http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 12 (23): 7289-7296, 2013
ISSN 1812-5638 / DOI: 10.3923/itj.2013.7289.7296

Toward Automatic Analyzing Statechart with Petri Net

[1,2]Ma Wei-gang, [3]Cao Yuan, [2]Wei Wei, [4]Lu Wei, [1]Ma Jian-feng and [2]Hei Xin-hong
[1]School of Computer Science and Technology, Xidian University, Xi'an, China
[2]School of Computer Science and Engineering, Xi'an University of Technology, Xi'an, China
[3]School of Electrical Engineering, Beijing Jiaotong University, Beijing, China
[4]Automation and Information Engineering, Xi'an University of Technology, Xi'an, China

Abstract: In view of utilizing Petri net to perform analysis of UML model, this paper proposes a transformation methodology from UML statechart to Petri net in terms of the semantics. Firstly, it gives the formal definition of the UML statechart, uses state sets, transition sets, event sets, guard sets, object sets and refinement functions of the state to describe the features of the UML statechart and designs transfer sets of state concurrency, transfer sets of collision and transfer priorities. Based on the strict formal semantics of the Petri net, the study studies the equivalency between them and proposes a transformation algorithm from UML statechart to Petri net which lays the foundation of automated transformation. Finally, in order to validate the proposed methodology, a novel decentralized railway interlocking system which ensures train safety in stations is modeled and analyzed.

Key words: UML statechart, petri net, distributed railway interlocking system

INTRODUCTION

Unified Modeling Language (UML) has become the industry standard. It not only supports object-oriented analysis and design, but also supports the whole process of software development from requirement analysis period. Applying the UML modeling approach brings a number of advantages: First of all, as UM L is a popular modeling language, the requirement models are understandable for both software and system engineers, who do not worry about misunderstanding due to different technical backgrounds; Secondly, UML offers multiple view-points for diflerent stakeholders and is more easily applied to modeling integrated architectures; Thirdly, The architecture models can be smoothly turned into software application designs and even into excusable simulation models by means of some emerging excusable UML approaches (Dong *et al*., 2012).

Because of UML is a semi-formal modeling language, the description of model is lacking in precise semantics. Although most of the static semantics have been defined, most of the dynamic semantics are described by natural language that often occurs vague or ambiguous. Currently there are two main ideas about the formal methods of UML. First, formulates formal of the UML core syntax and makes it in line with the formal specification language. In reference (France *et al*., 1998) a formal language on the meta-model level of the UML was proposed. Second, formulates transformation methods that use formal language for UML formal analysis under the premise of no or little loss information. In reference (Guo and Yao, 2007), GUO proposed a formal model SC_Net which can precisely describe the dynamic features of UML statechart.

Petri net is a graphical tool which can clearly describe the internal interactions of a system, such as concurrency, conflict, distribution etc. Meanwhile, Petri net has mature mathematical analysis methods (Gu *et al*., 2010), such as reachability, deadlock and other formal analysis methods and validation tools. These merits make Petri net become popular to modeling, analysis as well as optimization of complex systems, such as train dispatching system (Hei and Na, 2011). In reference, a new modeling method of extended hybrid Petri nets (EHPNs) was proposed to describe a semiconductor wafer fabrication flow available.

This study uses the Petri net to describe semantics of the UML statechart, defines the formal syntax of the UML statechart, studies the semantics of UML statechart and Petri net, then proposes an algorithm for transforming statechart into Petri net. This work will be the basis of the automated analysis of the UML statechart, which is significant for software specification, in safety-critical systems, such as train control system.

Formal definition of UML statechart: Main elements of UML statechart include: state, transition, event and so on.

Corresponding Author: Ma Wei-gang, School of Computer Science and Technology, Xidian University, Xi'an, China

State describes a period of time of a class object existence. Transition means that when a particular event occurs or satisfies some certain conditions, the state will transform a source state into target state. Event describes the state change in a particular time or space. In the UML statechart, triggers event of transition may send messages to some collaboration objects, so this study includes object sets in definition UML semantics.

Definition 1: UML statechart is defined as the six-tuple SC = (S, T, E, G, O, Func), S: state sets; T: transfer sets; E: event sets; G: guard sets; O: object sets; Func: state refinement function. Including the following functions:

- **State name:** S→String, state name
- **Child state:** S→2S, layer (parent/child) relation of between the states
- **Sub state:** S→p(S), defines p(S) is all sub-state of a state, there is a unique root state root∈S and ∀s∈S, root∈p(S) in the UML Statechart
- **Initial state:** S0→(0, 1)
- **Source:** T→S, source state of transition
- Target: T→p(S), target state of transition
- **Event:** T→E, triggers event of causing transition
- **Guard:** T→**G**, guard condition
- **Object:** (T→E)→O, occurrence of each transition may have a corresponding object
- **Entry:** enters action of the state
- **Exit:** exits action of the state
- Func:

$$S \to \{BASIC, AND, OR, PSEUDO\}$$

Which:

$$?(S) \neq F \wedge Func(S) = \{BASIC\}$$

S is basic state (including simple and final), because of initial state is a marking, so this study will not include initial state in the state sets; When:

$$?(S) \neq F \wedge Func(S) = \{AND\}$$

and in the state S, in fact is all the sub-state of state S; When:

$$?(S) \neq F \wedge Func(S) = \{OR\}$$

And in the state S, in fact is a unique sub-state of state S; When:

$$\rho(S) \neq F \wedge Func(S) = \{PSEUDO\}$$

S is RSEUDO state.

Definition 2: For $t, t' \in T$, if:

$$Event(t) = Event(t'),\ ?(Source(t)) \wedge\ ?(Source(t')) = F$$

So t and t' will cause concurrent transition set.

Definition 3: For t, t'∈T, if:

$$Event(t) = Event(t'),\ ?(Source(t)) \wedge\ ?(Source(t')) \neq F$$

So, t and t' will cause conflict transition set.

Definition 4: For conflict transition set, if :

$$?(Source(t) \wedge\ ?(Source(t')) \neq F$$

and:

$$?(Source(t)) \subset ?(Source(t'))$$

So:

$$t'\ priority > t\ priority$$

Definition 5: The state of UML statechart describes with pattern, pattern includes current all active sub-state and also includes the composite state active sub-state, by the function Patf: S→2S definition, all possible pattern of a state is as follows:

$$Patf(s,t) = \begin{cases} \{\lambda\} & if\ Func(s) = final \in \{BASIC\} \\ \{S\} & if\ Func(s) = simple \in \{BASIC\} \\ \cup_{s' \in Child(s)} Patf(s') & if\ Func(s) = AND \wedge s' \in Child(s) \\ (\otimes_{s' \in Child(s)} Patf(s')) & if\ Func(s) = OR \wedge s' \in Child(s) \end{cases}$$

Definition 6: Transition sequence of the state: When trigger event of the statechart triggers the state transition, it first executes exiting action of Source(t), the event E triggers Source(t), to Target(t). Then executes entering action of Target(t). The transition sequence of execution is:

$$Exit(Source(t)) \bullet Trans(E \& G) \bullet Enter(Target(t))$$

SEMANTIC TRANSFORMATION OF UML STATECHART TRANSFORMED TO PETRI NET

Based on the above, some simple semantics definition of the UML statechart is proposed and state, transition, event and so on elements of the statechart will be transformed into Petri net related elements from the semantic perspective. Place/Transition Petri net defined as a six-tuple:

$$PN = \left(P, T, F, K, W, M_0\right)$$

UML statechart will be transformed into Petri net as following:

State transformation: In Petri net, place describes the state of the system, these states and the state of the UML statechart have the same semantic that should be mapped to place of the Petri net. However, when initial state of the statechart will mapped to place of the Petri net, should include token, State (intial)→Place(token), while others state will be mapped to:

$$State\{s \in \{simple, final\}\} \rightarrow Place(p)$$

Transition processing: Transition of the statechart connects pattern of the two stable states, which the object from one state to another state, so should be mapped to transition of the Petri net:

$$T = \left\{ t_{from,t,to} \middle| \begin{array}{l} form \in Exit(Source(s',t)) \\ s' \in Patf, \quad t \in Transition(E \& G) \\ to \in Entry(Target(s',t)), \end{array} \right\}$$

Arc processing: In the Petri net:

$$F \subseteq (P \times T) \cup (T \times P)$$

is flow relation,so should be mapped to arc of the Petri net:

$$F \subseteq (< p, t_{from,t,to} > | p \in Place, t_{from,t,to} \in T)$$
$$U(< p, t_{from,t,to}, p > | t_{from,t,to} \in T, p \in place)$$

In order to accurately analyze nature of the model, this study will semi-formal UML statechart transform into formal Petri net from the semantic perspective. In the semantic definition process, a method is simple defined and transformed from the both concepts perspective. For some complex issues in the transition process, such as communication between the statechart will be researched in the next step.

Instance analysis

Distributed railway interlocking system: Distributed railway interlocking system (DRIS) ensures train run safely in the station by the cooperation of all related devices. Same to the traditional railway interlocking system, DRIS still includes three key devices: signal, point and track unit. But these three key devices in DRIS work independently to complete tasks and there no longer need a traditional central interlocking computer and they communicate with each other directly without centralized interlocking computer. The DRIS features strong independency, standardization of software and hardware, system decision decentralized and so on (Hei and Nam, 2011), as shown in Fig. 1.

Consider each kind of interlocking devices has the same functions, it is considered as an object and the object has its own control logic. These objects combine the logic relation or station yards data to execute the defined actions. For instance, the signal object include execution flow which is designed beforehand, such as reading route request messages, preparing destination device objects for messages, sending messages to related device objects, receiving and analyzing response messages from related device objects, executing route setting, canceling route requests and so on. When interlocking device object are initialized, the centralized data are downloaded to the device objects, which are then activated and interact with each other to complete the route interlocking.

The UML statechart model of DRIS: As a safety-critical system, the reliability and safety of DRIS are expected to

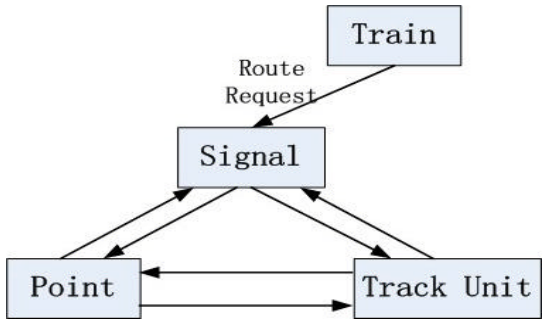

Fig. 1: Concept of DRIS

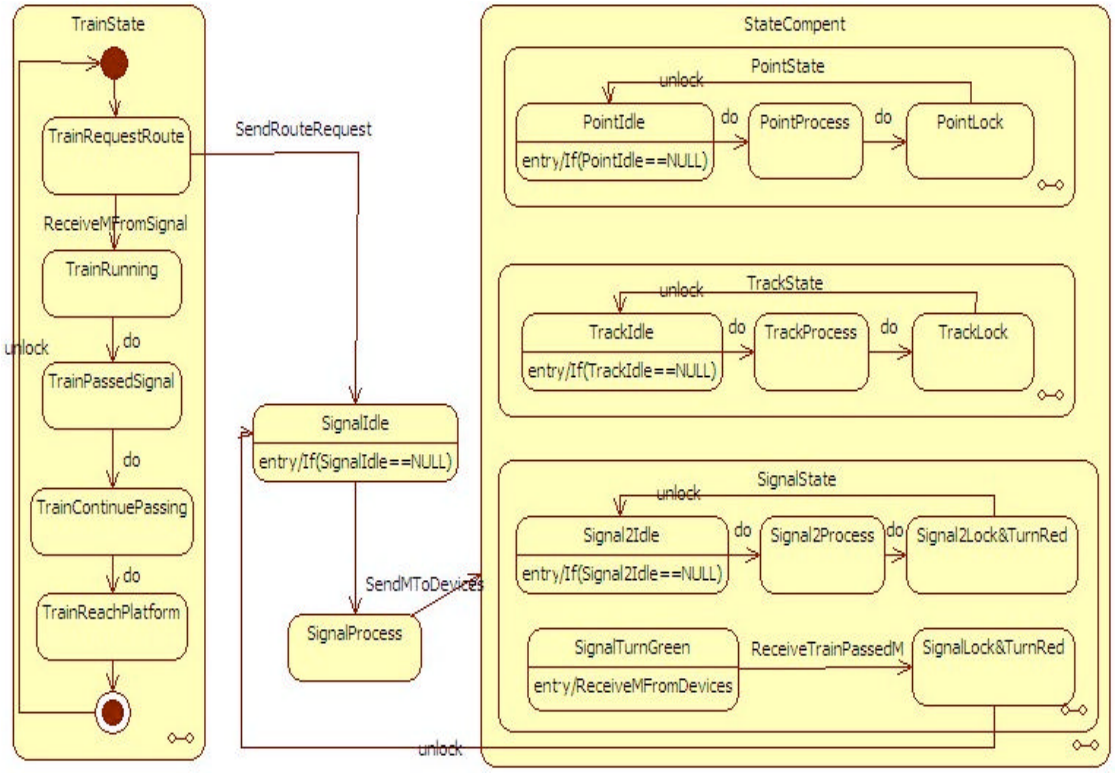

Fig. 2: DRIS UML statechart

ensure. Based on concept of DRIS, its UML statechart can be shown as Fig. 2.

Figure 2 includes the most of the statechart contents, which contains initial state, simple state, composite state (OR state, AND state) and the final state. The transition also includes various transfers in the statechart: transition of the simple state, transition of the OR state and transition of the AND state. From the UML statechart, it is considered that the signal, the point and the track circuit are concurrent and independent.

Transformation UML statechart into Petri net and Verification: According to the transformation rules in Section III, the UML statechart will be transformed into Petri net as shown in Fig. 3. In the statechart, the state will be mapped to the place of Petri net and transition between the states will be transformed into transition of the Petri net. However, some transitions between the states are restricted by the guard condition, such as: if the Singleidle = NULL and the guard condition = NULL, which indicating there is no car, the SingleIdle→Place. Otherwise wait.

In order to ensure the safety of train route, the correctness of Petri net model will be checked. Petri net analysis methods mainly include: reachability graph and coverability tree, incidence matrix and state equation, invariants and so on. This study uses Petri net analysis tools Pipe 3.0 to analyze distributed railway interlocking system.

As shown in Fig. 4, the state space analysis is used to analyze the Petri net model. From the Fig . 4, the Petri net model translated from the UML model of distributed railway interlocking system is bound, safe and not deadlock. It is concluded that the UML model of Fig. 2 is bound, safe and has not deadlock.

As shown in Fig. 5, the invariants method is used to analyze the Petri net model.

And P-Invariant equations of Petri net model are showed as flows:

- M(P0)+M(P1)+M(P11)+M(P12)+M(P13)+M(P14)+M(P15)+M(P16)+M(P17)+M(P18)+M(P2)+M(P3)+M(P8)= 1
- M(P0)+M(P1)+M(P13)+M(P14)+M(P15)+M(P16)+M(P17)+M(P18)+M(P2)+M(P3)+M(P4)+M(P5+M(P6)=1

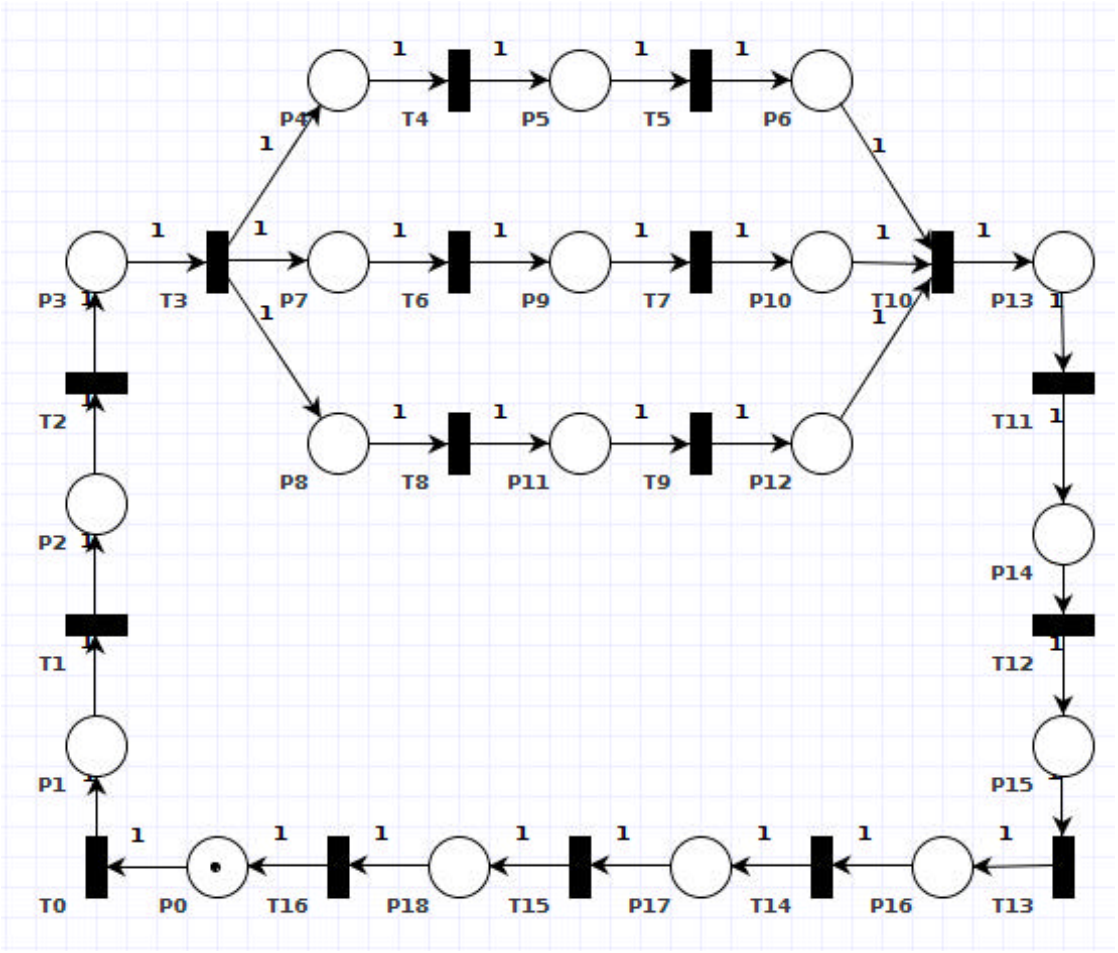

Fig. 3: Generated Petri net model

Petri net state space analysis results

Bounded	true
Safe	true
Deadlock	false

Fig. 4: Petri net state space analysis results

- M(P0)+M(P1)+M(P10)+M(P13)+M(P14)+M(P15)+ M(P16)+M(P17)+M(P18)+M(P2)+M(P3)+M(P7)+M(P9)= 1

From the Fig. 5 and P-Invariant equations, it is concluded that P-Invariants is a bounded and live net and T-Invariants is also bounded. The result confirms the safe process of signal scheduling specification.

As show in Fig. 6, Petri net simulation is performed to analyze the Petri net model. The average number of tokens of each place is calculated and shown in the figure. According to the average number of tokens of place, the efficiency of actions designed in the UML statechart is evaluated.

As show in Fig. 7, the reachability graph is used to analyze the generated Petri net model. All states in the graph are vanishing states. It is also concluded that the Petri net model is not deadlocked.

For the generated Petri net in Fig. 3 from the UML model of DRIS which is shown in Fig. 2, we use four different methods to analyze it respectively, include: space analysis method, invariant analysis, simulation as well as reachability graph. The conclusion is consistent, i.e., the model is bound, safe and has not deadlock.

Petri net invariant analysis results

T-Invariants

T0	T1	T10	T11	T12	T13	T14	T15	T2	T3	T4	T5	T6	T7	T8	T9	T16
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

The net is covered by positive T-Invariants, therefore it might be bounded and live.

P-Invariants

P0	P1	P10	P11	P12	P13	P14	P15	P16	P17	P18	P2	P3	P4	P5	P6	P7	P8	P9
1	1	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	1	0
1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0
1	1	1	0	0	1	1	1	1	1	1	1	1	0	0	0	1	0	1

The net is covered by positive P-Invariants, therefore it is bounded.

Fig. 5: Petri net invariant analysis results

Place	Average number of tokens	95% confidence interval (+/-)
P0	0.0495	0
P1	0.06931	0
P10	0.14851	0.03996
P11	0.14851	0.04753
P12	0.12871	0.08969
P13	0.05941	0
P14	0.05941	0
P15	0.05941	0
P16	0.05941	0
P17	0.05941	0
P18	0.0495	0
P2	0.05941	0
P3	0.05941	0
P4	0.14851	0.07465
P5	0.11881	0.04144
P6	0.14851	0.08799
P7	0.13861	0.04753
P8	0.13861	0.07992
P9	0.12871	0.05704

Fig. 6: Petri net simulation result

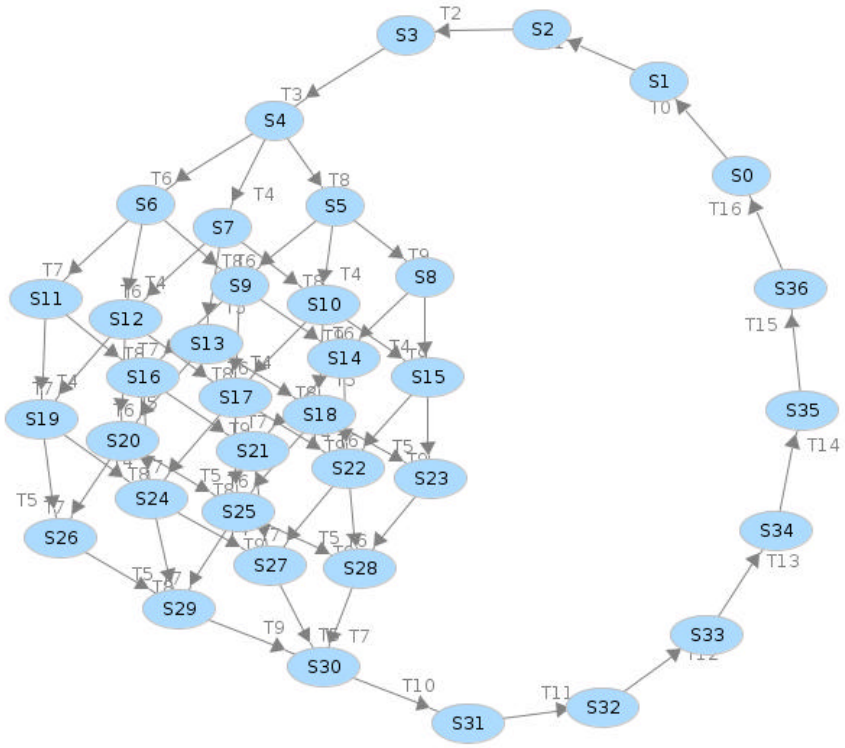

Fig. 7: Reachability graph results

With the proposed method, more complex UML statechart can be analyzed in a strict way, which reduces the human errors and ensures the effect of the verification.

CONCLUSION

Consider it is extremely important to confirm software specification in safety-critical systems, such as train control system, this study proposes to combine UML and Petri net. And because the UML statechart is semi-formal, it is difficult to analyze and validate its model. So this study gives a semantic definition for the UML statechart and Petri net. Based on these definitions, proposes a method which can transform UML statechart into Petri net. Finally, uses Petri net mature analyses methods to analyze the model, including T-invariants, P-invariants, simulations and reachability graph. As a case study, a distributed railway interlocking system is modeled and analyzed. By the obtained conclusion that the net is bounded and live, safety and correctness of the UML model are verified.

Our future work is to specify the elements identification of UML statechart and Petri net based on previous analysis and further express some complex semantics between the UML statechart and Petri net, such as communication problems by referring CSP (Communication Sequential Processes).

ACKNOWLEDGMENT

The authors wish to acknowledge the support of the National Natural Science Foundation of China (No. 61100173, No. U1334211), Fundamental Research Funds for the Central Universities(2012JBZ014) and fund of ShaanXi Province Education Department (No: 11JK1038). This project is also supported by China Postdoctoral Science Foundation (No.2013M542370). This work is also supported by Scientific Research Program Funded by Xi'an University of Science and Technology (Program No.201139).

Foundation item: This work was supported by the National Natural Science Foundation of China (No. 61100173),

REFERENCES

Dong, Q.C., Z.X. Wang, G.Y. Chen, X. Jiang and T.T. Zhang, 2012. Domain-Specific modeling and verification for C4ISR capability requirements. J. Central South Univ., 19: 1334-1340.

France, R., A. Evans, K. Lano and B. Rumpe, 1998. The UML as a formal modeling notation. Comput. Stand. Interfaces, 19: 325-334.

Gu, Y.W., Z.T. Wang and Q.D. Wu, 2010. Application of object-oridented Petri-nets in system modeling. J. Tongji Univ. Nat. Sci., 38: 437-441.

Guo, F. and S.Z. Yao, 2007. Formal model of UML statechart based on petri nets. J. Beij. Hangkong Hangtian Daxue Xuebao, 33: 248-252.

Hei, X.H. and O.Y. Na, 2011. The scheduling strategy of concurrent request in distributed railway interlocking system. ICIC Express Lett. B Appl., 2: 43-48.