

# A Component-Based Simulation Model

Weigong Lv, Fanchao Meng, Ce Zhang, Zhipeng Chen, Chang Yuan, Kun Wan, Yiran Zhao and Jianan Jiang  
School of Computer Science and Technology Harbin Institute of Technology at Weihai, Weihai, China

**Abstract**—The implementation of simulation system requires the support of effective simulation models. In this paper a new simulation model based on the low granularity is proposed. Starting from the quadruple parts composing the component, the component can be divided as the static component, non-trigger type component and trigger type component. And the relationships in components, among components and the corresponding time constraint are described formally. Under the basis a new simulation describing language is proposed, which can describe the action logic of and among components explicitly, form component models and then develop a new rapid constructing mode of the simulation system. Meanwhile, it can be used to construct flexible and effective simulation systems of different coupling degree.

**Keywords**—component; simulation; system; simulation describing languages

## I. INTRODUCTION

With the development of computer, system engineering and other related subjects, simulation technologies have permeated into lots of areas. More and more simulation systems are constructed. The present simulation system usually apply some composable simulation and service-oriented simulation methods. The composable simulation method is supposed to realize the rapid combination of models and simulations<sup>[1]</sup>, featuring highly customization and mutual operation. Most of its simulation components are tight coupling. The main idea of service-oriented simulation method is to build a distributed component model and wrap different units of application as service. And it is suitable for the loosing coupling simulation system<sup>[2]</sup>.

From the aspects of recent developments in simulation system, the research direction of composable simulation focuses on problems of the composability in simulation components[3][4][5], but the service-oriented simulation pays more attention to how to develop a simulation model framework by adding service to the components[6][7]. Most of its research is to describe the simulation system from the abstract level. Although the constructing foundation of simulation system is the simulation component from the low level, there is little research from the aspects of components.

The languages which are used for the simulation system develop from the advanced coding languages like C++ and Java to the professional simulation languages such as GPSS[8], SLAM[9] and DYNAMO[10]. The professional simulation languages solves the mutual operation problems, realizing the integration among heterogeneous models, but they cannot fully support the description of models[11][12]. Otherwise, the common modeling languages like UML could provide full

support, but they lack in the specific support for simulation systems, so they cannot be used for the utility.

In this paper, a component-based simulation language based on the low granularity is proposed, it can support the flow control based on the components, describe the combining logic of components explicitly and form component models. The language provides the explicit description of the concurrency in thread level and supports construction modes in flexible and effective simulation systems of different coupling degree. And it was put into use in PetroChina's compressor simulation system[13].

## II. SIMULATION MODEL DESCRIPTION

### A. Component

The component is regarded as the core element of the simulation design. It is responsible for some important jobs such as the construction of different parts in simulation system and the change of workflow. Its performance will affect the later work directly, next according the features of components we will define an abstract logical component.

**Definition 1:** Suppose that there is one component named Component, then it can be described abstractly as quadruple parts like below:

Component= $\langle$ Form, Characteristic, Action, Event $\rangle$

The Form is the external manifestation of the component, also called the external character, the Characteristic is the internal character of the component, the Action represents the behavior of the component, the Event is the event which components could respond to.

Starting from the abstract definition of components, one component can be concrete. The external manifestation Form can be divide into a lot of external characters, represented by an external characteristic set named F. As for the Characteristic, it can be divided into internal characters, represented by an external characteristic set named C, and one Action can be described as an action flow with the constraint condition, and one Action corresponds to one action set or an action flow set, represented by A. For the events which components could respond to, they can be simply concluded as one event set E.

**Definition 2:** Suppose that there is one component named Component, then it can be defined as a set of 4 subsets:

Component= $\{F, C, A, E\}$

After analysis of the four subsets of the component, you can find that a component-based simulation model contains two closely related parts, one is the part in the component, the other is the part among the components. The part in the component could be reflected using the relationship between F and C,

which is also called form-featured model. The other part could be revealed by the action A triggered by the E, which is called action event model.

### B. Form Feature Model

The Form feature model of component reflects the relationship between inner and outer characteristics, which is the fundamental of the component's integrity and independent. Such relationship should be bound into the inner place of the component through the process of component building. To build the relationship, as well as establishing the right form feature model, is the first step to build the component.

Definition 3: Given a component contains  $n$  outer characteristics as  $F=\{F_1, F_2, \dots, F_{n-1}, F_n\}$  and  $m$  inner characteristics as  $C=\{C_1, C_2, \dots, C_{m-1}, C_m\}$ , then the form feature of the component is the relationship that any state in  $F$  has and only has one counterpart in  $C$ .

With the form feature model, we could then build components based on the logical relationships. Set  $C$  might have different element counts with set  $F$ , so the logical relationship might be very complicated, which would then affect the complexity of the component.

However, through the real process of component building, we found that the relationship between set  $F$  and set  $C$  might also be as simple as one to one mapping. For example, given a component that has an outer characteristic height and an inner characteristic degree, then the relationship between these two characteristics is obviously a one to one map. If each element in  $F$  and  $C$  has such kind of relationship, then there is a one to one map between  $F$  and  $C$ .

Definition 4: Given a set  $F$ , which has  $n$  outer characteristics, and a set  $C$ , if we could build a one to one map between  $F$  and  $C$  (which means that for all  $1 \leq i \leq n$ , we could set up a relationship between  $F_i$  and  $C_i$ ), we call it a simple form feature model.

The construction of component would be easy if we could build up a simple form feature model, Figure 1 show the maps of normal feature form and simple feature form.

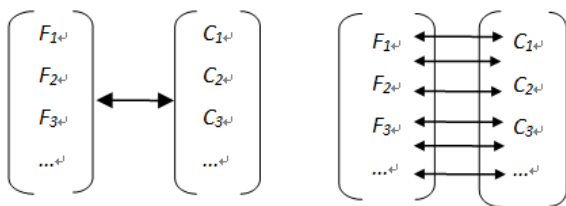


FIGURE I. NORMAL FEATURE FORM MODEL AND SIMPLE FEATURE FORM MODEL

### C. Event Behavior Model

The status of simulation system is determined by the component's status, which is represented by set  $F$  and set  $C$ . As we discussed before, set  $F$  is the outer feature form, and for a determined status of set  $C$ , we have and only have one corresponding status of set  $F$ , so the component's status could be represented by the status of set  $C$ , and the status of a simulation system is ultimately determined by the statuses of all the components it contains.

The changes of a component's status need to be triggered by trigger conditions, which are called events. These events together constitute the event set  $E$  as we mentioned before. The component must have the ability to respond events. When an event is responded, a series of actions of components will be triggered, and the simulation system will transfer from one status to another just like the real system. The actions triggered by one event are called a behavior. For each of the events, a component has a determined behavior. So a component must have a set of behaviors. When the component and all the behavior logic are determined, the simulation is then accomplished.

Compared with the form feature model, the event behavior model is more sophisticated, thus it becomes the nexus of simulation studying. In this model, the only way to relate one component to another is to trigger the behavior through events. The behavior will accomplish the relationship. The composable simulation mentioned before is to combine the components through behaviors to accomplish simulation tasks. However, in the service oriented simulation, events are all kinds of service requests, and behaviors could be regarded as a process of implementation.

As for the action flow of the behavior, this model offers four basic process, through which more complicated processes could be made up. These four process are shown in Figure 2, they are parallel sequence, serial sequence, serial branch and serial circle.

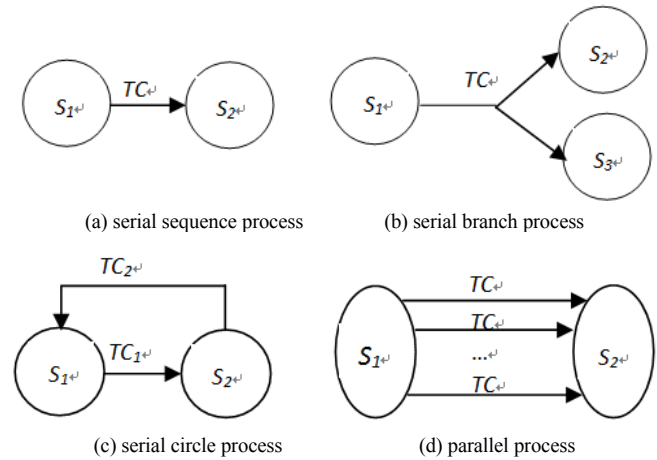


FIGURE II. FOUR BASIC PROCESSES OF EVENTS

Figure 2(a) describes a process a serial sequence execution. At the beginning status (the left-most one), the status set is  $S_1$ , and with the action of event  $TC$  it transforms into set  $S_2$ ; Figure 2(b) indicates the process of serial branch execution. The initial status is  $S_1$ , with the action of  $TC$ , it transforms into  $S_2$  or  $S_3$ ; Figure 2(c) indicates the process of a serial circle execution, the initial status is  $S_1$ , with the action of  $TC_1$  it turns to be  $S_2$ , and through event  $TC_2$ , it turns back into  $S_1$ ; Figure 2(d) indicates a process of parallel execution, to begin with the initial status  $S_1$ , it transforms into status  $S_2$  through 2 or more than 2 parallel actions.

Corresponding to the work logic of simulation system, any event could be described as a serial of transformation of status.

Given the initial status  $S_1$ , the transformation process could be described as below:

$$S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow \dots \rightarrow S_{n-2} \rightarrow S_{n-1} \rightarrow S_n$$

#### D. Time Constrains

There are two kinds of time constrain, one for events, another one for behaviors. The time constrains for events mean that the trigger conditions are based on time. The time constrains for behaviors mean that the action procedure should be constrained by time, and time should be a kind of constrain factors to make sure that the behavior is correct.

Definition 5: Under specific time criterion and time units, we define time point as time remark explicitly.

Definition 6: We define the time constrains of simulation models to be the set of time remarks through which we represent all the time conditions of event triggers and behavior constrains. Given  $p$  time remarks, the time constrain could be represented as the following set:

$$T = \{T_1, T_2, \dots, T_{p-1}, T_p\}$$

### III. THE CONSTRUCTION OF SIMULATION SYSTEM

#### A. The Classification of Simulation Components

According to the different role of components in simulation system, we classify the components into three major categories. The first category is static component, which doesn't change in the simulation system; the second is none-trigger components, the none-static components that do not participate in the event triggers but initiate the behavior; the third category is trigger components that participate in the event triggers and initiate behaviors.

As a result, given a component set  $I$ , we could divide it into three sub sets. The description of them is as below:

$$I = \{S, P, D\}$$

$S$  represents the static components' set, which is static in the process of simulation. Such as the texts for hints and the graphic components that do not change throughout the hole process. The designing of these components do not need to concern about the interaction between components, or the inner characteristics. What need to be concerned about is just the outer form (F we mentioned before) of the components.

Set  $P$  includes those elements that do not participate in the event trigger or initiate the behaviors. Such as the real-time status of some automated valve and thermometers, the changes of them are triggered by the low layer trigger components, which is indirect. What set  $P$  concerns is the form feature model of the components, which includes  $F$  and  $C$  in the component's elements.

$S$  indicates the trigger components' set, which describes the variable elements of the simulation objects, such as the button and the textbox. Events could trigger the behaviors of such components, change the status of its own or others, and ultimately change the status of the hole systems. The

components of this category includes at least elements  $A$  and  $E$ , and all of the four elements are also possible to be included together.

In the real simulation system, some components are physical entities, such as simulated compressor consoles. Those physical entities need a component (usually OPC component) to drive and control [14][15]; The time constrains could be achieved through a time component, which could set the time criteria and units, make time remarks of the time constrains, and trigger the time based events according to time remarks or constrain behaviors with time remarks. The two components are special kinds of trigger components.

#### B. Event

As mentioned before, only trigger components are related to event sets' designation. The occurrence of an event can change component's state, and then change simulation system's state. Therefore, it is necessary to analyze the reason of trigger event. The following are 3 types of events.

The first type is interactive event, which implies environment's trigger behavior to components. For example, a button can be a component, clicking the button can cause an event, and change corresponding components' states. Component and environment cause an interactive behavior in this process; The second type is transfer event, it is triggered by other events.

Events can cause behaviors, and actions which consist behaviors can be "trigger event". The first type interactive event can trigger other components' transfer event by causing it action, timer components can trigger components' event on fixed time. For instance, the system collects temperature and pressure data from control cabinet by PLC on fixed time, sequential execution of system start-up, can be triggered by timer, and presented by components in set  $P$  and  $D$ , trigger behavior process.

The first type has other special situation, that is a component build and destroy event. For example, a simulation system can have a primary component (can be triggered), when it is built, all static components' state are set; when it is destroyed, it can change other components' state.

In sum, building and running simulation model is actually building simulation component set's state transfer model. Its prerequisite is to build components' event model. Components in set  $S$  don't need event support in model, components in set  $P$  are the same, their state can be changed by other components' event. So we only need to build event model for components in set  $D$ .

Generally speaking, we can determine the initial event, destroy event and time-based event for components, and then complete other events' construction. Every components can build various events. In real system, events' type is much less due to the small granularity components.

For every event, its response is a series of changes in the status of components which conform to the working logic of the simulation system. The main task of establishing simulation model is to extract the logic relation of event model, the logic

relation is applied to the components set in the simulation model, and then shows a series of state changes.

For a system to be simulated, if it is a relatively independent system, this model can construct the corresponding component set and the event based state transition set, so as to complete the simulation system; For large systems, there may be a number of relatively independent subsystems to be simulated, each has its own set of components and event based state transition set, the model needs to be used many times to complete the simulation.

### C. Process Simulation Description Language

From the analysis of simulation system above, we can know that we need to operate the designed components in some way after the designation of components. This paper defines a process description language SL-1. We use this language and take components as unit to describe the working process of simulation SYSTEM.

```

<SL-1 program> ::= CSS <SL-1 program name> {
    [<public variable definition>;]
    [<procedure definition>;]
    <component set definition>; }
<component set definition> ::=
    <component definition>; [<component definition>;]
<component definition> ::= COMPONENT <component name> {
    <attribute set definition>;
    [<event set definition>;] }
<attribute set definition> ::=
    <attribute definition>; [<attribute definition>;]
<attribute definition> ::= <attribute type> <attribute name>;
<event set definition> ::= EVENT <event name> {
    [<private variable definition>;]
    <statement definition>; [<statement definition>;] }
<statement definition> ::= [<if statement definition>;]
    [<while statement definition>;]
    [<parallel statement definition>;]
    [<procedure call statement>;]
    [<statement>;]
<if statement definition> ::= if <condition expression> {
    <statement>; [<statement>;] }
<while statement definition> ::=
while <condition expression> {
    <statement>; [<statement>;] }
    do { <statement>; [<statement>;] }
    while <condition expression>
<parallel statement definition> ::=
parallel <procedure name> [<procedure name>]
<procedure call statement> ::= call <procedure name>
<procedure definition> ::= PROCEDURE <procedure name> {
    [<private variable definition>;]
    <statement definition> }
<statement> ::=
    <component name>.<attribute name>|<variable name>
    =<numerical expression>;<condition expression>;

```

FIGURE III. BNF NORMATIVE DEFINITION OF PROCESS DESCRIPTION LANGUAGE SL-1

Same as other languages, SL-1 language has a normative grammatical and semantic description. The program written by this language is called SL-1 program, furthermore, the simulation system generated by SL-1 program is called CSS(Component Simulation System). Fig 3 is BNF normative definition of process description language SL-1, it uses context-free grammar.

Due to space constraints, some details, such as the corresponding legal symbols of various names, expression operators, variables and attribute types, are omitted. Public variable definition and private variable definition which are similar to public and local variables in other languages are not illustrated at length.

SL-1 doesn't have an instruction of F, which is one of the three features of components. This is because components' external form is built based on target platform development environment, and target platform development environment always has the needed components, such as buttons, timers and driver hardware' OPC components, only a few components needs to be designed by developers. Using components designed in target platform can accelerate development greatly.

SL-1 language has to illustrate all components' internal characteristics except static components, the definition are in attribute set definition; Event set definition defines trigger components' event behavior features. All components' event behavior features form the logic among components, this logic follow the conduct code of simulation system, realizing system's dynamic simulation.

SL-1 language operates two kinds of data – attributions and variables, which are represented as <component name>.<attribute name> and <variable name>, subroutine definition are <procedure definition>. BNF offers branch, loop and parallel support, their definitions are <if statement definition>, <while statement definition> and <parallel statement definition>. Many processes in parallel support definitions represent simulation's parallel behaviors.

Designing separate simulation description language to separate components' behavior descriptions and interpreter programs can better improve portability and design flexibility.

### D. Some Design Discipline

A The design disciplines in this model are low granularity, reusability and tight coupling. Component operation with low granularity can guarantee the flexibility of component model, reusability makes model more widely used, and functional independent components with tight coupling are more efficient. In the case of valve and pipe pressure value's simulation, valve and pipe pressure can be seen as two components not as one. So in this model, we use two components with low granularity so that the connection between two components can be described in upper language.

Those components separated in logic and function should be divided separately, which make a solid foundation for using language to operate components. Meanwhile, we should avoid components being scattered too much, increasing the workload in the production and extraction component set, the loss outweighs the gain. The two is a contradictory process, which needs to be measured according to the actual situation, and then find the balance point.

In the divided components, the building rules of S set element are brief and simple, which only use F in simulation model. Non-trigger P components use F, A, these components are generally designed to change F, usually we use different A to distinguish effects on F, i.e. using A to describe F's change;



simple trigger type P components use F, A, E all 3 aspects, the core design of this kind of components are usually in E, E induces internal action coupling in internal components and external action can be described by SL-1 language; entity components generally use A and E, it focus on A's change, physical entities are operated, induce E, and E characterize the physical state by A, the induced actions can also be described by SL-1 language.

#### E. Actual Construction of Simulation System

The construction process of simulation system in simulation model is shown in Figure 4.

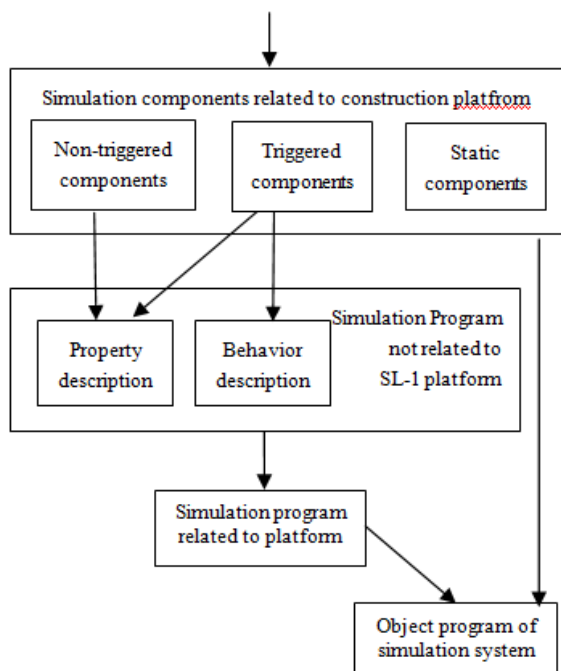


FIGURE IV. THE CONSTRUCTION PROCESS OF SIMULATION SYSTEM

At first, a platform related simulation system development environment needs to be chosen to design components. Design of static components are easiest, non-trigger components' construction is the logic relation between components' internal attributions and external form. Their attributions can be altered passively, and then change their external form. Trigger components need, event trigger behaviors aren't need definition, which are defined in upper description language.

There are 2 special trigger components whose external forms are not important, one is timer component, which is defined to describe the time constraint of simulation system. Another is hardware device driver component, OPC component is often used to complete this work, external device can trigger it to work, and its attribution determines device's control flow, state flow and data flow.

After the construction of components, we need to write SL-1 codes on platform independent layer using SL-1 language. We don't need to describe static components for SL-1 program, only need to describe non trigger components and trigger components, the external form of the components are also not

described in the program, what needs describing are two components' attributions and different trigger event behavior corresponding to triggered components, they have to characterize of relationship between components in the simulation system accurately.

SL-1 is a platform independent language, when used in real application, it needs to be transformed into a form that can be recognized by platform-dependent simulation system. So an interpreter program is needed to transform SL-1 language into a platform-dependent program which can operate the components defined earlier.

In the construction of Petro China compressor simulation system process, we choose windows platform visualization VB development environment, VB offers a large amount of custom controls, interpreter program transform SL-1 codes into VB codes. We finally complete the Petro China compressor simulation system by using simulation model and interpreter program, and it runs well. Figure 5 shows simulation system platform and simulated compressor system interface.

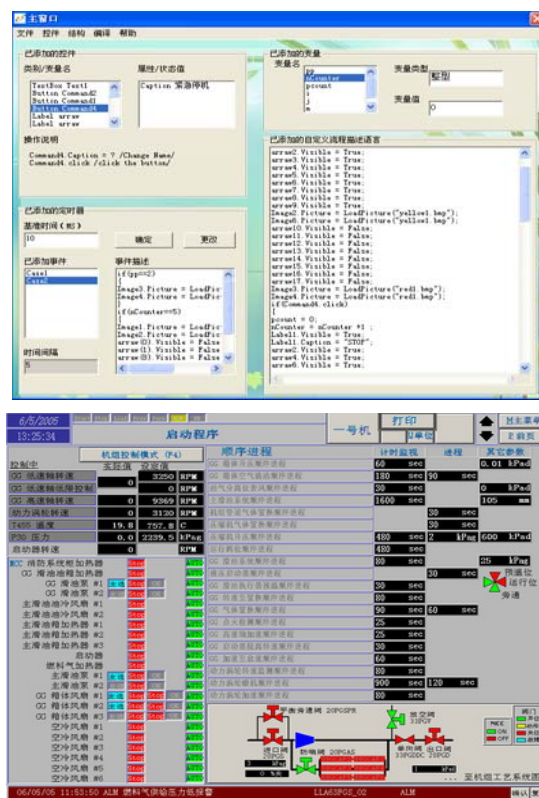


FIGURE V. THE SYSTEM UI OF THE SIMULATION PLATFORM AND THE SIMULATED COMPRESSOR

#### IV. CONCLUSION

With the development of simulation system, more and more efficient simulation models are needed. This paper present a components based simulation model, which formally describes the relationship in and among components, and put up with a process description language based on this, thus form a pattern to build simulation system quickly. Model has the following features:

## REFERENCES

- [1] Mikel D Petty, Eric W Weisel, A Composability Lexicon, Proceedings of the Spring 2003 Simulation Interoperability Workshop.2003. No. 023.
- [2] JIA Li, ZHANG He-ming. Research on Service Oriented Distributed M&S Framework[J]. Journal of System Simulation,2007,19 (20) :4680-4687.
- [3] Szabo, Claudia; Teo, Yong Meng. On Syntactic Composability and Model Reuse. First Asia International Conference on Modelling & Simulation.2007: 230 – 237.
- [4] Yifan Zhu; Qun Li; Chao Chen. Context-driven testing approach for composability of simulation model. Asia Simulation Conference - 7th International Conference on System Simulation and Scientific Computing.2008:89-93.
- [5] SONG Li-li, ZHANG Can, LI Qun, WANG Wei-ping.SOA-based Approach for Composable Simulation.Journal of System Simulation. 2009:3626-3630.
- [6] Yifan Zhu; Weiping Wang; Dongxiang Zhou. Conceptual framework of composable simulation using multilevel model specification for complex systems. Asia Simulation Conference - 7th International Conference on System Simulation and Scientific Computing.2008:84-88.
- [7] Qiang He; Yong Peng; Ming-Xin Zhang; Jian-guo Hao. A composable modeling framework base-on BOM in war simulation. 2011 International Conference on Computer Science and Service System.2011:1900-1904.
- [8] Kwak N K, Schmitz H H. The GPSS Simulation of Scheduling Policies for Surgical Patients[J]. Management Science, 1976, 22(9):982-989.
- [9] Montemerlo M S. Fastslam: a factored solution to the simultaneous localization and mapping problem with unknown data association[M]. Carnegie Mellon University, 2003.
- [10] Eastwood E. Industrial Dynamics and the Use of the Dynamo Language[J]. Operational Research Quarterly (1970-1977), 1970, 21(3):365-365.
- [11] Weisel E W, Petty M D, Mielke R R. A Survey of Engineering Approaches to Composability[C]. Proceedings of the Spring Simulation Interoperability Workshop. Arlington, VA, 2004.
- [12] Petty M D, Weisel E W, Mielke R R. A Formal Approach to Composability[C]. Proceedings of the Interservice Industry Training, Simulation and Education Conference Orlando, FL, 2003.
- [13] CAI Deyu, WANG Shengyue. Design of Simulation Training System for Gas Transmission Pipeline Compressor Unit [J]. CONTROL & AUTOMATION, , 2010, 26(25):169-171.
- [14] OPC Foundation. OPC Data Access Custom interface Standard Version3.00[S]. 2000, 03, 04.
- [15] Lucas, K.Yuan, Boone. Logic design for printability using OPC methods[J]. Design & Test of Computers, IEEE, 2006.

(1) It describes the constitution of simulation components, presents complete components' four elements pattern, its external characteristic is form, internal characteristic is feature, in this way, components' internal relation can be described as internal and external connection, the two tight coupling is exact independent components construction's essence. According to simulation components' events behavior model, relationship between components is defined, events explicitly describe components and external liaison channels, and actions present a process to achieve multiple actions, and the process is further standardized and decomposed into parallel, serial, serial branche and circular.

(2) Simulation components are classified as static components and non triggered components and triggered components. Explicit static components are focus on form, non triggered components concerns in form and characteristics, and triggered components focus on all four element characteristics. Those make the tasks and principles of the decomposition of the components in the process of construction in simulation system clear, and provide a framework for various components to achieve embodiment .

(3) Time constraints on the relationships between components are described. The time marks, as the specific constraints, are applied to the behavior of the component processes, time constraints, as a special component shown finally, their task is to set required time markers in simulation in a certain time benchmark. The timing constraint realization method explicitly separate the restriction time point mark and a trigger behavior, clear the time constraint logic, simplifies the realization.

(4) For the completion of the simulation component model, we define a components-oriented simulation description language, which can manipulate characteristics of abstract components, describe logic relation and time constraints among components, build and correct simulation model between components flexibly. The simulation model is platform independent, and it needs to be converted to the component operating language of the target platform when it is applied to the specific platform, and the conversion is not difficult because of the similar characteristics of the components operation.

As a general framework for simulation system development, there are still a lot to be improved and expanded in this model. Firstly, component implementation is platform dependent, which depends on development environment the platform provided. Realization of component model is restricted by the development environment, making it unable to accurately assess the workload and difficulty on different component development platforms; secondly, process description language is platform independent, different conversion procedures are needed corresponding to different platforms. To achieve cross platform support, there is still a lot of work to do.

## ACKNOWLEDGMENTS

This paper is Supported by the National Key Technology R&D Program of China (2014BAF07B02) and the Shandong Province Science and Technology Program of China (No.2015ZRA10032).