Cooperative View Mechanisms in Distributed Multiuser Hypermedia Environments

Shin’ichi Konomi, Yusuke Yokota, Kazuhiro Sakata and Yahiko Kambayashi
Department of Computer Science, Kyoto University
Yoshida-Honmachi, Sakyo, Kyoto 606-01, Japan
{konomi,yy,sakata,yahiko}@kuis.kyoto-u.ac.jp

Abstract

Distributed multiuser hypermedia environments provide not only information sharing mechanisms but also user collaboration/communication facilities. The provision of integrated views of heterogeneous information resources is necessary to create common understanding among the users who are possibly distributed in terms of geography and time. However, requirements of customization must also be considered since such diverse users would want to personalize views of shared information. In order to integrate views considering requirements of flexible customization, we propose cooperative view mechanisms of Dexter-based hypermedia systems, introducing environmental objects and their participation relationships. The mechanisms instantiate deputies of hypermedia on screen using participation relationships of users, hypermedia components, and environments. The relationships are also used for the purpose of supporting awareness. Using a novel user collaboration facility, the relationships are visualized so that users can easily recognize other users and/or user groups having the same, slightly different, or very different views. Attributes of environment objects are discussed so that the mechanisms can be effectively utilized in computer-supported cooperative work settings such as distance presentations, virtual offices and virtual classrooms.

1. Introduction

Computer-supported cooperative work (CSCW) systems are rapidly becoming a key to realizing successful virtual organizations in the age of global information infrastructure. In many CSCW research papers, the issues concerning user communication and distributed information sharing are addressed. Hypermedia technology is often used to provide user-friendly interfaces to distributed information sharing for cooperative workers.

Certain multiuser hypermedia systems [4, 10] provide views based on the WYSIWIS (What You See Is What I See) principle, i.e., all users share the same view, which would be useful for them to create common understanding. Conventional approaches to information sharing in CSCW systems generally focus on the provision of the same environment for all users, utilizing techniques such as data replication or common data models.

Customization is also an essential feature in many CSCW systems [3]. Cooperative work such as a distance presentation involves participants of different roles. In a distance presentation scenario, speaker participants and audience participants would dynamically customize various aspects of shared hypermedia. For example, while the speaker talks, each audience participant would customize the shared hypermedia materials by adding comments, by reducing the window size, and/or by autonomously navigating on hyperlinks to visit a digital library.

Certain customization should be visible to a group of users. In the distance presentation scenario, audience participants actually include ten English participants and five Japanese participants. The English participants display presentation materials in English (in this case, the original language), while Japanese participants customize the presentation materials so that they are displayed in Japanese, using English-Japanese translation software. When the speaker forces all participants to display a document, everyone displays the same document, however, in different languages. Only a few Japanese participants who can read both Japanese and English are permitted to change the languages used on their screen.

In this paper we address not only the issue of synchronizing views but also the issue of customizing views in distributed multiuser hypermedia environments. Customization of hypermedia can be performed by adding, removing,
aggregating and/or decomposing hypermedia components (nodes and links), by adding and removing comments to/from hypermedia components, by applying navigation histories to hypermedia components, and/or by tailoring user interfaces. Users should be able to dynamically create, remove and/or change customized views and also dynamically control permissions and degrees of customization since it is desirable that users and their groups can change the ways of working together whenever they want.

In the distance presentation scenario, the English participants start group discussions by two groups, each of which includes five people. In the group discussions, hypermedia materials are synchronized in each group, and when the discussions are finished, all participants’ display is synchronized with the speaker again.

In order to realize such dynamically customizable hypermedia, cooperative view mechanisms of hypermedia are proposed in this paper. Cooperative view mechanisms of hypermedia should be able to:


2. Dynamically instantiate views considering the specifications of collaborative work environments, and

3. Provide communication aids to minimize the difficulty of cooperation which can be introduced by customization.

Current hypermedia systems lack flexible virtualization mechanisms since their models are too simplistic to deal with different representations of hypermedia at the user interface level. [26] and [7] discuss virtualization of hypermedia, focusing on the use of logical queries as well as physical links. In [18] we presented basic concepts for applying the deputy model of objects [19] to the Dexter hypertext reference model (Dexter model) [8, 12] in order to realize advanced flexible virtualization by modeling incremental virtualization of hypermedia and customization of user interfaces. Here we extend the concepts so that deputy hypermedia components and deputy hypermedia sets provide such flexible reorganization mechanisms that fulfill requirements of certain important CSCW scenarios.

Each user and each group of users would often change their view definitions while participating in sessions of hypermedia-based collaborative work. In our cooperative view mechanisms, user group and dynamic changes of view definitions are managed by the use of environmental objects. Views of hypermedia are dynamically generated based on the participation relationships of users and the environments.

Cooperative view mechanisms should allow users not only to customize shared hypermedia but also to facilitate communication among users. However, customization of shared hypermedia makes communication difficult as a result of permitting each user to have different views of the shared hypermedia. For instance, two participants of distance presentation, each reading the presentation materials in Japanese or English, would have difficulty in discussing a specific topic in the presentation even if they both speak English fluently. We provide communication aids by supporting awareness of customized hypermedia. By generalizing the basic ideas presented in [23], participation relationships of users and environments are visualized together with deputy hypermedia.

Basic functions of the cooperative view mechanisms are implemented in the current prototype of VIEW Media, a collaborative hypermedia system. Based on the experiences of the prototype, we are improving VIEW Media so that advanced functions can be used efficiently and most of the essential functions of cooperative view mechanisms are realized.

2. Components of Multiuser Hypermedia Environments

First we introduce the three-layers structure of the Dexter hypertext reference model. In Figure 1, the storage layer corresponds to databases (DB) containing hypermedia nodes and links. Presentation specifications (PSpec) and anchors (Anchor) provide interfaces to the runtime layer and the within-component layer, which correspond to user interfaces (UI) and external applications (Application), respectively.

![Figure 1. The Dexter Model](image)

In the Dexter model, hypermedia documents are simply sets of component objects in the storage layer. Here we define hypermedia set $H$ to be a set of hypermedia components $c_{01}, c_{02}, ..., c_{0n}$:

$$H = \{c_{01}, c_{02}, ..., c_{0n}\}$$

(1)

Component object $c_{0i}$ ($1 \leq i \leq n$) consists of a unique
identifier (uid), a presentation specification (ps), a set of anchors (an), m miscellaneous attributes (a1, a2, ..., am) and the contents (cont). ps, an, and a1, a2, ..., am are called component information (ComponentInfo). A component object looks like the following:

\[ co_0 = (uid, ps, an, a_1, a_2, ..., a_m, cont) \] (2)

cont is the primary data of co0, which is data of various media types such as texts, images, sounds, videos, link specifiers, or composites. Note that miscellaneous attributes in ComponentInfo may include keywords and node/link types. Hypermedia components can be classified into three types, i.e., node, link, and composite.

In the following part of this section we discuss extensions of the Dexter model to support multiuser functions. Functions required for multiuser hypermedia systems have been investigated in certain research work. Platform and application issues in multiuser hypertext are discussed in [4]. Others discussed shared object service for collaborative work and concurrency control in a collaborative hypermedia authoring system [10]. Certain commercial systems and research prototypes including VirtualPlace and The Sociable Web [5] use an independent server for the management of users’ participation in the hypermedia space. Also, HypertextMOO and the recent VRML standardization allow implementations of the seamless user interfaces with which users can both manipulate hypermedia documents and communicate with others. However, the issues of dynamic or flexible views are not discussed in conventional research on multiuser hypermedia systems.

User deputy object \( o^u \) and environmental object \( o^e \) are introduced in our multiuser hypermedia architecture. User deputy objects have the attributes such as surrogate (srgt), which is a bilateral link to co defining how the user appears to others, self-description (sd), group membership (grp), access control list (acd), status (st) and s (0 ≤ s) miscellaneous attributes. Status st includes current status, navigation/operation history and future plans.

\[ o^u = (uid, srgt, sd, grp, acd, st, a_1, a_2, ..., a_s) \] (3)

Environmental object \( o^e \) provides multiuser environments for the sharing of hypermedia. Object \( o^e \), which interacts with the objects it contains, defines environmental components (ec), administrative information (adm), a set of bilateral links to participating objects (p), environment specification (es) and t (0 ≤ t) miscellaneous attributes.

\[ o^e = (uid, ec, adm, p, es, a_1, a_2, ..., a_t) \] (4)

Objects participating in environments are either user deputy, hypermedia component or environmental objects. Objects \( o_1 \), \( o_2 \), ..., \( o_t \), which participate in \( o^e \), interact with one another indirectly via the environment and/ or via direct communication channels. Environmental component ec is a bilateral link to a hypermedia component. The hypermedia component which is associated with environmental object \( o^e \) by ec is utilized by users to enter/leave the environment. It is also used to display information about the environment. Object \( o^e \)'s participation in environment \( o^e \) is denoted by \( o^e \). Note that a user can participate in more than 2 environmental objects at the same time.

Attribute p representing a set of participating objects is added to the scheme of hypermedia component objects. Thus, they have the following values:

\[ co_1 = (uid, ps, an, p, a_1, a_2, ..., a_m, cont) \] (5)

When users display a hypermedia component on screen, they participate in the component. Note that a user can participate in more than 2 components at the same time. We use dot notation to access attributes of \( co_1, o^e \) and \( o^e \). For example \( co_1.p \) is used to access participating objects of \( co_1 \).

3. Deputy Hypermedia

This section introduces incremental virtualization mechanisms with which deputy hypermedia sets and deputy hypermedia components are derived. The deputy model of objects [19] is used to flexibly represent the difference produced by the environments. The object deputy model allows an object to have multiple deputy objects which belong to different classes. The deputy objects have their own identifiers and may have additional attributes and methods, which realizes certain independence of the deputys. As shown in Figure 2, more than two deputy objects can be derived from an object. Also, deputy objects can be derived from other deputy objects.

![Deputies of Deputies Diagram](image)

**Figure 2. Hierarchy of Deputy Objects**

The specification of deputy hypermedia is stored as environment specification (ESpec) in environmental objects.

Figure 3 shows the process in which screen representation of hypermedia components is obtained.

Using the two different kinds of databases, each containing PSpec objects and hypermedia component objects, screen representation is dynamically computed. Binding
relations combine the hypermedia component objects and the PSpec objects which describe display-related attributes including window size, fonts, colors, shapes, and display metaphors. Using PSpec objects and the history databases, spatial and temporal memory of the users will be effectively utilized in the process of retrieval.

The process of creating screen representation is made flexible by incremental derivation of component objects and PSpec objects. The instantiator function creates screen representation by using two different kinds of the deputy objects which suit the requirements of the environments.

Figure 4 illustrates the incremental derivation of deputy hypermedia, where encircled directed graphs represent original/deputy hypermedia sets. There are two deputy hypermedia sets obtained from the original hypermedia set, one generated by selecting a subgraph and the other by selecting a subgraph, adding a link and changing direction of a link.

The deputy model of objects [19] defines six algebraic operations, Select, Project, Extend, Union, Join, and Grouping for the derivation of deputy objects. Select, Union and Grouping can be used to derive deputy hypermedia sets without changing attributes or values of each component.

Select: Derives a hypermedia set by selecting the objects which satisfy a certain selection predicate.

Union: Derives a hypermedia set by computing union of original hypermedia sets.

Grouping: Derives a hypermedia set by grouping the components of original hypermedia set.

Project and Extend can be used to derive deputy hypermedia sets by hiding/adding attributes $a_i$ of each component. For example, these operations can be used to add/remove keyword fields or labels/colors of hypermedia components. Note that it is not permitted to apply such operations to the attributes except for $a_i$ (1 $\leq$ $i$ $\leq$ $m$) where $co_i = (uid, ps, a_n, a_1, a_2, ..., a_m, cont)$ since the resulting objects should also be hypermedia component objects.

Project: Derives a hypermedia set by hiding certain attributes (and methods).

Extend: Derives a hypermedia set by adding certain attributes (and methods).

The union of all the hypermedia sets is called the global hypermedia set $G$.

$$G = \bigcup_{i} H_i$$

To avoid the inconsistency caused by the dangling links, we provide operations trim and extend to eliminate dangling links using the global hypermedia set.
4. Dynamic Instantiation Based on Environments

When object o participates in environmental object o', deputy object o" is created, which is denoted by o" = dep(o, o') where o < | o'.

4.1 Dynamic Instantiation

Let co ~ cd' denote navigation from hypermedia component co to cd'. When user o" displays hypermedia component co, o" participates in component co, i.e., o" < | co. Navigation co ~ cd' changes the user's participation, from o" < | co to o" < | cd'. If co and cd' are environmental components where co = o1.eo and cd' = o2.eo, co ~ cd' changes the user's participation to environments, from o1 to o2.

When users' participation to environments change,

1. Different hypermedia components are displayed with the same views or,
2. Different views of the same components are displayed.

Environments themselves participate in other environments, i.e., there exist environmental objects o1 and o2 such that o1 < | o2. Note that attributes are inherited from o2 to o1.

Figure 4 also shows derivation of a composite hypermedia component, which is performed not by creating deputies for the object. The original component consisting of two sub-components is incrementally reorganized by hiding, adding, decomposing or composing, from the top to the bottom. Hypermedia sets H2 and H4 are generated by converting a hypermedia set to be a hypermedia component, or vice versa. Hypermedia set H5 is derived by applying navigation history to H4. Application of navigation derived partially linearized hypermedia sets.

4.2 Discussions on Environments

Attributes including environment specification (ESpec), administrative information (AdminInfo) and participating
objects (PO) should be defined in creating environmental objects.

Environmental specification ESpec can be used to virtualize hypermedia components, hypermedia sets and attributes of hypermedia components.

When hypermedia component co participates in environment o', environmental specification (ESpec) is used to derive o' = dep(co, o') and thus composite hypermedia components are reorganized as shown in Figure 4.

Similarly, ESpec can be used to derive deputy hypermedia sets HD from environment o' and hypermedia set H, i.e., HD = dep(H, o').

Each attribute of co (o') references objects such as contents objects, PSpec objects, anchor objects, keyword objects and label objects. ESpec is used to derive deputies of these kinds of objects.

Deputy hypermedia components, deputy hypermedia sets and other deputy objects can be materialized. Materialized deputy objects are referenced by ESpec.

When environmental specifications are applied to hypermedia components, there can be conflicts between the owners: the owners of the environments and the owners of the components. In order to make it easier to resolve such conflicts, AdminInfo defines administrative information including the following information:

**Owner:** Bilateral link to the user deputy object of the owner.

**In-Out:** Set of user deputies who can enter/leave the environment.

**Administrator:** Set of the users deputies who can change attributes of environmental objects.

**Creator of Environments:** Set of user deputies who can create sub-environments in the environment. The sub-environments are considered as deputies of the environment.

Since environments function as places for not only information sharing but also communication and collaboration, there can be two environments whose difference is only the participating objects (PO). In this sense, participating objects themselves define parts of their environments.

When a user participates in more than two environments, it is possible for the user to appear differently in different environments. To do that, deputies of surrogate o'.srft, which is a hypermedia component object, are utilized.

Environmental specifications, administrative information and participating objects can be defined in a top-down fashion. However, bottom-up methods of defining environments are also required. Defining environments in a bottom-down fashion requires a sort of community mining based on the user profiles, hypermedia contents and navigation histories.

It is possible that multiple hierarchies of environments coexist in a system. For example, if there is a temporary task-force group in a virtual company, a position hierarchy of a company and a temporary hierarchy of a task-force group influence employees’ cooperation. Discussions on multiple hierarchies are also provided in the following section.

### 5. Awareness of Customized Views

In systems based on the WYSIWIS principle, users can work assuming that all other users see the same screen as theirs. So each user can say "the window on the left side of the current window," "the phrase starting from line #5 of the text," or "the button at the upper left corner of the window" to point any parts of the materials displayed on screen. However, in the systems which are not based on the WYSIWIS principle, the materials on a user’s screen may not be displayed on other users’ screen.

Even if the materials are on screen, their locations, sizes, and/ or structures may be different. The difficulties in introducing diversity in collaborative systems are observed since participants presuppose that their own activity and domain are visible to others [3]. Multiple environments can be introduced only if users are provided with natural ways to know other users’ identities, activities, environments, etc. Participants of collaborative work subsuming multiple environments are either in the same environment or distributed in different environments.

**Collaboration in the same environment:** When all the participants of collaborative work are in the same environment, they can be given WYSIWIS view of the shared information. If each environment provides participating users with WYSIWIS view, the system is based on intra-environment WYSIWIS principle.

**Collaboration in different environments:** If a set of environments provides participating users with WYSIWIS view, the system is based on inter-environment WYSIWIS principle.

In many cases, it is impossible to assume inter-environment WYSIWIS view. Therefore, it is required to gracefully degrade qualities of users’ communication by providing users with natural ways to exchange meta-information concerning sharing of information. To do that, meta-information should be stored for users to retrieve it. Also, it should be displayed on screen in such a way that they can understand it in natural fashions.

The following information about users and environments is used:

**Attributes of user deputies:** Private information of each user such as names, ID, addresses, conditions, current work, etc.
Attributes of environmental objects: Information of each environment, such as ID, administrative information, participating objects, display specification objects, etc.

Visualization of environmental objects should provide structural and/ or semantic information:

Structure of users and groups: Users should be able to easily recognize organizational hierarchy, users in the same group, etc.

Semantics of groups/environments: For instance, participants of a virtual classroom should be able to easily recognize group of teachers, students, people in a math class, people in an English class, etc.

Users can be represented using symbolic representations including user icons as well as video pictures. When using user icons, dynamic changes of user’s states can be represented by using slightly different user icons as shown in Figure 6.

Environments can be represented using symbolic representation of environments. Since environments usually do not have corresponding real-world entities, they should be associated with appropriate metaphors. If the hierarchies of environmental objects are represented on a screen, it can be used as a sort of maps of environments that allow users to know differences of environments in natural ways. Therefore, in choosing metaphors of environments, their capability of representing containment relationship should be considered.

Multiple hierarchies of environments are created for independent concepts. For example, a student who is a good student in a mathematics class can be an excellent student in an English class. In this case, s/he is in environments of good students and of excellent students where each environment is in a different hierarchy of environments.

When representing relationships of users and environments, their dynamic changes should be considered since users may migrate from one environment to another environment during real-time interactions. The migration might cause changes of symbolic representations of users on screen. Also, environments can be created dynamically causing changes of screen. Some of the changes of environments can be automatically processed using containment relationships of environments.

In Figure 6, environments are represented as rooms. Users are displayed in rooms as user icons that are changed by their states such as idle, busy, overloaded, etc. They can change their environments by moving user icons. One of the advantages of using rooms as a metaphor of environments is their capability of representing trees. Since hierarchies of environments are trees, any of them can be represented by the use of rooms.

Figure 6. Visualizing users and environments

Figure 7 represents the same hierarchy of environments as Figure 6, however, environments are represented as not only rooms but also costumes. Though it is difficult to represent hierarchical structures of environments by costumes, one of the advantages of using them is that user icons can be moved anywhere on a screen; they are not blocked by the walls. Open doors in Figure 6 represent environments where anybody can go in and come out. They correspond to costumes in the closet in Figure 7. We can use more than two kinds of costumes such as hats, ties, pants, earrings, belts, etc.

Combination of rooms and costumes can be used to visualize multiple hierarchy of environments. In the real world, people work in rooms. If there exist several projects, we usually don’t create a room for each project, instead uniforms or project ID card may be used. Combinations of rooms and costumes can realize a similar environment. In such combinations, we use rooms to represent primary hierarchy of environments when there are more than two independent hierarchies since costumes do not contain rooms.

Glasses are used to represent personal environments. If users wear glasses, they are in their own personal environments. If there is no user icon wearing glasses in a room,
one can understand that the environment represented by the room provides intra-WYSIWIS view. Users can move their user icons from one room to another or change their costumes in order to change their environments. Then the hypermedia components and their views are automatically changed on screen.

The awareness support mechanisms discussed in this section has the following characteristics:

**Awareness of virtual entities:** It is possible to support awareness by visualizing virtual entities such as environments. Conventional CSCW systems support awareness of real-world entities.

**Flexible views:** Environments can be visualized based on WYSIWIS principle, however, visualization methods can be personalized when necessary. For example, visualization methods such as focus-and-context can be used when the number of environments and users are large and their relationships are complicated.

6. Development of Cooperative View Mechanisms

Basic functions of the cooperative view mechanisms are implemented in the current prototype of VIEW Media. VIEW Media is a multiuser hypermedia system for collaborative applications such as interactive presentations, virtual offices, virtual classes and electronic conferences.

The first prototype was developed focusing on customization functions required to support interactive distance presentations. The current prototype, developed in Distributed Smalltalk, implements a comment/ interface customization facility, environmental objects, and visualization function of users and environments.

Figure 8 shows screen dump images of VIEW Media, where the larger screen dump image on the left side shows user Yokota’s screen, while the smaller image on the right side shows user Sakata’s screen. In each screen, there are three windows, i.e., a VIEW Media launcher window (a small window at the top-left corner), a hypermedia window, and an environmental window (rooms and user icons).

Hypermedia windows are customized by each user. Yokota adds a comment to the hypermedia component and also changes size/color of the hypermedia window as you can see in Figure 8. The comment is displayed in a small comment window at the hypermedia window. He can also emphasize parts of the text and/or add personal links on the shared hypermedia component.

In the environment window, Sakata is in environment #2 owned by Sakata; Yokota and two other users are in environment #3 which is owned by Yokota. Both of environments #2 and #3 are sub-environments of environment #1 whose owner is Sakata. Comments and interface customization in environment #1 are inherited to environment #2 and #3 when the owners of environments require.

Currently we are designing and developing the next version of VIEW Media based on the first and the second prototyping experiences. The system is being developed using Java as well as VisualWorks 2.5/ Distributed Smalltalk 5.5 under Solaris 2.5 UNIX/SUN Ultra I environment.

Figure 9 shows an overview of the system. The mechanisms presented in this paper will be handled by Hypermedia Reorganization Processor, Environment Manager and other system elements associated with the Environment Manager via participation relationships.

We also collect history information including navigation histories and reaction information such as questions and other kinds of feedback from students in virtual classrooms. History information is used by Hypermedia Reorganization Processor, while reaction information is used to improve hypermedia components.

The VIEW Media browser can display deputy hypermedia, users, environments and communication equipment such as chat, audio and video tools. Communication tools are also controlled by environments so that appropriate tools are automatically invoked in each environment.

Presentation Specification Object Manager store window sizes, colors, display metaphors, etc. Appropriate presentation specifications are selected and sent to Instantiator so that appropriate user interfaces of hypermedia components are automatically generated in each environment.

7. Conclusion

Since users create and participate in environments in a dynamic fashion, conventional static view mechanisms are not sufficient to manage virtualization of hypermedia in CSCW environments. We have proposed cooperative view mechanisms of distributed multiuser hypermedia environments in order to realize:

1. Virtualization and customization of hypermedia documents in a flexible fashion,

2. Dynamic instantiation of views considering the specifications of collaborative work environments, and

3. Communication aids to minimize the possible difficulty of cooperation introduced by (1) and (2).

In the proposed mechanisms, users’ participation in hierarchically organized environmental objects instantiates views and thus visualizes deputies of hypermedia on screen. We also described a novel user collaboration facility, which
supports users’ awareness of the situations concerning sharing and customization of views.

Applications of the cooperative view mechanisms include virtual offices, electronic conferences, and distance learning/education. Cooperative view mechanisms and their application systems are being developed in our research group using JAVA and Distributed Smalltalk.

In the future work we will realize a cooperative hypermedia system having the view mechanisms proposed in this paper. Also, we will further investigate the problems of consistency, security and updates, while augmenting the power of the view mechanisms.

Acknowledgements

We gratefully acknowledge Zhiyong Peng, Hironori Kamada, Chihiro Ito, Hajime Iwamoto and other researchers in VIEW group for useful discussions on some of the topics covered in this paper.

References

Figure 8. Screen Dump Image of VIEW Media

Figure 9. Overview of VIEW Media