13

A Novel Modeling Method for Cooperative Multi-robot Systems Using Fuzzy Timed Agent Based Petri Nets

Hua Xu
State Key Laboratory of Intelligent Technology and Systems, Tsinghua National Laboratory for Information Science and Technology, Department of Computer Science and Technology, Tsinghua University, Beijing P. R. China

1. Introduction

Characterized as cooperation and high efficiency, cooperative multi-robot systems (CMRS) have emerged as usual manufacturing equipments in current industries (Cao et al., 1997). Differing from generic control systems, the cooperation needs to be considered in the realization of CMRS (Cao et al., 1997). So the system modeling, analysis and refinement always meet with difficulties. As one of the typical multi-agent systems (MAS), CMRS can be regarded as one MAS in distributed artificial intelligence (Jennings et al., 1998). For modeling MAS, some attempts to use object-oriented methodology have been tried and some typical agent objects have been proposed, such as *active object*, etc (Guessoum & Briot, 1999). However, agent based object models still can not depict the structure and dynamic aspects of MAS, such as cooperation, learning, temporal constraints, etc (Jennings et al., 1998).

On one hand, as one of the most proper and promised realization technology, object-oriented concurrent programming (OOCP) methodology is always used to realize MAS (Guessoum & Briot, 1999). In OOCP realization, one special object called *active object* is proposed (Guessoum & Briot, 1999), which can be used to model generic agent architectures and behaviors with OO methodology. Although OOCP can solve MAS realization problem favorably, the modeling problem mentioned above still exists.

On the other hand, as a kind of powerful formal description and analysis method for dynamic systems (Murata, 1989), Petri net (PN) has become a bridge between practical application and model theories (Murata, 1989). With object-oriented concepts introduced into PN, typical object Petri nets such as HOONet (Hong & Bae, 2000) etc, al. are presented, which are strict OPNs. Then for providing the ability of modeling time critical complex systems, timed hierarchical object oriented Petri net (TOPN) (Xu & Jia, 2006) is proposed on the base of HOONet (Hong & Bae, 2000). It supports temporal knowledge description and object-oriented concepts. Modeling features in TOPN support describing and analyzing dynamic systems such as MAS and CMRS. Recently, some attempts have been conducted on modeling MAS by using OPN (Chainbi, 2004). Although Petri nets can be used to model and analyze different systems, they have failed to model learning ability and the aging effects in

Source: Recent Advances in Multi-Robot Systems, Book edited by: Aleksandar Lazinica, ISBN 978-3-902613-24-0, pp. 326, May 2008, I-Tech Education and Publishing, Vienna, Austria

dynamic systems. Recently, fuzzy timed Petri net (FTPN) (Pedrycz & Camargo, 2003) has been presented to solve these modeling problems. As a kind of reasoning and learning ability, fuzzy reasoning in FTPN can be considered as supporting autonomous judging or reasoning ability in MAS. In order to solve the reasoning ability and other modeling problems in large-scale MAS, fuzzy timed object-oriented Petri net (FTOPN) (Xu & Jia, 2005) is proposed on the base of TOPN (Xu & Jia, 2006) and FTPN (Pedrycz & Camargo, 2003). In FTOPN, agent can be modeled as one FTOPN object with autonomy, situatedness and sociality. However, in FTOPN every agent should be modeled from common FTOPN objects. To model CMRS—"one of the typical MAS" needs generic FTOPN agent objects on the base of *active object*s (Guessoum & Briot, 1999).

This chapter proposes a high level PN called fuzzy timed agent based Petri net (FTAPN) on the base of FTOPN (Xu & Jia, 2005). As one of the typical *active objects*, ACTALK object is modeled by FTOPN and is introduced into FTAPN, which is used as one generic agent object in FTAPN. The aim of FTAPN is to solve the agent or CMRS modeling ability problem and construct a bridge between MAS models and their implementations.

This chapter is organized as the following. Section 2 reviews the relative preliminary notations of active objects, TOPN and FTOPN quickly. Section 3 extends FTOPN to FTAPN on the base of ACTALK model. Section 4 discusses the learning which is important for representing dynamic behaviors in CMRS. Section 5 uses FTAPN to model one CMRS in the wafer etching procedure of circuit industry and makes some modeling analysis to demonstrate its benefits in modeling MAS. Finally, the conclusion and future work can be found in section 6.

2. Preliminary Notations

In this section, the basic concepts of ACTALK are firstly reviewed. Then the definitions of TOPN are introduced quickly. Finally, the concepts of FTOPN are reviewed formally.

2.1 ACTALK

2.1.1 Active Objects (Guessoum & Briot, 1999)

Object-oriented concurrent programming (OOCP) is one of the most appropriate and promising technologies for implementing or realizing agent based systems or MAS. Combining the agent concept and the object-oriented paradigm leads to the notion of agent-oriented programming (Shoham, 1993). The uniformity of objects' communication mechanisms provides facilities for implementing agent communication, and the concept of encapsulating objects or encapsulation support combining various agent granularities. Furthermore, the inheritance mechanism enables knowledge specialization and factorization.

The concept of an active object has been presented, which makes it possible to integrate an object and activity (namely a thread or process). It also provides some degree of autonomy for objects in that it does not rely on external resources for activation. Thus, it provides a good basis for implementing agent based systems or MAS. However, similar to common objects in Object-oriented systems, an active object's behavior still remains procedural and only reacts to message requests. More generally, the main feature of agent-based systems or MAS is autonomous. Agents should be able to complete tasks autonomously. That's to say, agents must be able to perform numerous functions or

activities without external intervention over extended time periods. In order to achieve autonomy, adding to an active object a function that controls message reception and processing by considering its internal state is one of effective realization methods (Briot, 1996) (Maruichi et al., 1990).

On one hand, for modelling and realizing MAS, there are two basic questions regarding how to build a bridge between implementing and modelling MAS requirements (Castelfranchi, 1995) (Gasser, 1992). On the other hand, the facilities and techniques OOCP provides (Gasser & Briot, 1992):

- How can a generic structure define an autonomous agent's main features?
- How do we accommodate the highly structured OOCP model in this generic structure?

The active-object (or actor) concept has been introduced to describe a set of entities that cooperate and communicate through message passing. This concept brings the benefits of object orientation (for example, modularity and encapsulation) to distributed environments and provides object-oriented languages with some of the characteristics of open systems (Agha & Hewitt, 1985). Based on these characteristics, various active object models have been proposed (Yonezawa & Tokoro, 1987), and to facilitate implementing active-object systems, several frameworks have been proposed. Actalk is one example.

When Actalk is used to model and realize the MAS, there still exist the following shortcomings:

- Active object is not an autonomous agent. It only manifests the procedural actions.
- Although active object can communicate, they do not own the ability to reduce the decision to communicate or order other active objects.
- If one active object has not received the information from other active objects. It is still in none-active state.

In order to overcome the shortcomings mentioned above, the concept of active object has been proposed and a general agent framework (Shoham, 1993) (Maruichi et al., 1990). An universal agent architecture has also been proposed so as to fulfil the modelling requirements of MAS (Gasser & Briot, 1992), which can be used to model and analyze MAS deeply. For either agent-based systems or MAS, the method mostly is on the base of active objects. So in this chapter, the concept of active object is firstly reviewed quickly.

2.1.2 ACTALK (Guessoum & Briot, 1999)

One of the typical active objects is Actalk. Actalk is a framework for implementing and computing various active-object models into a single programming environment based on Smalltalk, which is an object-oriented programming language. Actalk implements asynchronism, a basic principle of active-object languages, by queuing the received messages into a mailbox, thus dissociating message reception from interpretation. In Actalk, an active object is composed of three component classes (see Fig. 1), which are instances of the classes.

- Address encapsulates the active object's mailbox. It defines how to receive and queue messages for later interpretation.
- Activity represents the active object's internal activity and provides autonomy to the actor. It has a Smalltalk process and continuously removes messages from the mailbox, and the behavior component interprets the messages.
- ActiveObject represents the active object's behavior—that is, how individual messages are interpreted.

To build an active object with Actalk, we must describe its behavior as a standard Smalltalk (OOCP) object. The active object using that behavior is created by sending the message active to the behavior:

active
"Creates an active object with self as corresponding behavior"
^self activity: self activityClass address: self addressClass

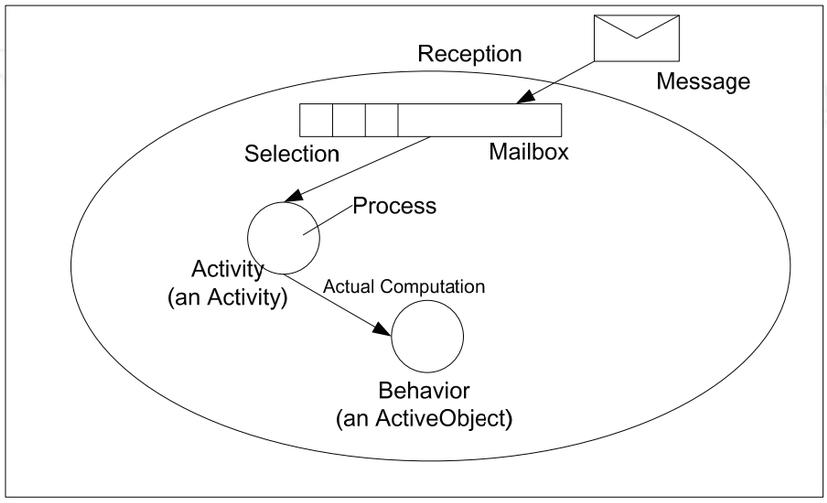


Figure 1. Components of an Actalk active object

The activityClass and addressClass methods represent the default component classes for creating the activity and address components (along the *factory method* design pattern). To configure the framework of Actalk means to define the components of its sub-classes. That's to say, it lets users to define special active object models. So Actalk is the basis to model agent based systems or MAS.

2.2 Timed Object-oriented Petri net (TOPN)

Formally TOPN is a four-tuple (OIP, ION, DD, SI), where (OIP,ION,DD) is an ordinary object Petri net-"HOONet" (Hong & Bae, 2000) and SI associates a static (firing) temporal interval SI: $\{o\}\rightarrow[a, b]$ with each object o, where a and b are rationals in the range $0\leq a\leq b\leq+\infty$, with $b\neq+\infty$. The four parts in TOPN have different function roles. Object identification place (OIP) is a unique identifier of a class. Internal timed object net (ION) is a net to depict the behaviors (methods) of a class. Data dictionary (DD) declares the attributes of a class in TOPN. And static time interval function (SI) binds the temporal knowledge of a class in TOPN. There are two kinds of places in TOPN. They are common places (represented as circles with thin prim) and abstract places (represented as circles with bold prim). Abstract places are also associated with a static time interval. Because at this situation, abstract places represent not only firing conditions, but also the objects with their own behaviors. So, abstract places (TABP) in TOPN also need to be associated with time intervals. One problem to be emphasized is that the tokens in abstract places need to have two colors at least. Before the internal behaviors of an abstract place object are fired, the color of tokens in it is one color (represented as hollow token in this paper). However, after fired, the color becomes the other one (represented as liquid token in this paper). At this time, for the following transitions, it is just actually enabled. There are three kinds of transitions in TOPN. The

timed primitive transition (represented as rectangles with thin prim) (TPIT), timed abstract transition (represented as rectangles with bold prim) (TABT) and timed communication transition (represented as rectangles with double thin prim) (TCOT).

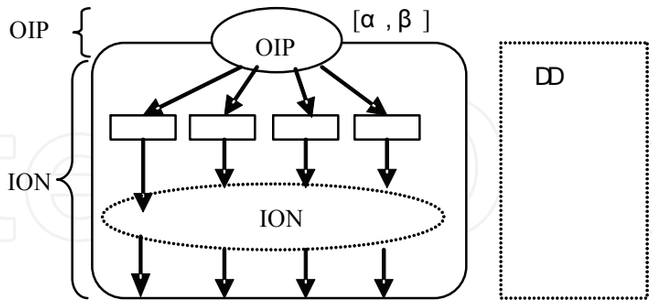


Figure 2. The General Structure of TOPN

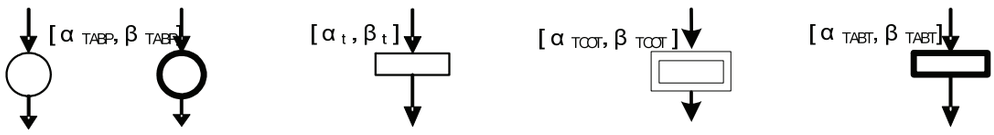


Figure 3. Places and Transitions in TOPN

Static time intervals change the behavior of TOPN just similar to a time Petri net in the following way. If an object o with SI(o)=[a, b] becomes enabled at time I_0, then the object o must be fired in the time interval [I_0+a, I_0+b], unless it becomes disabled by the removal of tokens from some input place in the meantime. The static earliest firing time of the object o is a; the static latest firing time of o is b; the dynamic earliest firing time (EFT) of t is I_0+a; the dynamic latest firing time (LFT) of t is I_0+b; the dynamic firing interval of t is [I_0+a, I_0+b].

The state of TOPN (Extended States-"ES") is a 3-tuple, where ES=(M, I, path) consists of a marking M, a firing interval vector I and an execution path. According to the initial marking M_0 and the firing rules mentioned above, the following marking at any time can be calculated. The vector--"I" is composed of the temporal intervals of enabled transitions and TABPs, which are to be fired in the following states. The dimension of I equals to the number of enabled transitions and TABPs at the current state. The firing interval of every enabled transition or TABP can be got according to the calculation formula of EFT and LFT in TOPN (Xu & Jia, 2006).

For enabling rules in TOPN, two different situations exist. A transition t in TOPN is said to be enabled at the current state (M, I, path), if each input place p of t contains at least the number of solid tokens equal to the weight of the directed arcs connecting p to t in the marking M. If the TABP object is marked with a hollow token, it is enabled. At this time, its ION is enabled. After the ION has been fired, the tokens in TABP are changed into solid ones.

An object o is said to be fireable in state (M, I, path) if it is enabled, and if it is legal to fire o next. This will be true if and only if the EFT of o is less than or equal to the LFT of all other enabled transitions. Of course, even with strong time semantics, o's being fireable in state (M, I, path) does not necessarily mean that t will fire in the time interval I.

2.3 Fuzzy Timed Object-oriented Petri net (FTOPN)

Similar to FTPN (Xu & Jia, 2005-1), fuzzy set concepts are introduced into TOPN (Xu & Jia, 2006). Then FTOPN is proposed, which can describe fuzzy timing effect in dynamic systems.

Definition 1: FTOPN is a six-tuple, FTOPN= (OIP, ION, DD, SI, R, I) where

1. Suppose OIP=(oip, pid, M_0, status), where oip, pid, M_0 and status are the same as those in HOONet (Hong & Bae, 2000) and TOPN (Xu & Jia, 2006).
   - oip is a variable for the unique name of a FTOPN.
   - pid is a unique process identifier to distinguish multiple instances of a class, which contains return address.
   - M_0 is the function that gives initial token distributions of this specific value to OIP.
   - status is a flag variable to specify the state of OIP.
2. ION is the internal net structure of FTOPN to be defined in the following. It is a variant CPN that describes the changes in the values of attributes and the behaviors of methods in FTOPN.
3. DD formally defines the variables, token types and functions (methods) just like those in HOONet (Hong & Bae, 2000) and TOPN (Xu & Jia, 2006).
4. SI is a static time interval binding function, SI: {OIP}→Q*, where Q* is a set of time intervals.
5. R: {OIP} → r, where r is a specific threshold.
6. I is a function of the time v. It evaluates the resulting degree of the abstract object firing. □

Definition 2: An internal object net structure of TOPN, ION = (P,T,A,K,N,G,E,F,M_0)

1. P and T are finite sets of places and transitions with time restricting conditions attached respectively.
2. A is a finite set of arcs such that P∩T=P∩A=T∩A=Φ.
3. K is a function mapping from P to a set of token types declared in DD.
4. N, G, and E mean the functions of nodes, guards, and arc expressions, respectively. The results of these functions are the additional condition to restrict the firing of transitions. So they are also called additional restricting conditions.
5. F is a special arc from any transitions to OIP, and notated as a body frame of ION.
6. M_0 is a function giving an initial marking to any place the same as those in HOONet (Hong & Bae, 2000) and TOPN (Xu & Jia, 2006). □

Definition 3: A set of places in TOPN is defined as P=PIP□TABP, where

1. Primary place PIP is a three-tuple: PIP =(P,R,I), where
   - P is the set of common places similar to those in PN (Murata, 1989) (Peterson, 1991).
2. Timed abstract place (TABP) is a six-tuple: TABP= TABP(pn, refine state, action, SI, R, I), where
   - pn is the identifier of the abstract timed place.
   - refine state is a flag variable denoting whether this abstract place has been refined or not.
   - action is the static reaction imitating the internal behavior of this abstract place.
   - SI, R and I are the same as those in Definition 1.

Definition 4: A set of transitions in TOPN can be defined as T= TPIT□TABT□TCOT, where

1. Timed primitive transition TPIT = TPIT (BAT, SI), where
   - BAT is the set of common transitions.

2. Timed abstract transition TABT= TABT (tn, refine state, action, SI), where
   - tn is the name of this TABT.
3. Timed communication transition TCOT=TCOT (tn, target, comm type, action, SI).
   - tn is the name of TCOT.
   - target is a flag variable denoting whether the behavior of this TCOT has been modeled or not. If target="Yes", it has been modeled. Otherwise, if target="No", it has not been modeled yet.
   - comm type is a flag variable denoting the communication type. If comm type ="SYNC", then the communication transition is synchronous one. Otherwise, if comm type="ASYN", it is an asynchronous communication transition.
4. SI is the same as that in Definition 1.
5. refine state and action are the same as those in Definition 3.

Similar to those in FTPN (Xu & Jia, 2005-1), the object t fires if the foregoing objects come with a nonzero marking of the tokens; the level of firing is inherently continuous. The level of firing (z(v)) assuming values in the unit interval is governed by the following expression:

$$z(v) = (\overset{n}{\underset{i=1}{T}}(r_i \rightarrow x_i(v'))\, sw_i)\, t\mathrm{I}(v) \tag{1}$$

where T (or t) denotes a t-norm while "s" stands for any s-norm. "v" is the time instant immediately following v'. More specifically, x_i(v) denotes a level of marking of the i^{th} place. The weight w_i is used to quantify an input coming from the i^{th} place. The threshold r_i expresses an extent to which the corresponding place's marking contributes to the firing of the transition. The implication operator (→) expresses a requirement that a transition fires if the level of tokens exceeds a specific threshold (quantified here by r_i).

Once the transition has been fired, the input places involved in this firing modify their markings that is governed by the expression

$$x_i(v)=x_i(v')t(1-z(v)) \tag{2}$$

(Note that the reduction in the level of marking depends upon the intensity of the firing of the corresponding transition, z(v).) Owing to the t-norm being used in the above expression, the marking of the input place gets lowered. The output place increases its level of tokens following the expression:

$$y(v)=y(v')sz(v) \tag{3}$$

The s-norm is used to aggregate the level of firing of the transition with the actual level of tokens at this output place. This way of aggregation makes the marking of the output place increase.

The FTOPN model directly generalizes the Boolean case of TOPN and OPN. In other words, if x_i(v) and w_i assume values in {0, 1} then the rules governing the behavior of the net are the same as those encountered in TOPN.

3. Agent Objects and Fuzzy Timed Agent Based Petri Nets

The *active object* concept (Guessoum & Briot, 1999) has been proposed to describe a set of entities that cooperate and communicate through message passing. To facilitate implementing *active object* systems, several frameworks have been proposed. ACTALK is

one of the typical examples. ACTALK is a framework for implementing and computing various *active object* models into one object-oriented language realization. ACTALK implements asynchronism, a basic principle of *active object* languages, by queuing the received messages into a mailbox, thus dissociating message reception from interpretation. In ACTALK, an *active object* is composed of three component classes: *address*, *activity* and *activeObject* (Guessoum & Briot, 1999).

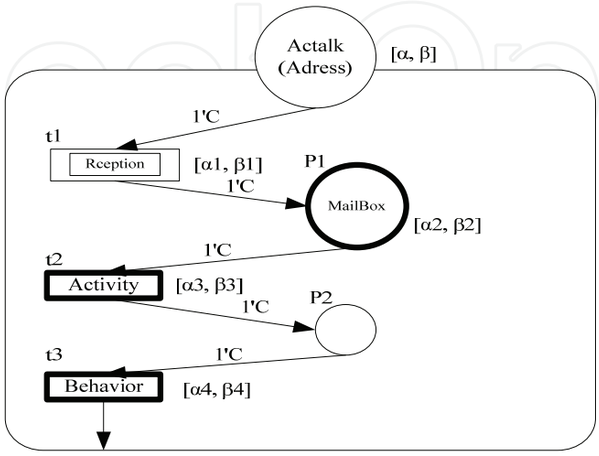


Figure 4. The FTOPN Model of ACTALK

ACTALK model is the basis of constructing *active object* models. However, *active object* model is the basis of constructing multi-agent system model or agent-based system model. So, as the modeling basis, ACTALK has been extended to different kinds of high-level agent models. Because of this, ACTALK is modeled in Fig.4 by FTOPN.

In Fig.4, OIP is the describer of the ACTALK model and also represents as the communication address. One communication transition is used to represent as the behavior of message reception. According to the communication requirements, it may be synchronous or asynchronous. If the message has been received, it will be stored in the corresponding mail box, which is one first in and first out queue. If the message has been received, the next transition will be enabled immediately. So mail box is modeled as abstract place object in FTAPN. If there are messages in the mail box, the following transition will be enabled and fired. After the following responding *activity* completes, some *active behavior* will be conducted according to the message.

Fig.4 has described the ACTALK model based on FTOPN on the macroscopical level. The detailed definition or realization of the object "*Activity*" and "*Behavior*" can be defined by FTOPN in its parent objects in the lower level. The FTOPN model of ACTALK can be used as the basic agent object to model agent based systems. That is to say, if the agent based model—ACTALK model is used in the usual FTOPN modeling procedure, FTOPN has been extended to *agent based modeling methodology*. So it is called *fuzzy timed agent based Petri net* (*FTAPN*).

4. Learning in Fuzzy Timed Agent Based Petri Nets

The parameters of FTAPN are always given beforehand. In general, however, these parameters may not be available and need to be estimated just like those in FTPN (Xu & Jia,

2005-1). The estimation is conducted on the base of some experimental data concerning marking of input and output places. The marking of the places is provided as a discrete time series. More specifically we consider that the marking of the output place(s) is treated as a collection of target values to be followed during the training process. As a matter of fact, the learning is carried out in a supervised mode returning to these ***target*** data.
The connections of the FTOPN (namely weights w_i and thresholds r_i) as well as the time decay factors α_i are optimized (or trained) so that a given performance index Q becomes minimized. The training data set consists of (a) initial marking of the input places $x_i(0),\ldots,x_n(0)$ and (b) target values—markings of the output place that are given in a sequence of discrete time moments, that is target(0), target(1),…, target(K).
In FTAPN, the performance index Q under discussion assumes the form of Eq.(4).

$$Q=\sum_{k=1}^{K}(t\arg et(k)-y(k))^2 \tag{4}$$

where the summation is taken over all time instants (k =1, 2,… , K).
The crux of the training in FTOPN models follows the general update formula in Eq.(5) being applied to the parameters:

$$\mathbf{param}(\text{iter}+1)=\mathbf{param}(\text{iter})-\gamma\nabla_{\mathbf{param}}Q \tag{5}$$

where γ is a learning rate and $\nabla_{\mathbf{param}}Q$ denotes a gradient of the performance index taken with respect to all parameters of the net (here we use a notation **param** to embrace all parameters in FTOPN to be trained).
In the training of FTOPN models, marking of the input places is updated according to Eq.(6):

$$\tilde{x_i} = x_i(0)T_i(k) \tag{6}$$

where $T_i(k)$ is the temporal decay. And $T_i(k)$ complies with the form in Eq.(7). In what follows, the temporal decay is modeled by an exponential function,

$$T_i(k)=\begin{cases}\exp(-\alpha_i(k-k_i)) & \text{if} \quad k>k_i,\\ 0 & \text{others}\end{cases} \tag{7}$$

The level of firing of the place can be computed as Eq.(8):

$$z=(\overset{n}{\underset{i=1}{T}}((r_i\rightarrow\tilde{x_i})sw_i)) \tag{8}$$

The successive level of tokens at the output place and input places can be calculated as that in Eq.(9):

$$y(k) = y(k\text{-}1)\mathbf{s}z,\ x_i(k) = x_i(k\text{-}1)\mathbf{t}(1\text{-}z) \tag{9}$$

We assume that the initial marking of the output place y(0) is equal to zero, y(0)=0. The derivatives of the weights w_i are computed as the form in Eq.(9):

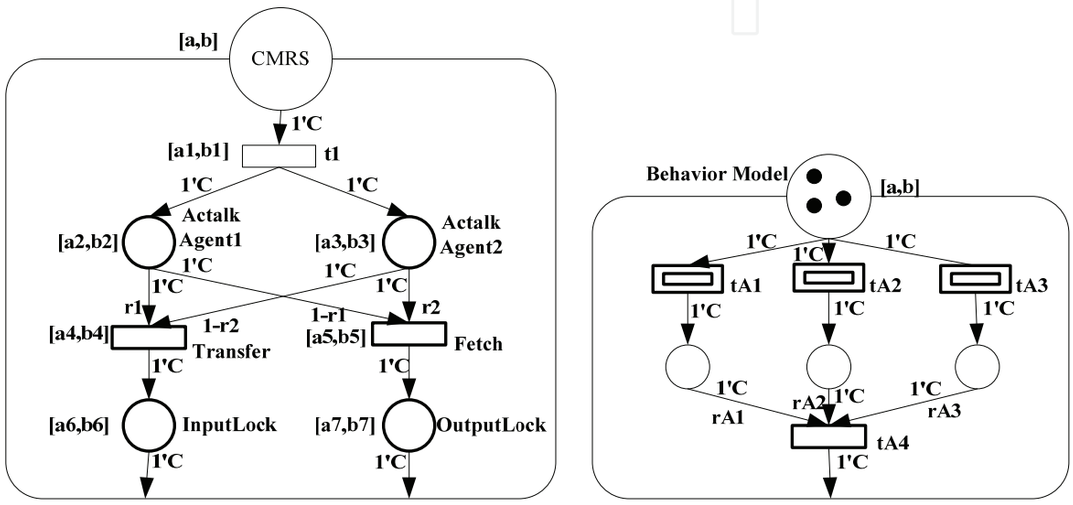
$$\frac{\partial}{\partial w_i}(t\arg et(k)-y(k))^2=-2(t\arg et(k)-y(k)\frac{\partial y(k)}{\partial w_i}) \tag{10}$$

where i=1,2,…, n. Note that y(k+1)=y(k)**s**z(k).

5. A Modeling Example

5.1 A CMRS Model

In the etching tools, usually there is a Brooks Marathon Express (MX) (Lee & Lee, 2004) CMRS platform made up of two transferring robots. These two cooperative robots are up to complete transferring one unprocessed wafer from the input lock to the chamber and fetch the processed wafer to the output lock. Any robot can be used to complete the transferring task at any time. If one robot is up to transfer one new wafer, the other will conduct the other fetching task. They will not conflict with each other. Fig. 5 depicts this CMRS FTAPN model, where two agent objects (ACTALK) are used to represent these two cooperative robots.



(a) The Agent Based FTAPN Model (b) The Behavior Model in Every Agent

Figure 5. The FTAPN Model

Figure 6. The Relevance

Fig.5 (a) has depicted the whole FTAPN model. The agent object—"ACTALK" is used to represent every robot model. Different thresholds are used to represent the firing level of the behavior conducted by the corresponding robot (agent). They also satisfy the unitary requirements and change according to the fuzzy decision in the behavior of every agent in Fig.5 (b). In the model of Fig.5 (b), three communication transition objects are used to represent the behavior for getting different kinds of system states. These states include the state of the other robot, its own goal and its current state, which can be required by the conductions of the communication transitions tA1, tA2 and tA3. When one condition has been got, the following place will be marked. In order to make control decisions (transition object tA4) in time, all of these state parameters are required in the prescriptive time interval. However, the parameter arrival times complies with the rule in Fig.5 (a). The other two kinds of information comply with that in Fig.5 (b). After the decision, a new decision

command with the conduction probability will be sent in this relative interval and it also affects which behavior (transfer or fetch) will be conducted by updating the threshold in Fig.5 (a).

5.2 Application Analysis

Table.1 summarizes the main features of FTAPN and contrast these with the structures with which the proposed structures have a lot in common, namely MAS and FTOPN. It becomes apparent that FTAPN combines the advantages of both FTOPN in terms of their learning abilities and the glass-style of processing (and architectures) of MAS with the autonomy.

Characteristics	MAS	FTOPN	FTAPN
Learning Aspects	Significant dynamic learning abilities. Dynamic learning and decision abilities are supported in every autonomous agent.	Significant learning abilities. Distributed learning (training) abilities are supported in different independent objects on various system model levels.	Significant dynamic learning abilities. Distributed dynamic learning and decision abilities are supported in every autonomous agent.
Knowledge Representation Aspects	Transparent knowledge representation (glass box processing style) the problem (its specification) is mapped directly onto the topology of the agent model. Additionally, agents deliver an essential feature of continuity required to cope with dynamic changes encountered in a vast array of problems (including autonomous decision tasks)	Glass Box Style (Transparent Knowledge Representation) and Black Box Processing is supported at the same time. The problem (its specification) is mapped directly onto the topology of FTOPN. Knowledge representation granularity reconfiguration reacts on the reduction of model size and complexity.	Glass Box Processing Style and Black Box Processing style are all supported. The problem (its specification) is mapped directly onto the topology of FTAPN, which can not only represent dynamic knowledge, but also deal with dynamic changes with well-defined semantics of agent objects, places, transitions, fuzzy and temporal knowledge.

Table 1. MAS, FTOPN and FTAPN: a comparative analysis

5.3 Application Aspects of FTAPN

Owing to the nature of the facet of temporal knowledge, fuzzy sets and object-oriented concepts in this extension of PN, they become viable models in a wide range of engineering problem augmenting the already existing high level Petri nets, cf. (Hong & Bae, 2000) (Xu & Jia, 2005-1). Two main and commonly categories of models are worth elaborating here.

5.3.1 Models of Multi-agent Systems

The multi-agent paradigm and subsequently a variety of models are omnipresent in a number of areas. In a nutshell, in spite of the existing variety of improved models, it still lacks of a powerful modeling method, which can bridge the gap between model and practical implementations. Petri nets come with objects, temporal knowledge and fuzzy sets can use active objects to model generic agents with situatedness, autonomy and flexible. This helps us to use the object to reduce the complexity of MAS systems and the dynamic learning and decision to support the autonomy of agents.

5.3.2 Models of Complex Real-time Systems

In models of complex real-time systems as usually encountered in industrial practice, the scale of the system module may be too complicated to be analyzed and the readings of different system state or sensors may be available at different time. The former may lead to the state explosion, while the latter needs adjustment of relevance of the information gathered at different time scales. The object models with temporal information degradation (aging) helps to abstract complicated model and quantify the confidence of the inferred results.

6. Conclusions and Future Work

CMRS is a kind of usual manufacturing equipments in manufacturing industries. In order to model, analyze and simulate this kind of systems, this chapter proposes fuzzy timed agent based Petri net (FTAPN) on the base of FTOPN (Xu & Jia, 2005-1) and FTPN (Pedrycz & Camargo, 2003). In FTAPN, one of the active objects— ACTALK is introduced and used as the basic agent object to model CMRS, which is a typical MAS. Every abstract object in FTOPN can be trained and reduced independently according to the modeling and analysis requirements for OO concepts supported in FTOPN. The validity of this modeling method has been used to model Brooks CMRS platform in etching tools. The FTAPN can not only model complex MAS, but also be refined into the object-oriented implementation easily. It has provided a methodology to overcome the development problems in agent-oriented software engineering. At the same time, it can also be regarded as a conceptual and practical artificial intelligence (AI) tool for integrating MAS into the mainstream practice of software development.

State analysis needs to be studied in the future. An extended State Graph (Xu & Jia, 2005-2) has been proposed to analyze the state change of TOPN models. With the temporal fuzzy sets introduced into FTAPN, the certainty factor about object firing (state changing) needs to be considered in the state analysis.

7. Acknowledgement

This work is jointly supported by the National Nature Science Foundation of China (Grant No: 60405011, 60575057) and the China Postdoctoral Foundation for China Postdoctoral Science Fund (Grant No: 20040350078) in China.

8. References

Agha, G., Hewitt, C.(1985). Concurrent Programming Using Actors: Exploiting Large Scale Parallelism, Lecture Notes in Computer Science, No. 206, S.N. Maheshwari,ed., Springer-Verlag, New York, 1985, pp.19–41.

Battiston, E., Cindio, F.D., Mauri, G.(1988). *OBJSA Nets: a class of high-level nets having objects as domains,* Proceedings of APN'88, Lecture Notes in Computer Science, Vol.340, pp.20-43

Briot, J.P. (1996). *An Experiment in Classification and Specialization of Synchronization Schemes*, Lecture Notes in Computer Science, No. 1107, pp. 227–249.

Cao, Y.U.; Fukunaga, A.S.; Kahng, A.B.; Meng, F. (1997); *Cooperative Mobile Robotics: Antecedents and Directions*, Autonomous Robots, Vol.4, pp.7–27

Castelfranchi, C. (1995).*A Point Missed in Multi-Agent*, DAI and HCI, Lecture Notes in Artificial Intelligence, No. 890, pp. 49–62

Chainbi, W.(2004) ; *Multi-agent systems: a Petri net with objects based approach*. In: Proceedings of IEEE/WIC/ACM International Conference on Intelligent Agent Technology, Beijing, pp.429 – 432

Gasser,L.(1992). *An Overview of DAI*, Distributed Artificial Intelligence, N.A. Avouris and L. Gasser, eds., Kluwer Academic, Boston, 1992, pp. 1–25.

Gasser, L, Briot,J.P.(1992). *Object-Oriented Concurrent Programming and Distributed Artificial Intelligence*, Distributed Artificial Intelligence, N.A. Avouris and L. Gasser, eds., Kluwer Academic, Boston, 1992, pp. 81–108.

Guessoum, Z., Briot, J.P.(1999). *From active objects to autonomous agents,* IEEE Concurrency, Vol.7, No.3, pp. 68 – 76

Hong, J.E., Bae, D.H.(2000). *Software Modeling And Analysis Using a Hierarchical Object-oriented Petri net,* Information Sciences, Vol.130, pp.133-164

Jennings, N.R., Sycara, K., Wooldridge, M.(1998). *A Roadmap of Agent Research and Development,* Autonomous Agents and Multi-Agent Systems, Vol.1, pp.7–38

Lee , J.H., Lee, T.E.(2004). SECAM: *A Supervisory Equipment Control Application Model for Integrated Semiconductor Manufacturing Equipment*, IEEE Robotics & Automation Magazine,Vol. 11, No. 1, pp.41 - 58

Maruichi, T., Ichikawa, M., and Tokoro, M. (1990). *Decentralized AI*, Modeling Autonomous Agents and Their Groups, Elsevier Science, Amsterdam, 1990, pp. 215–134.

Murata, T.(1989). Petri Nets: Properties, Analysis and Applications. *Proceedings of IEEE,* Vol.77, No.4, (April 1989) pp.541-580

Pedrycz, W., Camargo, H.(2003). *Fuzzy timed Petri nets*, Fuzzy Sets and Systems, Vol.140, No. 2, pp. 301-330

Peterson, J.L.(1991). *Petri Net Theory and the Modeling of Systems*, Prentice-Hall, N.Y., USA

Shoham, Y. (1993). *Agent-Oriented Programming*, Artificial Intelligence, Vol. 60, No.1, pp. 139–159.

Xu, H., Jia, P.F.(2005-1). *Fuzzy Timed Object-Oriented Petri Net,* Artificial Intelligence Applications and Innovations (Proceedings of AIAI2005), Springer, pp.148-160, N.Y., USA

Xu, H., Jia, P.F.(2005-2). *Timed Hierarchical Object-Oriented Petri Net*, GESTS International Transactions on Computer Science and Engineering, vol. 24, No.1, pp.65-76

Xu, H., Jia, P.F.(2006). *Timed Hierarchical Object-Oriented Petri Net-Part I: Basic Concepts and Reachability Analysis,* Lecture Notes In Artificial Intelligence (Proceedings of RSKT2006), Vol. 4062, pp.727-734

Yonezawa, A., Tokoro. M.(1987), Object-Oriented Concurrent Programming, A. Yonezawa and M. Tokoro, eds., The MIT Press, Cambrige, Mass., 1987.

Recent Advances in Multi Robot Systems

Edited by Aleksandar Lazinica

ISBN 978-3-902613-24-0
Hard cover, 326 pages
Publisher I-Tech Education and Publishing
Published online 01, May, 2008
Published in print edition May, 2008

To design a team of robots which is able to perform given tasks is a great concern of many members of robotics community. There are many problems left to be solved in order to have the fully functional robot team. Robotics community is trying hard to solve such problems (navigation, task allocation, communication, adaptation, control, ...). This book represents the contributions of the top researchers in this field and will serve as a valuable tool for professionals in this interdisciplinary field. It is focused on the challenging issues of team architectures, vehicle learning and adaptation, heterogeneous group control and cooperation, task selection, dynamic autonomy, mixed initiative, and human and robot team interaction. The book consists of 16 chapters introducing both basic research and advanced developments. Topics covered include kinematics, dynamic analysis, accuracy, optimization design, modelling, simulation and control of multi robot systems.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen