Visual Integration Approach for Parallel Discrete-Event Simulation Applications Based On Object-Interaction Graph

Y. Liu, Y.P. Yao, W.J. Tang, F. Zhu, F. Yao
College of Information System and Management, National University of Defense Technology, Changsha 410073, China

***Abstract*—Visual Development Technology has been more and more attractive in the field of Modeling and Simulation with its intuitive features. As the scale of Parallel Discrete Event Simulation (PDES) enlarges, Researchers are becoming focused on how to integrate existing simulation entity models into PDES Application efficiently, by adopting the intuitive visualization technology. In this paper, the research status of the PDES visualization integration technology is discussed; and the authors have found the limitations of PDES visual integration technology, like the complex and not intuitive process of integration, the difficulty in reusing "basic integration module", and the difficulty in adjusting the application structure. Then a Visual integration approach for PDES Application based on Object-Interaction Graph was proposed for the "object interaction" features of PDES. This approach provides the developers with a more natural viewing angle with its more understandable graphical way, so that we can develop PDES Application more easily. The more flexible structure of PDES makes it possible to quickly adapt to the rapidly changing requirements of the simulation.**

Keywords-parallel discrete event simulation; visualization; simulation objects; Object-Interaction Graph; simulation object event

I. INTRODUCTION

Along with the rapid development of computer simulation technology, the PDES technology has been widely used in the Simulation of large-scale Discrete-Event System [1], such as the national energy planning, population management, traffic management, and combat systems. It is becoming a focus of study on how to integrate existing simulation entity models into PDES Application efficiently based on the large number of reusable simulation model.

Large-scale parallel discrete event system often contains a mass of entities among which there are numerous intricate interactions. Modeling and Simulation for those systems is achieved by modeling the entities and the interactions among them to a great extent. Currently, modeling theory and technology for simulation entities has tended to perfect, and a large number of reusable entities have been built [2-11], but the theory and technology research of application integration is relatively less. Developers traditionally integrate entities into PDES Application step by step from the bottom to top by writing code manually with the General Programming Language. The disadvantage here is obvious: high technical threshold, poor intuition, low efficiency of integration, inflexible PDES Application, and the difficulty of modification. With the expansion of PDES Application scale, traditional integration approach becomes increasingly powerless. In this case, the visual PDES integration technology has emerged. It gradually packages all the details irrelevant to the application logic, and converts graphical application model to specific language automatically, thus lowering the threshold for application integration and improving the behaviour level of developers, and making the dream of "designing is coding" come true. Therefore, the visual PDES integration approach has become the development trend of current integration technologies for simulation application. In this paper, we first discuss the limitations of the existing visualization integration technologies, and then analyze the advantages of PDES Application integration from an object perspective based on the feature of "objects-interaction" of PDES.

The structure of this paper is as follows. Section 2 presents the research status of visual integration technology. Section 3, we analyze the objects interaction features of PDES, then give the definition of "Objects-integration Graph" as well as some technique problems in implementing focused on here. We introduce the Visual integration approach for PDES Application based on Object-Interaction Graph in Section 4, with a demo shown here. Finally, we draw the conclusion and point out possible future works in Section 5.

II. RELATED WORKS

There are already a variety of PDES visual integration technologies, such as FSA, Event Graph, and ACD. These technologies are widely used in various fields of simulation and modeling with their good modeling characteristics. However, they are not the simple map of the real world, so the developers cannot observe the PDES Application with actual meaning.

The state diagram based modeling techniques focuses on the system's "state". They accomplish the modeling task with describing the state transition of system, including the establishment of the state set and the transition set. For systems which have smaller state space, the state diagram has good modeling effect. However, when it comes to the massive PDES, the integration difficulty becomes much greater because the dimension of state vector will be great and the state space will be in explosive growth. Therefore, the state diagram is unsuitable for PDES Applications integration.

The Event Graph-based modeling method concentrates on "events"; entities appear only implicitly as event attributes. The state variables of system are changed by the event, while the occurrence of future events probably results in the current event. Event Graph-based modeling describes the behavior of the system by "events" and "event scheduling", when it is applied in the PDES Application integration, the first work is seeking the system "events", and the next is describing the relationship of event scheduling. In this way, a system model is a directed graph composed of event nodes and scheduling edges. [3, 12] As the "basic integration module" is "event", and there is no physical counterpart with it, there exists a great gap between people's understanding of the real world and the graph model, which adds the difficulty of understanding. On the other hand, searching "events" from the entire system will be a very difficult task. What is more, the event graph scale and complexity will increase rapidly, which makes it difficult to understand the behavior of system.

As there are defects of Event-Graph, the researchers proposed the concept of Activity. Activity, usually meaning the process between two adjacent events, begins due to the occurrence of event, and ends due to the occurrence of the next event. Activity Cycle Diagram (ACD) is graphical modeling method. It creates an activity cycle diagram for each entity of the system, which can represent the status of the entity and the interactions between them in an intuitive way. It can fully reflect the behavior pattern of various types of entities, and describe the system status changes by individual state set. Therefore, it can better express the concurrent and collaborative between the entities of the system. However, it only describes the steady state of system, and does not describe the transient of system; that is to say, it only considers the relationship of the status transition between the entities without considering the events that causes the beginning and ending of the activity. Moreover, it is not suitable for large systems integration because of its "one entity, one diagram" pattern. The integration process is very complex, so developers have to add some auxiliary diagrams to finish the integration. Moreover, because of the close relationship between Activity and simulation scenario, it is difficult to reuse the "basic integration module".

The above visual integration technology concentrates more on how to describe the system's inner logical in a graphical way. There are obvious shortcomings in reusability, and you have to write code by hands especially, so they are not the intuitive way for the integration.

III. OBJECT-INTERACTION GRAPH

Large-scale Parallel Discrete Event System always contains a large number of entities among which there are numerous complex interactions. Namely, the large number of objects and the complexity of interactions are the fundamental characteristics of PDES system. Therefore, with the object-oriented thinking [15], describing the PDES system from the "object-interaction" perspective is very natural. Applying the object-oriented technology to the process of simulation and modeling, we can easily map the "object" to the Logical Process that is the basic element of the parallel discrete event simulation. Moreover, with the reuse level of the "basic integration module", the application can have a real parallel execution on the simulation object level, which is more useful for the improvement of integration and execution. The object is the objective reality of the world, so there are two aspects of static and dynamic contents. The static content refers to state of object, while dynamic content refers to the behavior of object. To realize the visual integration of application, the first problem we encounter is how to define the "simulation object" visual representation that contains the two aspects of content. The traditional technology uses UML Class diagram to describe the objects. However, the Class diagram is a description of the static properties and functions of objects, reflecting the correlation and dependency relationship between classes. That is to say, it is a static description of the system's structure, when it comes to the dynamic interaction between objects, we will have to seek the help of "state diagram", or "activity diagram", or "sequence diagram". What's more, the standard UML class diagram does not define intuitive graphical representation of the object [7]. If we simply copy the standard UML graphical to complete the PDES Application integration, we will not only fail to provide a natural visual experience of, but also increase the complexity of the integration. So the UML and relevant visualization tools have been restricted in the field of PDES Application integration. Then it is imperative to design simple-to-use graphical language that has the capacity to describe the static properties and dynamic behavior of object according to the characteristics and PDES Applications. In this section, we will give the definition to the "object interaction diagrams".

A. Object-Interaction Graph

Object-Interaction Graph is used to describe PDES Application from "objects interaction" perspective, composed of nodes and directed edges. Each node corresponds to a "simulation object", while each edge corresponds to the relationship of event scheduling between objects. We ensure the complete ability of describing PDES Applications on the facts that PDES Application is established through realizing large number of entities and interactions. In other words, "entities" and "interactions" are the fundamental elements of PDES Application, and also the only elements. Luckily, the basic elements of "object-interaction graph" are "objects" and "interactions", so the "object-interaction graph" has the ability to express PDES system completely. The basic "object interaction graph" is shown in Figure 1.

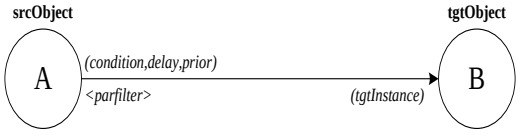

FIGURE I. OBJECT-INTERACTION GRAPH.

The purpose of establishing "object interaction graph" is not only to describe PDES system in an intuitive way, but to use the "object interaction graph" to generate code directly.

This requires "object interaction graph" to have sufficient expression ability, fully presenting the whole information including "static information", "dynamic behavior", and "interaction logical". Therefore, this information must be attached to each node and each edge.

For each edge, referred as "event scheduling edge", the information needed to attach are: objects associated with it-including the source object (srcObject) and the target object (tgtObject), events associated with it-including the source event (srcEvent) and the target event (tgtEvent), scheduling condition (condition), scheduling latency (delay), scheduling priority (prior), target instance (tgtInstance), transfer parameter (parfilter), etc. Therefore, the edge means that the srcEvent of the srcObject will schedule the tgtEvent of the tgtObject at the time of delay milliseconds later, of course the schedule condition must be true, the priority of the scheduling is prior, and parameters are parfilter.

For each node, the primary dissimilarity with all the other graphical language is that the node of the "object interaction graph" has its own structure, so it can represent complex simulation object. We call the structured node "Simulation Object Cell". It is the structure of the Simulation Object Cell shown in Figure 2, including static part (static units) and dynamic part (dynamic units), corresponding to "static content" and "dynamic content" of the "simulation object" respectively. Dynamic unit is generally referred to as "port" of the Simulation Object Cell, and is associated with a specific event of the simulation object. From the aspect of scheduling direction, ports of the Simulation Object Cell are divided into "scheduled port" and "scheduling ports", and installed on the left side and right side of the "static units" respectively. "Scheduling port" means that the simulation object can schedule other object's event through this port; while "Scheduled port" means that the event of the port associated with may be scheduled by other object's event. For the aspect of this object, the other object's event that can be scheduled by other objects is referred to as "external event". From the interaction type, scheduled ports installed on the left of the static units are divided into "state sharing port" and "event scheduled port". "State sharing port" is supplied for the convenience of state information sharing between the objects, it means that objects can use this port to publish self state information to public, and by the mechanism named "subscribe" the others can get this information whenever it wants. Though the port is installed on the left, this "publish" is active, and we use different shapes to separate from the "event scheduled port", as publish/subscribe port in the figure 2. This classification makes it not only easy to use "Simulation Object Cell" to integrate the PDES Application, but also to configure the interactions between objects.

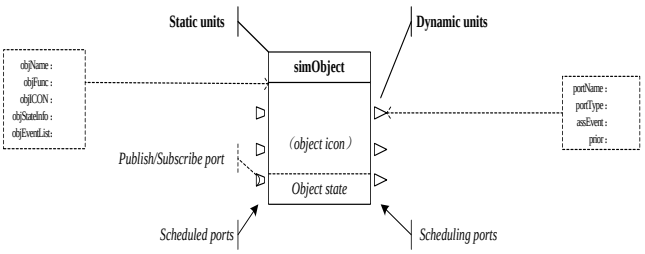

FIGURE II. SIMULATION OBJECT CELL.

Both the "static units" and "dynamic units" carry information. The "static units" carries the state information of the object, while the "dynamic units" carries the dynamic behavior information of the object, including the event, parameter, priority and so on.

The "dynamic behavior", namely the event logical process, will be descried in an XML file which is the other form of the Simulation Object Cell. The XML schema is introduced in the next section. The final "Object Interaction Graph" is as follows.

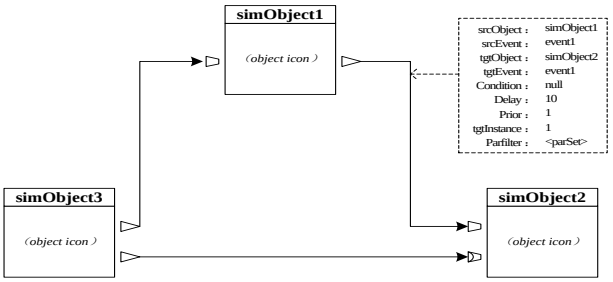

FIGURE III. PDES APPLICATION OBJECT-INTERACTION GRAPH.

The port type and quantity can be freely selected according to the specific situation in the object modeling. As the division of port's type is not for specific platform, the Simulation Object Cell is in good independence and universality.

B. Event Processing

In addition to the above static information, there is also object's dynamic behavior information, which is the event processing, it should be described by graph too. However, the uncertainty of event processing causes it almost impossible to achieve the requirement. To solve this problem, event processing should be formalized. This is based on the fact that any event processing includes only two parts named "computing" and "calling". So, the event processing can be classified as "computing" and "scheduling". The "computing part" completes the logical calculation, and the "scheduling part" completes the scheduling of external events and their local events, also including the general method. The calculation of the "Computing part" is done by calling a special "module" to complete; the "module" is a separate executable code that can be invoked, such as ".dll" files. This division makes the processing of events not change the state of object any time, it means that the event processing only includes parameters passing, state data obtaining, scheduling computational model to calculate, saving state data, and scheduling

external event based on the results or condition, and so on. Event processing is shown in Table 1.

TABLE I: THE FORMALIZED EVENT PROCESSING.

1) obtain the incoming parameters of the event ;
2) obtain the state data of object;
3) i=1;
4) whether the condition of scheduling "computing module i" is true; otherwise, go 9 ;
5) prepare parameters of "computing module i", then schedule "computing module i";
6) save the current state of the object , get the output of "computing module i";
7) determine whether should schedule other events based on the results or condition; otherwise, go 9;
8) Prepare parameters of the event scheduling, then schedule the event. (End of the current event)
9) i++;
10) if there is a "computing module i" needed to be scheduled, repeat the process 4-8 for the "computing module i", otherwise, the current event processing ends.

Obviously, to describe the process of event, the information need to be recorded includes the input parameters of the events, local variables, scheduling modules and events, and the order, etc. That is helpful for using XML to describe the event processing. Table 2 is an XML template of the event processing.

TABLE II: XML TEMPLATE OF EVENT PROCESSING.

<event name="" type="" func="">	*event processing*
<inputParaSet>	*input parameters set*
<para name="" type="" desc=""/>	*input parameter*
</inputParaSet>	
<localvarSet/>	*local variables set*
<scheduleCalModelSet/>	*scheduling modules set*
<scheduleLocalEvent/>	*local events of the object*
</event>	

C. XML Description Of The Object-Interaction Graph

"Object-Interaction Graph"not only carries intuitive visual features, but is able to describe all the information of object including static properties, dynamic behavior, and interaction between objects. However, in order to generate code automatically through it, it must be converted to non-graphical representation for computer to handle. Using XML to describe "Object-Interaction Graph" is a suitable solution. To this end, an XML description file is designed for each "Simulation Object Cell". Table 3 is the template of the file. Obviously, the XML template of Event Processing is the portion of it.

TABLE III: XML TEMPLATE OF SIMULATION OBJECT.

<simObject name="" func="" icon="">	*simulation object*
<stateInfo>	*state information*
<statevar var="" type="" desc=""/>	
</stateInfo>	
<statePublishSet/>	*Publish state information*
<eventList>	*events*
……	***event***
</eventList >	
<portList>	*ports*
<port name="" type="" event="" relation=""/>	
</portList >	
</simObject>	

An XML template for the description file of the "Object-Interaction Graph" defined is shown in Table 4, the most important information is the event scheduling relationships between objects.

TABLE IV: XML TEMPLATE OF OBJECT-INTERACTION GRAPH.

<objintgraph name="" desc="" >	
< simObjectSet>	*Objects set*
<statevar var="" type="" desc=""/>	
</simObjectSet>	
<eventscheduleSet>	*Scheduling events set*
<SchEvent srcobject="" tgtobject="" ……>	
<InputPara>	*Input parameters*
<para serial="" paraname="" ……/>	
<para serial="" paraname="" ……/>	
</InputPara>	
<SchConditioin>	*Scheduling condition*
<cond var="" oper="" value="" />	
</SchConditioin>	
</SchEvent>	
</eventscheduleSet>	
<statePublishSubscribeSet>	*Subscribe state informaiton*
<port name="" type="" event="" relation=""/>	
</statePublishSubscribeSet>	
</objintgraph>	

Two XML files contain all the information of "Object-Interaction Graph". As the actual "Object-Interaction Graph" contains many objects, the number of "Simulation Object Cell" XML file will be large, but the "Object-Interaction Graph" XML file is only one. The next work is to convert these XML files to code automatically.

IV. VISUAL INTEGRATION FOR PDES APPLICATIONS BASED ON OBJECT-INTERACTION GRAPH

Modeling with "Object-Interaction Graph" considers the system as a set of interactional Simulation Objects, using the "Simulation Object Cell" to assemble the application. Developers can intuitively and easily achieve PDES visualization application integration by "dragging and dropping" with the support of relevant tools. The developers no longer confined to computer professionals who are familiar with the simulation platform, but the general domain experts can do it easily too, even a general computer operator. What's more, with the "object assembling" way, the application's structure can be changed at any time, which can be well adapted to the rapidly changing needs of the simulation.

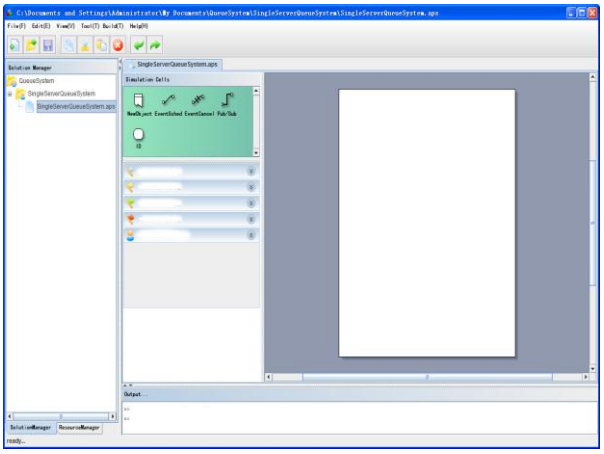

FIGURE IV. PDES APPLICATION INTEGRATION TOOL.

In order to realize the visual integration of PDES Application based on "Object-Interaction Graph", we developed the "PDES Application Integration Tool". With this tool, the process of application integration is simplified as follows: selecting or developing objects, connecting the objects, configuring event scheduling, and generating code. Figure 4 shows the interface of the "PDES Application Integration Tool". In this section, we show the steps of the PDES Application integration using the tool.

(1)Analysis and model selection

In this phase, the developers make detailed analysis for seeking objects, and then checking the object library to make sure they are in it. If there is none, we should develop it by self or gain from the supplier. We should make detailed analysis of all the interaction information too, preparing for the next step.

Queuing system is a classic discrete event systems [16], mainly related to three types of objects: helpdesk (Server), queue (Queue), and customer (Customer), therefore, three simulation objects should be defined. There are five types of interactions between the three objects. The customer asked for queuing when he/she arrives, it is represented by event scheduling from evtArrive event of the Customer to evtInQueue event of the Queue. The server finishes its work and informs customer to leave, at the same time notifies the Queue that "I'm idle". These are described by two times of event scheduling: one is from evtEndServer event of the Server to evtLeave event of the Customer, and the other is from evtEndServer event of the Server to evtOutQueue event of the Queue. The Queue informs the first customer in it to quit the queue and to accept service; this is presented by event scheduling from evtOutQueue event of the Queue to evtAccepted event of the Customer. Then, the customer required the server to start work, which is represented by the event scheduling from evtAccepted event of the Customer to evtStartServer event of the Server.

(2)Establish the Object-Interaction Graph

The only work that developers need to do is to select and drag the objects into the application assembly area of the tool, then to connect them with the interaction line. The most important work is configure the scheduling between objects, including the conditions of interaction occurs, priority, instance number, parameters, and so on.

(3)Code Generating

The tool can generate XML file from the "Object-Interaction Graph" automatically, then the code generation tool which is designed for the specific platform, can generate platform-specific code. It can be compiled to run directly. Of course, you can make changes to it if you want.

V. CONCLUSION AND FUTURE WORK

In this paper, we analyze the basic characteristics of visual integration for PDES Application, and point out the problems of current graphical modeling languages when they are applied in the PDES Application integration. Then, the Object-Interaction Graph is proposed for the "object interaction" features of the parallel discrete event system. Based on these work, we propose a Visual integration approach for PDES Application, and introduce the tool that works based on this. In the future, we will improve the ability of the "Object-Interaction Graph" expression and the process of the integration approach. Code automatical generating will be implemented. We also plan to research the method of redundancy checking for the "Object-Interaction Graph". Providing better integration framework and improving user experience with the tool are also our future work.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science foundation of China (NSF) Grant (No.61170048) and the National Key Technology R&D Program (2012BAH08B02).

REFERENCE

[1] Yiping Yao, Ying-xing Z. Solution for Analytic Simulation Based on Parallel Processing. Journal Of Systems Simulation 2008.

[2] Sanchez PJ. A Gentle Introduction to Simulation modeling. Winter Simulation Conference; 2006.

[3] Schruben LW. Building Reusable Simulations Using Hierarchical Event Graphs. Winter Simulation Conference; 1995.

[4] Buss A, Blais C. Composability and Component-based Discrete Event Simulation. Winter Simulation Conference; 2007.

[5] Maria A. Introduction to Modeling and simulation. Winter Simulation Conference; 1997.

[6] Eden AH, Gasparis E, Nicholson J, Kazman R. Modeling and Visualizing object-oriented programs with Codecharts. Form Methods Syst Des 2013.

[7] Brockmans S, Volz R, Eberhart A, Loffle P. Visual Modeling of OWL DL Ontologies Using UML. 2004.

[8] Balci O, Bertelrud AI, Esterbrook CM, Nance RE. Visual Simulation Environment. Winter Simulation Conference; 1998.

[9] Qun L, Chao W, Wei-ping W, Yi-fan Z. Design and Implementation on Simulation Engine Compliant with SMP2.0. Journal of System Simulatioin 2008.

[10] Howell F, McNab R. Simjava : A discrete Event Simulation Library for Java.

[11] Muetzelfeldt R, Massheder J. The Simile Visual Modeling Environment. Europ. J. Agronomy 2003.

[12] Schruben L. Simulation Modeling with Event Graphs. Ssimulation Modeling and Statistical Computing 1983.

[13] Cota BA, Sargent RG. A Modification of the Process Interaction World View. ACM Transactions on Modeling and Computer Simulation 1992.

[14] Banks J, Carson JS. Process-interaction simulation languages. Simulation 1985.

[15] L.Y Liu, Ioannou PG. Graphical Object-Oriented Discrete-Event Simulation System. Winter Simulation Conference; 1992.

[16] Law AM. Simulatioin Modeling and Analysis(Fourth Edition); 2009.