Database Support for Computer Supported Cooperative Work

Yahiko Kambayashi
Shin’ichi Konomi
Osami Kagawa
Sang-Hoon Lee
Werner Winiwarter

Department of Information Science
Kyoto University

Michiyoshi Hayase
Okayama Prefectural University

It is possible to realize systems supporting cooperative work among users in distance location utilizing recent network technology. Most current computer supported cooperative work (CSCW) systems, however, offer identical display views to all the users through video in order to simulate real world. We use database functions and non-video based communication tools to realize such cooperative work functions by providing different display view to each user, in order to realize powerful CSCW systems. Applications in VIEW(Virtual Interactive Environment for Workgroups) project include VIEW Office, an office system, VIEW Classroom, a distance education system, and VIEW Conf, a conference system. VIEW Media is a multi-user distributed hypermedia system for interactive presentation that provides media services as a basic communication facility in VIEW applications.

1. Introduction

Due to the down-sizing of high performance computers and the progress of computer network technology, distributed computer systems have been used widely. One of the promising applications of such systems is support of user cooperation, called CSCW (Computer Supported Cooperative Work). We have been developing VIEW (Virtual Interactive Environment for Workgroups) to explore effective use of database management systems in CSCW. Applications in VIEW include VIEW Office, an office system (it was called VirtualOffice in [11] but renamed since virtual office becomes a common term), VIEW Classroom, a distance education system[4, 6, 12], and VIEW Conf, a conference system[7]. VIEW Media[6] is a multi-user distributed hypermedia system for interactive presentations that provides media services as a basic communication facility in VIEW. Various new functions are added to conventional hypermedia systems utilizing database capability with extended view functions and also utilizing the concept of deputy objects[5, 9, 10].

Conventional CSCW systems utilize computer-mediated communication tools such as window sharing systems, text, audio and video communication tools, bulletin board systems, electronic mail systems, or file transfer systems. VIEW offers various functions which are not central issues of conventional CSCW systems. Some of them are as follows.

(1) Extensive use of database functions to realize flexible sharing and reuse of information.
(2) Use of non-video technology for information selection.

Most of conventional CSCW systems utilize video based technology, which is suitable to realize awareness. We cannot use only video technology, since there are the following problems. (a) high communication costs, (b) use of video may cause some security/privacy problems, (c) it is hard to find out necessary information. We will use non-video technology to select information to handle security/privacy problem. Furthermore, we will try to use symbolic representation to realize awareness in an abstract manner, which can be easier to be understood than direct representation by video systems. We believe a proper combination of these functions with conventional CSCW systems utilizing video mechanisms will be useful to realize practically useful systems.

Some new functions developed in VIEW project are as follows.
- Personalization facilities using generalization of database view functions
- New model suitable for flexible cooperation
- Control functions and observe functions for cooperative work
- Combination with active functions
2. VIEW Office

As requirements for VIEW Office [11], a virtual office system, are rather easy to understand, we will describe organization of VIEW Office in this section. The main objective of VIEW Office is to support personal and cooperative work in distributed environment. The following pairs of opposite properties are handled in integrated way.

1) Real office and virtual office
2) Public space and personal space
3) Physical location and logical location
4) Real and virtual equipment, real and virtual users

Public Space and Personal Space: Public space in the prototype of VIEW Office is shown by a floorplan. The interface resembles a map of an office and is what we refer to as the Floorplan. The Floorplan is a bird's eye view of the office layout with desks, walls, equipment, etc. represented by rectangles, lines, and icons (see Figure 1 (b)). Different users may have different views of the Floorplan, due to the fact that some objects, or even rooms may, for security reasons, be hidden from some users. User status may be shown by the corresponding user icon, by which other authorized user can know whether he is busy and what kind of work he is engaging in. The Floorplan can include virtual rooms and virtual objects. A virtual room may not physically exist in the real office, but be added in VIEW Office so that a group of users have a common working area which provides an environment for cooperation in. Each user has one or more personal spaces which consists of desks, bookshelves, communication equipments etc. It is similar to a user's private desk in a real office. A user can arrange everything as he/she pleases by e.g. placing files in drawers or folders, leaving them out on his/her desk as a reminder, etc. In VIEW Office a user may have several different personal spaces, for example, one for each project.

Important Issues: The following issues have been emphasized in VIEW Office.
1) User friendly interface
2) Integration with communication equipment
3) Communication among multiple users
4) Construction of personal office space
5) Security
6) Location transparency of office elements

In order to realize above characteristics, the object-oriented data model was selected.
- Private data and public procedures (methods) are encapsulated as an object, data can be accessed only through the object's methods. The definition of an object type is called a class. - A class can inherit the types of data and methods from other classes. This mechanism forms class hierarchies.
- An object is created as an instance of a class.
- Computation proceeds by sending messages between object's methods.

Office Elements: Office elements are as follows.

Space: An element which other elements can be put on, or can include other spaces. For example, rooms or document folders are Space.

Equipment: An element which realizes some functions. Equipment perform active actions. For example, FAX machines or telephones are Equipment.

Data: A form of information to be manipulated by Equipment. Data is created, modified, referred to or deleted by Equipment. Simple documents or documents with Figures are Data. Data are also stored in Space.

User: An element which enters a Space and manipulates Data by operating Equipment in the Space.

Composite Element: An element which integrates the functions of different office element types, i.e. Space, Equip, Data and User. This element can not be defined by itself, it is described by specifying the corresponding classes. The FAX machine mentioned above is defined as a FAXMachine inheriting from the Folder class and the FAX class. Another example is TelwithAddressbook which is defined as a class which inherits from the Tel class and the Document class.

Methods: The above classes form a hierarchy. Methods corresponding to each class are as follows.

Methods for Data: Primitive methods such as read, write, update or access privilege control.

Methods for Equipment: Active methods to manipulate the data. For example, a FAX machine object communicates with the data object and reads the contents of it, then send the data using a FAX modem.

Methods for Space: Passive methods to manage the contents changed by user activities. For example, when the user opens a Space using the open command, methods of it display the contents included at the last closed time. Furthermore, it can control whether the objects included in it should be shown to a user or not.
Methods for Users: These are not defined clearly because a User can be also a high-level Equipment. Although these definitions are required when coordination among users and programs are realized, they are not handled in this paper.

Office Space Description: In order to define office space, office space description language based on object-oriented model is introduced. It is difficult for low-end user to describe them. Each user needs to describe his/her own office space using parts provided in the parts definition part. In this way, it is possible to provide the environment where a user buys telephone and a document folder, put them on his/her desk and use them as if they were real office elements. In addition, this office description makes it easy to interchange office space among users. Office space constructed by a user can be shared by other users with this language interface. Furthermore, GUI to describe this language is supported. This is because the user interface of the description of language is text-based and it is not suitable for low-end users.

The followings are special features.

Shared Objects: Different from a real office, the same object can be shared by more than one space in the virtual office. An example is a case when a document used by all members of one group can be put on each member’s desk.

Virtual Space: In a real office, there are many cases when multiple projects are processed in one room. In VIEW Office, however, a user can assign one room to each project he/she is working on by defining a virtual space using the office space description language. Then a user can work in the virtual rooms, where only documents required for the project are available.

Security Management: In a real office, a secret document can be put into a locked drawer so as not to be seen by other persons. However, the existence of the secret thing cannot be hidden in this way because a person who finds the locked drawer may suspect that there might be a secret thing in it. In VIEW
Office, a mechanism for hiding even the existence of a secret thing should be realized.

3. Action View and Action History View Mechanisms

Generalized view mechanisms called action view and action history view are introduced in VIEW to support coordination among users. View for databases is intuitively defined as results of composition of selected data. View for objectbases can be similarly defined, where objectbases store objects which consist of data and methods. Database actions are defined as a combination of data and how operations are applied to the data. Database actions at other locations are required for user coordination. A user may perform various operations, but another user working with him may be able to see a subset of such operations due to security/privacy constraints. For example, assume that users A and B are working together for project 1, user A also works for project 2 while user B does not, operations related to project 2 performed by user A should not be seen by user B. Action history is defined by a history of such user actions. A part of such history is called an action history view.

Action history can be realized by video technology. We can record display contents of each workstation. Using conventional data compression technology such as MPEG, we can reduce the storage size for action history. It is not possible, however, to realize flexible views, since the selection of data and operations is almost impossible. Thus we will use symbolic recording for action history. Operations are stored in symbolic form in the history together with data when modified.

Typical advantages of symbolic recording method over video based recording method are as follows.

1) Part of operations and data can be selected so that security/privacy constraints can be realized.
2) Elimination of logically redundant operations is possible. A pair of an operation and its undo operation can be eliminated together. Intermediate locations of a mouse are not required to be recorded.
3) Retrieval of a partial history can be specified by a high-level language. For example, "find the part where operation a is applied to objects b and c".

Although the amount of data required to store action history by symbolic recording is much smaller than that by video recording, we need to erase unnecessary history as much as possible. Recording of operation history is not new, but the authors believe that introduction of view for actions is not considered before.

There are the following two well-known methods to support coordination of users working at remote workstations.

Video cameras: Awareness can be realized easily. Communication cost for video is much higher than that for characters.

Control and observation of remote workstations: With this system, two users can work together by displaying the screens of the both workstations.

Although the both methods used widely, neither of them offers view functions. We cannot select part of objects in the screen. It is important, since some window may display user's personal mails, which should be kept secret from others.

Selection of objects to be displayed: If a part of displayed objects should not be seen by others due to security or privacy reason, we must hide these objects.

Change of object locations: Each user may have his/her own preference on locations of objects. For example, object arrangement in one workstation can be different from object arrangement in another workstation.

Change of display method: For example, if a user wants to see large characters due to eye problem, he/she can change character size as well as window size.

These kinds of view functions are not supported by the above two methods. These are related to personalization facilities to be discussed in the next section. The concept of deputy objects is introduced to define strongly related, but not identical, objects. Deputy object b corresponding to object a can have different values, different attributes and different methods. For cooperation among users, object view is not enough. Changing process of objects is also important to be displayed. This kind of view is called action view. It should be noted that if selection of objects is defined by a logical expression, an object may disappear or appear by the result of modification. We also define action history view, which is a part of action history. Action history view is defined by a language based on regular expression. Right side of Figure 1 (b) shows usage of such a language.

4. Model for VIEW Systems

We showed two typical applications of VIEW in order to discuss requirements for VIEW model.

Requirement for Personalization: In VIEW Office it is shown that each user can have his/her own personal work spaces. We will use another example, distance education systems, to explain how personalization is important. We assume that distributed hypermedia documents are used for teaching. The following functions are required.

1) Students cannot see some of teaching materials prepared for teachers.
2) Each student can have his own notebook to write his private comments.
3) During a class each student can visit digital library in order to know some details of the subject being taught
4) For each user, environment is defined in order to specify the power of each user. Users can share identical environment. For example, a teacher can be in one environment and the students can be in another environment.
5) Environment can be changed by the owner of the environment. For example, when the teacher wants give a test, students cannot visit digital library nor open the teaching material. In this case the teacher must be the one of student environment.
6) In the student environment each student can define his own environment. There can be a hierarchy of environments. If the owner of the environment permits one can visit his environment to see his own notebook.

In conventional systems, it is possible to share identical data by all the users due to the WYSIWIS (What You See Is What You Get) principle. Also it is possible to define personal data for each user. The purpose of the personalization facility of our system is to support intermediate data sharing among users in a group. Part of data is shared and their visualization process can be different as shown in the previous section.

It is important to realize adaptability for various application domains as well as various kind of users. We have developed object deputy model to realize personalization facilities. The model was introduced to handle some important problems uniformly, multiple inheritance, object role and object migration, of object-oriented databases.

**Deputy Objects**: An object can have its deputy object. Each deputy object can have its own attributes and methods as well as an object identifier. Since a deputy object can also have its own deputy objects, there is a hierarchy of deputy objects as shown in figure 2. A set of deputy objects with similar property can form a class. Class hierarchy of deputy objects is also defined.

**User objects**: User objects are objects that represent users in a hypermedia system. User objects are different from users in a sense that they can respond to messages automatically, however, some messages are delegated to real users. For example, when someone need to know your name, he/she might either send message “name” to your user object or send an email massage asking your name to your user object. In the latter case, it actually forwards the message to you.

Attributes of user objects include:
- **Identity**: full names, gender, profiles, portraits, addresses, participating groups
- **Security**: access control lists
- **Current activities**: live data (video, audio, text, pointing, navigation), busy/idle status
- **Past activities**: historical data (video, audio, text, pointing, navigation)
- **Future activities**: plans, schedules
- **Personal data**: personal comments, underlines, scribbles, notebooks

An access control list of a user object controls access to the user object itself. It associates with groups of users and certain levels of security to attributes of user object. Current activities of users produce values of attributes online. Thus, the attributes are almost always updated. An example of this kind of attributes is the current x-y co-ordinates of the pointer of a speaker.

**Environment**: Environments are also defined as objects. Environment objects can specify some attribute values of deputy objects instead of the original objects. Environment objects can even override values included in display objects. For example, users can force word "databases" to be displayed in red. In this case, whenever term "databases" appear on their screens it is displayed in red regardless of its original color, i.e., environment objects defined environments and can override attributes of objects that visit the environments. Environments that overrides attribute values are called strong environments and environments that does not override attribute values and just fulfill empty attribute values are called weak environments. As environment also defines restrictions on method applications, if a user moves from one environment to another, his capability to apply methods to data may be modified. There can be sub-environment in an environment and each sub-environment inherits characteristics of the parent environment. Definition of views can be also put into one environment and it is modified by the influence of the environment.

There are the following users for each environment.
- **Owner**: The user who has a right to change the following three sets.
- **In-Out**: The set of users who can get into the environment.
**Administrator:** a set of users who can customize the environment.

**Creator of the sub-environment:** Each user in this set has a right to create sub-environment in the environment.

---

**Visualization:** In order to cooperate with other users, we may need to see these uses in some extent. We have the following visualization facilities, since using video is not always the best way to realize cooperation, since if the number of users is more than some value, it is difficult to display all the users. Furthermore to watch user actions becomes also difficult.

The following visualization facilities are being developed.

1) Visualization of user status
2) Visualization of environment
3) Visualization of relationships among users and environments

---

Two typical methods to represent environment and user status are shown in Figure 3.

**Display Environment Objects:** As a special environment, display environment objects are defined, which specify characteristics of displays and requirements of users together with global constraints on display objects.

1) Characteristics of displays: Monitors limit display-related attributes such as numbers of colors, number of pixels, resolutions, and vertical/horizontal ratio. Also, CLUT (color look-up table) constrain available colors. Display environment objects of this category are applied to displays of a certain group of users.

2) Global constraints on display objects: Some privileged users such as owners of channels can specify global display environments. For example, speakers of presentations can specify global display environments for presentation materials that are shared with audiences. Display environment objects of this category are applied to displays of a certain set of semantic objects.

3) Requirements of users: Each user can define his own display environments. Display environment objects of this category are usually weaker than those of the other categories.

**Observability:** Visualization facility are important tools to observe other users' work. The concept is generalized as observability, which consists of the followings.

1) User status observability
2) Method application observability
3) Environment observability

If all information is obtained by observability facility, it is completely observable, but usually only partial information is observable (partially observable). As we cannot express some information by visual methods, usually observation through visualization facilities is partial observable. Synchronous and asynchronous observability can be defined to support synchronous and asynchronous awareness, respectively.

**Controllability:** In order to restrict user operations, control functions are prepared. There are typically the following kinds of controllability.

1) Method application controllability
2) Environment controllability:
   - There also should be time constraints on controllability.
Active Mechanisms: In order to realize dynamic property utilizing such control, active mechanisms are required. Such mechanisms are realized by ECA(Event Condition Action) mechanism. Variables in event and condition definitions must be observable and only controllable actions must be used in the process of defining ECA mechanisms.

Figures 4 and 5 show display copies of VIEW Media and VIEW Classroom, respectively.

5. Concluding Remarks

In order to realize hypermedia systems for distributed applications, several new concepts are required, such as deputy objects, environment, action views, action history view, visualization and observability, controllability. Motivation of these concepts is discussed in this paper. As many users work together, we need to handle conflicts among users and systems, such as conflict between data in an object and data inherited from others, conflict between environment objects and objects inside. Such a problem is one of the important future research topics.

Acknowledgement

The authors thank to Mr. K. Sakata, Miss C. Ito and Mr. Y. Yokota for their help in developing the system. The work is supported by the fund from Information Technology Promotion Agency, Japan.

References


