Composable Modeling Method for Generic Test Platform for Cbtc System Based on the Port Object

WAN Yongbing
Shanghai Rail Transit Technical Research Center
Shanghai, China

WANG Daqing
Shanghai Rail Transit Technical Research Center
Shanghai, China

MEI Meng
School of Electronic & Information Engineering
Tongji University
Shanghai, China

***Abstract*—The Communications-based train control(CBTC) system has gradually become the first choice for signal systems of urban mass transit. However, how to guarantee its safety has become a research hotspot in safety fields. The generic test system with high efficiency has become the main means to verify the function and performance of CBTC system. This paper discusses a composable modeling method for the generic test platform for CBTC system based on the port object. This method defines the port object(PO) model as the basic component for composable modeling, verifies its port behavior and generates its compositional properties. Based on the port description and the test environment description, it builds port sets and environment port cluster, respectively. Then it analyzes and extracts possible crosscutting concerns, and finally generates a variable PO component library. It takes the modeling of block port objects in line simulation of generic test platform for CBTC systems as an example to verify the feasibility of the method.**

Keywords—composable modeling; test platform; CBTC; port object; line simulation

I. INTRODUCTION

By allowing trains to operate safely at closer headways, CBTC system can permit more effective utilization of rail transit infrastructure. It has become the preferred standard of urban rail traffic signal system. As a kind of safety critical systems (SCS), however, CBTC system, with the highest safety requirements, is directly responsible for the train operation[1]. Once the system fails, it will lead to a great or even a catastrophic loss of lives, property, and environment.

As one of the important means to improve the safety and assure the quality of the system, testing plays an important guiding role in the process of researching and developing the CBTC system. However, the key issue of the implementation of system testing is how to build a simulation and test environment in accordance with the real operational scenario of CBTC system under test(SUT)[2,3]. Hence, the more attention paid on establishing the test platform for CBTC system, the higher the requirement. The test platform for CBTC system is developing to the direction of network, intelligence and generalization. The test platform is a virtual environment which is used to verify the correctness and reliability of system design. It generally includes the input, processing, validation, and output of signal data, which can not only meet the needs of SUT for functional verification but also some non-functional verification, such as performance test, pressure test and safety test[4].

II. TEST PLATFORM FOR CBTC SYSTEM

A perfect test platform for CBTC system is the combination of simulation technology and testing technology. The simulation activity builds up the desired external scenario and simulates the external environment for SUT. The key to the following testing process is whether the design of a simulation environment is successful or not. And the testing process is a continuation of the simulation activity and its sequential execution[5,6]. A full process of building the test platform for CBTC system is shown in Figure 1.

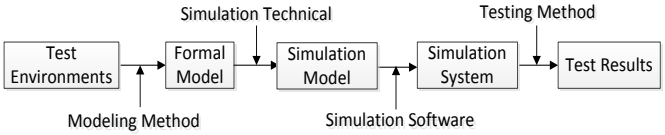

Fig. 1. The process of building the test platform for CBTC system

So far, a widely used and generic test platform for CBTC system hasn't been built. Besides the challenge of developing CBTC system, there are also some problems of building the test platform for CBTC system. One is the architecture of CBTC system differs between the vendors, requiring the test platform makes quick adjustment to adapt to the test of different venders; another is some of the CBTC interfaces differ between the vendors, requiring the test platform has variable interfaces and is easy to switch and roam seamlessly from interfaces to interfaces.

By aiming at establishing a generic test platform for CBTC systems developed by multiple vendors, it focuses on how to build a generic test model for CBTC system, and how to design effective test platform architecture for CBTC system in this paper. Although [7], [8] and [9] have discussed the port-based approach to integrated modeling and simulation of SCS, respectively, they consider the SUT more without the property to safety, real-time and was not suitable to build the test system for SCS. This paper further the evolution towards a seamless integration of simulation for SCS test platform with the idea of a PO.

III. COMPOSABLE MODELING BASED ON THE PORT OBJECT

Unlike traditional closed-loop system, the need for designing and modeling generic test platform for CBTC system

This research was supported by the Shanghai Engineering Research Center of Rail Transit Network Operation number 15DZ2283000.

comes from the external environment[10,11]. Furthermore, the system is designed and modeled merely based on external interface documents and its modeling method should both consider port evolvable and module replaceable, renewable and reusable, which pose a great challenge to the modeling method for the test platform. The concept of Port Object is given below, and a composable modeling method based on Port Object is proposed.

A. Port Object

The Port Object, PO is a kind of novel software abstract that is configurable and replaceable, and the basic unit that generates the system components. As is shown in figure 2, it incorporates the concept of objects and the port automata model used for concurrent processing, and meet the needs of the communication mechanisms in safety-critical areas[12].

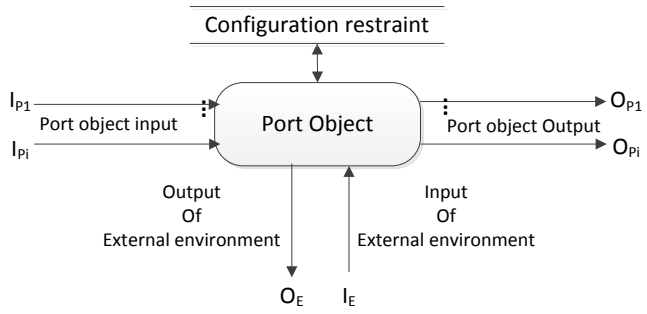

Fig. 2. Port Object

Definition 3.1 (Port Object) is an eight-tuple: $P =< S, S_0, I_P, O_P, I_E, O_E, \tau, R >$, where S is a set of states; S_0 is an initial state and contains at most one state; I_P is a set of inputs from other PO; O_P is a set of outputs to other PO; I_E is the input event from external environment, which can be null, namely $I_E = \varnothing$; O_E is the output event to external environment, which can be null, namely $O_E = \phi$; $\tau \subseteq S \times I_P \times O_P \times I_E \times O_E$ is the transaction relationship; R is the configuration constraints of PO.

To better understand the PO, and further illustrate its properties, it defines the following relations. Defining that the Port Object P ; two states $s_1, s_2 \in S$; $A_P = I_P \cup O_P$, where A_P is the set of events between POs; $A_E = I_E \cup O_E$, where A_E is the set of events between POs and the CBTC system; $A = I_P \cup O_P \cup I_E \cup O_E$, where A is the set of all events; if $a \in A_P$, $b \in A$, then two finite action sequences $\alpha = a_1 a_2 a_3 \cdots a_n \in (A_P)^n$ and $\beta = b_1 b_2 b_3 \cdots b_n \in (A)^n$. Under these conditions, it has:

1) $s_1 \xrightarrow{a} s_2$, if and only if $(s_1, a, s_2) \in \tau$;

2) $s_1 \xrightarrow{\tau} s_2$, if and only if there is an action b which makes $s_1 \xrightarrow{b} s_2$;

3) $s_1 \xrightarrow{\alpha} s_2$, if and only if $s_1 \xrightarrow{a_1} p \xrightarrow{a_2} p \cdots \xrightarrow{a_n} s_2$, especially $s_1 \xrightarrow{\varepsilon} s_1$;

4) $s_1 \xrightarrow{\alpha} p*$, if and only if there is a state $s_2^{'} \in S$ which makes $s_1 \xrightarrow{\alpha} s_2^{'}$;

5) $s_1 \xrightarrow{*} s_2$, if and only if there is a finite action sequence $\gamma \in A$ which makes $s_1 \xrightarrow{\gamma} s_2$;

6) $s_1 \overset{b}{\Rightarrow} s_2$, if and only if $s_1 \overset{\varepsilon}{\Rightarrow} p \xrightarrow{b} p \overset{\varepsilon}{\Rightarrow} s_2$;

7) $s_1 \overset{\varepsilon}{\Rightarrow} s_2$, if and only if $s_1 (\xrightarrow{\tau} p) * s_2$;

8) $s_2 \overset{\beta}{\Rightarrow} s_2$, if and only if $s_1 \overset{b_1}{\Rightarrow} p \overset{b_2}{\Rightarrow} p \cdots \overset{b_n}{\Rightarrow} s_2$.

Where $*$ is the reflexive transitive closure, and the union operation means the combination of two relations[13].

B. Ports' Behavior Description of PO Model

The ports' behavior elements of PO model is described as below: $M := \{m\}$, where m is the input identification, which means messages; $R := \{r\}$, where r is the output identification, which means responses; $C := \{c\}$, where c is the Boolean expression.

The internal port behavior of the PO has four different situations:

1) *Null output*: $f : M \to \varepsilon$ which denoted by $m_1 \mapsto \varepsilon$;

2)*Only one output*: $f : M \to R$ which denoted by $m_2 \mapsto r_1$;

3)*Sequential output*: $f : M \to R^*$, $R^* = \{r_{k_0}, r_{k_1}, \cdots r_{k_n} \mid r_{k_i} \in R \cup \{\varepsilon\}$ which denoted by $m_3 \mapsto r_2 r_3 r_4$, where $k_i \in \{0,1,2,3,\cdots\}$;

4)*Branch output*: $f : C \circ M \to C \circ R^*$ or $f : C' \circ M \to C' \circ R^*$, where $C' := C \cup \{\varepsilon\}$

To ensure security and real-time communication for CBTC test system, it usually needs timeliness and order calibration. Here, it assumes that the PO model in simulation port as a receiver, and then it receives secure data sent by CBTC system periodically. After receiving the secure data, it will carry out a real-time inspection of the time sequence. If it isn't synchronized, then returns the timeliness calibration request and waits to receive the response from timeliness calibration. If synchronized and passed validation, then it shall proceed with receiving secure data[14]. The internal port behavior of this PO

model is shown in Figure 3. The branch output expression is derived as : $c_1 m_4 \mapsto c_1 r_5$, $c_2 m_4 \mapsto c_2 r_6$.

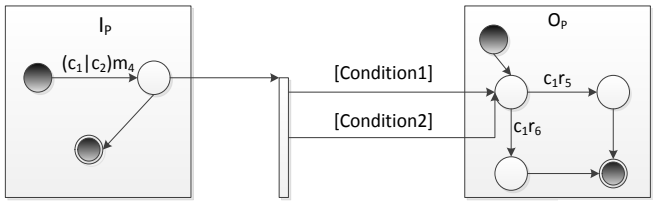

Fig. 3. The internal port behavior of PO mode

C. Configuration Constraint of PO Model

The PO configuration constraint is one of the important parts of the PO model which describes the configuration instruction, external environment interfaces, internal interfaces between objects and properties of PO[15].

- Configuration Content

The structure of PO configuration constraints is shown as Figure 4.

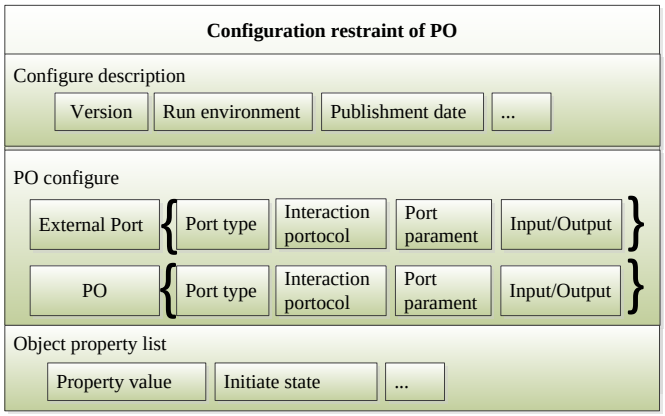

Fig. 4. The structure of PO configuration constraints

1) Configuration Description

The Description of the current configuration constraints, describing the current configuration, such as version, release date and operating environment for PO, mainly facilitates the management of PO, avoids confusion version, as well as the combined behavior or debugging failure due to operating environment errors. Also, configuration description is the main information when components added into the component library.

2) Interface Configuration

Interface configuration comprises the interface configuration with external environment and between POs. The external interface refers to the interfaces with the CBTC system and testers, which includes the interface type, interaction protocols, interface parameters and Input/Output(I/O) description, etc. Depending on its interface mode, the described interface properties shall be different with different modes.

3) Object property table

Object property table describes the relation between interface properties or between properties and values. There can also be a description of the objects' initial states in the table.

- Configuration Description Based on XML

By nesting and referencing the hierarchical relations between the specific elements, the extensive markup language (XML) uses elements and properties to describe data. The XML configuration description is given below based on the structural characteristics of the configuration constraints in PO model[16]. The XML configuration template is described in Figure 5.

```
<?xml version="1.0" encoding="GB2312"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="Configuration">
    </xsd:sequence>
    </xsd:complexType>
    </xsd:element>
    <xsd:complexType name="Description">
    <xsd:sequence>
    <xsd:element ref="Version"/>
    <xsd:element name="Runtime_Environment" type="xsd:string"/>
    <xsd:element name="Release_Date" type="xsd:string" >
    </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="Ports">
    <xsd:sequence>
    <xsd:complexType name="Environment_Ports">
    <xsd:sequence>
    <xsd:element name="Port_Name" type="xsd:string"/>
    <xsd:element name="Port_Type" type="xsd:string"/>
    <xsd:element name="Port_Protocol" type="xsd:string"/>
    <xsd:element name="Port_Output" type="xsd:string"/>
    <xsd:element name="Port_Iutput" type="xsd:string"/>
    <xsd:element name="Port_Property" type="xsd:string"/>
    ......
```

Fig. 5. The XML configuration template

D. Composability of PO Model

Definition 3.3 (Composability of PO) The two PO models *M* and *N*, if the following conditions are met:

$$A_{P_M} \cap A_{P_N} = \phi, I_{P_M} \cap I_{P_N} = \phi, O_{P_M} \cap O_{P_N} = \phi$$

Then M and N are called composable, and are defines as: $shared(M, N) = (I_{P_M} \cap O_{P_N}) \cup (O_{P_M} \cap I_{P_N})$. To give compatibility and replaceability of the PO model, the related definition is given below.

Definition 3.4 (Product of PO) If PO models *M* and *N* are composable, and therefore their product $M \otimes N$ is defined as:

$$S_{M \otimes N} = S_M \times S_N \; ; \; S_{0_{M \otimes N}} = S_{0_M} \times S_{0_N} ;$$

$$I_{P_{M\otimes N}} = I_{P_M} \times I_{P_N} \setminus shared(M,N);$$
$$O_{P_{M\otimes N}} = O_{P_M} \times O_{P_N} \setminus shared(M,N);$$
$$I_{E_{M\otimes N}} = I_{E_M} \times I_{E_N} \setminus shared(M,N);$$
$$O_{E_{M\otimes N}} = O_{E_M} \times O_{E_N} \setminus shared(M,N);$$
$$\tau_{M\otimes N} = \{((v,u),a,(v',u)) \mid (v,a,v') \in \tau_M \wedge a \notin shared(M,N) \wedge u \in S_N\} \cup \{((v,u),a,(v,u')) \mid (u,a,u') \in \tau_N \wedge a \notin shared(M,N) \wedge v \in S_M\} \cup \{(v,a,v') \in \tau_M \wedge (u,a,u') \in \tau_N \wedge a \in shared(M,N)\}$$

Definition 3.5 (Illegal State) If PO models M and N are composable, and therefore the illegal state set $Illegal(M,N) \subseteq (S_M \times S_N)$ in $M \otimes N$ is defined as:

$$Illegal(M,N) = \left\{ (v,u) \in S_M \times S_N \mid \exists a \in shared(M,N) \left(\begin{array}{c} a \in O_{P_M}(v) \wedge a \notin I_{P_N}(u) \\ \vee \\ a \in O_{P_N}(u) \wedge a \notin I_{P_M}(v) \end{array} \right) \right\}$$

Definition 3.6 (Environment of PO) The environment E of the PO model M shall meet the following conditions：1) E and M is composable; 2) E is non-null; 3) $I_{P_E} = O_{P_M}$; 4) $Illegal(M,E) = \phi$.

Definition 3.7 (Legal Environment of PO) If PO models M and N are composable, E is the environment of $M \otimes N$, the state of $Illegal(M,N) \times S_E$ is unreachable in $(P \otimes Q) \otimes E$, and then E is called a legal environment of (M,N),

Definition 3.8 (Compatibility of PO) If PO models M and N are non-null and composable, additionally, there is a legal environment (M,N), and then M and N are called compatible.

Definition 3.9 (Replaceability of PO) N is the PO model, E_N is the environment of N in system S, I is the bind action set of N and environment E_N, $C\ Replaced(N,S)$ is the replace action of N in system S, and therefore $C\ Replaced(N,S) = (P_N / I) / / (P_{E_M} \downarrow \overline{I})$.

IV. Composite-Oriented Modeling Process of PO Model

A. Acquisition of PO model

There are three main sources of demands for the generic test platform for CBTC system: interface document, test environment description and test requirements. The interface document describes all the external interfaces of tested CBTC system, including the interface type, interface parameters and interaction protocols. The test environment description includes all the required external system, infrastructure and environmental constraints. The test requirement mainly describes the needed tests for CBTC system, including the functional test and performance test. The key to composable modeling a generic test platform for CBTC system is doing requirements analysis based on the interface document, test environment description and test requirements, acquiring the PO model and then generating the component library.

The PO model mainly focuses on the port, so does acquiring the PO model. The acquisition process of PO model is described in Figure 6.

Step1: Based on the interface document, the test system is divided into different ports, and generate port sets.

Step2: Based on the test environment description, the environmental resources is it analyzed, then it is added to those covered by the interfaces to the appropriate port set, and the environment port cluster is generated.

Step3: Based on the test requirements, the man-machine interface is extracted and it is added to the appropriate environment port cluster or as an independent environment port.

Step4: After the environment port cluster is analyzed, and the possible crosscutting concerns is extracted, the problem of requirement distraction and requirement entanglement will be resolved.

Step5: Based on the crosscutting concerns, the environment port cluster is divided into different PO.

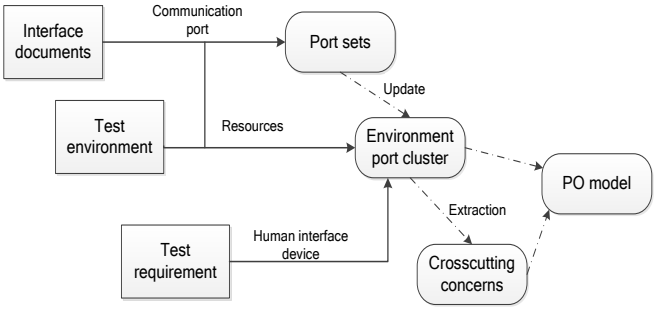

Fig. 6. Process of acquiring the PO model

B. Generate Port Cluster

Description of the test environment comprises all the resource sets needed by the test platform. These resource sets are the state-description objects in PO model. *Resources'* dependency on the interface comprises.

1) Call, resources call interface to communicate in operation.

2) Interrupt, resources interrupt interface communication in operation due to state changes.

3) Modify, resources (interfaces) modify resources (interfaces) parameters or states in initial start when it is needed.

4) Update, resources update their states due to interaction information.

5) Mutually exclusion, this relationship is less likely to appear, which means the interface will not be activated if resources participated in operation at the very beginning.

Define the resource set as *ResourceSet*, which comprises all the resources of the test system. Define the port set as *PortSet*, which comprise all the communication interfaces of the test system. Define the port cluster as *PortCluster*. *Resources'* dependency on the interface denotes as *r*. The way of generating port clusters is given below, as shown in Figure 7.

```
InPut: ResourceSet,PortSet;
OutPut: PortClusters;
PortCluster=ϕ;
RS∈ResourceSet;
Do
{
  PortSet′ = PortSet;
  Do
  { RS′ ∈ResourceSet′;
     PC∈ PortCluster;
      If (RS′ !=RS∨PCrRS=true)
        PortCluster= PortCluster+{ RS′ };
      Endif
        PortSet′ = PortSet′ - {RS};
        Until {PortSet′ =ϕ};
        ResourceSet=ResourceSet-{RS};}
        Until{ ResourceSet=ϕ}
  }
}
```

Fig. 7. The way of generating port clusters

C. Extract Crosscutting Concern

In the built environment port clusters, some resources will appear in different clusters, which are both related to one PO and another, in other words, exist in the intersection of different environment port clusters.

The crosscutting concerns of a resource in two environment port clusters can be written as CC={Re|Re∈*PortCluster*1∧Re∈*PortCluster*2}, where CC is the crosscutt-ing concern, *Re* is the resource and *PortCluster* is the environment port clusters.

In an environment port cluster, if there is a resource which has dependency on several ports, then this crosscutting concern can be expressed as: CC = {*R*e | *R*erPortCluster1 ∧ *R*er PortCluster2∧ ⋯ ∧*R*erPortCluster}.

Here an algorithm used seeking all the possible crosscutting concerns is introduced, where *CS* is the cluster set of all the possible crosscutting concerns, *RS* is the set of all resources, *RRS* is the resource relationship set. *ExistR*(*Re*, *Po*, *r*) is a boolean function, if there is a relationship *r* between the resource Re and the port cluster Po, then its value is true. If not, its value is false. The way of extracting crosscutting concerns is described in Figure 8.

```
CS=ϕ;
RRS=ϕ;
For each RS∈ ResourceSet and Po∈ PortClusters
  If Exist(Re,Po,r)=True
    Then RRS={(Re,Po,r)}∨RRS;
  Endif
For each two (Re1,Po1,r1),(Re2,Po2,r2) ∈ RRS
  If r1=r2& re1=re2
    Then CS={Re1}∨CS
  Endif
```

Fig. 8. The way of extracting crosscutting concerns

D. Generate PO Model

Once the environment port cluster is formed and the crosscutting concerns which causes distraction and entanglement are extracted, the PO model can be generated based on environment port cluster and crosscutting concerns. Define the port object model set as POMS and CS is the cluster set of all the possible crosscutting concerns. The way of generating the PO model is described in Figure 9.

```
POMS=ϕ;
For each PortCluster
  If PortCluster ∉ CS
    Then PortCluster → PO
  Else
     PortClust=PortCluster-{CC}
    AddObjectPort(CC,PortCluster);
    PortCluster → PO
  If CC ∉ POMS
    Then
    AddObjectPort(CC,PortCluster);
    CC → PO
  Endif
    POMS=POMS+{PO}
  Endif
```

Fig. 9. The way of generating the PO mode

V. Instance Analysis

An example of the PO in the line simulation of test platform for CBTC system is presented, and a detailed modeling process is introduced. Some of the descriptions of the three documents are tabulated in Table 1 for further illustration.

Table 1 only tabulates some of the details about the feature points and appropriate PO models can be built based on these details. The modeling process is shown in Figure 10.

1) First of all, the port set which obtained through the port description only describes the ports' <type, constraint>, such as 24V, two relay states (0 and 1) and relay connection method.

TABLE I. PORT DESCRIPTION, ENVIRONMENT DESCRIPTION AND TEST REQUIREMENTS

Number	Document	Content description
1	Port description	Wayside signal system connected to the wayside equipment by 24V relay, 24V relay.
2	Wayside description	Wayside equipment including: section, switch, Signaling, platform screen door, platform emergency stop button, depot/park and drivers protect button, etc. Section includes two states, free and occupation.
3	Test requirement	When the train operation to a track section, CBTC wayside signal equipment should be able to collect this information. When a section is occupation because of a failure, CBTC wayside signal equipment should be able to collect the fault information.

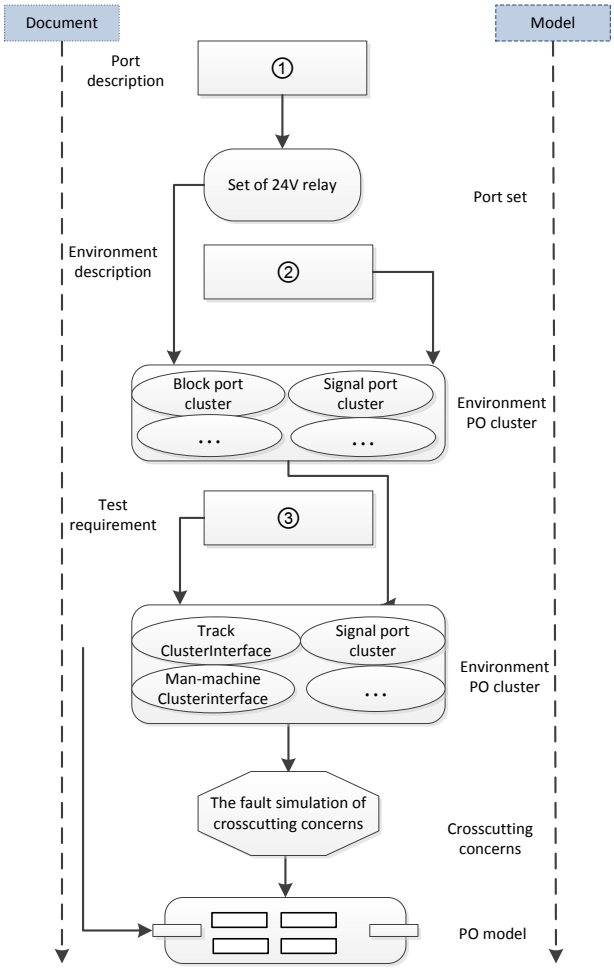

Fig. 10. The modeling process of PO of line simulation

2) After environment description is added, a connection object <port type, port constraint, resource description, dependency> of the 24V relay is generated, where the port type and the port constraints integrate the description of the port set. Resources description includes the section number, the section length and the starting point. Dependency is described as the following. Assume that one section connected with one relay (ignore reacquisition). When the section is occupied and the drive relay is energized, the output is "Occupy", or the section is vacant and the drive relay de-energized, output is "Idle".

3) After test requirements are added, the man-machine interface is added to the environment port cluster. The sector port cluster has two external inputs: the man-machine interface and the train motion simulation PO. Moreover, both of them are POs. The output of the sector port cluster is the environment port of 24V relay.

4) With the further analysis on test requirements, the fault simulation will be appeared in many port clusters. Thus, they are extracted and are named as crosscutting concerns. All of them are POs, their inputs are man-machine port clusters and outputs are other environment port clusters.

5) It is much easier to convert the section port cluster to the section port. Define that the PO's state $S = \{0,1\}$ *where "0" is "Occupied" and "1" is "Idle" and its initial state is 1. Each of the section ports* $I_P = \{I_{P1}, I_{P2}\}$ *has two POs* $I_{P1} = \{Train\_Position\}$, $I_{P2} = \{Fault\_Occupy, Fault\_Idle\}$ *and one environment port* $O_E = \{Occupy, \mathrm{Idle}\}$.

The configuration constraints can be easily generated based on the port constraints, as shown in Figure 11.

```
<Cofiguration>
 <Description>
  <Version>V1.0</Version>
  <Runtime_ Environment>WondowsXP,VS2010,
        SQL Server</ Runtime_ Environment>
  < Release_Date>2013-09-01</ Release_Date>
 </ Description>
 <Ports>
  < Environment_Ports>
    < Port_Name>XL_LS_Port< /Port_Name>
    < Port_Output>Occupy/Idle</ Port_Output>
    <Port_Property>24V</ Port_Output>
    <Port_Count>128</Port_Count>
 </ Environment_Ports>
   < Object_Ports>
    < Port_Name>Fault_Port< /Port_Name>
    < Port_ Iutput >Fault_Occupy/Fault_Idle</ Port_ Iutput >
    < Port_ Output >Fault_Occupy/Fault_Idle</ Port_ Output>
    <Port_ Protocol > Shared Memory </ Port_ Protocol >
 </ Object_Ports>
   < Object_Ports>
    < Port_Name>Train_Port< /Port_Name>
    < Port_ Iutput >Train_Position</ Port_ Iutput >
    < Port_ Output >Occupy/ Idle</ Port_ Output >
    <Port_ Protocol > Shared Memory </ Port_ Protocol >
 </ Object_Ports>
 </Ports>
< Property_List>
<Track>
   <Name>T2115</Name>
   <ID>0x1C000009</ID>
   <StartE>50</DeviceWidth>
   <EndE>360</EndE>
 </Track>
...
 <Track>
   <Name>T2116</Name>
   <ID>0x1C000010</ID>
   <StartE>360</DeviceWidth>
   <EndE>558</EndE>
 </Track>
< /Property_List>
```

Fig. 11. The configuration constraints

VI. CONCLUSION

The test and verification of CBTC system has become one of the important means of ensuring system security. While building the generic test platform for CBTC system is the prerequisite for test and verification. Nowadays, CBTC system having a more complex architecture, more diversified interfaces and more often upgrading, which poses a greater challenge to the modeling and simulation technology. In this paper, some preliminary study on how to build and realize the generic test platform for CBTC system will be done, a composable modeling method for CBTC simulation and test system based on the port object will be suggested, and integral composite-oriented modeling process as well as verify the feasibility of the method will be illustrated

In this paper, a meaningful technology method are put forward for resolving the problems in the process of development for test platform of CBTC system, but there are still many works to be further researches, such as another important aspects with the combination of system modeling, namely the assembly based on component technology. In view of the PO model based assembly technology, a best method of assembly needs further discussion and analysis.

REFERENCES

[1] Z. Yujun, X. Zhongwei, M. Meng. “Port-based Composable Modeling and Simulation for Safety Critical System Testbed”. Lecture Notes in Computer Science, Vol.7529, pp.51~58, 2012.

[2] A. Speck, E. Pulvermuller, M. Jerger, et al. Component Composition Validation. International Journal of Applied Mathematics and Computer Science, Vol.12, pp.581-589 , 2002.

[3] G.V. Bochmann, S. Haar, C. Jard, et al. Testing Systems Specified as Partial Order Input/Output Automata.Testing of Software and Communicating Systems. Springer Berlin Heidelberg, Vol.5, pp.169-183 , 2008.

[4] L. Bin, W. Xin, F. Hernan, M. Antonello. A Low-Cost Real-time Hardware-in-the-loop Testing Approach of Power Electronics Controls. IEEE Transactions on Industrial Electronics, Vol.57, pp.919-931 , 2007.

[5] L. Chen, W.P. Wang, Y.F. Zhu. Research on SEB Composable Modeling methodology for system-of-system Combat Simulation. Journal of System Simulaiton, Vol.19, pp.644-656, 2007.

[6] G.Y. Wang, Q.W. Hu, W. Liu. Study on Composable Modeling and Simulating technology for equipment battlefield damage. Binggong Xuebao/Acta Armaentarii, Vol.10, pp.1266-1275, 2012 .

[7] C.B. Peter. Port-based Modeling of Mechatronic Systems. Mathematics and Computers in Simulation, Vol.66, pp.99-127 , 2004.

[8] Y.L. Lei, Q. Li, F. Yang, W.P. Wang. A Composable modeling framework for weapon system effectiveness simulation. System Engineering Theory and Practive, Vol.11, pp.2954-2966, 2013.

[9] C. Yuan, N. Ru, T.H. Xu, T.Tang. Wireless Test Platform of Communication based Train Control(CBTC) System in Urban Mass Transit. Proc. of the 2007 IEEE International Conference on Vehicular Electronics and Safety, pp.39-14, 2008.

[10] D.I. August, S. Malik, L.S Peh, P. Willmann. Achieving Structural and Composable modeling of Complex Systems. International Journal of Parallel Programming, Vol.2, pp.81-101, 2005.

[11] C.J.J. Paredis, R. Sinha, P.K. Khosla. Composable Models for Simulation-based design. Engineering with Computers, Vol.2, pp.112-128 , 2001.

[12] C. Berger, M. Chaudron, R. Heldal, O. Landsiedel. Model-based, Composable simulation for the development of autonomous miniature vehicles. Simulation Series, Vol.4, pp.118-125, 2013.

[13] Z. Zhu, Y.L. Lei, Z. Ning, Y.F. Zhu. Composable modeling frameworks for networked air and missile defense systems. Journal of National Univeristy of Defense Technology, Vol.5, pp.186-192 , 2014.

[14] X.X. Chen, D. Wang, H. Huang. Design of Simulation Testing Platform for CBTC System. Railway Computer Application, Vol.8, pp.50-56, 2011.

[15] N.N. Chen, J.Xu, X.Z. Yin. Desing and Implementation of Eurobalise Simulation Test Platform for Urban Transit CBTC System. Railway Computer Application, Vol.12, pp.59-61, 2013.

[16] R. Srinon, S. Ramakrishnan. Distributed Simulation Modeling for Manufacturing Systems Design Using XML. Proc. 18th International Conference on System Engineering, pp.395-400, 2005.