OPMSE: An Object Petri Nets based Modeling and Simulation Environment

Luo Xueshan, Qiu Dishan, Rao Xianhong, Bao Weidong
Research Center of C^3I Systems
National University of Defense Technology
Changsha , Hunan , 410073 P.R.China
Tel: +(731) 4575856 Fax: +(731) 4575856
Email: xsluo@nudt.edu.cn xsluo@public.cs.hn.cn

Abstract: In this paper, the methodology of integrated simulation is proposed. The simulation conceptual framework with modeling, experiment and analysis is discussed. And an object model with integrating static, dynamic and representation characteristics is presented.

Under the direction of the above-proposed methodology, an Object Petri net-based model description language OPDL is designed and implemented. Furthermore, an integrated simulation environment OPMSE is developed. OPMSE is constituted of two packages: Model Developer and Simulator. The architecture of OPMSE is described.

Classification: Computer tools for nets
Keywords: Petri Nets, Object-Oriented, Model, Simulation, Integrated Environment

1 Integrated Simulation Framework

Most research of system simulation put emphasis on the experiment. That is they lay stress on how to obtain the tracks of performance parameters varied with time going. In this mode, the model data, experiment conditions and data managing are put together. That makes it difficult to reuse the models and requires the uses to have both the knowledge of systems and the skill of programming and simulation.

In view of above drawback, the integrated simulation framework is proposed. One of its characteristics is the separation of the modeling, experiment and analysis. In modeling phase users focus on the system description without thinking of experiment conditions. During the experiment phases, users set the parameters of the mode and run the model to get the simulation data. Finally according to analysis of the simulation data users educe the results.

The framework of this simulation mode is shown as Fig.1.

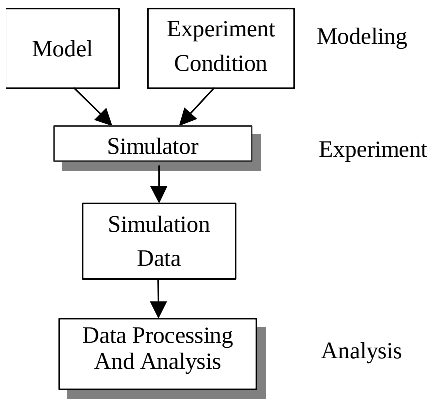

Fig. 1 Integrated Simulation Framework

Because of the separation of the modeling and experiment, the model data and the running mechanism are divided. It makes the real-time interaction and multimedia demonstrating of simulation process possible. At every end of inferential cycle the simulation environment can get the model's state to demonstrate. At the same time user can modify the model data. In next inferential cycle the modification will be operated. It is shown as Fig.2.

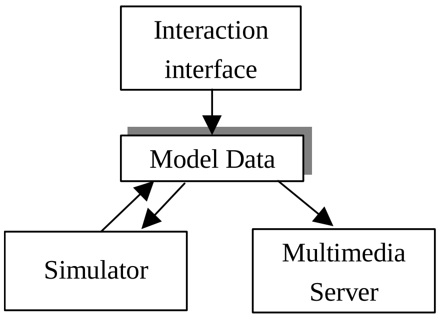

Fig.2 Interaction, Inference and Demonstration of Simulation

2 Object Model

To fit the framework of integrated simulation, An object model with integrating static, dynamic and representation characteristics is designed. It is defined as follows:

$$O=< X, S, Y, t_s, \sigma_s, \delta_s, \lambda, \delta_\pi, A_m>$$

$$X=\{ (p_i, m_e) \mid m_e \in M_e , p_i \in P_I \}$$

$$Y=\{ (p_o, m_e) \mid m_e \in M_e , p_o \in P_O \}$$

$$t_s : S \rightarrow E$$

$$\sigma_s : M_e \rightarrow E$$

$$\delta_s : S \times E \rightarrow S$$

$$\lambda : S \times E \rightarrow Y$$

$$\delta_\pi : S \times E \rightarrow A_m$$

In above expression,

M_e means the set of messages among the objects;

P_I, P_O respectively means the input ports and output ports of the objects;

E means the set of events resulted by the operations and messages;

X means the set of messages imported from input port p_i;

Y means the set of messages exported to output port p_o;

S means the set of object states;

t_s is the action function under state s. When object state is s the action function will be executed and results in new events.

σ_s is the message conversion function. It converts the input messages to corresponding events.

$\delta_s(s,e)$ is the state conversion function. It defines the next state when object is in the state s and event e occurred;

λ is the message output function. It determines the message exported to other objects when object is in the state s and event e occurred;

A_m is the multimedia attribute of the object. It defines the media types, material and operations, which needed by the representation of object;

δ_π is the media conversion function. It determined the representation when the object is in the state s and the event e occurred.

3 Object Petri Nets based Description Language (OPDL)

To implement the object model, an Object Petri Nets based Description Language is designed. OPDL extents Petri nets from two aspects:

1. Introduces object-oriented methods into Petri nets to form Object Petri Nets(OPN). Object is the basic unit of model and reusable module;

2. Adds new elements including switch and inhibitory arc to Petri nets and attributions to all of the elements to enhance the description capability.

Using OPDL the basic unit of a

model is the object, which generally correspond to the realistic parts of the systems. The similar objects are abstracted as a class, and all the objects that belong to the class have the same basic features. An object is generated dynamically from the class that it belongs to in the execution process. Model building and organizing bases on the class structure. Therefore modeling is the process of designing and using classes. The classes are organized in the form of class library. The mechanism of the class inheritance and quotation supports the reuse of model with high independence.

The definition of the class includes four parts:

- Property table: Each item is composed of property name and property value. Via the name the value can be accessed. It is a data space used by the object;
- OPN description: It is an OPN graph. Different graphs connect through input ports and output ports. There are two types of ports: the external ports and the internal ports. The external ports are used to connect with brother objects or parent object. The internal ports are used to connect with son objects and can not be accessed from outside.
- Initialization function: It is used to initialize the instance when the object is created;
- Post-instancing function: It will be executed after the instance has been created.

To create an object from a class the instance function is needed. First the environment copies the graph and creates the property table for the object. Then the initialization function, the instance function and the post-instancing function are executed in sequence. All the functions are written with VBScript. Furthermore the external module can be linked to the model.

The expansion of the elements of Petri nets is as follows:

- **Place**: The place is a structure, which has queues to buffer tokens. Each place corresponds to a color. Only the tokens with same color can enter the place. The place has the attributions such as principle of queue, capacity and the event processing function. When a token enters or gets out the place the event processing function will be executed.
- **Port**: The port is a kind of special place and has the same attributions as place. But the successor of the output port must be the input port and the predecessor of input port must be the output port.
- **Transition**: We add the priority and delay function, predicate function, action function, event processing function to transition. Priority is used in handling the conflict. Delay function is used in determining the processing time of transition. Predicate is a necessary condition that must be satisfied for the transition to be enabled. Action is a sequence of operations that a transition carries out when it fires. In action function the multimedia demonstrating can be defined. At the beginning of transition firing the event processing function will be executed. All the functions are written with VBScript.
- **Switch**: the switch is a special kind of transition. The difference between switch and transition is that switch does not export tokens at the end of

firing. How the tokens to be distributed must be dedicated by users.

- **Arc**: Arc is used to connect the place and the transition just as Petri nets. Otherwise it is used to connect the input port and the output port.
- **Inhibitory Arc**: Inhibitory Arc is a special kind of arc. It must start from the place and end at transition. If a place and a transition are connected with a inhibitory arc, the transition is enabled only if there is not token in the place.
- **Token**: Token is defined as a structural data and corresponds to a color. Token has a property table. In the functions of transition the items of property table can be insert and accessed.

4 The Architecture of OPMSE

OPMSE is an integrated modeling and simulation Environment including Model Developer and Experiment Manager. The architecture of OPMSE is shown as Fig.3:

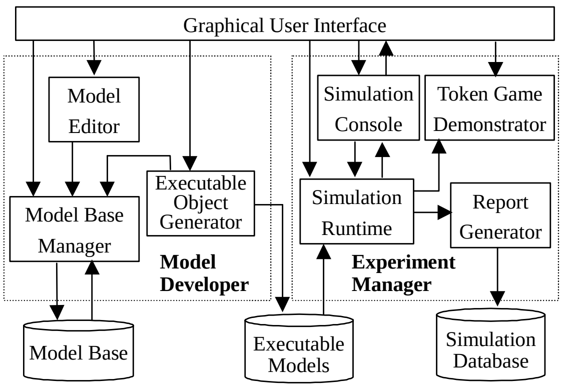

Fig.3 The architecture of OPMSE

- **Model Editor:** It is a visual and integrated editor that can edit graphs and texts. The editor has the ability of syntax-guided. It does not respond the operations that don't satisfy the syntax rules and prompts with warnings.
- **Model Base Manager:** It manages OPDL class libraries. It supports insert, delete and extract operations of classes and the merger of libraries.
- **Executable Object Generator:** It compiles the OPDL models and generates the executable objects.
- **Simulation Runtime:** It provides a framework for objects running and simulation. The inference mechanism scans the data structure of objects, dealing with the conflict, constructs the enabled set, firing the transitions and executes the functions.
- **Simulation Console:** It reads the states of the running model and represents the simulation process according to the user's definition. At the same time it receives and responds the interaction commands of user. The inferential cycle is independent, that is, the inferential process has nothing to do with the previous ones and the state of model is discrete. So user can interrupt the inference at any time and modify the model's state. Inference mechanism can continue working from the new state.
- **Token Game Demonstrator:** It demonstrates the running process of Petri net by changing the color of transitions and arcs. The number of tokens in the place also varies with the running to show the flowing of tokens.
- **Report Generator:** It records the simulation parameters and generates the simulation report. The parameters of a transition include average service time, maximum service time, minimum service time and times of firing. The parameters of a place include maximum and minimum queuing lengths, maximum and minimum waiting time of tokens, average waiting time of tokens, total numbers of input and output

tokens.

5 Conclusion

As its major application area OPMSE addresses discrete event dynamic systems. It is suitable for the simulation engineers and developers of discrete event systems such as C^3I system, CIMS, FMS, communication protocols and automatic control systems. A criticism that is often raised against Petri net is the unmanageable size of the models of complex system. This drawback can be reduced by using Petri net structured and layered within an object-oriented framework and executing on a network. Distribution also rises the extent of concurrence. Our farther works focus on distribution and model verification.

References:

1. Bao Weidong, Study on Integrated Simulation and Its Support Environment for C^3I Systems, Doctoral Dissertation of National University of Defense Technology, Changsha, 1999;
2. Bao Weidong, Sha Jichang, Luo Xueshan, Design of a Distributed Simulator Based on OPNs, Proceedings of the 1997 IEEE International Conference on Intelligent Processing Systems (ICIPS'97), Beijing, 1997.10;
3. Luo xueshan, Analysis and Design of C^3I Simulation and Demonstrating System, Reference Periodical of National Technology, vol. 18 No.3 1997;
4. Yu Yuncheng, Rapid Prototype of C^3I Systems, Reference Periodical of National Technology, vol. 15 No.2 1994;
5. Yang Kyulee, OPNets: An Object-Oriented High Level Petri Net Model for Real-Time System Modeling, J. System Software, 1993(20):56-67;
6. Wang weiping, Study on Object-Oriented Multimedia Simulation, Doctoral Dissertation of National University of Defense Technology, Changsha, 1998.