Automatic Transformation from UML Statechart to Petri Nets for Safety Analysis and Verification

Xinhong Hei, Lining Chang, Weigang Ma, Jinli Gao
School of Computer Science and Engineering
Xi'an University of Technology
Xi'an, China
heixinhong@xaut.edu.cn

Guo Xie
College of Science and Technology
Nihon University
Chiba, Japan

Abstract—**As a powerful and object-oriented graphical modeling language, Unified Modeling Language (UML) has been introduced into the design and development of safety-critical computer systems. However, UML model is difficult to reflect the concurrency and consistency of constraint rules between objects and impossible to completely demonstrate the dynamic behavior characteristics of a system. Further, it cannot be directly expressed and analyzed by mathematical tools. Petri net modeling technology uses graphical element place and transition to clearly describe the internal interactions of a system. Meanwhile, Petri net has mature mathematical analysis methods. We have proposed to combine these two tools for designing and analyzing a system with a set of transformation rules from UML statecharts to Petri net. In this paper, an automatic conversion tool is introduced. The tool performs the transformation with three steps as follows: (1) Transforming UML model into XML document, which makes information extraction of the UML model become easier. (2) Analyzing semantics of UML model and Petri net model, then designing data structure for transformation, and completing the transformation based on the transformation rules. (3) Analyzing the transformed Petri net model to verify the characteristics of the UML model. The transformation rules and the tools are developed based on a new railway interlocking system.**

Keywords-UML; petri net; DRIS; XML; automatic conversion tool

I. INTRODUCTION

More and more functions are implemented by software in the safety-critical systems. Therefore ensuring the software reliability and safety has become a very important issue. Also the design and expression of software specification are key problems. As a well-defined and powerful graphical modeling language including new ideas, new methods and new technologies of software engineering domain, UML has become an indispensable part in software design, especially object-oriented design [1]. However, the software requirements described by UML are difficult to analyze and verify, it is a serious defect to the high reliability requirements. In [2], model transformation for the formal analysis of railway interlocking models is given. Model transformation is a key process in Model-Driven Engineering (MDE). Once a model is specified, executable code for a computing platform can be automatically generated by means of model transformation. Regretfully, the paper does not provide a transparent verification methodology.

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, such as reachability, deadlock and other formal analysis methods and validation tools [3]. These merits make Petri net become popular to modeling, analysis as well as optimization of complex systems, such as train dispatching system[4], scheduling discrete events, combination of hardware-software cooperative mechanism [5], and concurrent processing mechanism [6], etc.

In this paper we introduce an automatic transformation tool which transforms UML model to Petri nets via XML (Extensible Markup Language) in order to analyze and verify the original UML model. The tool is developed based on semantic transformation rules we have proposed in previous work. And as a case study, both the rules and the tools utilize a distributed railway interlocking system which ensures train running safely in the station.

II. BACKGROUND

A. Unified Modeling Languae (UML)

Unified Modeling Language UML is a powerful, object-oriented visual system analysis and modeling language. The main contents of UML can be defined by five class graphics (total nine kinds of graphics): (1) Use case diagram, describes the system function from the consideration of users, and points out actor of the each function. (2) Static diagram, including class diagram, object diagram and package diagram. (3) Action diagram, describes the system dynamic model and interaction between the component objects, including state diagram and activity diagram. (4) Interactive diagram, describes the interactions between objects, including sequence diagram and collaboration diagram. (5) Implementation diagram, including component diagram and deployment diagram.

B. Extensible Markup Language (XML)

XML can easily describe and organize data. However, in practice we need resort to XML parser to extract data users need from application [7]. At present, TinyXML is the most popular and open source XML parser which is based on DOM model and applied in C++ platform. The parser analyze XML file to generate DOM model in memory, so that we can convenient traverse the XML tree.

C. Petri Net

Petri net analysis method is a mathematical model and graphics technology. A place/transition (P/T) system consists of a six-tuple $\Sigma = (S, T; F, K, W, M_0)$:

1) (S,T; F) is a network, S elements is place, T elements is transition;

2) $K: S \to N+ \cup \{\infty\}$ is location capacity function;

3) $W: F \to N+$ is arc weight function;

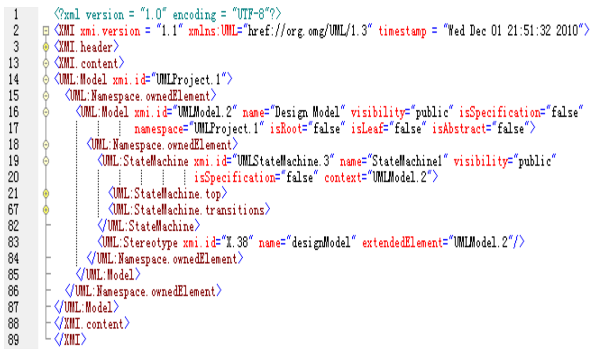
4) $M_0: S \to N$ is initial marking, it satisfy the condition: $\forall s \in S: M_0 s) \leq K(s)$.

P/T_system has a special case of $K = \omega$ and $W = 1$, which means that the capacity of all place are infinite and weight of all arcs are 1.

III. Transformation from UML Statecharts to XML Document

In previous work, we have completed the system modeling of the railway interlocking system with UML statechart. Furthermore, we have defined corresponding transformation rules from UML statechart to Petri net based on main elements of statechart. With these rules, a UML model can be transformed into Petri net manually. The purpose of these work are to give a formal analysis of UML model by means of Petri net tools.

This paper presents an automatic conversion tool from UML statechart to Petri net. Consider it is difficult to get the main information from UML model directly, we propose to transform UML statechart into XML documents firstly, then read it through the XML documents. UML modeling tools provide a function to export UML statechart to XMI document which is an XML-based metadata exchange mechanism and defines a mapping form UML to XML. By doing this it needs not to create a new XML express. Therefore, first we will correspond to UML model with XMI model, in order to transform UML model into XML model description smoothly. XMI document is shown as Fig. 1.



```
1   <?xml version = "1.0" encoding = "UTF-8"?>
2   <XMI xmi.version = "1.1" xmlns:UML="href://org.omg/UML/1.3" timestamp = "Wed Dec 01 21:51:32 2010">
3   <XMI.header>
13  <XMI.content>
14  <UML:Model xmi.id="UMLProject.1">
15    <UML:Namespace.ownedElement>
16      <UML:Model xmi.id="UMLModel.2" name="Design Model" visibility="public" isSpecification="false"
17                namespace="UMLProject.1" isRoot="false" isLeaf="false" isAbstract="false">
18        <UML:Namespace.ownedElement>
19          <UML:StateMachine xmi.id="UMLStateMachine.3" name="StateMachine1" visibility="public"
20                           isSpecification="false" context="UMLModel.2">
21            <UML:StateMachine.top>
67            <UML:StateMachine.transitions>
82          </UML:StateMachine>
83          <UML:Stereotype xmi.id="X.38" name="designModel" extendedElement="UMLModel.2"/>
84        </UML:Namespace.ownedElement>
85      </UML:Model>
86    </UML:Namespace.ownedElement>
87  </UML:Model>
88  </XMI.content>
89  </XMI>
```

Figure 1. XMI document

- The first line is XML declaration;
- The second line is the XMI root element, which must contain XMI version property;
- The two sub-elements XMI.header and XMI.content of XMI root element are placed from the third line to the eighty-eighth line;
- The third line to the twelfth line store the model information;
- The thirteenth line to the sixty-sixth line store the actual model information;
- The sixty-seventh line to the eighty-first store the location information of model elements, and finally by means of TinyXML parse XML and obtain valid information of UML model.

IV. Design an Automatic Transformation Tool from UML to Petri Net

A. Distributed Railway Interlocking System (DRIS)

Distributed railway interlocking system 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, as shown in Fig. 2.

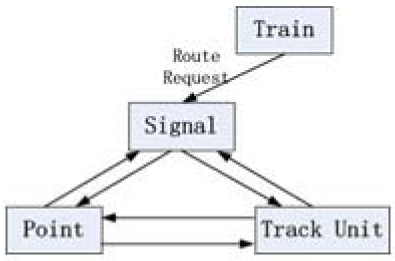


Figure 2. Concept of DRIS

Consider each kind of interlocking devices has the same functions, we may consider them as a 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.

B. The UML Statechart Model of DRIS

As a safety-critical system, the reliability and safety of DRIS are expected to ensure [4, 8]. Based on concept of DRIS, its UML statechart can be shown as Fig. 3.

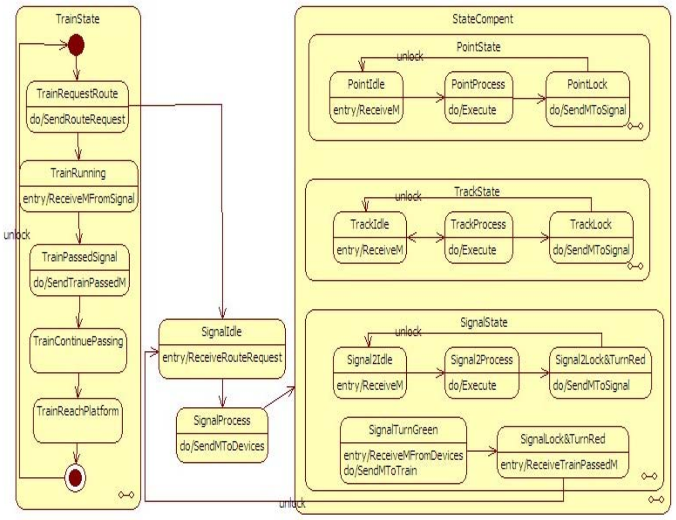


Figure 3. DRIS UML statechart

It can be seen from the UML statechart that signal, point and track are parallel and independent. So when designing the automatic conversion tool, train, signal, point and track will be modeled respectively. Based on the proposed transformation rules in the previous study, read the valid information from the XML documents, design data structure for conversion, lastly realize the automatic conversion tool.

C. *The Automatic Conversion Tool Block Diagram*

The software block diagram is shown as Fig. 4.

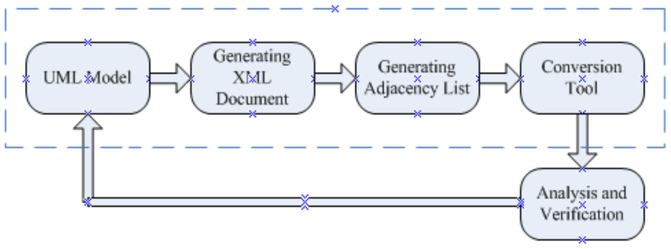
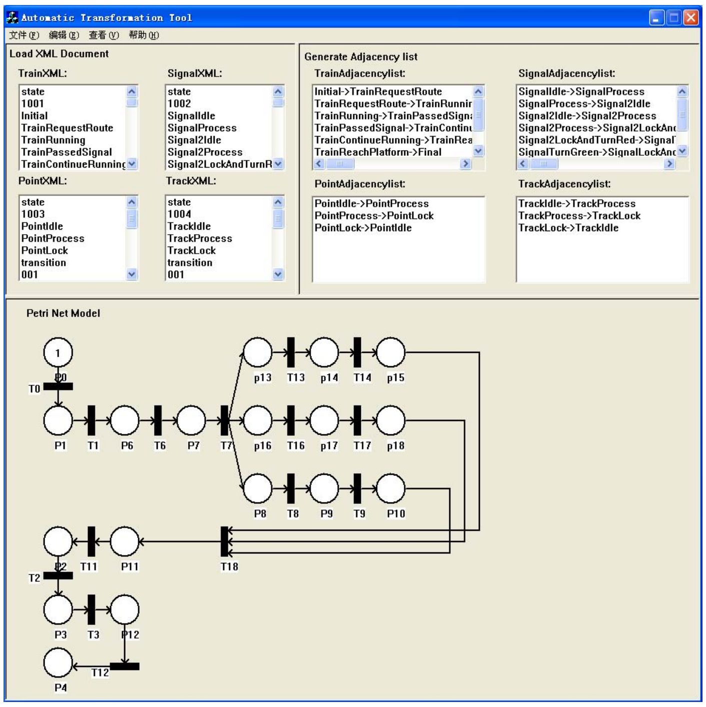


Figure 4. The software block diagram

The whole tool can be divided as five parts. The four parts labeled with dashed box are main contribution of this paper. First, we load XML document to the tool. The XML document which generates from UML model contains the status and jump information between the status of each device. Then, the XML document generates adjacency list with pre-designed data structure. The adjacency list contains the relationship between the status. Finally, unifies the four devices by messages sent to each other, and generates a Petri nets model.

D. *The Automatic Conversion Tool Interface*

The automatic conversion tool interface is shown as Fig. 5, where the transformation process and all steps are displayed in the windows.



TrainXML:

```
state
1001
Initial
TrainRequestRoute
TrainRunning
TrainPassedSignal
TrainContinueRunning
```

SignalXML:

```
state
1002
SignalIdle
SignalProcess
Signal2Idle
Signal2Process
Signal2LockAndTurnR
```

PointXML:

```
state
1003
PointIdle
PointProcess
PointLock
transition
001
```

TrackXML:

```
state
1004
TrackIdle
TrackProcess
TrackLock
transition
001
```

TrainAdjacencylist:

```
Initial->TrainRequestRoute
TrainRequestRoute->TrainRunnir
TrainRunning->TrainPassedSign
TrainPassedSignal->TrainContinu
TrainContinueRunning->TrainRea
TrainReachPlatform->Final
```

SignalAdjacencylist:

```
SignalIdle->SignalProcess
SignalProcess->Signal2Idle
Signal2Idle->Signal2Process
Signal2Process->Signal2LockAnc
Signal2LockAndTurnRed->Signal
SignalTurnGreen->SignalLockAnc
```

PointAdjacencylist:

```
PointIdle->PointProcess
PointProcess->PointLock
PointLock->PointIdle
```

TrackAdjacencylist:

```
TrackIdle->TrackProcess
TrackProcess->TrackLock
TrackLock->TrackIdle
```

Figure 5. Automatic conversion tool

The meanings of some important place and transition of Petri net model are listed in Table 1.

TABLE I. MEANINGS OF THE MAIN PLACES AND TRANSITIONS

Place	Description	Transition	Description
P1	The train requests route.	T1	The train sends route request to signal.
P6	Signal idle.	T7	Signal sends messages to related devices.
P13	Point idle.	T13	Point triggers the messages.
P16	Track idle.	T16	Track triggers the messages.
P8	Protection signal idle.	T8	Protection signal triggers the messages.
P11	Signal turns green.	T18	Related devices trigger signal to turn green.
P3	The train passed signal.	T3	The train sends messages to signal, let it turn red.

V. CONCLUSION

In order to ensure the safety of train operation in station, the correctness of the designed UML model needs to be checked. In this paper we present an automatic transformation tool from UML model to Petri net by using the transformation rules we have defined. Finally, we analyzed and verified the Petri net model. With the proposed tool, it is feasible to check and verify the UML model indirectly.

Our future work will focus on: 1) Semantically, verify equivalency of UML model and Petri net model by counter-example. 2) Add automatic analysis and verification function of Petri net model to the developed tool.

ACKNOWLEDGMENT

The authors wish to acknowledge the support of State Key Laboratory of Rail Traffic Control and Safety (Contract No. RCS2008K008）and Natural Science Basic Research Plan of Shaanxi Province (2009JQ8010), as well as Education Department Research Plan of Shaanxi Province (2010JK698), and The SRF for ROCS, SEM.

REFERENCES

[1] M. Cai, H. Xu and B. Huang, UML Foundation and Rose Modeling Tutorial. The People's Posts and Telecommunications Press, 2008.

[2] T. Xu, O. M. Santos, X. Ge and J. Woodcock, "Use of model transformation for the formal analysis of railway interlocking models," Computers in Railways, vol.7, 2010.

[3] C. Lin, Stochastic Petri nets and Performance Evaluation (In Chinese). Beijing: Tsinghua University Press, 2005.

[4] X. Hei, O. Na, "The scheduling strategy of concurrent request in distributed railway interlocking system," ICIC International 2011, vol. 2, no 1, 2011.

[5] H. Yoshikawa, K. Akama and Hiroshi Mabuchi, "ET-based distributed cooperative system," ICIC International 2010, vol. 5, no. 12, pp. 4655–4666, 2009.

[6] K. F. Jea and J. Y. Wang, "Probabilistic service partition for parallel and distributed computing," ICIC International 2010, vol. 6, No. 9, pp. 3887–3909, 2010.9.

[7] L. Fan, XML Practical Tutorial. The People's Posts and Telecommunications Press, 2009.

[8] X. Hei, et al., "Modeling and analyzing component-based distributed railway interlocking system with petri nets," Institute of Electrical Engineers of Japan (IEEJ) Transactions on Industry, vol. 129, no. 5, pp. 455-461, 2009.