

# Design an Interoperable Mobile Agent System Based on Predicate Transition Net Models

Junhua Ding<sup>1</sup>, Dianxiang Xu<sup>2</sup>, Yi Deng<sup>1</sup>,  
Peter J. Clarke<sup>1</sup>, Xudong He<sup>1</sup>

<sup>1</sup> School of Computer Science, Florida International University  
Miami, FL 33199, USA

{jding01, deng, clarkep, hex}@cs.fiu.edu

<sup>2</sup> Computer Science Department, North Dakota State University  
Fargo, ND 58105, USA

dianxiang.xu@ndsu.nodak.edu

## Abstract

*Mobile agents provide an effective and flexible style to develop advanced distributed systems. In order to promote interoperability and ensure the quality of mobile agent systems, it is necessary to formalize software architecture of mobile agent systems. In this paper, we not only define the software architecture of interoperable mobile agent systems using predicate transition nets, but also analyze the interoperability between agents. In addition, a formal model-driven design method to develop mobile agent systems is described with examples. Our method naturally integrates formal methods and practical approaches in the agent system design phase. The method can be used to develop other complex software systems as well.*

## 1. Introduction

Mobile agents have become one of the most active research areas in distributed systems since early 1990s. Mobile agent technique provides a uniform infrastructure so that many distributed applications can be implemented easily, efficiently and robustly [2], [4]. Mobile agent systems are widely different in the architecture and implementation, which is impeding the interoperability and rapid proliferation of mobile agent techniques. Interoperability is of importance for mobile agent systems due to its impact on security (the most concerned issue) and mobility (the unique characteristic) of mobile agent systems. In reality, we cannot expect all mobile agent systems are homogeneous. If agent systems do not support interoperability, agent migration between systems is impossible. Since agents need to interoperate, it is important to provide necessary security mechanisms in the interoperability facility. In order to solve the interoperability issue, a standard for mobile agent systems is desirable. One of these standards is the Mobile Agent System Interoperability Facilities

(MASIF) defined by Object Management Group (OMG) [4]. MASIF specifies basic functions and facilities for constructing mobile agent systems, and common interfaces for interoperability between them. In this paper, we formally model the interoperable software architecture of mobile agent systems based on MASIF using Predicated/Transition (PrT) nets, and discuss the PrT net model-driven method for designing an interoperable mobile agent system.

Formalizing mobile agent systems is helpful to analyze system properties, research related theories, and develop high quality systems. There are several different approaches. The closest works include modeling mobility of mobile agents using reference nets [3] or two-layer PrT nets [7]. These models implement “nets within nets” paradigm [6], where tokens might be other nets wrapped as tokens in system nets. The “nets within nets” paradigm naturally captures the physical architecture of mobile agent systems. Their mobility mechanism is by reference passing in the reference net models. The two-layer PrT net models define agents as agent nets, and agent systems as system nets. Agent nets are wrapped as tokens in system nets, and connectors are introduced to facilitate the communication between different nets. It keeps the basic semantics of PrT nets in each net, so that its models are clear and it does not add more complexity to analyze models [7]. In [8], a multi-agent system is defined using G-nets, and a method to develop agent systems based on G-nets was discussed in [9], but they described intelligent agent systems without mobility, and the interoperability was not discussed. A brief comparison of this work to existing works is listed in Table 1.

	Ref. [3]	Ref. [7]	Ref. [9]	This paper
Interoperability	No	No	No	Yes
Mobility	Yes	Yes	No	Yes
Design method	No	No	Yes	Yes
Modeling language	Reference Nets	Two-layer PrT Nets	G-Nets	Two-layer PrT Nets

**Table 1 A comparison of related works**

In the next section, we first introduce mobile agents and requirements for the interoperability of mobile agent systems, and then model an interoperable architecture of mobile agent systems using PrT nets. Section 3 discusses the analysis of the interoperability of agent systems, and the method for developing mobile agent systems based on the PrT net models. Summary and future work is discussed in section 4.

## 2. Modeling an Interoperable Mobile Agent System

An agent is an object that acts autonomously on behalf of a person or organization. Mobile agents can move themselves from hosts to hosts in networks according to their plans. An agent system is a platform that can create, interpret, execute, transfer and terminate agents [4]. In the following subsections, we describe the basic functions and facilities for an agent system and the requirements for interoperability between agent systems.

### 2.1. Requirements for Interoperability

Interoperability between mobile agent systems requires that mobile agents in different mobile agent systems can recognize, communicate and find each other [4]. The fundamental functions of the interoperability between mobile agent systems include: (1) One agent system can accept and support the running of agents coming from other agent systems. (2) One agent system can support the transferring of agents to other agent systems. (3) One agent can find other agents, agent systems and places.

In order to meet these requirements, the following areas of mobile agent technology need to be formally defined [4]: (1) Agent management, which is used to manage different agent systems via standard operations. (2) Agent transfer, which is used to transfer mobile agents among compatible agent systems. (3) Agent and agent system name. Mobile agents need to cooperate with other agents or systems, and move continuously between different systems. Therefore, agents and agent systems need standard name syntax to communicate and identify each other. (4) Agent system types. Each type of agent systems has different functions and interfaces, so that the compatibility between agent systems can be judged from their types. (5) Location syntax. The location syntax must be standardized so that an agent can access system type information from desired destination agent systems, and so that the source and destination agent systems can recognize each other. (6) Authority. Agents or agent systems can interoperate with each other only

when they pass the mutual authorization. Therefore, standard authorization is required.

In MASIF, communication between agent systems is through the communication infrastructure, which provides communication transport services, naming, and security services for agent systems. MASIF has three parts, which are CORBA Services, MAF and ORB. The CORBA Services provide the common services, and the ORB provides the communication infrastructure. The MAF defines the fundamental functions of a mobile agent system, which includes MAF Finder and MAF Agent System. MAF Finder is used to register/unregister agents or agent systems, and dynamically locate agents or agent systems. MAF Agent System is used to request services and provide services. Requesting services includes requesting data, classes, RMI, creating agents, moving agents out, and proving services includes transferring agents, creating agents, assigning authority, authentication, terminating or resuming agent running, running RMI, transferring data, classes, forwarding data or classes to agents, processing received data or classes.

All fundamental functions and requirements for interoperability between agent systems are defined in finders and agent systems. Finders collect and provide interoperability information for agents and agent systems, and agent systems support the running of visiting agents. Agent management and agent transfer are defined in finders and agent systems, and the agent or agent system name, type, location, and authority are defined in standard messages and supported by middleware such as CORBA services. Therefore, we only define the finder and the agent system models in this paper.

### 2.2. PrT Nets

The PrT net model on mobile agents can be found in [7]. In this paper, we focus on modeling interoperable mobile agent systems based on MASIF. We choose PrT nets to model the architecture of mobile agent systems because they are more suitable for agent modeling due to the similarity to a logic system, and their reachability analysis can be efficiently preformed through the compact structure of planning graphs [7]. To bridge the gap between tokens and agents, a two layer approach is chosen, which is introduced in [7].

**Definition (PrT net):** A PrT net is a tuple  $(P, T, F, \Sigma, L, \varphi, M_0)$ , where: 1.  $P$  is a finite set of predicates (first order places),  $T$  is a finite set of transitions ( $P \cap T = \emptyset, P \cup T \neq \emptyset$ ), and  $F \subseteq (P \times T) \cup (T \times P)$  is a flow relation.  $(P, T, F)$  forms a directed net. 2.  $\Sigma$  is a structure consisting of some sorts of individuals (constants) together with some operations and

relations. 3.  $L$  is a labeling function on arcs. Given an arc  $f \in F$ , the labeling of  $f$ ,  $L(f)$ , is a set of labels, which are tuples of individuals and variables. The zero tuple indicating a no-argument predicate (an ordinary place in Petri nets) is denoted by the special symbol  $\langle \phi \rangle$ . 4.  $\varphi$  is a mapping from a set of inscription formulae to transitions. The inscription on transition  $t \in T$ ,  $\varphi(t)$ , is a logical formula built from variables and the individuals, operations, and relations in structure  $\Sigma$ . 5.  $M_0$  is the initial or current marking. Each token is a tuple of symbolic individuals or structured terms constructed from individuals and operations in  $\Sigma$ .

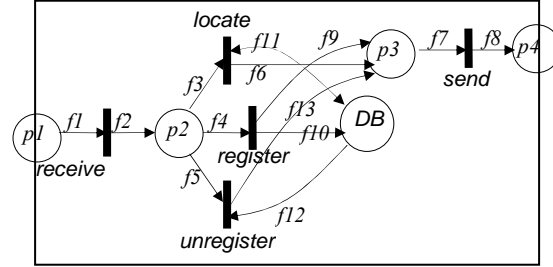
In [7], a two-layer PrT net model consisting of system nets and agent nets and connector nets, to model the behaviors of the environments, mobile agents, and connectors, respectively. Each agent is defined as a PrT net, called agent net. The interface, behavior, and state of an agent are modeled by some input/output predicates for incoming/outgoing messages, the transitions, and the predicates of the agent net, respectively. Each environment is modeled as a PrT net, called system net, and each component includes a system net and an internal connector net. Each system net has internal input/output interfaces that connect to internal connectors, which transfer messages between agents and the system. A group of systems is connected via external connectors, and arcs of connector nets are supposed to be properly labeled so that a migrating agent is always transferred to a proper destination.

### 2.3. Modeling an Agent System using PrT Nets

Each host is made up of an agent system, a finder, and an internal connector. PrT nets are used to model the behaviors of mobile agent systems, mobile agents, finders, as well as connectors. A system net is used to model the mobile agent system; a finder net is used to model the finder, whereas an agent net is used to model the behaviors of a mobile agent, and is taken as a token of the system net. Each agent system communicates with the finder in the host directly, but agents within the host communicate with the finder through the agent system.

Since the agent net, internal connector, external connector are similar to those in [7], we do not duplicate them here. We define the models for systems and finders using PrT nets, which define fundamental functions and requirements for the interoperability between agent systems. The finder receives messages from external connectors or the local agent system. If the message is to locate an agent or agent system, the related information is retrieved from the database and sent back to the requester; if the message is to register

an agent or agent system, then the related information is added to the database and its updated information is forwarded to the destination. If the message is to unregister an agent or an agent system, the related information is removed from the database, and the updated information is forwarded to its destination. The PrT net model for a finder is defined in Fig. 1.



**Fig. 1 A PrT net model for a Finder**

In the following definition for Fig. 1,  $sa$  is the agent, where the token comes from,  $sl$  is the source host of the token,  $da$  is the destination agent, and  $dl$  is the destination host,  $ai$  is agent ID,  $si$  is agent system ID where the agent  $ai$  is in, and  $info$  is the agent system properties.  $cmd$  represents one of commands:  $LOC$ ,  $REG$ ,  $UNREG$ ,  $DAT$ , or  $AGENT$ ,  $param$  is the parameter for  $cmd$ .  $CL$  is current system location,  $\phi$  means empty, operator  $\leftarrow$  is used to assign the right side value to the left parameter, and operator  $\leftrightarrow$  is used to exchange data.

The place definitions:

$$\begin{aligned} \varphi(p1) &= \varphi(p2) = \varphi(p3) = \varphi(p4) = \wp(\langle sa, sl, da, dl, cmd, param \rangle) \\ \varphi(DB) &= \wp(\langle ai, si, info \rangle) \end{aligned}$$

The arc definitions:

$$\begin{aligned} L(f1) &= L(f2) = L(f3) = L(f4) = L(f5) = L(f6) = L(f7) = L(f8) = \\ &= \langle sa, sl, da, dl, cmd, param \rangle \\ L(f9) &= \{ \langle sa, sl, da, dl, cmd, param \rangle \} \\ L(f13) &= \{ \langle sa, sl, param.ai, param.si, REG, param \rangle \} \\ L(f11) &= L(f12) = \{ \langle ai, si, info \rangle \} \\ L(f10) &= \{ \langle param.ai, param.si, param.info \rangle \} \end{aligned}$$

The constraints of transitions:

$$\begin{aligned} R(receive) &= (dl = CL) \\ R(locate) &= ((cmd=LOC) \wedge (ai=param.ai) \parallel (cmd= LOC) \\ &\wedge (si=param.si) \wedge (cmd \leftarrow DAT \wedge param \leftarrow \langle ai, si, info \rangle \wedge (sl \\ &\leftrightarrow dl) \wedge (da \leftrightarrow sa) ) \\ R(register) &= (cmd = REG) \wedge ( (param.ai \neq \phi) \wedge (cmd \\ &\leftarrow AGENT) ) \parallel ( (param.ai = \phi) \wedge (cmd \leftarrow DAT) ) ) \\ R(unregister) &= (cmd=UNREG) \wedge (ai=sa) \wedge (si = sl) \wedge (cmd \\ &\leftarrow REG) \wedge (da \leftarrow param.ai) \wedge (dl \leftarrow param.si) \wedge (param \leftarrow \\ &param.info) \\ R(send) &= (cmd \neq DAT) \parallel ( (cmd = DAT) \wedge (param.info \neq \phi) \parallel \\ &(cmd = DAT) \wedge (param.info = \phi) \wedge (dl \leftarrow param.si) ) \end{aligned}$$

Each agent system receives messages from other systems through external connectors, and the messages are forwarded to the destination through the destination

finder. Each agent communicates with its system through the internal connector, and communicates with the finder through the system. Agent systems can control agent running, create an agent using an agent template, send data to agents. Each agent is staying or running in a particular place in its agent system. When an agent is moved out, it is removed from the place.

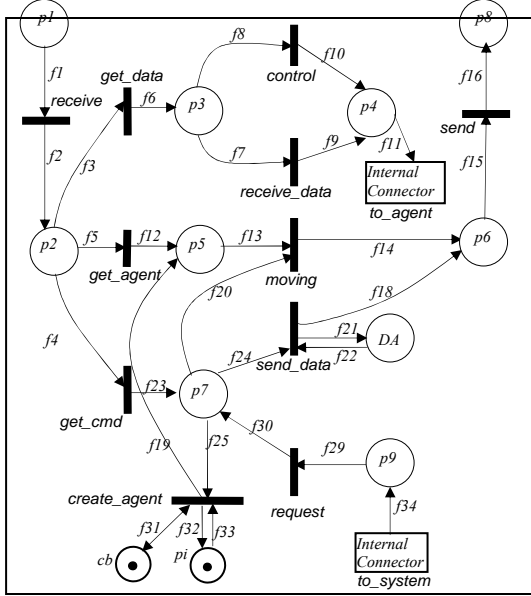


Fig. 2 A PrT net model for an agent system

In the following definition for Fig. 2,  $sa, sl, da, dl, type, param, ai, si, info, UNREG, \phi,$  and  $\leftarrow$  have the same meaning as in Fig.1.  $type$  represents one of commands:  $STOP, RESUME, DAT, REQ\_DAT, AGENT, MOV,$  and  $CREATE$ . If the  $type$  is  $STOP$  or  $RESUME$ , then the  $param$  is  $\phi$ ; if the  $type$  is  $DAT$ , then the  $info$  is the content of data, the  $type$  is  $AGENT$ , then the  $info$  is  $\langle an \rangle$ , and  $an$  is the agent net of  $sa$ ; if the  $type$  is  $REQ\_DAT$ , then the  $info$  is for the description of data or classes; if the  $type$  is  $MOV$ , then the  $param$  is  $\langle ai, si, \langle \phi, sl, info \rangle \rangle$ .

The place definitions:

$$\begin{aligned} \phi(p1) &= \phi(p2) = \phi(p3) = \phi(p4) = \phi(p6) = \phi(p7) = \phi(p8) = \phi(p9) = \varphi(\langle sa, sl, da, dl, type, param \rangle) \\ \phi(p5) &= \varphi(\langle ai, an \rangle) \\ \phi(DA) &= \varphi(DATA) \\ \phi(cb) &= \varphi(AN), AN \text{ is agent template} \\ \phi(pi) &= \varphi(STRING) \end{aligned}$$

The arc definitions:

$$\begin{aligned} L(f1) &= L(f2) = L(f3) = L(f4) = L(f5) = L(f6) = L(f7) = L(f8) \\ &= L(f11) = L(f15) = L(f16) = L(f20) = L(f29) = L(f30) = L(f34) \\ &= L(f23) = L(f24) = L(f25) = \{ \langle sa, sl, da, CL, type, param \rangle \} \\ L(f9) &= \{ \langle sa, sl, da, CL, DAT, \langle \phi, \phi, data \rangle \rangle \} \\ L(f10) &= \{ \langle sa, sl, da, CL, STOP/RESUME, \phi \rangle \} \\ L(f12) &= L(f13) = L(f19) = \{ \langle ai, an \rangle \} \\ L(f14) &= \{ \langle ai, CL, da, CL, UNREG, \langle da, dl, an \rangle \rangle \} \end{aligned}$$

$$\begin{aligned} L(f18) &= \{ \langle \phi, CL, da, dl, DAT, \langle \phi, \phi, data \rangle \rangle \} \\ L(f33) &= \{ ai \}, L(f21) = \{ data \} \\ L(f22) &= \{ data \} \\ L(f31) &= \{ an \}, L(f32) = \{ \phi \} \end{aligned}$$

Constraints of transitions:

$$\begin{aligned} R(receive) &= (dl = CL) \\ R(get\_data) &= (type = STOP) \parallel (type = RESUME) \parallel (type = DAT) \\ R(get\_agent) &= (type = AGENT) \\ R(get\_cmd) &= (type = CREATE) \parallel (type = REQ\_DAT) \parallel (type = RMI) \\ R(control) &= (type = STOP) \parallel (type = RESUME) \\ R(receive\_data) &= (type = DAT) \\ R(moving) &= (type = MOV) \wedge (ai = ai') \wedge (type \leftarrow UNREG) \\ R(send\_data) &= (type = REQ\_DAT) \wedge (param.data \leftarrow data') \parallel (type = LOC) \\ R(request) &= R(send) = true \\ R(create\_agent) &= (type = CREATE) \wedge (ai' = ai) \end{aligned}$$

### 3. Designing an Interoperable Mobile Agent System

The purposes of formalizing the interoperability requirements of mobile agent systems are: (1) providing a solid basis to formally analyze interoperability between agent systems, and (2) providing sound models to develop high quality interoperable mobile agent systems. In the following sections, we first analyze the interoperability between agents, and then we discuss how to design an agent system based on the PrT net models.

#### 3.1. Interoperability Analysis

The interoperability between agents in different systems is implemented through moving agents to the same system, and then these agents interoperate with each other within the same agent system. As long as agent systems meet interoperability requirements, they can support interoperability among agent systems.

Let us consider one scenario of interoperability between agents in different systems. Agent  $X$ , which initially resides in agent system  $A$ , needs to communicate with agent  $Y$ , which resides in agent system  $B$ . The agent  $X$  decides to move itself to the place of agent  $Y$  to take advantage of locality. First, Agent  $X$  sends a message  $\langle X, A, \phi, A, LOC, \langle Y, B, info \rangle \rangle$  through internal connector to system  $A$  to retrieve agent  $Y$  properties, and it is sent to the local finder. If  $Y$  is already registered in the local finder, related information is retrieved from the  $DB$  and the message  $\langle X, A, X, A, DAT, \langle Y, B, info \rangle \rangle$  is sent back to system  $A$ , which sends  $info$  to the requested agent  $X$ . If system  $Y$  is not registered in the local Finder, reply information from the  $DB$  should be  $\langle Y, B, \phi \rangle$ , then the request message  $\langle X, A, \phi, B, LOC, \langle Y, B, info \rangle \rangle$  is sent to the

finder in system  $B$ , which sends a reply message  $\langle \phi, B, X, A, REG, \langle Y, B, info \rangle \rangle$  back to the finder in system  $A$ , and system  $Y$  is registered into  $DB$ , and message  $\langle \phi, B, X, A, DAT, \langle Y, B, info \rangle \rangle$  is sent back to agent  $X$ . As soon as system  $A$  receive a positive reply in the  $info$ , system  $A$  sends a moving request message  $\langle X, A, \phi, B, MOV, param \rangle$  to system  $A$  to move out agent  $X$ , a message  $\langle X, A, \phi, A, UNREG, \langle \phi, B, info \rangle \rangle$  is sent to the finder to unregister  $X$  from current system  $A$ , and  $\langle X, A, \phi, B, REG, \langle \phi, A, info \rangle \rangle$  is sent to the finder in system  $B$ . Then agent  $A$  is registered in the finder of system  $B$ , and message  $\langle X, A, \phi, B, AGENT, \langle \phi, A, info \rangle \rangle$  is forwarded to system  $B$ . Agent  $X$  will run within the place  $p_5$ , where agent  $X$  communicate with agent  $Y$  that is also in  $p_5$  through the internal connector. We use  $M(p)$  to represent the place  $p$  has tokens add and it causes following transitions firing,  $M_s$  for system  $A$ , and  $M_f$  for finder  $A$ ,  $M'_s$  for system  $B$ , and  $M'_f$  for finder  $B$ , and ... represents firing actions in connectors. One firing sequence  $Seq$  ( $Y$  is not registered in  $A$ ) in the finder net and the system net for above steps is:

$M_s(p_9)[request > M_s(p_7)[send\_data > M_s(p_6)[send \dots$   
 $M_f(p_1)[receive > M_f(p_2)[locate > M_f(p_3)[send \dots$   
 $M'_s(p_1)[receive > M'_s(p_2)[locate > M'_s(p_3)[send \dots$   
 $M_f(p_1)[receive > M_f(p_2)[register > M_f(p_3)[send \dots$   
 $M_s(p_1)[receive > M_s(p_2)[get\_data > M_s(p_3)[receive\_data \dots$   
 $M_s(p_9)[request > M_s(p_7), M_s(p_5)[moving > M_s(p_6)[send \dots$   
 $M_f(p_1)[receive > M_f(p_2)[unregister > M_f(p_3)[send \dots$   
 $M'_s(p_1)[receive > M'_s(p_2)[register > M'_s(p_3)[send \dots$   
 $M'_s(p_1)[receive > M'_s(p_2)[get\_agent > M'_s(p_3)$

### 3.2. Designing a Mobile Agent System

The PrT net models for mobile agent systems can be served as high-level designs for mobile agent systems. Here we discuss a formal model-driven method for designing an interoperable mobile agent system. The general rules for deriving design models based on high level PrT net models are: (1) The PrT net models specify the software architecture and essential properties of a system. Therefore, each PrT net might be implemented with many classes in the design, but attributes and methods in these classes are traceable or related to predicates, transitions, and arcs in the PrT net models. (2) Predicates in PrT nets are translated into attributes in classes, and transitions are translated into functions; arcs are reflected in the relationship between classes and input/output of related functions, and labels are translated into parameters of related functions. (3) Each firing sequence in PrT net models can be designed as a sequence diagram in the design models. Each sequence diagram reflects a firing sequence in the PrT net models. (4) Object diagrams, state diagrams and other diagrams in design also have obvious relationship with PrT net models. Even it is

difficult to generalize a formal transformation algorithm for translating PrT nets into design models, defining formal models of a system in the early phase provides a sound basis to accurately understand the system, and the design procedure directed with formal models is great helpful to construct high quality design models. Since design models have close relationship with high level PrT net models, the test adequacy criteria and test sets generated from PrT net models might be reused to test corresponding designs and implementations. The PrT net models also bridge the gap between the informal requirements and system designs. Each development phase (from the requirement analysis, system design, implementation, and testing) is driven by the formal PrT net models. This approach naturally integrates formal methods to system development. In order to illustrate our approach, we define a class diagram and a sequence diagram based on PrT net models for a mobile agent system.

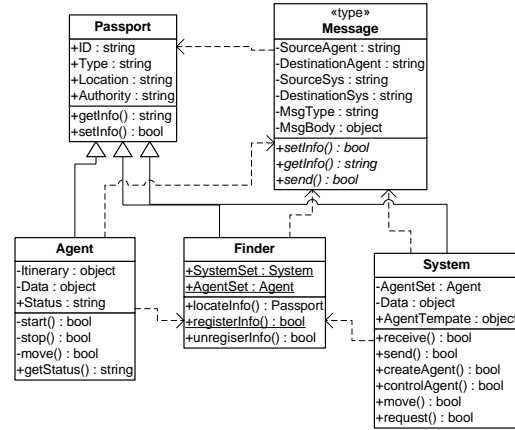
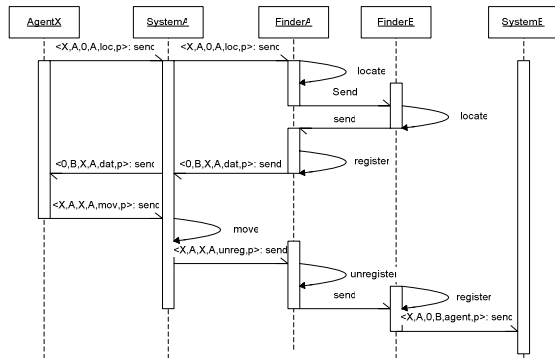


Fig. 3 A class diagram of an agent system

Based on the PrT net models, we understand that a mobile agent system consists of a finder, a system and other CORBA facilities if we develop the system based on CORBA. Agents are the most important attribute of a mobile agent system; therefore, we add an agent class into our class diagram. The interoperability between agents is based on the agent type, the source and destination agent system type, and the authority, so that we define a passport class for these interoperability properties. The communication between agent systems is through message passing so that we add a message class in the design. Fig. 3 is a simplified class diagram.

Communication messages have the format from the PrT net models. Passport information that is used to determine the interoperability is wrapped into messages. The functions, attributes and relations in the class diagram are derived from the PrT net models

directly, and the class diagram can be refined according to other requirements and the development environments.



**Fig. 4 A sequence diagram of an agent system**

Sequence diagrams have obvious relationship with firing sequences in PrT nets. As soon as we defined a class diagram, firing sequences in PrT net models can help us to define the sequence diagram. Transitions in the firing sequence are translated into active methods in the sequence diagram, and the transition firing order is reflected in the activity sequence in the sequence diagram. Firing sequence in PrT net models would be helpful to analyze sequence diagram so that to improve design quality. The simplified sequence diagram for firing sequence *Seq* in section 3.1 is showed in Fig. 4.

#### 4. Summary and Future Work

In this paper, we model the software architecture of interoperable mobile agent systems using two-layer PrT nets. Based on the PrT net models, we analyze the interoperability of mobile agents, and discussed the formal model-driven method to develop high quality interoperable mobile agent systems with examples. Through modeling the high-level models of interoperable mobile agent systems using PrT nets, and designing an interoperable mobile agent system, we naturally introduce formal methods into software design process. Defining PrT net models in the early phase of software development provides us a sound basis for developing other models. The PrT net models help us analyze important system properties as early as possible so as to reduce development cost and improve product quality. In addition, through modeling and design of interoperable mobile agent systems, we found that PrT nets are a nice tool for defining high level system models because of their easy-to-understand graphic representation, high level modeling capability, and efficient formal analysis algorithms. Our future work includes: (1) developing a mobile

agent system development kit based on PrT net models, (2) developing a clinical information system using mobile agents, and (3) developing methods for testing PrT nets and system designs.

#### Acknowledgements

This research was supported in part by the National Science Foundation of the USA under grant HRD-0317692, and by the National Aeronautics and Space Administration of the USA under grant NAG2-1440.

#### References

- [1] H.J. Genrich: Predicate/Transition Nets. *Petri Nets: Central Models and Their Properties*, W. Brauer, W. Resig and G. Rozenberg, eds., pp. 207-242, 1987
- [2] R.S. Gray, G. Cybenko, D. Kotz, and D. Rus: Mobile Agents: Motivations and States of the Art. *Handbook of Agent Technology*, J. Bradshaw, ed., 2001
- [3] M. Kohler, D. Moldt, and H. Rolke, "Modeling mobility and mobile agents using nets within nets." In W.M.P. van der Aslst and E. Best, eds, *Proc. of Int. Conf. on Applications and Theory of Petri Nets, LNCS*, vol. 2769, pp. 121-139, June 2003
- [4] OMG, MASIF—Mobile Agent System Interoperability Facility, *Technical Report*, OMG, 1998
- [5] T. Murata: Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, vol.77, no.4, pp. 541-580, 1989
- [6] R.G. Valk: Petri Nets as Token Objects: An Introduction to Elementary Object Nets. *Application and Theory of Petri Nets*, J. Desel and M. Silva, eds., LNCS 1420, pp. 1-25, 1998
- [7] D. Xu, J. Yin, Y. Deng and J. Ding: A Formal Architecture Model for Logical Agent Mobility. *IEEE Trans. on Software Engineering*. vol. 29, No. 1, pp. 31-45, Jan. 2003
- [8] H. Xu, S.M. Shatz: A Framework for Model-Based Design of Agent-Oriented Software. *IEEE Trans. on Software Engineering*. vol. 29, No. 1, pp. 15-30, Jan. 2003
- [9] H. Xu, S.M. Shatz: ADK: An Agent Development Kit Based on a Formal Model for Multi-Agent Systems. *Automated Software Engineering*, vol. 10, No. 4, pp. 337-365, Oct. 2003