Abstract

This paper analyzes the architecture of virtual experiment and discusses the possibility of building a standard interface for it. We also introduce an agent-based VE system, which can integrate easily apparatus developed independently and using which a teacher can just focus on his own field even without any programming technique. This model may greatly eliminate many problems involved in software engineer and make distributed experiment possible. 

Key words-- Virtual Experiment, COM, Agent

I. INTRODUCTION

Virtual Experiment (VE) is a kind of efficient means of e-learning, it can save money due to expensive apparatus, ensure safety because of dangerous chemical powders and so on. However, nowadays many VEs are developed in different entities though their functions are same.

The components of VE, virtual apparatus, are difficult to reuse. On the other hand, the border between teacher and developer is not clear and it is usually not possible for a teacher to design an experiment with his own knowledge. So it is an urgent requirement for us to devise a standard interface of virtual apparatus and an intelligent mechanism for teacher to design a VE without much unnecessary effort.

II. ARCHITECTURE OF VIRTUAL EXPERIMENT

There is no doubt that VE is a complex system, so different people may have different perspectives on it. These people relevant to VE can be roughly divided into three groups: teachers, programmers and students. On the basis of top-down analysis, VE consists of three parts: programming by programmers, designing by teachers and experimenting by students (see Figure 1). We should keep in mind the fact that not all of the teachers are programmers at the same time, which implies that different people with different background should have different responsibilities. A successful VE system must ensure that designing an experiment would be as simple as to such an extent that the teacher can design VE using no other expertise but his own instructional field.

III. VIRTUAL APPARATUS

VE is made up of many virtual apparatus. To make them reusable and easy to integrate, the best technology to adopt might be Component Object Model. Though COM has many merits of independece, scripting skill is still needed to glue these virtual apparatus together. The aim of our VE framework is to gain the advantage of two features:

1. A standard programming interface
2. Minimum interference into the inner structure of each virtual apparatus

If we achieve this goal, we will never be trapped into the embarrassment of software engineer. The basic appearance of each virtual apparatus is shown in figure 2.[1]

Figure 1 – Different Perspectives

Figure 2 – Virtual Apparatus

Each virtual apparatus is implemented as a COM object with some interfaces exposed to the outside and a self-
described XML file for registration. One of the interfaces, called IDropSource, must be inherited to enable the teacher to interact with the apparatus. The IAgent interface will be discussed shortly. The special XML file is in fact a media for the virtual apparatus and VE to interact and has two sections: one is a readable section for VE to obtain the information of the COM object including its dynamic library file name, CLSID and IID, etc.; the other is a writable section filled by VE, where the object can get the necessary run-time information such as its agent identity number, server name and port number.

IV. AGENT STRUCTURE

Since only minimum interference is done to each apparatus, it must do lots of work on its own, such as changing its behavior according to the environment condition. This character makes an apparatus look like an agent, so every one should implement the IAgent interface. Though several characteristics of agents have been discussed in the literature, we assume that agents exhibit the following three important properties in our modeling approach[2][3].

1. Autonomy - an agent can make decisions about what to do based on its own state, without the direct intervention of other entities.
2. Adaptation - an agent can perceive its environment, and respond to the changes in the environment in a timely fashion.
3. Cooperation - an agent can interact with other agents through a particular agent-communication language, and typically has the ability to engage in collaborative activities to achieve its goal.

Once activated by the environment, the agent goes into the adaptation state. Then it receives messages from other agents, judges the condition variables and sees if they are relevant. After the sense phase, it can produce some plan by arranging a series of actions, which may result in the changed appearance or its inner states. The agent can also give feedback to the environment by changing the experiment parameters. The agent comes to the end of its life cycle if deactivated by the outside.

V. COMMUNICATION

Only a minimum number of functions in an agent are exposed to the environment, such as Activate and Deactivate, but they cannot be accessed by other agents. The messages between agents are passive, i.e., they do not contain methods because agents never know each other's structure. Once activated, an agent just sends out massages containing measurement parameters according to its own state transitions, which are sent to another agent connected to it or to the environment. The parameters might be temperature, pressure, and so on. Because the message is different from traditional method invocations, we name it as passive message or PMessage. The communication mechanism is implemented on the basis of socket technique. At the beginning of its life cycle, every agent must create a listening socket with the server address and port number assigned by the environment on registration. Those who want to communicate with it must bind to that address and port number. Such a mail-like communication mechanism can make distributed experiment possible. If agents want to have a conversation, they must send their identity numbers to each other via run-time environment. Figure 3[4] shows such scenario.

VI. CONCLUSION AND FUTURE WORK

Since our virtual apparatus component is highly autonomous and possesses a clean programming interface, potential trouble for integrating them into a virtual experiment system is avoided. At the same time, we have achieved the feature for teacher to design an experiment with little effort. Future work will focus on the concurrency and responsiveness of our agent model. We will also pay much attention to the analysis of heterogeneous experiment platforms and realize the distributed virtual experiment system.

VII. REFERENCES

1. COM Specification http://www.microsoft.com