Joint Conference of 1996 International Computer Symposium
December 19~21, Kaohsiung, Taiwan, R.O.C.

Designing a Multi-user Interactive Script Language for CSCW Systems

Jia-Sheng Heh, Wen-Han Lin
Dept. of Info. & Comp. Eng., Chung Yuan Christian Univ., Chung Li
Jwu-Hwa Ho
Telecommunication Lab., Chung Li, Taiwan

Abstract

This paper proposes an interactive script model of CSCW systems and then develops the corresponding script language, called MISL (Multi-user Interactive Script Language). By the study of multi-user interaction and multi-user interface in CSCW systems, the interactive script model treats different multi-processes working environment as a scene and suggests the scene layout-switching graph in a script. To establish CSCW systems, a Multi-user Interactive Script Language (MISL) is designed and its interpreter is implemented. To prove such idea, a distance education system with five learning modes is built as a real CSCW case.

Keywords: CSCW (Computer-Supported Cooperative Work), interaction, user interface, script language.

1. Introduction

As the technology of computer multimedia and network transmission progress, it is important to establish application systems integrating these techniques to aid a group of people to accomplish a specific objective. These softwares are called *groupwares* and this kind of working style is known as *CSCW (Computer-Supported Cooperative Work)*. [6] Up to now, there are many application systems to support different group works. [9][7] However, a CSCW system probably includes different social norms and working environments. For example, in a practical distance teaching system, whiteboard and audio phone are both necessary. Therefore, a complete CSCW system is not only a multi-user cooperative system, but also a serial multi-processing working environments which provide different interaction styles in a computer network.

When a receiver gets the message sent from a transmitter, it will proceed in response to this received message and send the result back to the sender. Based on traditional propagation theory, these interpretation and response form an effective two-way communication, called *interaction.* [14] By such process, an interaction cycle include message source, receiver, message and response, as shown in Figure 1.

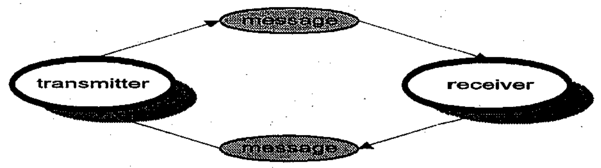


Figure 1 Propagation model proposed by Schramm (1954)

As computers are widely used, people become more and more dependent on them; therefore, studying friendly *human-computer interaction* emerges in the research areas of programming and human factor. From the viewpoint of propagation theory, human-computer interaction [5] is one kind of interactive process between the human beings and machines. However, a machine can not transmit and receive messages as free as people, but it has to be designed carefully; more specifically, it can only provide some limited interaction methods. Schwier [14] mentions that different user interfaces have different interaction levels, including responsive interaction, active interaction and two-way interaction. Here, we can simply define *human-computer interaction* as an interactive process between person and computer and this interaction is one kind of two-way communication with immediate-response interactive process, as shown in Figure 2. From this point of view, the studying of human-computer interaction is emphasized on the communication method between people and machine, including how a machine receives the message from a person, how this person receives those message generated by this machine, and how the machine processes and responds to the person.

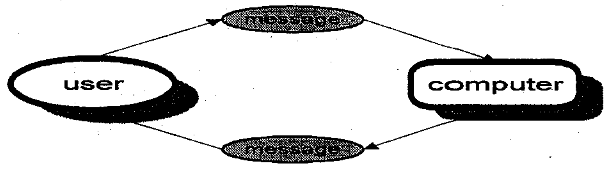
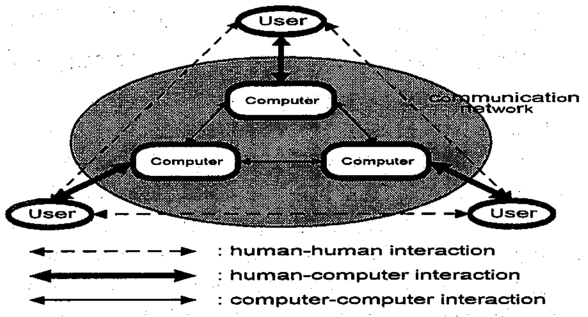


Figure 2 Human-computer interaction model

A CSCW system is a multi-user multi-machine system, by which people can effectively communicate with other users to attain some specific objective. From this standpoint, the interactions of CSCW systems can be categorized into three forms: human-human interaction, human-computer interaction and computer-computer interaction, as Figure 3 shown. For a CSCW system, *human-human interaction* comes from its special social norms. On the other hand, its *human-computer interaction* is much more complex than that of single user system, since a computer receives not only the messages delivered by the user of this computer, but also the messages sent by other remote user. Finally, the *computer-computer interaction* is composed of the network transmission among different computers as well as the inter-process communication among those processes within a computer. In such multi-user multi-computer systems, the human-human interaction has to be accomplished by the coordination of human-computer interaction and computer-computer interaction.


User
Computer
Communication network
Computer
Computer
User
User
: human-human interaction
: human-computer interaction
: computer-computer interaction
Figure 3 Interactions in a CSCW system

Based on the study of the above three kinds of interactions, this paper proposes and analyzes an *interactive script model* of CSCW systems, then develops a Multi-user Interactive Script Language (MISL) to build such system. Section 2 describes the development of this interface model. Then the interactive script and its scene layout-switching graph are investigated in Section 3. The design of MISL and its interpreter are illustrated in Section 4. Finally, Section 5 applies this language to construct a real CSCW system, the distance education system with five learning modes. Section 6 is a conclusion.

2. Interactive script model

It has been mentioned that in a CSCW system, the interactions among multiple users are carried out through the network and its computers. Since the users can not see other users' responses directly, these computer must take the responsibility of displaying other users' status and information. This makes a proper user interface design for presenting the multi-user interactions much more important.

In the traditional research, *Seeheim model* [8] divides a user interface into three parts: presentation, dialog control and application, as Figure 4 shown. *Presentation control* deals with the real exhibition in screen, including input-output devices, screen layout, interaction methods and display skills. *Dialog control* takes charge of the dialog (interaction) between user and computer. This means that the user has to follow the pre-defined procedures to interact with the computer; if not, the computer will ignore his/her operations. Finally, *application interface module* defines the interface between user interface and other programs and manages the function calls of the application.

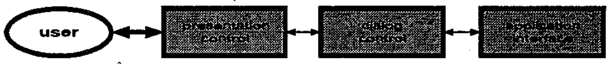


Figure 4 User interface model (Seeheim model)

In this model, presentation control and dialog control interact closely. This is because presentation control provides proper user interface and different kinds of human-computer interactions and dialog control follows pre-defined syntax to check the legality of user operations. When the user's operation is legal, some message will be passed to application interface module and a corresponding response will be displayed on screen. Human-computer interactions are done through a series of such interaction processes. In this sense, the representation of human-computer interaction can be treated as a sequential screen switching process in response to user's operations.

Furthermore, extending this model into multi-user interface needs some more considerations. [7] For examples, one application program maybe possesses different kinds of operations or displays for different users, and the presented screen has to suitably display some related information and responses of other users. Many researchers have made many efforts on multi-user interface model, two of which are admitted as typical: centralized architecture and replicated architecture. [4] In *centralized multi-user interface model*, as shown in Figure 5, many users share one single application program and all their operations and screen responses are managed by this program. On the other hand, every user has his/her own application program in replicated architecture, as shown in Figure 6. Since there is only one single application in the former architecture, any error in this program may bring the whole system a disaster. By comparison, the replication architecture possesses more flexibility and error independence. This paper will accept the replicated one as our model of CSCW systems.

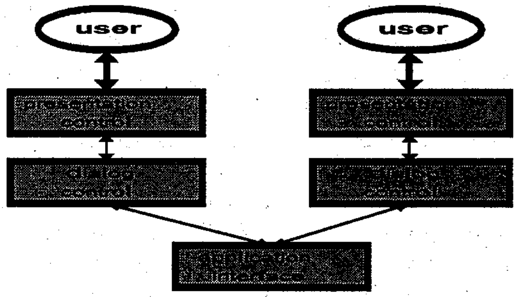


Figure 5 Centralized architecture of multi-user interface

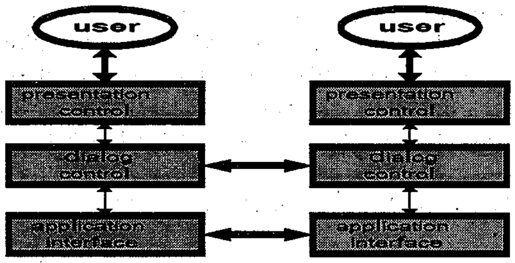


Figure 6 Replicated architecture of multi-user interface

This paper combines multiple user-machine interactions and multi-user interfaces to propose an *interaction script model* of CSCW systems. For a CSCW system, it often has to provide many complex working environments. Each *working environment* contains more than one application, with which users make their interactions. For examples, an electronic classroom can be used as an environment of teaching, discussion or test, and a teaching environment contains many teaching tools, such as blackboard, textbook and notebook. All these works can not be accomplished by simple applications, but have to be controlled through some complex screen switchings and their coordination. Thus, the studying point of CSCW systems will not fall on the display and response of simple objects (applications) as human-computer interface researches, but focus on interactive script flow.

Interactive script flow control is one kind of dialog control with high-level interaction control, whose position in the above multi-user interaction and multi-user interface can be understood from Figure 7, the *interaction script model* of CSCW systems. In Figure 7, more than one application tool can work together corresponding to its own application module window. Compared to Seeheim model, interactive script flow control is related to dialog control, different application tools is to application program module, and the screen constituted of application module windows is to presentation control. Moreover, there are networking communication among interactive script flow controls in different machines.

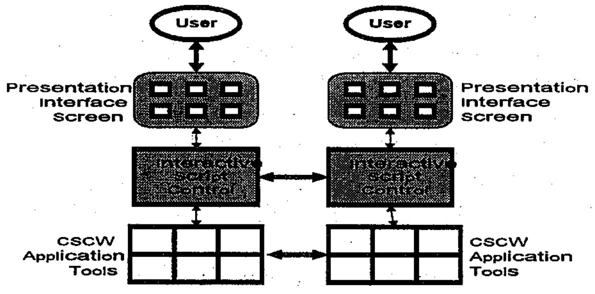

User
User
Presentation Interface Screen
Presentation Interface Screen
Interactive Script Control
Interactive Script Control
CSCW Application Tools
CSCW Application Tools

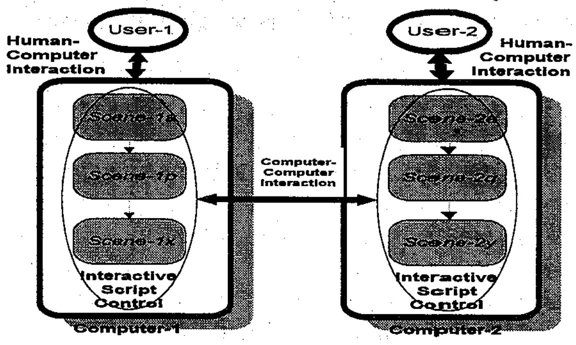
Figure 7 Interactive script architecture of multi-user interface

When a user works in some environment of a CSCW system, this environment and the corresponding screen will be affected by the interactions among users. The interactive script flow control will take charge of all the switching of complex working environments and working states. To regulate the switching well, the tasks of this interactive script flow control should be carefully analyzed and clearly defined.

3. Analysis of interactive script model

There are two basic elements of interactive script: tools and scenes. *Tools* are a set of application program modules, each of which possesses specific functions, such as textbook and electrocardiogram display. A *scene* is a set of windows of applications in the same screen, which forms a working environment. Practically, different tools are integrated to be a working environment to attain some specific goal. Take teaching as an example: the teacher may move the contents on textbook to whiteboard; whereas, every student records what the teacher show on whiteboard downto its own notebook.

In interactive script model, only two kinds of three above interactions exist: human-computer interaction and computer-computer interaction. To accomplish the whole objective of a CSCW system, the scenes of different computer users should operate cooperatively. When a user interacts with computer to switch his/her working environment, the user's screen in remote site will be changed accordingly through computer-computer interaction, as shown in Figure 8. This forms *scene switching* phenomenon among multiple users, which accomplishes human-human interaction indirectly.


User-1
User-2
Human-Computer Interaction
Human-Computer Interaction
Scene-1a
Scene-1p
Scene-1x
Scene-2a
Scene-2q
Scene-2y
Computer-Computer Interaction
Interactive Script Control
Interactive Script Control
Computer-1
Computer-2

ɪɢᴜre 8 Scene switching in interactive script model

The human-computer interactions come from user's operations, which introduce many kinds of scene switchings. Different human-computer interactions produce different *control messages*, then trigger different responses. Shneiderman [14] proposes that there are five types of human-computer interactions: menu selection, form filling, command language, natural language and direct manipulation. The first one (menu selection) is responsive interaction, both form and command are active interactions, and the last two are two-way interactions. In the primary study of this paper, we consider only the simplest type: responsive menu selection, including the selections from menu and button. Therefore, one specific kind of applications, called *control tool*, is designed to represent these responsive interaction. Any interaction from user's selection, no matter whether menu or button, can make the whole CSCW system switch to another working environment.

On the other hand, computer-computer interactions indicate data transmission among different computers and processes. When a computer receives the control message from local user, it has to make some processing and sometimes to pass this message to remote user in suitable way. This message transmission incurs a corresponding scene switching. To design a CSCW system, all these message transmission method, message format and response method are necessary to be carefully chosen. The overall description of these interactions are shown in Figure 9, where *Scene-1p* and *Scene-2p* are the present scenes and *Scene-1q* and *Scene-2q* are other scenes. All these scene switchings are controlled by an *interactive script control (ISC)*.

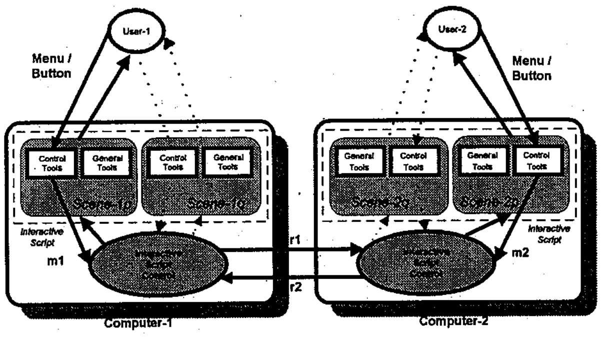


Figure 9 Interactions through messages passing

Before the development of analyzing tools of scene switching, it is necessary to investigate the characteristics of this kind of scene switching in CSCW systems. Firstly, the control flow of scene switching is a *concurrent* process; that is to say, the computer screens of different users may be switched at the same time. [9] During the interactions of scene switching, more than one user may concurrently deliver his/her scene switching request. If there is not any ordering among these switching requests to ensure every user to deal with his/her scene switching in the same order, the inconsistency among the scene switchings of different users may occur, then their interactions are not able to keep going on. Therefore, we face to solve the concurrency problem of scene switching, which can be overcome by the concurrency study in distributed system.

The scene switching of multi-user interactive script proceeds among several computers, then involves many networking transmission mechanisms. In usage, transmission mechanisms include networking transmission among different computers and inter-process communication among different processes within a computer. Also, there are two kinds of networking transmissions among computers: synchronous transmission and asynchronous transmission. [11][13][16]. For synchronous transmission, when a computer transfers data to another, it has to suspend and to wait for response. Whereas for asynchronous transmission, it will not pause but continue its successive works. All these transmission methods induce the problem of *data inconsistency*. Practically, the synchronous mechanism gives better assurance of data consistency.

Finally, the last characteristic of scene switching is *distributed computing*. This is because several autonomous concurrent processes do not share a single memory, but exchange information and cooperate through message transferring. [1] [15]

In the research area of distributed systems, there have developed many analyzing tools, the most popular of which is *Petri net*. Petri net is a directed graph with its nodes as *places* representing system states and *transitions* representing triggering conditions among states. Through Petri net, all the above concurrency, data consistency and distributed computing problems can be taken into consideration. The following will extend Petri net to develop our describing tool to analyze the dynamic behaviors of interactive script.

To develop the analyzing tool, it is natural to map the above scenes to be our places in Petri net, then the transitions among states are the above scene switching. As Figure 9 shown, there are two *triggering conditions* for a scene to be switched: one is the operation of local user, which is the human-computer interaction of *control message (m1 or m2)*, and the other is controlled by some remote user, which is the computer-computer interaction of *remote message (r1 or r2)*. In both interaction cases, when the interactive script receives some triggering message, it have to check what the message is, and to switch the screen to the requested scene.

With the above definitions, a Petri-net-like graph, called *scene switching graph*, can be constructed to describe the scene and scene switching in CSCW systems. An example graph is shown in Figure 10. However, this graph is not able to exhibit the layout of tools in a scene, not to say the interactions emitted from control tools. Based on such consideration, we extend scene switching graph into *scene layout-switching graph*, with the symbols defined in Table 1.

In scene layout-switching graph, the token in Petri net is replaced by some tools indicating the scene layout, while the transition is merged with interaction transmitter (control tool) and interaction receiver (message buffer). A corresponding scene layout-switching graph of Figure 10 is shown in Figure 11 to depict their differences.

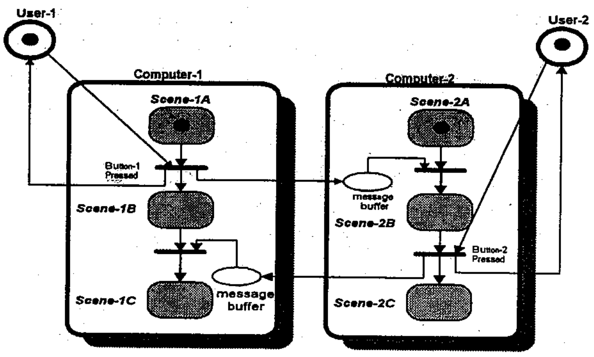


Figure 10 Scene switching graph of multi-user interactive architecture

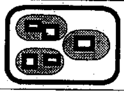
Functions	Symbols in Petri net	Symbols in Scene layout-switching graph
a general tool	not defined	▭
a scene	a place ⬬	
a computer containing several scenes	not standard	
a user's operation	a place ⬭	a control tool ⬭
a message buffer for remote message passing	a place ⬭	⬭
a (remote) control message	a token ●	scarely used
a (remote) message passing	a directed line ↓	a directed line ↓
a user's operation / a message	a transition	merged with the control tool / message buffer

Table 1 Symbol definitions in Petri net and scene layout-switching graph

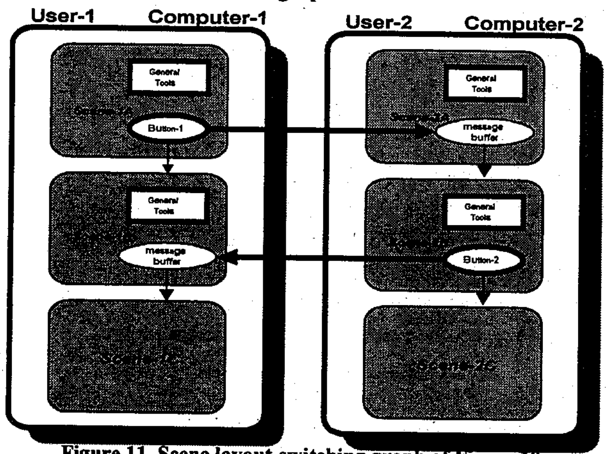


Figure 11 Scene layout-switching graph of Figure 10

With the above scene switching and scene layout-switching graphs, the scene switching of a CSCW system as well as its interactions can be easily analyzed. By the interactive script control, there are two basic scene switching models: centralized scene switching and interactive scene switching.

In centralized scene switching mechanism, all scene switchings are controlled by a certain user. Other users have no rights to determine their own scenes, all of which are all governed by this special user. The use occasions of such architecture is teaching and medical diagnosis. This kind of design is simple and easy to be obtain consistent scenes, but is lacking of flexibility and unable to attain two-way interaction. Figure 12 shows an example scene switching graph of such architecture.

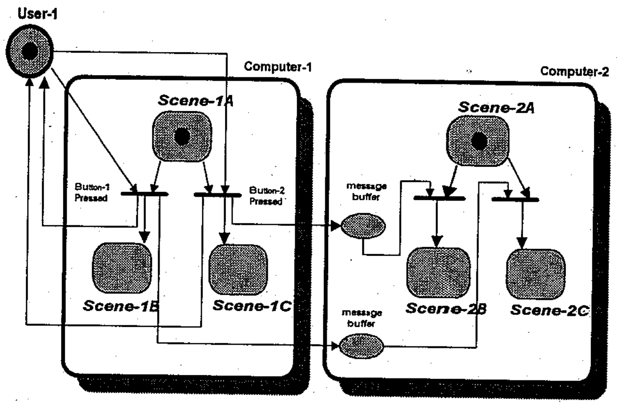


Figure 12 Scene switching graph of centralized scene switching model

On the other hand, in interactive scene switching model, each of the users is able to influence other users' scene switchings, as Figure 10 shown. This makes effective interaction and communication among users, but may cause inconsistency at scene switching. Such scene inconsistency can be avoided by some coordination mechanisms. [17]

4. MISL (multi-user interactive script language) design

After analyzing scene switching mechanism in interactive scrip of CSCW systems, it is possible to design a script language to employ the above multi-user interactions. Here, a *multi-user interactive script language* (abbreviated as MISL) based on the previous model is proposed to construct CSCW systems. The syntax of MISL is listed in Table 2.

CATEGORIES	FUNCTION DESCRIPTIONS	SYNTAX
Script structure	beginning of a script	BEGINSCRIPT
	end of a script	ENDSCRIPT
	stop script control	STOP
	comment	REM comment
Resource share	resource block	{RESOURCE}
	set server for tools	SERVER server
	set path for tools	PATH path
	alias of the filename	ALIAS alias file
[illegible]	beginning of the scene	SCENE {scene}
	end of the scene	ENDSCENE {#scene#}
	switching condition	SWITCHCONDITION
	a tool in a scene	{scene}.tool
[illegible]	newly set the tool	LOAD alias tool pos. [other-para.]
	remove the tool	UNLOAD {scene}.tool
	hide the tool	HIDE {scene}.tool
	show the tool	SHOW {scene}.tool
	move the tool	MOVE {scene}.tool

Scene switch	selective control	IF condition THEN
		ENDIF
	remote control message to other	REMOTEMESSAGE
	scene switching	SWITCHTO scene
Control message conditions	message from button	PRESSED = button
	message from timer	EXPIRED = timer
	message from menu item	CHOSEN = menuitem

Table 2 Basic Syntax design of MISL

With the above language syntax, it is possible to develop an interactive script program. As shown in Table 3, a general script program structure is divided into two parts: the initialized resource declaration block and a serial scene definition blocks. As mentioned, the *resource declaration block* defines several utilities of tools, including server, directory path and alias of tool's filename. Also, each *scene definition block* in a script is composed of two parts: scene layout and scene switching. The *layout* part of a scene illustrates what appearance the screen is for this scene. It arranges every tools appropriately on the screen by basic scene arranging actions (**LOAD** and **SHOW**). The *switching* part of a scene defines all its switching conditions and the corresponding next scenes (post-conditions). The *switching conditions* contain a series of selected statements **"IF condition THEN action-statements ENDIF selective statement"**, whose **conditions** have three *triggering conditions*: button, timer and menuitem and each **action-statement** may have the following three components:

(1).*next-scene switching*: using **SWITCHTO** instruction.

(2).*post-conditions*: including a series of scene clearing actions (**UNLOAD**, **HIDE** and **MOVE**).

(3).*remote message sending*: using **REMOTEMESSAGE** instruction.

```
BeginScript
    {Resource}
        Server
        Path c:\XXX\XXX
        Alias XXX c:\XXX\filename.exe

    Scene  {Scene1}
        Load  XXX "xx,yy,ww,hh"
        Load YYY "title1,xx,yy,ww,hh"
    SwitchCondition
        if PRESSED=button1 then
            SwitchTo {Scene2}
            RemoteMessage message1
            Unload XXX
        endif
        if CHOSEN=ch1 then
            SwitchTo {Scene3}
            RemoteMessage message2
            Unload YYY
        endif
    EndScene  {#Scene1#}

    Scene  {SceneN}
        Load  XXX "xx,yy,ww,hh"
        Load YYY "title1,xx,yy,ww,hh"
    SwitchCondition
        if PRESSED=buttonN then
            SwitchTo {Scene2}
            RemoteMessage messageN1
            Unload XXX
        endif
        if CHOSEN=chN then
            SwitchTo {Scene3}
            RemoteMessage messageN2
            Unload YYY
        endif
    EndScene  {#SceneN#}
EndScript
```

Table 3 MISL program structure

When executed, an interactive script starts with declaring resources, including server machine, tools' path and filenames' aliases. Subsequently, the scenes in the interactive script will interact with ISC to achieve the human-computer interactions and computer-computer interactions, as shown in Figure 9. In the beginning, the foremost scene block of the scene definition part is the first scene to be executed. However, the execution order of ensuing scenes depends not on the programmed order of scenes, but on the scene switching mechanism. When each scene is executed, only the preconditions (scene layout) are satisfied, i.e. the screen will be arranged as defined layout. Meanwhile, the control right of computer belongs to its user. If the user's operation (control message emitted from menu, button or timer-expiration) satisfies the switching condition of any scene, this scene will be the next scene and its corresponding actions, as mentioned post-conditions, remote message sending and new scene switching, will be executed.

The user's control message in the precondition is so-called human-computer interaction; whereas, the remote control message is computer-computer interaction. Both of these interactions should be analyzed by use of the above scene layout-switching graph beforehand.

Beside the above human-computer and computer-computer interactions, there still exist interactions between the scripts and ISC. When user operations are transformed into control messages (m1 or m2 in Figure 9), ISC should check each of the switching conditions to determine the next scene, then to proceed its post-conditions, including current-scene clearing, new-scene switching and remote-message sending. When sending remote messages, an ISC in one computer will connect with the ISC in another computer. This means the computer-computer interactions are ipso facto realized by the interaction between ISCs.

In summary, the tasks of ISC can be itemized as the following:

(1).Interpretation of all the preconditions and post-conditions read from the assigned scene in the script.

(2).Execution of these preconditions and post-conditions, including dismantlement of tools in old scene and establishment of tools in new scene.

(3).Communication with other ISCs through remote messages, and last but not the least.

(4).Suspension of control to let user operate the scene's tools amongst the scene switching, that is, waiting for the control message from control tools.

By this reason, the implementation of ISC is not suitable by a compiler, but by an interpreter, which is possible to meld the execution of the script with human-computer interaction.

In essence, ISC is an control program, which manages all the other programs (tools) in the screen. These management rule is written in the designated script. Hence, strategically this ISC program (interpreter) has to be run first of all. Then an interactive script is selected to execute. Sequentially, all the tools are loaded/cleared one by one with prescribed positions and attributes. In the meantime, each of the users/operators can get involved in front of his/her own computer.

The interpreter implementation of the script control brings out many difficulties in checking the legality of an interactive script. An interactive script for being smoothly executed has to satisfy the following constraints:

(1). All used tools in all scenes, either control or general, must be existent when being loaded.

(2). Each user interface of control tools must be effective, that is, all its post-conditions are executable.

(3). All scenes are reachable by some switching conditions.

(4). The destination for passing the remote control messages must be reachable.

The second and third constraints are related to the execution flow of the script and can be checked when analyzing through Petri-net or scene layout-switching graph. The next section will show examples. On the other hand, the first and fourth constraints can only be detected while executing. This suggests the user (the operator, not the programmer of the script) to check these execution environment before the use of interactive scripts.

5. Implementation and example system

This section builds a complete system to demonstrate the practicability of multi-user interactive scripts and their interactions with ISCs. This demonstration system is a distributed distance education environment in social learning, called *Electronic Classroom.* [10] Within this system, the ISC is implemented as an independent interpreter program and there are two kinds of scripts: teacher's and student's. The designated course is BCC (Basic Computer Concept), which can learned through several designated scenes (learning modes).

The platform of the above system is IBM compatible Personal Computer with CPU faster than Intel 486 DX2-66. The operating system can be Microsoft Windows NT or Windows 95. The basic networking mechanism is Ethernet, but WAN (Wide-Area Network) and PSTN (Public Switching Telephone Network) are both possible since the demonstration system is built on the popular TCP/IP protocol and Windows socket interface. There are more than three machines in this system, one for the teacher and others for students; this constitutes a classroom of some course in computer network, as Figure 13 shown. In general, this system allows more than one course held at the same time. Moreover, this implementation includes several media, including text, audio and image, and even video is possible only if the bandwidth is broad enough.

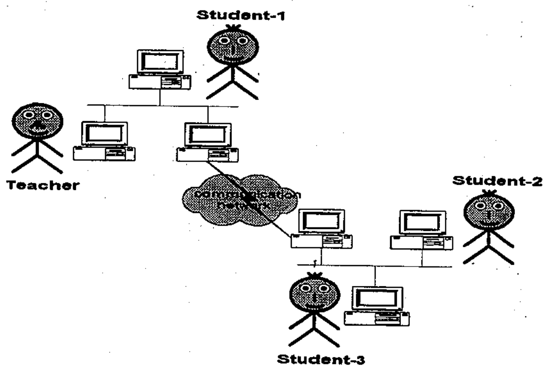


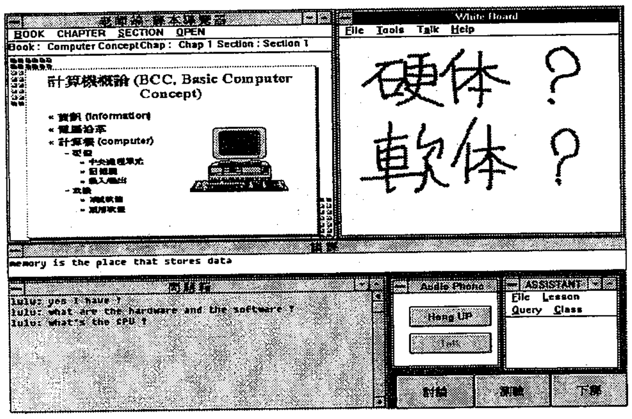
Figure 13 Electronic Classroom on computer network

The ISC interpreter is implemented in Borland C++ 4.0. As all tools of user interface are designed separately, they can be implemented in other programming languages. For the time being, some are implemented in Visual BASIC such as textbook, some are in Borland C++ 4.0 such as whiteboard and coordinator, and some are in Microsoft Visual C++ 2.0 such as audio phone. The integrated coordinator is a manager to perform the overall coordination works, such as group management, access control and floor control. [17] Many real-time multimedia toolboxes are devised to process some cooperative works, such as whiteboard, audio phone, question box. [12] All these tools work in the control of the ISC and its interactive script.

In general, distance education is a tutoring and learning style, in which teachers and learners are separate. Recently, social learning system [2] is proposed as one kind of environment where computer-simulated or real agents works at the same computer or across computer network to process learning activity. Hence, a distributed social learning system supports a group of persons in different locations tutoring/learning together through computer network.

According to OCTR learning model [3], four *learning modes* are used in our distance education environment: lecture (including inquiry), discussion, test and self-education. All these learning modes are designed as scenes in interactive scripts. To complete the operation of the whole system, a *login scene* is introduced in this design. Scenes of the teacher and students and their switching have to be kept consistent, and the corresponding screens can be carefully laid out. All these works are done through the previous scene layout-switching graph.

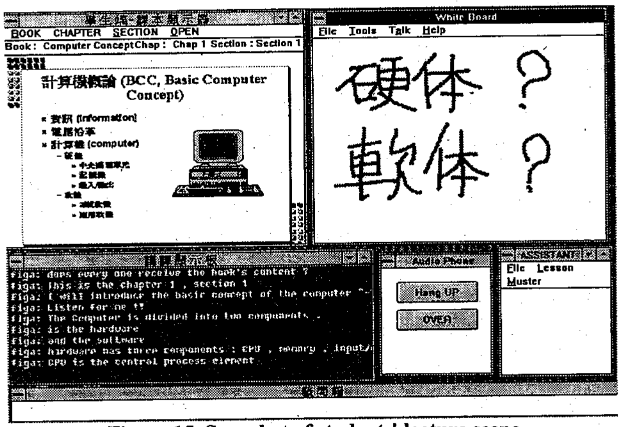
The above system provides a practical, feasible and low-cost distance education environment. By real experiment, this system can work well through network or telephone line. All the process is recorded as a video tape and Figures 14 and 15 shows the snapshots of teacher's and student's screen in lecture scene.



memory is the place that stores data

lulu: yes I have ?
lulu: what are the hardware and the software ?
lulu: what's the CPU ?

Figure 14 Snapshot of teacher's lecture scene



figa: does every one receive the book's content ?
figa: this is the chapter 1 , section 1
figa: I will introduce the basic concept of the computer ^~
figa: Listen for me !!
figa: The Computer is divided into two components .
figa: is the hardware
figa: and the software
figa: hardware has three components : CPU , memory , input/
figa: CPU is the central process element

Figure 15 Snapshot of students' lecture scene

6. Conclusion

Up to now, an interactive script model of CSCW systems is investigated and its script language is developed. The study of this interactive script model of CSCW systems includes the analysis of its interaction interface, the tools and scenes for different working environment. To represent the layout of a scene and the interactions among scenes, the scene layout-switching graph of a script is proposed. For the establishment of such CSCW system, a Multi-user Interactive Script Language (MISL) is designed and its interpreter is constructed as the interactive script control. Finally, based on such a developing environment, we build a real CSCW system, the distance education system with five learning modes. This demo system is built on Wintel (Windows and Intel) system and performs well on Internet and PSTN (Public Switching Telephone Network).

References

[1] H. E. Bal, J. G. Steiner and A. S. Tanenbaum, "Programming languages for distributed systems," *ACM Computing Surveys*, Vol.21, No.3, Selp.1989, pp.262-231

[2] T.-W. Chan, "A tutorial on social learning systems," *Emerging Computer Technologies in Education* (edited by J. Self and T.W. Chan), AACE, 1996

[3] T.-W. Chan, C. Lin, S. Lin and H. Kou, "OCTR: A model of learning stages," *AIED-93*, 1993, pp.257-264

[4] P. Dewan and R. Choudhary, "A high-level and flexible framework for implementation multiuser interfaces," *ACM Transaction on Information Systems*, Vol.10, No.4, 1992, pp.345--380

[5] A. Dix, J. Finlay, G. Abowd and R. Beale, *Human-Computer Interaction*, Unalis Co., Taiwan, 1993

[6] C. A. Ellis, S. J. Gibbs and G. L. Reln, "Groupware: some issues and experiences," *Communication ACM*, Vol.34, No.1, Jan. 1991, pp.36-58

[7] C. Ellis and J. Wainer, "A conceptual model of groupware," *CSCW'94*, Chapel Hill, 1994, pp.79-88

[8] M. Green, "A survey of three dialogue models," *ACM Transaction on Graphics*, Vol.5, No.3, July 1986, pp.244-275

[9] S. Greenberg and D. Marwood, "Real time groupware as a distributed system: concurrency control and its effect on the interface," *CSCW'94*, Chapel Hill, Oct. 1994, pp.207-217

[10] J.-H. Ho, W.-H. Lin, J.-S. Heh and T.-T. Wu, "An interactive distributed distance education environment," *ED-Media'96*, Boston, 1996, p.363

[11] T. Kirsche, R. Lenz, H. Luhrsen, K. M. Wegener, H. Wedekind, M. Bever, U. Schaffer, C. Schottmuller, "Communication support for cooperative work," *Computer Communication*, Vol.16, No.9, Sept.1993, pp.594-602

[12] J.-S. Lin, P.-C. Chang and J.-S. Heh, "Building a real-time multimedia system through an efficient replication algorithm," *HD-Media'95*, Taiwan, 1995, pp. OA-3-13-18

[13] J. Palme, "Standards for the asynchronous group communication," *Computer Communication*, Vol.16, No.9, Sept.1993, pp.532-538

[14] H. Peng, "A study of the nature of human-computer interaction," *Computer Graphic Workshop*, Taiwan, 1994, pp.121-125

[15] S. M. Shatz, *Development of Distributed Software Concepts and Tools*, Macmillian Pub. Co., 1993

[16] I. Tou, S. Berson, G. Estrin, Y. Eterovic and E. Wu, "Prototyping synchronous group applications," *IEEE Computer*, May 1994, pp.48-57

[17] C.-H. Yuan, *Research on Integrated Coordinator of CSCW Systems*, Ms. thesis, Dept. of Info and Comp. Eng., Taiwan, 1995