Modular and Hierarchical Modeling of Interactive Mobile Agents*

Roberto Sanchez-Herrera, Ernesto López-Mellado
CINVESTAV-IPN Unidad Guadalajara
Prol. López Mateos Sur 590, 45090 Guadalajara, Jal. Mexico
elopez@gdl.cinvestav.mx

Abstract - This paper deals with the modeling of mobile agent systems. In particular, we focus on the modeling of protocols for agent interaction. First, a multi-level Petri net based formalism, called n-LNS, in which tokens can be symbols or other nets is presented. Then a method for the modeling of this kind of systems using n-LNS is proposed. Finally, a methodology for implementing mobile agents from n-LNS models using the JADE framework is outlined.

Keywords: Mobile agent systems; Multi level modeling; Interaction protocols.

1 Introduction

Nowadays Multi Agent Systems (MAS) is a distributed computing paradigm that is attracting the attention of many researchers in AI applications such as electronic commerce, information retrieval, and distributed manufacturing systems. For this kind of systems formal methods are required for specifying and verifying the functioning requirements in the early stages of the development lifecycle.

Petri nets (PN) and their extensions have been widely used for modeling, validating, and implementing large and complex software systems. In the field of MAS, high level PN have been well adopted for modeling agents or parts of them, because these formalisms allow representing in a clear and compact manner complex behavior including concurrence, synchronization, resource allocation and information exchange.

In [2] Colored PN (CPN) were used for modeling MAS; however, it is difficult to model important elements such as the environment where the agents evolve or the agents mobility. In order to cope with these problems, a CPN extension was proposed for specifying MAS [3]; in this the agent mobility is modeled through the updating of references. This approach brings the specification near to software implementation despite the loss of clearness of the description.

Recently, the approach of "nets within nets" has been held in several works for modeling systems with mobile entities. In [13] a two level elemental object system (EOS) is proposed; the first level model is a PN state machine while tokens may be ordinary PN or integers. Similar definitions have been proposed in [9, 10, 11]. Following this approach, EOS has been extended; a less restrictive definition of a three-level net formalism for the modeling of mobile physical agents has been proposed [4, 12].

In this paper the definition included in [4] is extended; a multilevel PN system, n-LNS, is presented, where an arbitrary number of levels can be defined. Also, we added a more complete interaction mechanism allowing the description of conversations among agents. Through a case study a methodology for implementing mobile agent (MA) systems is shown. The remainder of this paper is organized as follows: in section 2 the definition of the multilevel system is presented; section 3 illustrates the application of the proposed formalism to a case study regarding an electronic marketplace; implementation issues using the JADE framework are briefly discussed.

2 A multi-level net system

An n-LNS model consists mainly of an arbitrary number of nets organized in n levels according to a hierarchy; n depends on the degree of abstraction that is desired in the model. A net may handle as tokens, nets of deeper levels and symbols; the nets of level n permits only symbols as tokens, similarly to CPN. Interactions among nets are declared through symbolic labeling of transitions.

2.1 n-Level net system definition

The definition of n-LNS includes the description of the components (structure and marking), declaring of interaction (transition labeling), and the enabling and firing rules (including the synchronization mechanism).

2.1.1 Petri net structure

Definition. A PN structure is a bipartite digraph denoted by a triple G = (P, T, F) where P and T are finite nonempty set of vertices called places and transitions respectively, P ∪ T = O, and F ⊆ P × T ∪ T × P is a flow relation of the net. Pictorially, places are represented as circles and transitions as bars or rectangles.
2.1.2 Type nets

Definition. A type-net of level i is a tuple type\textit{net}_i = (G, TOKEN_i, LABEL_i, VAR_i, \pi, \lambda, \pi) for 1 \leq i \leq n, where:
- G is a PN structure
- TOKEN_i is a finite non empty set of type-nets and symbols permitted into the places of a net level i:
  \( \text{TOKEN}_i = \{\text{type}\text{net}_i, 1 \leq i \leq n, 1 \leq s \leq r\} \cup \text{SYMBOL} \)
- \( n \) is the number of levels of a multi level net system,
- \( r \) is the number of different type-nets allowed into places of a net of level i.
- SYMBOL_i is a finite set of symbols allowed into the places of a net of level i.
- LABEL_i is a finite set of symbols defined for a net level i.
- VAR_i is a set of variables defined to a net of level i.
- Type\textit{net} is a PN structure with additional information that declares and handles data defined in TOKEN_i, according to the pre and post conditions established by \( \pi \) and \( h \).

2.1.3 Nets of level i

Definition. A net of level i is a type-net type\textit{net}, with a marking \( \mu: \text{NET}_i = (\text{type}\text{net}_i, \mu) \); 1 \leq i \leq n, where:
- type\textit{net} is a type-net of level i.
- \( \mu: \text{P}_i \rightarrow \text{M}_{\text{NET}, \text{TOKEN}_i, \text{SYMBOL}_i} \) is a marking function for the type-net of level i.
- \( \text{NET}_{\text{TOKEN}i} \subseteq \{\text{NET}_{1,i}, \text{NET}_{2,i}, \ldots, \text{NET}_n\} \)

2.1.4 Net system

A n-LNS model, called net system, is the set of all the defined nets at all the levels.

Definition. A n-level net system is a n-tuple NS= (NET_1, NET_2, ..., NET_n) where NET_i is the highest level net, and NET_i = {\text{NET}_{1,i}, \text{NET}_{2,i}, ..., \text{NET}_{ni}} is a set of r nets of level i.

Figure 1 sketches pieces of the components of a 4-LNS. The level 1 is represented by the net NET_1, the level 2 by the nets NET_2 and NET_2, the nets NET_3, NET_3, NET_3, and NET_4 compose the level 3, and the nets NET_4, NET_4, NET_4, NET_4 form the level 4.

2.2 Net system evolution

The components of a model may interact among them through synchronization of transitions. The synchronization mechanism is included in the enabling and firing rules of the transitions; it establishes that two or more transitions labeled with the same symbol must be synchronized. In order to define the enabling conditions and firing of transitions we introduce first the notion of variable binding.

Definition: A binding \( b \) on a variable set \( \text{VARS} = \{x, y, ..., \} \) is a function \( \text{VARS} \rightarrow \text{NETSTOKEN}_i \); for a \( v \in \text{VARS} \), \( b(v) \) is a lower level net whose the type is \( \text{Type}(v) \).

2.2.1 Enabling rule

Definition. A transition \( t \) of a net of level \( i \) \( \text{NET}_i \) is enabled with respect to a label \( \text{lab} \in \lambda(t) \) if:
- There exists a binding \( b_t: \text{VARS} \rightarrow \text{NETSTOKEN}_i \) where \( \text{VARS} \) is the set of variables appearing in all \( \pi(p, t), \text{lab} \).
- It must fulfill that \( \forall p \in \pi, \pi(p, t), \text{lab}b_{\pi} \subseteq \mu(p) \).
- \(<b,>_t \) is not necessary when the level net is \( n \).

The conditions of one of the following cases are fulfilled:
Case 1. If \( \text{lab} = (I, 0) \). The firing of this transition is autonomously performed.

Case 2. If \( \text{lab} \neq (I, 0) \) one must consider one of the following situations:

1. \( \text{lab} = (I, \{\!^\uparrow\!\}) \). It is required the simultaneous enabling of the transitions labeled with \( I \) belonging to other nets into the same place \( p' \) of the next upper level net. The firing of these transitions is simultaneous and all the (locally) synchronized nets remain into \( p' \).

2. \( \text{lab} = (I, \{\!^\uparrow\!\}) \). It is required the enabling of the transitions labeled with \( I \) belonging to lower level nets into \( \bullet t \). These transitions fire simultaneously and the lower level nets and symbols declared by \( \pi((p, t), \text{lab}) \) are removed.

3. \( \text{lab} = (I, \{\!^\uparrow\!\}) \). It is required the enabling of at least one of the \( t^* \) belonging to \( I \), labeled with \( I \), of the upper level net where the NETi is contained. The firing of this transition provokes the transfer of NETi and symbols declared into \( \pi((p^*, t^*), \text{lab}) \).

The rest of the subsets of ATTRIB represents combinations of these clauses. So, \( (I, \{\!^\uparrow\!\}) \) indicates that a transition must be synchronized locally, internally, respectively, with respect to the symbol \( I \).

### 2.2.2 Firing Rule

The firing of transitions in all level nets modifies the marking by removing \( \pi((p, t), \text{lab}) \) in all the input places and adding \( \pi((p, t), \text{lab}) \) to the output places.

In Figure 1, NET, is synchronized through the transition labeled with \( a \) with \( NET_{2,3} \), \( NET_{1,3} \), \( NET_{1,4} \) and \( NET_{4,3} \) by means the transitions (locally synchronized) labeled with \( a^1 \); all these transitions must be enabled to fire. The simultaneous firing of the transitions removes these nets from the input places.

\( NET_{2,3} \), \( NET_{1,3} \) and \( NET_{1,4} \) are synchronized through the transitions labeled with \( b \), \( b^* \), \( b^3 \) respectively; the firing of the transitions changes the marking of \( NET_{2,3} \) and \( NET_{1,3} \) removing \( NET_{1,4} \) from the place of \( NET_{2,3} \), \( NET_{1,3} \) is removed from the input place of \( NET_{2,3} \) and \( NET_{4,3} \) is removed from \( NET_{1,3} \); this interaction is established by \( c^4 \), \( c^4 \), \( c^4 \) respectively.

### 3 Modeling mobile agents

#### 3.1 General strategy

The use of nLNS induces a modular and hierarchical modeling methodology allowing describing separately the behavior of all the involved components and then to integrate such models into a global one through the transition synchronization. We consider that the minimum number of levels in a nLNS model is three: the first level structures the environment where the agents move through, the second level represents the behavior of the agents (mobile or stationary), and the third level describes the functioning of specific items of an agent, such as interaction protocols, plans, intentions, etc; the lower level may specify other entities not included into the agent model such as resources. According to the size of the MAS or depending on the adopted hierarchy, more levels can be defined.

#### 3.2 An agent marketplace

Consider a case study regarding an agent marketplace where agents sell and buy goods on behalf of remote users [1]. The user sends to the marketplace an agent with all the necessary information to buy or to sell a product. There, the agent begins a negotiation process with other agents. When this process finishes, it gets back to its origin to inform the result of the transaction.

Figure 2 shows a possible configuration of the system. The host that acts as marketplace contains seller and buyer MAS, as well as a stationary agent that controls the entrance to the host. Other hosts shown in the figure contain mobile and stationary agents. An arrow between hosts indicates physical connection: an agent can travel from a host to another through the net, or it can communicate with another agent in a different host. We suppose that just one kind of product is traded in this market.
### 3.2.2 Agents and other models

The behavior of **stationary agents** is simple: they receive access requests from mobile agents, decide to grant or deny the permission, and send the answer. A protocol similar to Request Interaction Protocol [7] is used for this purpose. Furthermore, stationary agents take the record of the agents in the host. Figure 4 presents the structure of the stationary agent of the market. Agents of other hosts are similar, but with different labels in some of their transitions.

![Figure 4. Stationary agent model](image)

Tokens marking the places of the stationary agent are nets of level 3: one net representing the **Participant role** in the Request Interaction Protocol (see figure 5) and other net of type **TypeControl**, where the record of the agents in the host is taken.

The behavior of the **mobile agents** is more complex; it is described by a net of level 2 that moves through the environment model. This behavior is guided by the plans of the agents. A plan describes the sequences of tasks to be executed by the agent for the achievement of its goal (in this case, to sell or buy a product).

Figure 6 shows the structure of the mobile agent. Its plans are contained into pl, the tasks are included into p2, and the protocols used by the agent to interact with other agents are contained into p4. When an agent wants to enter into the market, transition t2 is fired, removing the plan and the pertinent task from pl and p2 respectively. The task starts the necessary protocol to request access to the host, sending and receiving messages using t4. Once the protocol finished t3 fires, and when the task finishes, t6 fires, returning the plan and task to their corresponding places. If the access is granted, the agent informs its departure to the stationary agent of its host (t7), it moves to the market (t1), and informs its arrival (t7). Then the agent begins the negotiation process, firing t2 again, selecting the appropriate task, according to its plan (selling or buying); the task in execution selects the appropriate protocol.

![Figure 6. Mobile agent model](image)

The MAs have different plans, depending if they have to buy or sell a product. These plans are modeled with nets of level 3. Figure 7 depicts the plan of a buyer agent. The plan of the seller is similar, but in the case of it does not find a buyer, the agent tries to sell its product again. Roughly speaking, the buyer plan consists of the following actions: request permission to enter to the market, move to the market, try to buy, request permission to return to its original host, move to the host, and inform the result.

![Figure 5. Request Accept/Reject](image)

The tasks an agent can carry out are: buy, sell, and enter to a host. Figures 8 and 9 show the nets of level 3 that model the buying and the entering tasks, respectively. The selling task is similar to the buy task.
The interaction protocols used in this example are Request Interaction Protocol [7], and Contract Net Interaction protocol [6]. The former is used for requesting access to a host, and the latter for the trading of a product. Figure 5 depicts the net for the participant role in the Request Interaction Protocol. The initiator role is played by the MA, and it is described in figure 10. The initiator role in Contract Net is played by the buyer agent, and the participant role by the seller. The buyer sends the call for proposal (CFP), including the specification of the product it wants to buy. The seller can refuse the CFP, or accept it and then send a proposal. The buyer receives the proposals and the refuse messages, and then it evaluates the proposals, selecting the best one. Then, the buyer notifies his decision to the accepted and rejected sellers, and the protocol ends. If no agent sends a proposal, the protocol ends, the buyer returns to its original host, and informs that it failed in buying the good. Figures 11 and 12 show the models of the contract net protocol for the roles of initiator and participant respectively.

An agent model may interact with other agent models and with the nets of level 3 representing resources or other abstract entities such as blackboards.

3.3 Implementation issues

Multi mobile agent based software has been developed from models expressed in nLNS using the JADE, which is a middleware for the development of agent applications in compliance with the FIPA specifications for interoperable intelligent multi-agent systems [5]. Its set of APIs and tools for debugging and deployment allows the development of distributed MAS.
A methodology was defined for mapping nLNS elements into classes and concepts provided by JADE. For the time being it is considered three level models; an outline of this methodology is given below.

Level 1. Every place of the environment model, representing a host, corresponds to a container in JADE. The set of all the containers composes an agent platform.

Level 2. Every agent model is implemented as a class that extends the class Agent of JADE. The initial marking of every agent net corresponds to variable instances that represents plans, tasks, and protocols of the agents.

Level 3. The nets modeling plans, tasks, and protocols are implemented as Behaviours. The plans are implemented using some subclasses of CompositeBehaviours, in which the ChildrenBehaviours control the resuming of the agent tasks as well as the displacement of an agent net to another container. The tasks, according to their complexity, may be implemented as objects of the class SimpleBehaviour or the subclass CompositeBehaviour (when a task performs parallel operations or interacts with other agents through protocols). Protocols are directly implemented using the classes included in JADE; protocols that are not included may be implemented using the class SimpleBehaviour.

The DF service included in JADE is able to perform the function of a blackboard where the agents offer their services to other agents.

The case study presented in this paper was implemented following the described methodology; for every component of the model the corresponding classes were defined. The software was distributed into a set of PC interconnected through a LAN; several tests were performed using different number of agents.

4 Conclusions

The formalism n-LNS induces a modular and hierarchical modeling strategy in which the modules are first conceived separately. The labeling of transitions allows declaring the relationships among modules, and the nesting of nets establishes their hierarchy. Through several case studies regarding mobile agents that perform collaboration or negotiation, both n-LNS and the implementation methodology have demonstrated to be suitable for the development lifecycle of large and complex MAS.

References


• Acknowledgements

This work was partially supported by CONACYT in the scope of the project U41968Y.