equation 8 defines the action function.

TABLE III
DESCRIPTION OF STATES FOR AGENT A OF M_1.

States (S_A)	Description
s_1	Agent has object
s_2	Agent has no object and it is not at the end
s_3	Agent has no object and it is at the end

TABLE IV
DESCRIPTION OF ACTIONS FOR AGENT A OF M_1.

Actions (A_A)	Description
a_1	Walk to the end
a_2	Pick an object
a_3	Walk to start

$$action_A(s) = \begin{cases} a_1 & \text{if } s = s_2 \\ a_2 & \text{if } s = s_3 \\ a_3 & \text{if } s = s_1 \end{cases} \quad (8)$$

It should be noted that agent A has no actions for the exchange of an object at the intersection with agent B. Agent B is assumed to take over the object from agent A, thus changing the environment of agent A and incurring an indirect interaction.

Abstract model for agent B: Agent B can undertake a set of actions A_B, its environment is defined by S_B and the function $action_B$ defines the actions to undertake based on the changes in the environment. The environmental states and actions descriptions are presented in tables V and VI respectively; equation 9 defines the action function. Agent B is assumed to take over the object from agent A when they intersect. This action is reflected in A_B and function $action_B$.

TABLE V
DESCRIPTION OF STATES FOR AGENT B OF M_1.

State (S_B)	Description
s_1	Agent has no object
s_2	Agent has object and it is not at the start
s_3	Agent has object and it is at the start
s_4	Agent is at the intersection with agent A

TABLE VI
DESCRIPTION OF ACTIONS FOR AGENT B OF M_1.

Action (A_B)	Description
a_1	Walk to the end
a_2	Take over object from agent A
a_3	Walk to start
a_4	Put object down at start

$$action_B(s) = \left\{ \begin{array}{ll} a_1 & \text{if } s = s_1 \\ a_2 & \text{if } s = s_4 \\ a_3 & \text{if } s = s_2 \\ a_4 & \text{if } s = s_3 \end{array} \right\} \quad (9)$$

5) Discussion: The analysis performed in this section (case 1) shows that the multi-agent system under consideration has appropriate structural properties. This system will not go into a deadlock and will work correctly under the current modeling assumptions. Assumptions with respect to the agents' behavior may not hold if the assumptions in the agents environment change. The strongest assumption in this scenario case 1 (for abstract model M_1 and Petri net model N_1) is the fact that the number of objects to move is infinite. If this is changed to a finite number of objects, the current agent behavior may not work for the overall system objective; in fact, the multi-agent system will deadlock.

When analyzing the abstract model M_1 when this assumption does not apply, it can be seen that eventually, agent A will reach state s_3 and no object will be there to be picked. Then, action a_3 will have no effect and will not change the state of the local environment. In the case of agent B, it will eventually remain at state s_4 and action a_2 will not be executed as expected. In the case of the Petri net model M_1, t_2 will never fire even if it is active because the activity modeled by place p_{A2} cannot be concluded without objects to pick. This model fails and no longer represents the dynamics of the real system.

C. Case 2: Petri net model

The multi-agent system described in section IV-A is studied here considering now that there is a finite number of objects at the end of the path. Since the agents do not have the capability to handle a lack of objects to transport, it is expected that the system falls into deadlock. A Petri net model (N_2) is presented and analyzed in order to check that the system falls into deadlock.

1) Petri net model and analysis: Let $N_2 = (P, T, A, W, M_0)$ be the Petri net presented in figure 7 that models the multi-agent system under consideration. It is based on net N_1 presented in the previous section to model an infinite number of objects. An additional place was added to the previous model in order to include a representation of the finite number of objects to be moved from the end to the start of the path. This additional place will serve as a *buffer place*, and should be linked in a way to the action taken by agent A when picking the object at the end of the path. The buffer place p_b was added to model the finite number of objects. Transition t_2 is now active when there is a token in place p_{A2} (agent A is at the end of the path loading an object) and there is at least one token in place p_b (there is at least one object at the end of the path). From inspection of the Petri net it can be seen that eventually the net execution will reach a state with no enabled transitions. Consider the instance where $M(p_b) = 0$ (no more objects available at the end of the path) and $M(p_{A2}) = 1$. Transition t_2 can not be enabled anymore because p_b works as a source of tokens and have no incoming arcs from other transitions, therefore the net is not *live*.

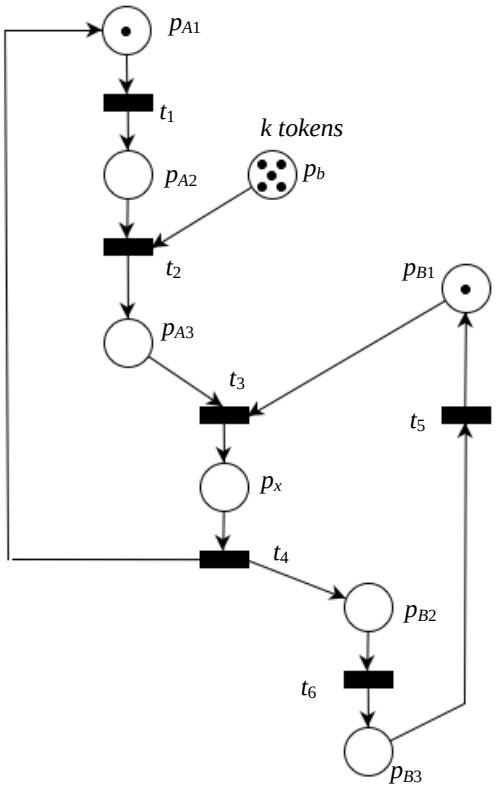

Fig. 7. Petri net model N_2 for system with finite number of objects.

The description of places $P = \{p_{A1}, p_{A2}, p_{A3}, p_x, p_{B1}, p_{B2}, p_{B3}, p_b\}$ for N_2 is the same as that of N_1 presented in table II, plus the buffer place p_b described above. The transitions $T = \{t_1, t_2, t_3, t_4, t_5, t_6\}$ have the same interpretation and the initial marking $M_0 = [1, 0, 0, 0, 1, 0, 0, k]$ where k is a positive integer denoting the number of objects in the path. The weights of the arcs are all one as in N_1.

The liveness and boundedness properties can be verified by inspection of the reachability graph or by analysis of the incidence matrix which is the approach presented for this case. The reachability graph will be large even for small k and it will be impractical to build by hand, but it is easy to visualize that it will contain at least one node with no outgoing arc, e.g., $M_i = [0, 1, 0, 0, 1, 0, 0, 0]$ (where $M(p_b) = 0$ and $M(p_{A2}) = 1$). This indicates that there is a marking $M_i \in R(M_0)$ with no enabled transition. As a result, the net is not live for the initial marking

M_0. A similar reasoning can be used to see that there is no unbounded place in $R(M_0)$. At any execution point of the net, the largest marking is that of place p_b, with $M(p_b) = k$ at the initial marking. As a result, any place of the network will have a marking less than or equal to k, hence the net is bounded.

The structural analysis can be carried out in this case. Let A_2 be the incidence matrix of the model. The order of the places in the matrix is now $P = \{p_{A1}, p_{A2}, p_{A3}, p_x, p_{B1}, p_{B2}, p_{B3}, p_b\}$, and for transitions it is $T = \{t_1, t_2, t_3, t_4, t_5, t_6\}$. The incidence matrix has no T-invariants since $A'y = 0$ has only the trivial solution $y = 0$. The P-invariants are $x_1 = [1, 1, 1, 1, 0, 0, 0, 0]^T$ and $x_2 = [-1, -1, -1, 0, 1, 1, 1, 0]^T$. Therefore, the net is neither covered by T-invariants nor P-invariants.

$$A_2 = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \end{bmatrix} \tag{10}$$

Since the net is not covered by P-invariants we cannot use this method to assess boundedness. Similarly, we cannot assess liveness directly, if it is assumed that the net is bounded, and taking into account that M_0 is finite, it can be concluded that the net is not live due to the fact that it is not covered by T-invariants and the necessary condition is not met.

2) Discussion: This model also shows how the analysis by visual inspection of the reachability graph quickly becomes cumbersome indicating the need for computational tools even for small systems. On the other hand, it also shows that the structural analysis theorems are not always useful and other techniques are required. Another way to perform the analysis will be to reduce the Petri net model into one with a smaller number of places and transitions that preserves the liveness and boundedness properties. The smaller model can then be analyzed by means of the reachability graph.

D. Case 3: Abstract and Petri net models

The abstract model M_1 presented in case 1 (section IV-B) can be modified to allow the system to behave properly when there are a finite number of objects to be moved. In Case 1 the Petri net model assumed an infinite number of parts, as a result the system never deadlocks. On the other hand, Case 2 considered a finite number of parts and the system deadlocked. The abstract model for case 1 did not consider the fact that there could be finite resources. Basically, we can increase the number of environmental states S of the agents A and B, so they will now allow the fact that there might not be any object to pick up at the end of the path.

There should be a new state for each of the agents of the multi-agent system and a set of actions to perform when there are no more objects at the end of the path. The state and action for each of the agents can be defined independently. Lets consider agent A to remain at the end of the line until more objects are available. In order to make use of the previous strategy now agent B must return to the start of the path when there is no part to take over in the intersection with agent A. Note that in the case of no parts left, the intersection will happen at the end of the path.

1) Abstract model for Case 3: Let M_3 be a multi-agent system with agents A and B modeled with the abstract architecture and let $P = S$ ($action : S \mapsto A$). Agent A can undertake a set of actions A_A, its environmental states are S_A, and $action_A$ defines the actions to undertake based on the changes in the environment. In addition, Agent B can undertake actions A_B, its environment is defined by S_B, and the action function $action_B$.

a) Abstract model for agent A:: The following tables describe the upgraded abstract model for agent A. There is one additional action and an additional state. Tables VII and VIII show the states and actions descriptions respectively, $action_A$ is presented in equation 11.

TABLE VII
DESCRIPTION OF STATES FOR AGENT A OF M_3.

State (S_A)	Description
s_1	Agent has object.
s_2	Agent has no object and it is not at the end.
s_{31}	Agent has no object, it is at the end of the path and there are available objects.
s_{32}	Agent has no object, it is at the end of the path and there are no available objects.

TABLE VIII
DESCRIPTION OF ACTIONS FOR AGENT A OF M_3.

Actions (A_A)	Description
a_1	Walk to the end
a_2	Pick an object
a_3	Walk to start
a_4	Wait for more objects

$$action(s) = \left\{ \begin{array}{ll} a_1 & \text{if } s = s_2 \\ a_2 & \text{if } s = s_{31} \\ a_3 & \text{if } s = s_1 \\ a_4 & \text{if } s = s_{32} \end{array} \right\} \tag{11}$$

b) Abstract model for agent B:: Agent B is assumed to take over the object from agent A when they intersect. The additional environment state allows agent B to behave properly when it is at the intersection with agent A and there is no object to take over. Table IX presents the description of the environmental states, table X is the description of the actions the agent can undertake, and equation 12 is the function $action_B$.

$$action(s) = \left\{ \begin{array}{ll} a_1 & \text{if } s = s_1 \text{ or } s = s_{42} \\ a_2 & \text{if } s = s_{41} \\ a_3 & \text{if } s = s_2 \\ a_4 & \text{if } s = s_3 \\ a_5 & \text{if } s = s_5 \end{array} \right\} \tag{12}$$

TABLE IX
DESCRIPTIONS OF STATES FOR AGENT B OF M_3.

State (S_B)	Description
s_1	Agent has no object
s_2	Agent has object and it is not at the start
s_3	Agent has object and it is at the start
s_{41}	Agent is at the intersection and there is a part
s_{42}	Agent is at the intersection and there is no part
s_5	Agent is at the end of the path (no objects available)

TABLE X
DESCRIPTION OF ACTIONS FOR AGENT B OF M_3.

Actions (A_B)	Description
a_1	Walk to the end
a_2	Take over object from agent A
a_3	Walk to start
a_4	Put object down at start
a_5	Wait

2) Petri net model: The multi-agent systems presented in this section will work properly when there are no more objects at the end of the path. The added capabilities make the agents wait at the end of the path until more objects are available.

Let $N_3 = (P, T, A, W, M_0)$ be the Petri net that models the multi-agent system under consideration. Figure 8 shows the Petri net model of the system and table XI presents the description of the places $P = \{p_{A1}, p_{A2}, p_{A3}, p_x, p_{B1}, p_{B2}, p_{B3}, p_b, p_{int}\}$. The transitions are $T = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7\}$ and the initial marking $M_0 = [1, 0, 0, 0,\ 1, 0, 0, k, 0]$. The new capabilities consist only of waiting for more objects to be available in order to be able to move. This capability is already modeled since agent A will remain in place PA_2 (waiting for more objects to be available). In the case of agent B, this will be reflected in place P_{int} but in the original model N_2 (figure 7) place PB_2 could also model this capability.

TABLE XI
PLACES DESCRIPTION FOR N_3.

Place	Description
PA_1	Agent A going to the end of the path to pick object
PA_2	Agent A is at the end of the path loading an object or waiting
PA_3	Agent A is going to start of the path to intersect with agent B
PB_1	Agent B going to the end of the path to intersect with agent A
PB_2	Agent B going to the start of the path to leave object
PB_3	Agent B unloading object at the start of the path
P_x	Both agents are exchanging the object
P_{int}	Agent B is at an intersection at the end of the line waiting for more objects

3) Discussion: The modeling approach used previously models agent activities as places. Having a token in a place means that an agent is executing the activity indicated by the place. The modeling approach is now important because previous results show that there is no uniqueness in the Petri net models obtained. The objective of modeling the multi-agent system as a discrete-event system is to analyze structural properties, and use those results to infer something about the behavior of the multi-agent system and its interaction protocol.

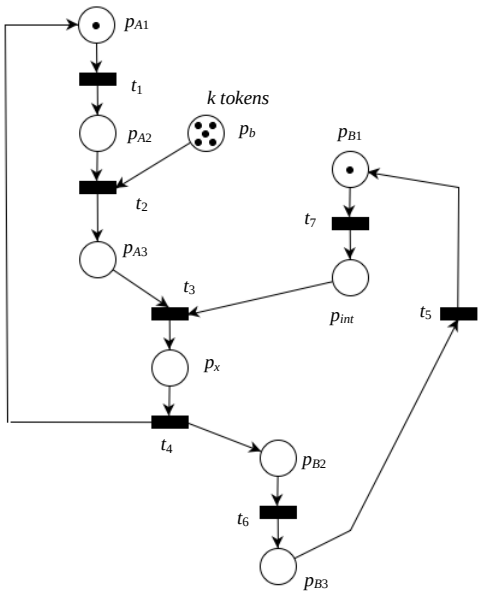

Fig. 8. Petri net model of N_3.

V. MODELING OF MULTI-AGENT SYSTEMS WITH INDIRECT INTERACTION

The methodology presented here consists of defining a simple multi-agent system based on the abstract architecture for intelligent agents (M_4). The abstract architecture is modeled as a discrete-event system using Petri nets (N_4) and structural and reachability analysis provides an assessment of the interaction properties.

The purpose of this work consists of the definition of an abstract architecture for multi-agent systems with indirect interaction, analogous to the abstract architecture for intelligent agents. The proposed architecture allows the description of agent-to-agent interactions via changes in the environment and serves as an initial description of the discrete-event dynamics of the multi-agent system. In addition, this work presents an algorithm (algorithms 5.1 and 5.2) to obtain a Petri net model of a multi-agent system by making use of the multi-agent system's abstract architecture. Finally, a methodology to ensure that the multi-agent system is deadlock free is presented; it is based on the analysis of the properties of the Petri net model.

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]. 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 [8]. By focusing on the interactions among agents as described above, it is natural to regard a multi-agent system as a discrete-event system.

A. 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].

Abstract model for purely reactive agents: In 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 agent environment changes based on the actions applied by the agent, as well as actions by other agents, and it may be dynamic in that it may change by itself.

The environment consists of a set of states $S = \{s_1, s_2, \cdots\}$. The agent can undertake a set of actions $A = \{a_1, a_2, \cdots\}$ and perceive a set of percepts $P = \{p_1, p_2, \cdots\}$. For a purely reactive agent, the behavior of the agent can be represented as the function $action : P \mapsto A$ and $perception : S \mapsto P$. The deterministic behavior of an environment can be represented by the function $environment : S \times A \mapsto S$. A detailed explanation of the abstract architecture was presented in section II-A.

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*, T is a finite set of *transitions*, $A \subseteq (P \times T) \cup (T \times P)$ is a set of *arcs*, $W : A \mapsto \{1, 2, 3, \ldots\}$ is a *weight* function, and $M_0 : P \mapsto Z_+^{|P|}$ is the initial *marking*.

1) Obtaining Petri net models from the abstract architecture: Places model the environmental state of the agent. Having a token in a place representing state s_i means that the agent is currently in such a state. *Transitions* model the actions of an agent. The environmental state is changed by actions so, 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.

Algorithm 5.1: (Petri net sub model for agent i) Let S_i 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 A_i be the set of actions of agent i, and $a_{ik} \in A_i$ be the k^{th} action of agent i.

1) Add a place for each element of the environment S_i and label each place using notation P_{ij} for s_{ij}.
2) Add a transition for each action in A_i and label each transition using notation T_{ik} for a_{ik}.
3) For each instance of the function $environment : S_i \times A_i \mapsto S_i$ say $s_{ij} \times a_{ik} \mapsto s_{il}$: a) add an arc leaving from place P_{ij} and ending in transition T_{ik}; b) add an arc leaving from transition T_{ik} and ending in place P_{il}; c) add a weight of 1 to each arc.
   If an arc from transition T_{ik} to place P_{il} already exists, add a new transition and label it $T_{ik}^{'}$; perform this step using $T_{ik}^{'}$ instead of T_{ik}.
4) Add a token in the place representing the initial state of the environment.

Algorithm 5.2: (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 a way that an agent i action 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 modeling the environmental states of agent j to the transition modeling the communication in agent i.

2) 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 [30].

B. Case 4: Multi-agent system modeling and analysis

This system consists of two agents moving objects from one end of a path to the other. Figure 3 shows 4 scenarios in the operation of the system. The system is defined by the abstract architecture of agents under the following assumptions to ensure that the overall system will be deadlock free: a) there are always available objects at the end of the path; b) agent A moves faster than agent B so it reaches the objects first (figure 3b); and c) agent B takes the object from agent A when they intersect in the path and it returns to leave the object (figures 3c and 3d). There is no direct interaction between the agents; they interact

via changes in their environments. The interchange of the object between the agents should be regarded as a result of a change in the environment of both agents. It should be noted that there is no way to avoid the interaction since it is assumed that the agents meet (see figure 3c) in the path and they cannot avoid each other. The two agents are regarded as purely reactive, which implies they do not have a record of history and they do not have an internal state.

The objective of agent A is to go to the end of the path, pick an object and return to the start of the path (figure 3a). Since it is faster than agent B, at the time they intersect in the path, agent A gives its object to agent B and returns to the end of the path to pick another object. In addition, the objective of agent B is to go in the direction of the end of the path, intersect with agent A and get its object. Once the agent has an object, it returns to the start of the path, places the object down and starts all over again.

The exchange of the object in the intersection of the two agents on the path is only a capability of agent B. As a result, agent A will only react to it but will have no intelligence on that matter. The decisions they make are about which actions to undertake and those decisions are directly influenced by the location of the agent in the path and whether or not the agent has an object.

1) Abstract architecture description: Under normal conditions, agent A will be walking to the end of the path to pick the object to be moved. The object is going to be taken away by the other agent at the intersection. Let M_4 be the multi-agent system with agent A and B modeled with the abstract architecture. Agent A has environmental states S_A, a set of actions A_A, an action function $action_A$, and an environment evolution function $environment_A$. On the other hand, Agent B has environmental states S_B, a set of actions A_B, an action function $action_B$, and an environment evolution function $environment_B$. Tables XII and XIII describe the possible environmental states (S_A) and the actions (A_A) that agent A can undertake.

TABLE XII
ENVIRONMENTAL STATES OF AGENT A OF M_4.

State (S_A)	Description
s_1	Agent has object
s_2	Agent has no object and it is not at the end
s_3	Agent has no object and it is at the end

TABLE XIII
ACTIONS FOR AGENT A OF M_4.

Actions (A_A)	Description
a_1	Walk to the end
a_2	Pick an object
a_3	Walk to start

Table XIV presents the mapping of the environment ($environment_A$) describing how it will be changing as the agent undertakes actions. It should be noted that the notion of exchanging the part with agent B at the intersection has not been considered explicitly in the description of the $environment_A : S \times A \mapsto S$ for agent A. The agents decision mechanism is described by the $action_A$ function as presented in (13).

TABLE XIV
ENVIRONMENT FUNCTION FOR AGENT A OF M_4.

A_A	S_A	S_A	Description
a_1	s_2	s_3	Walking towards the end of the path. Eventually will reach the end with no object.
a_2	s_3	s_1	Now the agent has an object.
a_3	s_1	s_1	Walking to the start of the path. Eventually will lose object to agent B.

$$action_A(s) = \left\{ \begin{array}{ll} a_1 & \text{if } s = s_2 \\ a_2 & \text{if } s = s_3 \\ a_3 & \text{if } s = s_1 \end{array} \right\} \quad (13)$$

Agent B will be walking toward the end of the path until it intersects with agent A which is on its way back to the beginning of the path carrying an object. At the intersection point, agent B takes over the object of agent A and proceeds to return to the beginning of the path to drop the object and start the cycle again. The abstract architecture of agent B is presented in Tables XIV and XV, and the $action_B(s)$ function is described in (14).

TABLE XV
ENVIRONMENTAL STATES FOR AGENT B OF M_4.

State (S_B)	Description
s_1	Agent has no object
s_2	Agent has object and it is not at the start
s_3	Agent has no object and it is at the start
s_4	Agent is at the intersection with agent A

TABLE XVI
ACTIONS FOR AGENT B OF M_4.

Actions (A_B)	Description
a_1	Walk to the end
a_2	Take over object from agent A
a_3	Walk to start
a_4	Put object down at start

$$action_B(s) = \left\{ \begin{array}{ll} a_1 & \text{if } s = s_1 \\ a_2 & \text{if } s = s_4 \\ a_3 & \text{if } s = s_2 \\ a_3 & \text{if } s = s_3 \end{array} \right\} \quad (14)$$

The mapping of the environment of agent B ($environment_B$) is presented in Table XVII. The interaction with agent A (in the exchange of the part) is implicitly modeled by action a_2 although there is no indication in its abstract architecture that it will change the environment of agent A.

TABLE XVII
ENVIRONMENT FUNCTION FOR AGENT B OF M_4.

A_B	S_B	S_B	Description
a_1	s_1	s_4	Will walk toward the end until intersection with agent A
a_2	s_4	s_2	Object exchange, now it has an object
a_3	s_2	s_3	Will walk towards start until it reaches it
a_4	s_3	s_1	Will drop the object at start

2) Petri net model: The Petri net model of the multi-agent system was obtained following the procedure described in algorithm 5.1.

Let $N_4 = (P, T, A, W, M_0)$ be the Petri net model of the complete system with places $P = \{p_{A1}, p_{A2}, p_{A3}, p_{B1}, p_{B2}, p_{B3}, p_{B4}\}$, transitions $T = \{t_{A1}, t_{A2}, t_{A3}, t_{B1}, t_{B2}, t_{B3}, t_{B4}\}$, and $M_0 = [0, 1, 0, 1, 0, 0, 0]$. Figure 9 shows N_4 and the interpretation of places and transitions is presented in Table XVIII. The Petri net sub-models of the individual agents are not presented due to space limitations but they can be observed from the complete model.

Transition t_{B2} which models an action of agent B, modifies the environmental state of agent A. The Petri net model can now be analyzed to assess the deadlock property in a systematic way.

The tokens in places p_{A2} and p_{B1} represent the initial conditions of agent A and B respectively. The tokens travel through the net representing different environment states for the agents as the system executes. The systems execution scenarios presented in figure 3 can be identified in the Petri net model, e.g., scenario d) of figure 3 will be represented in the Petri net model as having a token in p_{A2} and p_{B2}.

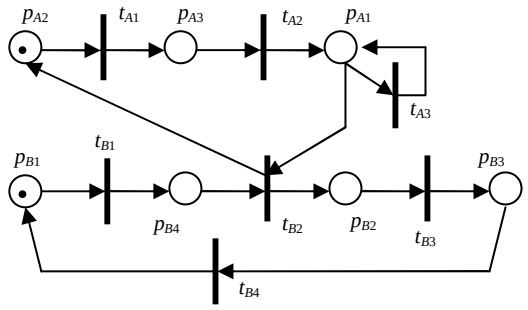

Fig. 9. Petri net model N_4.

3) Analysis of the Petri net model N_4: The Petri net model presented in figure 9 (N_4) can now be analyzed to assess the deadlock property of the underlying multi-agent system. The reachability graph of the model is presented in figure 10 and it shows that the net is live and bounded. Therefore, the multi-agent system is deadlock free. Using the incidence matrix (15) of the net and its invariants (16), it is shown that the net is bounded and that the necessary condition for liveness is satisfied; this is a slightly weaker conclusion on "liveness" than when both necessary and sufficient conditions are met.

TABLE XVIII
DESCRIPTION OF PETRI NET MODEL N_4.

Element	Name	Description
Places (P)	p_{A1}	A token in this place indicates the environment is at state s_1 for agent A
	p_{A2}	A token in this place indicates the environment is at state s_2 for agent A
	p_{A3}	A token in this place indicates the environment is at state s_3 for agent A
	p_{B1}	A token in this place indicates the environment is at state s_1 for agent B
	p_{B2}	A token in this place indicates the environment is at state s_2 for agent B
	p_{B3}	A token in this place indicates the environment is at state s_3 for agent B
	p_{B4}	A token in this place indicates the environment is at state s_4 for agent B
Transitions (T)	t_{A1}	Execution of action a_1 for agent A
	t_{A2}	Execution of action a_2 for agent A
	t_{A3}	Execution of action a_3 for agent A
	t_{B1}	Execution of action a_1 for agent B
	t_{B2}	Execution of action a_2 for agent B
	t_{B3}	Execution of action a_3 for agent B
	t_{B4}	Execution of action a_4 for agent B

$$A_4 = \begin{bmatrix} 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 1 \\ -1 & 1 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 & 0 \end{bmatrix} \quad (15)$$

$$\text{T -invariants} = \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \text{ and P-invariants} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \quad (16)$$

VI. DISCUSSION

An abstract architecture description for multi-agent systems with indirect interaction was presented. In addition, a methodology to obtain Petri net models of such systems was introduced. The deadlock avoidance property of a multi-agent system can be assessed systematically from the Petri net model. A simple example consisting of a multi-agent system with two agents was introduced and the proposed methodology was applied to it in order to show that the system was deadlock free. The results presented in this work show the potential of using Petri nets to assess key properties of multi-agent systems. Future work in this area will focus on the study of direct agent-to-agent interactions as well as Petri net synthesis methodologies to assess systems with a large number of agents.

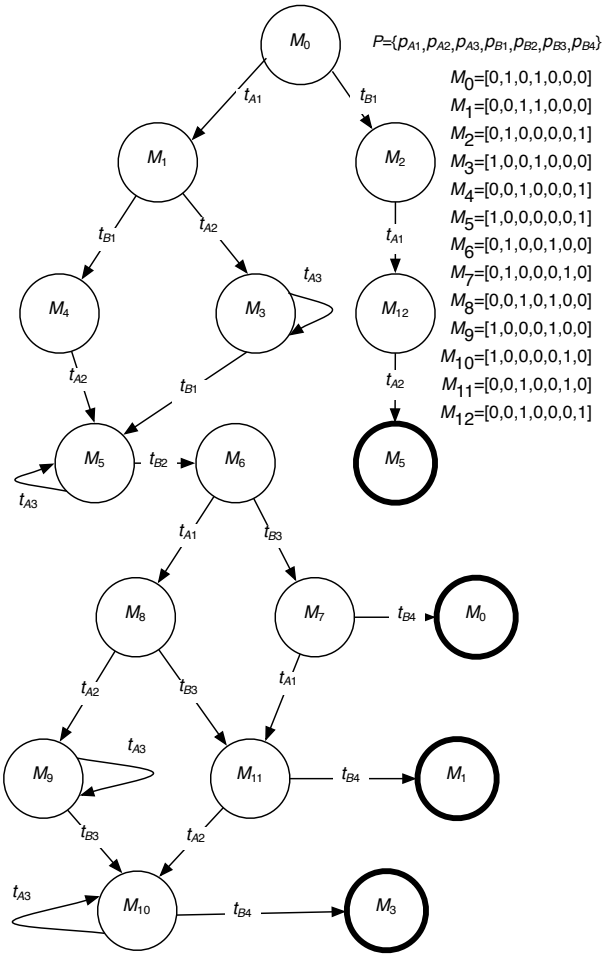

Fig. 10. Reachability graph of Petri net model N_4.

ACKNOWLEDGEMENTS

Portions of this work were supported by NSF grant DMII-0121902.

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.

[2] R. Khosla and T. Dillon, *Engineering intelligent hybrid multi-agent systems*. Kluwer Academic Publishers, 1998.

[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 Transactions on Systems, Man and Cybernetics, Part A*, vol. 32, no. 5, pp. 560–573, 2002.

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

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

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

[11] M. D. Jeng, "A petri net synthesis theory for modeling flexible manufacturing systems," *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, vol. 27, no. 2, pp. 169–183, April 1997.

[12] 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.

[13] M. Zhou, K. McDermott, and P. A. Patel, "Petri net synthesis and analysis of a flexible manufacturing system cell," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, no. 2, pp. 523–531, March 1993.

[14] T. Agerwala, "Putting petri nets to work," *Computer*, vol. 12, no. 2, pp. 85– 94, December 1979.

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

[16] T. Murata, P. C. Nelson, and J. Yim, "A predicate-transition net model for multiple agent planning," *Inf. Sci.*, vol. 57-58, pp. 361–384, 1991.

[17] 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.

[18] 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.

[19] P. Leitão, 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. Marík, 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.

[20] F.-S. Hsieh, "Model and control holonic manufacturing systems based on fusion of contract nets and petri nets," *Automatica*, vol. 40, no. 1, pp. 51–57, 2004.

[21] M. R. Lyu, X. Chen, and T. Y. Wong, "Design and evaluation of a fault-tolerant mobile-agent system," *Intelligent Systems*, vol. 19, no. 5, pp. 32–38, 2004.

[22] A. W. Krings, "Agent survivability: An application for strong andweak chain constrained scheduling," in *HICSS '04: Proceedings of the Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Track 9*. Washington, DC, USA: IEEE Computer Society, 2004, p. 90297.1.

[23] S. Bussmann, N. R. Jennings, and M. Wooldridge, *Multiagent Systems for Manufacturing Control*. Springer Verlag, 2004.

[24] J. M. Vidal. Fundamentals of multiagent systems with netlogo examples. [Online]. Available: http://www.multiagent.com/fmas

[25] S. Kraus, *Strategic negotiation in multiagent environments*. Cambridge, MA, USA: MIT Press, 2001.

[26] C. Tessier, L. Chaudron, and H.-J. Müller, Eds., *Conflicting agents: conflict management in multi-agent systems*. Norwell, MA, USA: Kluwer Academic Publishers, 2001.

[27] C. G. Cassandras, *Discrete Event Systems, Modeling and Performance Analysis*. Aksen Associates Incorporated, 1993.

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

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

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

[31] D. Gross and C. M. Harris, *Fundamentals of Queuing Theory*, ser. Wiley Series in Probability and Statistics. Wiley-Interscience, 1998.

Jose R. Celaya is a visiting scientist with the Research Institute for Advanced Computer Science at the Prognostics Center of Excellence, NASA Ames Research Center. He received a Ph.D. degree in Decision Sciences and Engineering Systems in 2008, a M. E. degree in Operations Research and Statistics in 2008, a M.

S. degree in Electrical Engineering in 2003, all from Rensselaer Polytechnic Institute, Troy New York; and a B.S. in Cybernetics Engineering in 2001 from CETYS University, México.

Alan Desrochers received the B.S.E.E. degree from University of Massachusetts, Lowell, in 1972 and the M.S. and Ph.D. degrees in electrical engineering from Purdue University, West Lafayette, Indiana, in 1973 and 1977, respectively.

During 1974 he was employed in the Guidance and Control Systems Department at the Lockheed Missiles and Space Company, Sunnyvale, California. From 1977-1980, he was an Assistant Professor in the Department of Systems and Computer Engineering at Boston University. In 1980, he joined the Electrical and Systems Engineering Department at Rensselaer Polytechnic Institute, Troy, New York. During the 1986-1987 academic year he was a Visiting Scientist in the Laboratory for Information and Decision Systems at MIT, Cambridge, Massachusetts. He is presently a Professor in the Electrical, Computer, and Systems Engineering Department at Rensselaer.

He is the author of Modeling and Control of Automated Manufacturing Systems, IEEE Computer Society Press,1990, a co-author of Application of Petri Nets in Manufacturing Systems, IEEE Press, 1995 (A. A. Desrochers and R.Y. Al-Jaar), editor of Intelligent Robotic Systems for Space Exploration, Kluwer Academic Publishers, 1992, and co-author of State Variables for Engineers, second edition, John Wiley & Sons, 1998 (P.M. DeRusso, C.M. Close, R. J. Roy, and A.A. Desrochers).

Dr. Desrochers is a past Editor for the IEEE Transactions on Robotics and Automation. In 1987 he was part of a six-faculty member team that received the LEAD award from the Society of Manufacturing Engineers. Since 1995 he has been a Fellow of the IEEE for contributions to manufacturing systems engineering and manufacturing systems education

Dr. Robert J. Graves is with the Thayer School of Engineering , Dartmouth College, Hanover, NH 03755 USA. (Robert.J.Graves@dartmouth.edu). Dr. Graves led Rensselaer's Electronics Agile Manufacturing Research Institute (EAMRI) since its inception in 1994. He has published over 200 books and technical papers on his research findings. He is an Associate Editor of the Journal of Manufacturing Systems and regularly reviews for other prominent journals in the field. He has received the David F. Baker Distinguished Research Award from IIE, the Reed-Apple Award from the Material Handling Education Foundation, and the RPI School of Engineering Research Professor Award in 2003. He is a Fellow of IIE and a Fellow of SME as well as a Senior member of IEEE.