World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:2, No:11, 2008

Interactive Model Based on an Extended CPN

Shuzhen Yao, Fengjing Zhao, and Jianwei He

***Abstract*—**The UML modeling of complex distributed systems often is a great challenge due to the large amount of parallel real-time operating components. In this paper the problems of verification of such systems are discussed. ECPN, an Extended Colored Petri Net is defined to formally describe state transitions of components and interactions among components. The relationship between sequence diagrams and Free Choice Petri Nets is investigated. Free Choice Petri Net theory helps verifying the liveness of sequence diagrams. By converting sequence diagrams to ECPNs and then comparing behaviors of sequence diagram ECPNs and statecharts, the consistency among models is analyzed. Finally, a verification process for an example model is demonstrated.

***Keywords*—**Consistency, liveness, Petri Net, sequence diagram.

I. Introduction

IN distributed systems; there are a lot of parallel components. The parallel components bring potential risks for system performance. One of goals for system analysis is to discover the potential risks by means of static analysis or simulation of system models. The deadlock checking of models has been the study focus for distributed applications. Deadlocks mostly accompany interactions. From system view, interaction means not only the exchange of data but also synchronization between processes. The exchange of data is possibly related to deadlock caused by resource competition, while unreasonable application of synchronization between processes probably results communication deadlock. For object-oriented applications, besides properties of individual models, the consistency analysis among models has become another focus of application research [1]-[3].

In UML, the interaction among objects in a sequence diagram depends on services provided by individual objects [4] [5]. All services are linked to behaviors of objects, which are clearly depicted in the statecharts. It can be observed that sequence diagrams and statecharts have significant overlap in terms of expressing some dynamic behaviors. A key concern is identifying the degree of consistency of these two models. With its formal representation and dynamic analysis techniques, Petri Net can be used to check the consistency of different models, such as a sequence diagram and a statechart, based on their dynamic behaviors. There has been a lot of study devoted to individual models [6]-[8], but relatively little effort on exploring the relationship among such models. This paper explores the liveness of individual models and the consistency relationship between a sequence diagram and a statechart.

In Section 2, we will define ECPN. The property analysis techniques will be provided in detail in Section 3. Section 4 concludes the paper and mentions further work.

Shuzhen Yao is with Software School of Beihang University, Beijing, China (phone: 86-10-82314660; e-mail: szyao@buaa.edu.cn).

Fengjing Zhao is with Electronics & Information School of Shanhai Dianji University, Shanghai, China.

Jianwei He is with Computer School of Beihang University, Beijing, China.

II. Extended CPN

Sequence diagrams show a detailed flow for a specific use case or even just part of a specific use case. They show the calls between the different objects in their sequence and can show, at a detailed level, different calls to different objects. A sequence diagram has two dimensions: The vertical dimension shows the sequence of messages/calls in the time order that they occur; the horizontal dimension shows the object instances to which the messages are sent.

The widely used form of interactive diagram is sequence diagram, which describes interactions by focusing on the sequence of messages that are exchanged, along with their corresponding event occurrences on the lifelines. Sequence diagrams are applied to model interactions and in various phases of the software development process (e.g. use case refinement, modeling of test scenarios, interactive model, detailed modeling of message exchanges or specification of interfaces).

There have been formal techniques to analyze sequence diagrams. A variety of Colored Petri Nets (CPNs) or stochastic Petri Net are applied to check properties of interactive models, especially communication features [9]-[11]. An Extended CPN (ECPN) is defined to provide a verification mechanism on both individual models and multi models in this paper.

A. Overview of ECPN

ECPN is described as $\sum = (P, T, F, W, M_0)$, where:

$P\{p_1,p_2,\ldots,p_n\}$ $(n\geq 0)$ is a finite set of places. There are 2 sorts of places: state places and event places. State places are for holding states, represented as circles; event places is for holding events, represented as dual circles for synchronous events and a circle with a nested square for asynchronous events;

$T=\{t_1,t_2,\ldots,t_m\}$ $(m\geq 0)$ is a finite set of transitions, represented as rectangles. There are 2 sorts of transitions. Action transitions are used for general actions, represented as rectangles; object transitions are used to create or destroy objects, represented as rectangles with a nested rectangle;

$F\subseteq(P \times T)\cup(T \times P)$ is a set of arcs. A special kind of arc, inhibitor is introduced into F, which is shown as a line with a

small circle at the end, specifying a restriction that only when the source place of an inhibitor is empty, the transition associated is enabled. Inhibitors are adopted to describe synchronous interaction and priority of transitions. W and M_0 are the weight function and the initial marking of $\sum$ respectively.

B. ECPN Models

As a high-level Petri Net, ECPN can be used to model sequence diagrams. For sequence diagrams, we discuss their three basic flow structures: alternative, parallel and loop, and four basic interactive actions: sending message, receiving messages, creating messages and destroying messages. We began our research with defining equivalent ECPN structures for these basic components of sequence diagram, seen in Fig. 1 and Fig. 2.

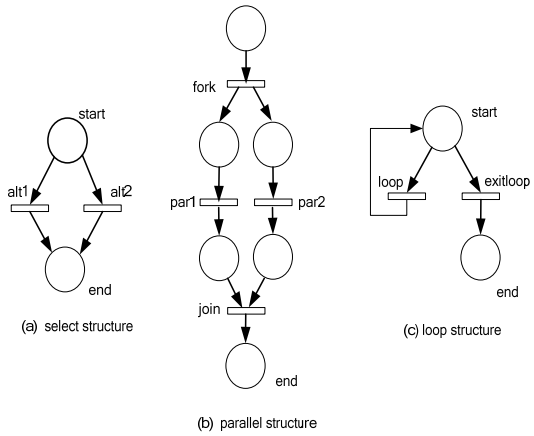
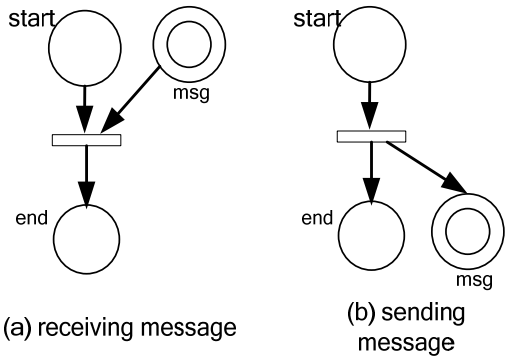


Fig. 1 Basic flow structures in ECPN



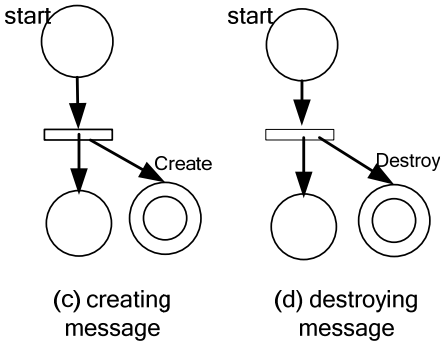


Fig. 2 Basic interactive actions in ECPN

We can get a corresponding ECPN from a sequence diagram by integrating the basic components. Let's take Fig. 3 as an example. Fig. 4 demonstrates two ECPNs corresponding to object A and object B in Fig. 3.

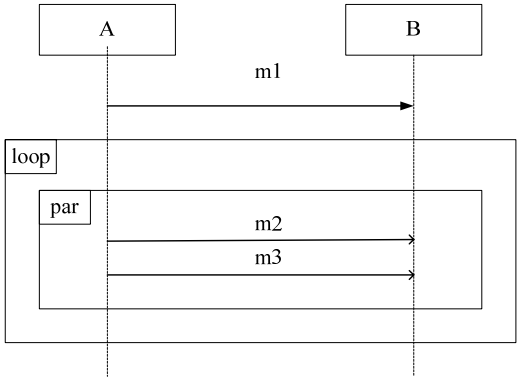
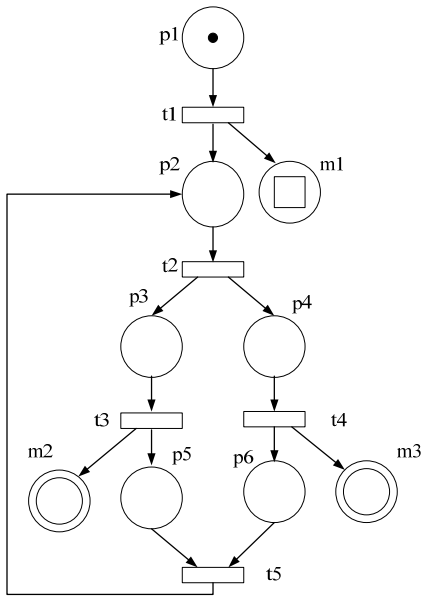


Fig. 3 An example of sequence diagram



(a) ECPN of object A

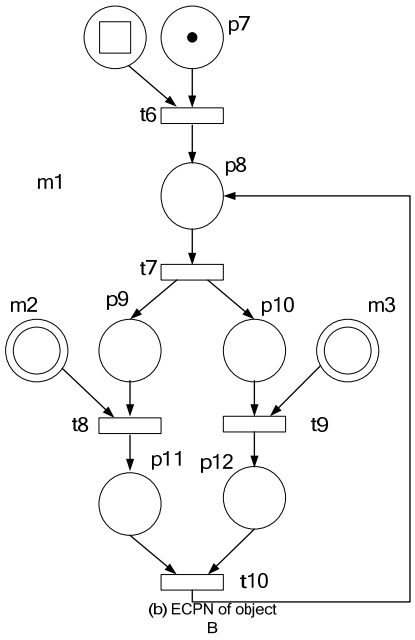


Fig. 4 ECPN of individual object

III. Properties Analysis

The next step is to merge all ECPNs into the overall ECPN of the sequence diagram. For objects in an interaction, some places from different objects may hold same kind of message. These places will be merging points of interactive objects. Fig. 5 is the ECPN merged from ECPNs in Fig. 4; Fig. 6 is the improved version of Fig. 5.

A. Merging of ECPNs

According to definition of ECPN, there are two sorts of places: state places and event places. For clear illustration, we predefine two functions over set of places as follows,

Definition 1 : *Type*

Type : P→{State,Event} is to get the type of a specific place.

Definition 2 : *Event*

Event : P→set of events, is to get the type of the message which is allowed to reside in a specific place.

Based on the definitions above, we give an operation .

Definition 3 : *Merging of ECPNs*

Assuming ECPN

$\sum 1=(P_1, T_1, F_1, W_1, M_{10})$, $\sum_2=(P_2, T_2, F_2, W_2, M_{20})$, $\sum=\sum_1 \oplus \sum_2=(P, T, F, W, M_0)$ is the merging of $\sum_1$ and $\sum_2$, where :

$P=P_1 \cup P_2$, and $\forall p_1 \in P_1, p_2 \in P_2$ if

$Type(p_1)=Type(p_2)=Event \wedge Event(p_1)=Event(p_2)$, then $p_1=p_2$ in P,

$T= T_1 \cup T_2$,

$F=\{f|f \in P \times T \cup T \times P,$ and $f \in F_1$ or $f \in F_2\}$,

$W=\{w|w \in W_1$ or $w \in W_2\}$,

$M_0= \{m|m \in M_{10}$ or $m \in M_{20}\}$

In order to represent the synchronous message m_1 to ensure that all event places hold asynchronous messages, an inhibitor from m_1 to t_2 is drawn as seen in Fig. 5 for the transformation from synchronous messages to asynchronous messages. When t_1 is fired, tokens are put into place p_2 and m_1.With the restriction of the inhibitor, t_2 is disabled until t_6 is fired and token in m_1 is taken. That is in accordance with semantic of synchronous message of sequence diagram.

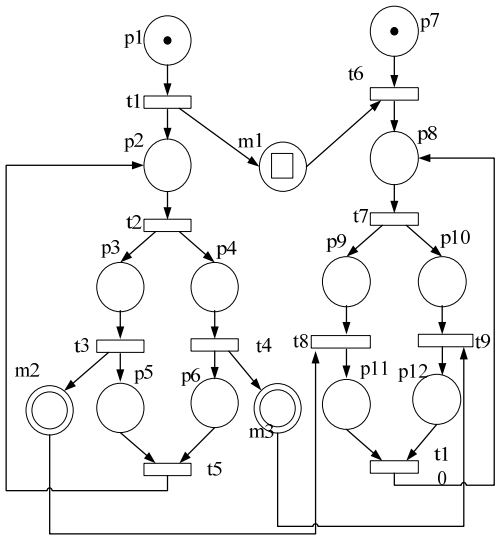


Fig. 5 Merging of ECPNs

B. Liveness Analysis

ECPN has a large expressive power to model a variety of interactive systems. But it is bound to have a high complexity, and it is not possible to develop a comprehensive theory that relates the structure of ECPN to its behavior. These limitations can be removed when we restrict our study to a specific subclass of Petri Net, where the result of the choice between two transitions can never be influenced by the rest of the system except for their common input places--in other words, choices are free. This subclass of Petri Net is called Free Choice Petri Net. Through analysis, it's found that, each basic interaction action of ECPN shown in Fig. 2 corresponds a Free Choice Petri Net. The typical structure for synchronization is a synchronized transition with multi input places. All of merging happens at the place which is either the starting place of an exclusive input arc to its post transitions or and the starting place of an exclusive output arc from itself. It's really the feature of a Free Choice Petri Net.

With the aid of Free Choice Petri Net theory, we could reason many properties of ECPNs, such as liveness, boundedness and cyclicity[12]. In Fig. 6, the feature liveness or deadlock-freedom is easily ruled out by means of Free Choice Petri Net theory about liveness. The analysis results are useful to enhance interactive models.

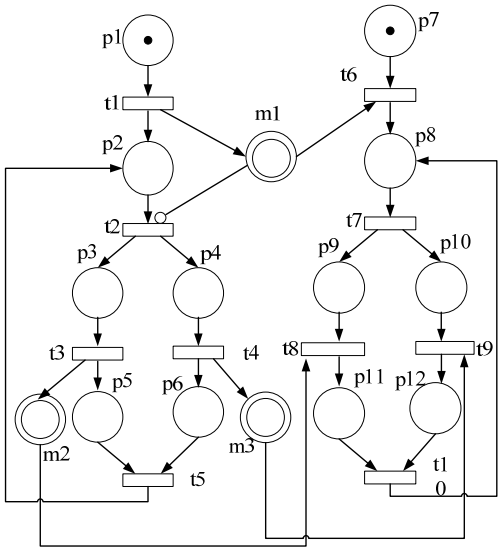


Fig. 6 Improved ECPN with synchronous messages transformed

C. Consistency Analysis

Another important property, consistency, is investigated in the paper. For a sequence diagram, related behavior sequences of an object can be formed from its ECPN's behavior sequence by picking up those transitions of receiving and sending messages by the object. Then we can interpret each transition related to message m as send m(short for sending m), or rec m(short for receiving m).

From Fig. 6 we get the behavior sequence of a sequence diagram

1) {(send m_1,rec m_1) (send m_2,rec m_2,send m_3,rec m_3)$^+$},
2) {(send m_1,rec m_1) (send m_2, send m_3, rec m_2,re cm_3)$^+$},
3) {(send m_1,rec m_1) (send m_3,send m_2,rec m_2, rec m_3)$^+$}
4) {(send m_1,rec m_1) (send m_2, send m_3, rec m_3,recm_2)$^+$}
5) {(send m_1,rec m_1) (send m_3, rec m_3,send m_2,rec m_2)$^+$}
6) {(send m_1,rec m_1) (send m_3,send m_2,rec m_3,rec m_2)$^+$}

In case 1)~3), the behavior sequence of object B is {rec m_1, (rec m_2,rec m_3)$^+$ }.In case 4)~6), the behavior sequence of object B is {rec m_1, (rec m_3,rec m_2)$^+$ }.So the behavior sequence of object B is {rec m_1, (rec m_3,rec m_2)$^+$ } or {rec m_1, (rec m_3,rec m_2)$^+$ }.

If the internal structure of the object B is like Fig. 7, we get the behavior sequence of single object B based on the algorithm for ECPN of statecharts as (rec m1,rec m_2,rec m_3)$^+$.

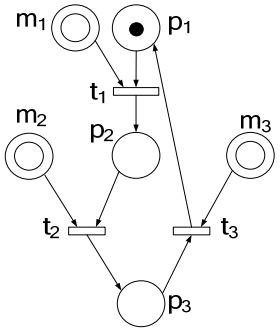


Fig. 7 One internal structure of object B

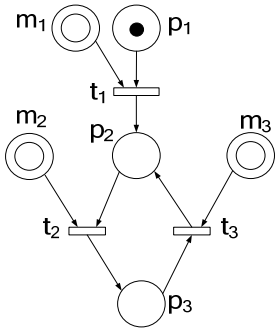


Fig. 8 Another internal structure of object B

Assuming we get the behavior sequence of a sequence diagram (rec m_1, rec m_2, rec m_3)$^+$, (rec m_1 ,rec m_2, ,rec m_3) is the only element included in {rec m_1, (rec m_3,rec m_2)$^+$ or {rec m_1, (rec m_3,rec m_2)$^+$,which is object B's behavior sequences. The comparison result indicates that from the second time on, there will be inconsistency.

This inconsistency detected by means of consistency checking will help to enhance the original models, interactive model or internal structure models. Considering the form of object B in Fig. 7, if we replace the arc from t_3 to p_2 with the arc from t_3 to p_1 , seen in Fig. 8, then the behavior sequence of the ECPN of the sequence diagram in Fig. 6 is included by the behavior sequence in the ECPN of the statechart in Fig. 8.

IV. Conclusion

In the paper, we outline the features of sequence diagrams in UML, and introduce ECPN to describe dynamic features of UML models. Our research covers UML modeling and property checking. With the wide application of Petri Nets, many modeling and analysis tools have been developed [13]. We have developped ECPN modeling tools to convert sequence diagrams and startcharts into analyzable ECPNs. The mechanism to analyze some important properties like liveness, consistency is demonstrated in the paper. We are processing on other features of the models and trying to find a way for automatic optimization of model behaviors.

References

[1] G. Engels, J. H. Hausmann, R. Heckel, and S. Sauer, "Testing the consistency of dynamic UML diagrams", Integrated Design and Process Technology, June 2002.
[2] S. K. Kim and D. Carrington, "A Formal Object-Oriented Approach to defining Consistency Constraints for UML Models", 2004 Australian Software Engineering Conference, 2004.
[3] T. Miyamoto, "A Survey of Object Oriented Petri Nets and Analysis Methods", IEICE Trans. Fundamentals, Vol. E88–A, No. 11, 2005
[4] J. Rumbaugh, I. Jacobson, and G. Booch, The Unified Modeling Language Reference Manual. Addison Wesley/Pearson, Jan. 2001.
[5] Tom Pender, UML Bible, Wiley, Sep. 2003.
[6] T. Schafer, A. Knapp, and S. Merz. Model Checking UML State Machines and Collaborations on the Workshop on Software Model Checking, Paris, Jul.2001.
[7] W. Dong, and Z. C. Qi, "Study on the checking of UML Models", Dissertation of National Defense University of Science and Technology, China, Oct. 2002.

[8] B.Simona,D.Susanna,M.Jose, from UML Sequence Diagrams and Statecharts to analysable Petri Net models,In Proc. of 3. rd. Int. Workshop on. Software and Performance (WOSP2002), .Rome, Italy, July 2002.
[9] E. Guerra and J. D. Lara, "A Framework for the Verification of UML models. Examples using Petri Nets", http://www.ii.uam.es/~jlara/investigacion
[10] S.Z.Yao, F.G.Tang, and Y.F. Liu, "An object-oriented model for parallel software", TOOLS27, China, Sep.1998.
[11] M.H.Hamza, Modelling, Identification, and Control (MIC 2004), Grindelwald, Switzerland, Feb.2004.
[12] T. Murata, "Petri Nets: Properties, Analysis and Applications", Proc. IEEE, Vol. 77, 1989.
[13] http://www.informatik.uni-hamburg.de/TGI/Petri Nets /tools.