Take CoVer: Exploiting Version Support in Cooperative Systems

Anja Haake, Jörg M. Haake

Integrated Publication and Information Systems Institute (IPSI)
Gesellschaft für Mathematik und Datenverarbeitung (GMD)
Dolivostr. 15, D-6100 Darmstadt, F.R.Germany
Tel.: ++49 / 6151 / 869 - 929
e-mail: {ahaake, haake}@darmstadt.gmd.de

ABSTRACT

Current CSCW applications support one or more modes of cooperative work. The selection of and transition between these modes is usually placed on the users. At IPSI we built the SEPIA cooperative hypermedia authoring environment supporting a whole range of situations arising during collaborative work and the smooth transitions between them. While early use of the system shows the benefits of supporting smooth transitions between different collaborative modes, it also reveals some deficits regarding parallel work, management of alternative documents, or reuse of document parts. We propose to integrate version support to overcome these limitations. This leads to a versioned data management and an extended user interface enabling concurrent users to select a certain state of their work, to be aware of related changes, and to cooperate with others either asynchronously or synchronously.

KEYWORDS: CSCW, versioning, cooperation modes, alternative object states, group awareness, hypertext

INTRODUCTION

Current Computer Supported Cooperative Work (CSCW) applications (see [3], [9] for an overview of CSCW systems) support a group of co-workers performing a task on a shared information base. Usually, these systems support a specific collaboration model, e.g., asynchronous collaboration through draft passing, or synchronous collaboration through joint editing. At any point in time, exactly one state of the overall information base exists.

However, collaboration in not computer-supported environments often includes a lot of parallel work resulting in alternative or conflicting drafts. Each co-worker produces a different state of the information base which must be merged with the other states later. In addition, co-workers use different collaboration models according to the needs of a specific task.

Current CSCW systems do not support different collaboration models at the same time, nor do they allow for parallel work on a shared information base or alternative states of a shared information base.

A first step towards a solution is to support different modes of collaboration allowing co-workers to work either asynchronously or synchronously on the same (part of the) information base. Specific support for different kinds of synchronous collaboration is required, and the smooth transition between the different modes must be supported. This is the approach taken by the SEPIA cooperative hypermedia authoring environment (see [6], [12]) which allows more parallelism for working on the same state of a hyperdocument.

To support different states of an information base we propose to integrate version support into CSCW systems. This still allows synchronous work on the same state of the information base but enables concurrent co-workers to select a certain state of the information base, to be aware of related changes, and to cooperate either asynchronously or synchronously. In such a system not only an extended data management is needed, but also extended user interfaces reflecting group awareness as well as the existence of alternative versions are required. This approach led to the development of the versioned cooperative SEPIA system described in this paper.

We will first explain our approach to support different modes of collaboration taking cooperative SEPIA as an example. After identifying a list of deficits of cooperative SEPIA we introduce basic version support. Next, we introduce our approach of integrating version support into cooperative SEPIA. Using an example of collaborative authoring we illustrate how the extended system overcomes the deficiencies of the original system. Finally, we discuss the differences to related work and the general applicability of our approach.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

© 1993 ACM 0-89791-575-5/93/0004/0406...$1.50

THE COOPERATIVE SEPIA SYSTEM

SEPIA is a cooperative hypermedia authoring environment. Its purpose is to support groups of authors who collaboratively create hyperdocuments [6]. A hyperdocument consists of three kinds of hypermedia objects: atomic nodes, composite nodes and links. Atomic nodes contain data (e.g., text, graphics, images, sound). Composite nodes contain a set of references to other hypermedia objects. Thus they allow the clustering of objects into subgraphs of the overall hyperdocument. Links represent relationships between the hypermedia objects.

Authoring Support in SEPIA

To support the general authoring activity SEPIA provides the notion of a project. A project contains all material needed for the production of a publication including the final hyperdocument. Working on a project includes different activities which are supported by four dedicated workspaces, the so-called activity spaces [11]. They comprise the Planning, Content, Argumentation and Rhetorical Space. The Rhetorical Space serves the final hyperdocument production while the others are used for preproduction activities. Associated with each activity space is a dedicated browser which offers a certain set of functions on the objects it presents. To display the subgraph contained in a composite node a new browser can be opened on the composite. In SEPIA all hypermedia objects are typed carrying certain attributes and each activity space provides activity-specific types for nodes and links [13].

Cooperation Support in SEPIA

To support the cooperation among co-authors cooperative SEPIA provides access to a shared information base by using the cooperative hypermedia server CHS [12]. SEPIA ensures a common view of the information base among all concurrent users relying on the CHS update notification service. The consistent state of the information base is maintained through the use of transactions. Access to the database is synchronized via activity markers signalling a specific status of an object (e.g., "about to be modified") which must be interpreted by the browsers before accessing or modifying the object.

Specific support is provided for the three modes of collaborative work:

- **Individual work** can be performed by opening a browser on a composite node which is currently not used by other authors.
- **Loosely-coupled work** allows several co-workers to concurrently access some part of the information base but still permits rather independent work. This work mode is supported by providing group awareness when several co-authors work simultaneously in the same composite node. Every co-author is able to work independently but sees the effects of other co-authors' activities immediately as they appear in the currently visible part of the subgraph. Activity markers are used to avoid collisions between authors accessing the same objects. This approach can be considered as the integration of general awareness – as proposed in the Portholes systems [2] for distributed office rooms – into CSCW tools themselves.
- **Tightly-coupled work** allows to cooperate and coordinate group work in synchronous conference-like "meetings". This work mode supports shared views among the concurrent browsers (the WYSIWIS principle: "What You See Is What I See" [10]), a telepointer for each co-author, and additional communication channels (e.g., audio and video conferences, and a shared drawing tool have been added to SEPIA. For details, see [6], [12]).

Actual collaboration proceeds by shifting between these three collaboration modes. The transitions are often prompted by needs for coordination which arise from individual work or through observing activities of co-workers in loosely-coupled work. Therefore, smooth transitions between the modes must be supported, i.e. natural and easy-to-use transitions in a non-disruptive manner have to be provided.

The smooth transitions between the modes are supported by binding these transitions to changes of the set of current users of a composite node: While working on a composite without concurrent users, an author works in individual work mode. When co-authors open a composite node already used by another author, all concurrent browsers shift into loosely-coupled mode. If some co-author in a loosely-coupled session feels the need for a synchronous conference, the co-author can initiate a tightly-coupled session between a set of specified loosely-coupled collaborators. Those who confirmed the conference request then shift into tightly-coupled mode while the others remain in loosely-coupled mode. At any time, co-authors can close their browsers, thus leaving the corresponding loosely- or tightly-coupled sessions. This way, transitions between the modes are triggered by the authors' navigational actions (open and close browsers) or by explicit conference requests.

Deficiencies

Early use of the system indicated a number deficiencies. First of all, the individual work mode, as defined above, includes two different cases which should be supported explicitly:

- **Isolated work**, where just one author should be allowed to work on and see the content of a specific document part (i.e., composite node).
- **Separate work**, where several co-authors may concurrently work on separate drafts of the same document part, but without interfering with each other.

The first case could be implemented in SEPIA using "lock activity marker" on composite nodes thus preventing others from entering the document part. The second case requires the management of alternative drafts created from one origi-

nal document part, which is possible in SEPIA (through copy functions and creating links among original and copied objects) but which is not explicitly supported.

Next, copying objects for use in other composites causes problems because of **loosing the object identity**. When the copy is edited this is not visible within loosely-coupled sessions viewing the original object. This may result in conflicting changes which cannot be easily detected by co-authors.

In addition, during conference-like tightly-coupled sessions aiming at the integration of several drafts or document parts it is necessary to be able to review each of the parts and to modify them to create an integrated result. But the **integrity of the individual contributions** (drafts) must still be maintained (at least for documentation purposes). In SEPIA this requires explicit copying of objects which put additional load on the co-authors.

Moreover, **history tracking** is completely missing. Thus co-authors rejoining work e.g. after holidays found it difficult to find out what happened in the meantime. Another situation requiring history tracking arises when a bug is identified and all places have to be found where the wrong information has been used or has been contributed to.

At least three of the above deficiencies are key issues of version management (i.e., isolated workspaces, management of alternatives for separated work, and history tracking). In the next chapter we explain our approach to versioning of hyperdocuments. After that we discuss the integration of versioning into cooperative SEPIA to overcome the mentioned shortcomings.

COVER: A CONTEXTUAL VERSION SERVER

CoVer is a hypermedia version server [5]. To provide for adaptable version management CoVer has been designed as an extension of the cooperative hypermedia server CHS [12]. Thus, the versioning concepts offered by CoVer can be used by hypertext applications to define application-dependent version support. CoVer maintains context information with the versions. Context information guides version creation and in particular helps in version identification. After introducing the basic versioning mechanisms, we explain the context-based versioning mechanisms derivation history and task.

Basic version management

CHS offers persistent nodes, links, and composites but does not preserve previous states of objects. The task of basic version management is, above all to define the notion of a versioned object.

CoVer represents versioned objects by so-called **multi-state objects** (mobs). A mob represents a versioned object by gathering all states of the versioned object in its version set. The states of a versioned object are called versions and are represented by individual nodes, links, and composites. A mob is realized on top of CHS as a composite holding references to all states of the versioned object it represents. CoVer maintains the creation time and author of each object. In order to preserve the states of versioned objects, versions of nodes, links, and composites can be **frozen** [5].

It is a key characteristic of CoVer that is does not impose a fixed structure on the versions of a versioned object. Version selection is based on viewing and browsing versions with respect to values of their attributes or relationships to other objects. Consequently, our notion of alternatives does not depend on the notion of revision (cf. below 'Derivation history') defined by a certain version graph, as found in many other approaches (see [1], [8] for an overview). Any two or more versions of a version set that match an equivalence predicate given by a query exploiting the versions' attributes are considered alternatives with respect to the equality characteristic expressed by the query.

Derivation history

CoVer's notion of revisions is subsumed by the **derivation history**. The derivation history induces a (possibly) unconnected graph structure over all versions, independently of their structuring into mobs. The derivation history is implemented using a specific link (derivation link) connecting the ascendant and the descendant version. Derivation links can be created explicitly by the application. Moreover, CoVer offers two derive operations creating either a new version as a copy of a version or a new object and installing a corresponding derivation link. Versions of the same mob connected by a derivation link may be considered revisions of a versioned object. But in addition, the derivation history records the reuse of material across document boundaries and can be navigated during version selection.

Tasks

A key concept for version creation is task tracking [14]: Users change their hypertext network to perform a task. For example, writing a document by a group of authors includes tasks such as proposing an outline, creating alternative proposals, or merging several contributions. These tasks can guide meaningful, automatic version creation. Stored persistently as contextual version information they serve version identification.

While mobs maintain the history of a single versioned object, tasks maintain the (versions of) various objects used and created in the context of performing a job. Eventually, a task maintains a state of the hyperdocument that fulfills the requirements of the respective jobs. To create such a new version of a specific object, tasks may profit from all available versions of all available objects in the overall system. Therefore any object may be included into a task by an include operation provided by CoVer. Changes to included objects will be recorded automatically in derived versions, i.e. CoVer

freezes the included objects, derives new versions of these objects and executes the changes on these derived versions.

Apart from a system maintained identifier, a task is described by a short name supplied by the application, start and completion date and time - i.e. if the completion date is undefined the task is still alive, otherwise terminated - , author information, and optional application defined attributes providing detailed task information. In particular, a task holds references to all objects constituting its current state. A task may have several subtasks. Tasks may be executed sequentially or in parallel. Thus, tasks form a task hierarchy providing a framework for managing the recursive decomposition of application processes. Keeping a log of all subtasks of a task, CoVer monitors the concrete work flow. A task is implemented on top of CHS as a composite holding references to the objects determining its current state. Moreover, special types of links are used to represent the various task relationships. An initial top level task maintained by CoVer records all tasks created by the applications.

If a task is completed by the application, its current state is frozen by CoVer and delivered as a result to the supertask. These versions are the only versions visible in the supertask, i.e. all intermediate versions of the subtask are hidden. If a task is aborted by the application, CoVer also aborts all direct open subtasks and connects the completed subtasks to the parent task of the aborted task. Versions belonging to incomplete tasks are only visible to their subtasks. But versions maintained as the result of a completed subtask are accessible for everybody since they represent a state of work consistent from the point of view of the task being performed by the completed subtask. Thus, CoVer provides an **encapsulated work area** for ongoing tasks and hides inconsistent system versions from other tasks. CoVer supports set-oriented access to tasks by querying their attributes as well as browsing the task history. Thus, it offers application access to versions of both whole hypertext networks and single objects via task-related information.

INTEGRATING VERSION SUPPORT INTO COOPERATIVE SYSTEMS

This section discusses how cooperative systems can be extended by version support to alleviate the deficiencies discovered. We call such systems versioned cooperative systems. First, we focus on extensions to the implementation of SEPIA on top of the CoVer version server. Next, we show how these extensions are reflected by the user interface of the versioned cooperative SEPIA.

The System's View

Hypertext applications exploit CoVer via CoVer's application interface. As described in the previous section, CoVer naturally extends the notion of hypertext objects to the notion of versioned hypertext objects. However, the applications have to decide how those objects should be versioned, i.e. the applications have to implement their specific versioning behavior (versioning style). For each type of operation the application has to define how it affects the versioned objects by declaring the respective operation as a task.

For cooperative systems, the behavior of the cooperation modes is of particular importance. This is why, after we have described the exploitation of the versioned data model to alleviate the detected deficiencies, we will discuss how the extended cooperation modes of SEPIA are implemented using tasks. To assure the smooth transition between different cooperation modes we designed extensions to SEPIA that can be generally applied to realize group awareness in versioned cooperative systems.

Exploiting the versioned data model. A prerequisite to overcome the main deficiencies is the application of mobs to the basic object representation. A mob represents a versioned object by gathering all states of the versioned object in its version set. This representation guarantees the **object identity** of the versioned object and simultaneously represents the versions as identifiable units. The former enables the detection of conflicting changes whereas the latter is the basis to manage alternative drafts, for example for **separate work**, and to maintain the **integrity of individual contributions** when creating integrated results in tightly-coupled sessions. In addition, CoVer offers several options for **history tracking**. The application can explore the historical development according to the creation date of versions, restrict the view to versions to those created by a specific author, or use any other attribute that suggests a sensible view over the version set. Exploring the derivation history, bugs propagated by cut and paste operations may even be detected across document boundaries. In particular, tasks monitor the work flow at different levels of detail and abstraction and ease the access to historical information.

Extending the cooperation modes. We propose to support the extended set of cooperation modes by applying tasks. **Separate work** corresponds directly to parallel tasks. If several co-authors want to work concurrently on separate drafts of the same document, each author should create a successor task of that task that created the original document. CoVer's task concept automatically preserves the original and keeps the changes in separate versions not interfering with each other.

Supporting the restricted mode of **isolated work** requires a combination of CoVer's task concept with the CHS activity marker concept. If just one author should be allowed to work on and see the content of a specific document part, the original version should be totally protected, as well as all new versions created in the context of the author's task. Since the original is protected, no parallel task may be started on the original. Since new versions are protected, nobody may join the author's task (cf. below for joining tasks). Thus, unin-

tended creation of alternatives that may lead to serious merge problems can be avoided.

Since authors may also want to modify the same version of a document, the loosely-coupled and tightly-coupled mode should also be supported in a versioned system. To enable authors to work independently in a loosely-coupled session on the same version of a document, we allow to **join tasks**. If an author is working on a task, anybody else may join this task thus shifting into loosely-coupled mode.

To monitor changes to objects performed by different authors involved in the session, the cooperative versioned SEPIA records each update to a version, which has not been performed by the version creator, in a new, derived version. Thus, revisions of objects by collaborating authors – which led to a loss of the original state if the original had not been saved in a backup copy in the cooperative SEPIA system – are automatically maintained by the versioned cooperative SEPIA. The same versioning style has been implemented for tightly-coupled sessions.

At any point in time, authors may initiate a tightly-coupled session. The versioned cooperative SEPIA records each tightly-coupled session in a task that is considered a subtask of all current tasks of users participating in this session. Beside using the backup copies saved by the system if different authors attempted to subsequently modify the same object version, authors engaged in a tightly-coupled session profit in particular from the **extended object identity** maintained by CoVer's mobs. Tightly-coupled sessions are used when several contributions developed during isolated or separated work should be merged into one group work result. During a tightly-coupled session, the authors may include their individual contributions into the task. Alternatives, i.e. versions of the same object, are identified by the system-maintained extended object identity. Since a task keeps changes to included versions automatically in derived versions, a new version covering the final result will be created automatically and the **integrity of individual contributions** is preserved.

Extending group awareness. The support of smooth transitions between the different cooperation modes is based on the notion of object identity: If users work on the same composite, they move into loosely-coupled mode. While working in loosely-coupled mode every co-author sees the effects of other co-authors' activities and activity markers are used to avoid collisions between authors accessing the same objects. These mutual observations of activities may create the need to initiate an unforeseen tightly-coupled session.

Exploiting version support enables the users to keep individual changes in alternative versions. Actually, these changes affect the same object but constitute different states of the object. Unintended generation of alternative versions may result in unforeseen merge problems or overall consistency problems that should have been prevented.

One way to prevent these problems, is to work in isolated mode. In many situations, this approach is too restrictive. If users want to create alternative versions of the same object for different purposes, or if users want to work in parallel assuming that a later merge can be performed without major problems, they may work separately. Nevertheless, unexpected alternative updates of the same object may be problematic. Therefore, users working on separate tasks should be informed about alternative versions created by other users. If required, concurrent users working at the same time may then initiate a tightly-coupled session to solve the conflict immediately, or concurrent users working at different times may schedule an early tightly-coupled session to fix the unforeseen conflict. This is true not only for individual separate work, but also for loosely- or tightly-coupled sessions occurring in parallel. To support this behavior, CoVer's notion of an encapsulated work area (cf. subsection about tasks) has been relaxed and the implementation of CoVer on top of CHS exploits the extended object identity provided by mobs to inform applications of alternative versions.

The User's View

The cooperative SEPIA user interface looks as follows: The SEPIA main control panel allows users to create, select, and open projects. After opening a project a SEPIA Project Launcher pops up and offers access to the four activity spaces. One scrollable browser window for each activity space and for each composite node is used to display and manipulate the composites' content. In general, the state of an object is signalled using colors.

To discuss the introduction of version support into cooperative systems at the user interface, we look at the cooperative writing of a hyperdocument by a group of four authors using the versioned cooperative SEPIA authoring environment. Following well-known patterns of cooperatively writing documents, we will pick up four different situations that illustrate the four cooperation modes supported by a versioned cooperative system.

Isolated work. At one point of time, one author may enter the SEPIA system to produce a new draft proposal from the individual contributions, alternative proposals, and annotations of the co-authors. After selecting the right project, the author has to select an appropriate task determining the state of work to be initially confronted with. To do so, the author investigates the Task Browser (cf. Figure 1) maintained for each project that shows the development of the project up to the current point of time. For example, Figure 1 shows the situation after each team member has revised and extended the annotated outline of the hyperdocument. Since producing a draft proposal is a new task, the author creates a new task succeeding the four previous tasks by selecting all predecessor task icons and using the corresponding menu. Then,

the author opens a new task in isolated mode to avoid changes made by others during the merge phase, i.e. a task in isolated mode assures an exclusive workspace and inhibits the creation of new versions by others.

Since the new task is a successor task of all four preceding tasks, the opened project prompts with four SEPIA Project Launchers named by a combination of project and task name representing the four different states of work. The author investigates each individual contribution and merges them into a new draft proposal.

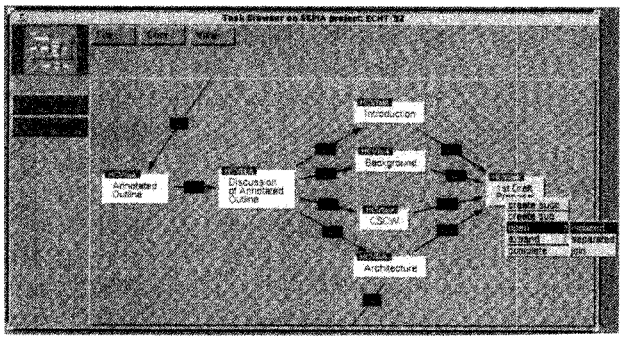

Figure 1 Task Browser
showing the development of work. One author has proposed an Annotated Outline. The group has discussed this outline and decided to split up in four groups. Now, one author merges the group works' results into a 1st Draft Proposal to be discussed eventually in a joint meeting.

While investigating a single contribution, the author may request the visualization of alternatives proposed by other contributions which will be piled beneath the selected contribution. Figure 2 shows the Rhetorical Space of the contribution provided by the task "Introduction" (cf. Figure 1 for names of tasks). Only two of the other three co-workers have also proposed a new version of the node named "Challenge of HM Authoring". To investigate these alternative proposals the author may open a Mob Browser (cf. Figure 2) on a versioned object that shows the relationships and attributes (of a specified subset) of the versions of the version set. Selecting two versions in the Mob Browser triggers the comparison of the versions' content (cf. Figure 2).

Based on the differences between the selected versions shown in the Mob Browser, the author decides to copy some sentences of the version of the node created by task "Background" into the version of the node created by task "Introduction". This update automatically leads to the creation of a new version belonging to the actual task. This new version is immediately prompted in the Mob Browser. Switching back to the Rhetorical Space, the system shows the new version of the node as an additional alternative. The author may explicitly exclude those alternatives that should not contribute to the draft proposal. Further investigation and merge may proceed in the same way leading to a new version of the hyperdocument.

Tightly-coupled work. Having finished the draft proposal, the author may schedule a synchronized session (per e-mail or any other suitable communication means) to discuss the draft proposal and eventually create a new common draft of the hyperdocument. All authors join the new task which has been initiated by the main author as a successor task of the merge task and enter the tightly-coupled session. During the meeting, the author of the draft proposal introduces the proposal. The tightly-coupled mode allows co-authors to interrupt the presentation if required (e.g., verbally using the audio channels provided by the cooperative SEPIA) and to bring up parts of their previous proposals to support the discussion of decisions made by the author of the proposal. To indicate which browsers show versions belonging to the current state of work and which browsers are visualizing previous versions, the former are presented in normal colors whereas the latter are presented in pale colors. During the discussion, the authors may jointly use the comparison facilities or edit the new proposal simultaneously. All updates will automatically be kept in new versions documenting the result of the meeting.

After finishing the draft, the team has decided to split up into three groups for further work. Two authors will work independently on different aspects of the article whereas two authors should elaborate another aspect together.

Loosely-coupled work. The group of two authors may decide to further separate the work (i.e. create separate subtasks for each author, then later merge their results in a tightly-coupled session) or to manipulate the same state of work synchronously. In the latter case, one author may join the task if the other author is already working in separate mode. Then, both move into loosely-coupled mode. Working in loosely-coupled mode, one author may unconcerned revise the changes made by the partner since alternating updates are maintained by the system, propose changes in an alternative version that will be visualized by the system, or initiate a tightly-coupled session to discuss the proposals immediately.

Separate work. Although the three different groups are working separately on different aspects of the hyperdocument, it may happen that some of them have modified the same object which led to the maintenance of alternative versions with respect to the different tasks. To check for the unintended generation of alternative versions, a user may request the system to show all alternative versions created by concurrent tasks. Icons representing these versions are presented in pale color to distinguish them from alternatives belonging to the actual state. If some alternatives seem to constitute conflicts, the involved users may initiate a tightly-coupled session to resolve the problem in a common subtask. The result of this task will be kept automatically in a new version of the object and may be used in the concurrent tasks during future work. In any case, the result of these three concurrent tasks are three new states of the hyperdocument that are kept as alternative versions that may be merged later.

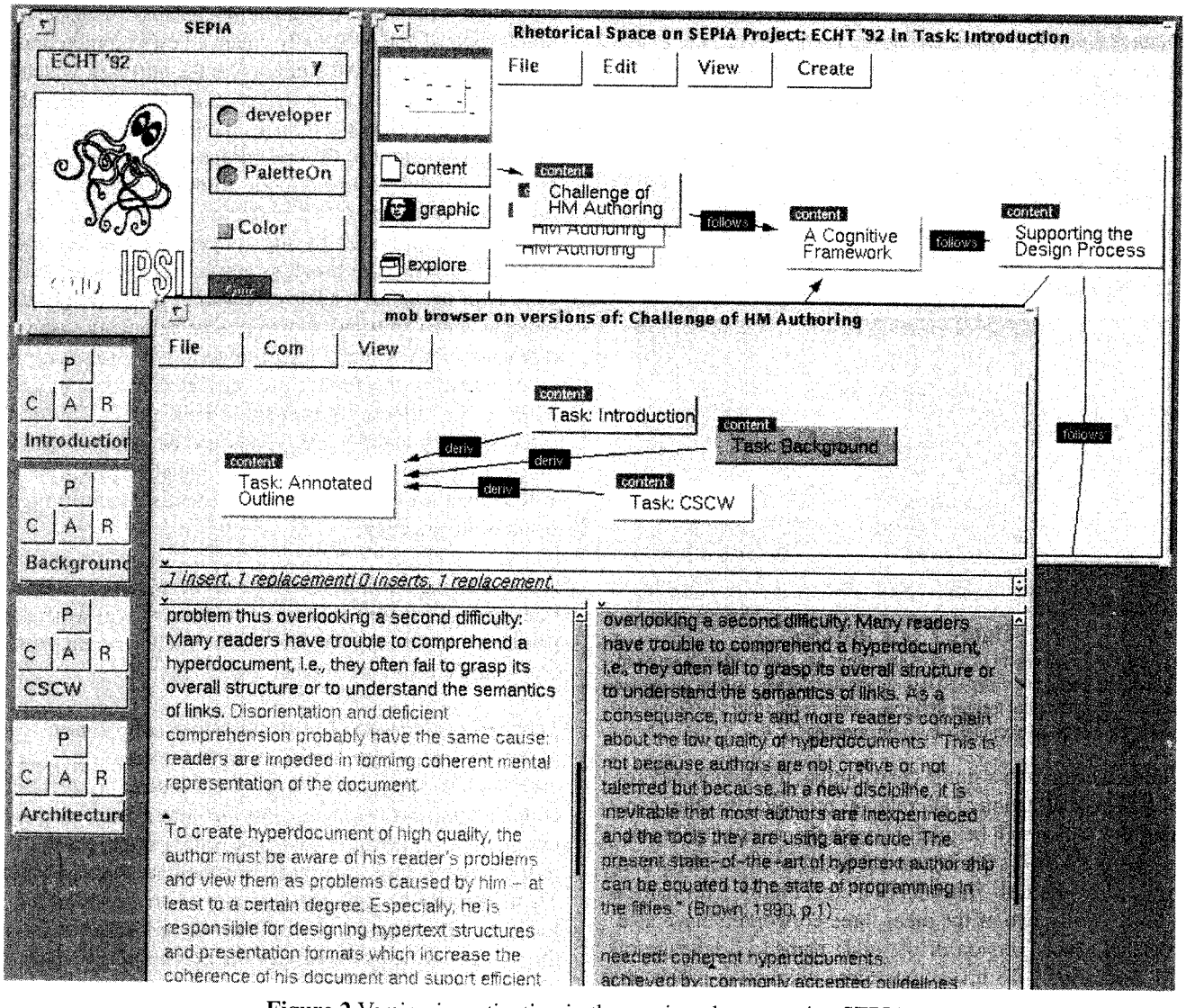

Figure 2 Version investigation in the versioned cooperative SEPIA.

RELATED WORK

Related work either focuses on cooperative work or on version support. This work has been discussed in [6] and [5], respectively. Related work on cooperative work either supports subsequent draft passing or synchronous work on a single system state only. Related work on version support allows to maintain different subsequent or parallel states of objects providing different degrees of history tracking to support the mutually intelligibility between cooperating users. In general, versioning approaches provide check-in / check-out mechanisms that may be used for enhanced draft passing as an extended means of collaboration. These mechanisms are introduced as single concepts and the integration of the concepts in a cooperative environment and user interface is not discussed.

We are aware of only two attempts to exploit version support directly to enhance cooperative work. Quilt [4] is a tool for collaborative writing of linear text. Whereas SEPIA's cooperation support also attempts to support unforeseen interaction and communication needs, a prerequisite to use Quilt is to define a cooperation style naming the participants and describing their social roles that define their rights and responsibilities in a certain collaboration task. Quilt allows to annotate work-in-progress (which is inherent in hypertext and thus in our approach), exchange messages, and track the status of work. It does not support synchronous work. Quilt maintains only one valid state of the linear document at a time. It is up to the users to determine when to save a version of the document. Moreover, Quilt records a complete activity log of the users' interactions that may be extended by more abstract descriptions of activities provided by the users. In our approach also basic operations may be declared as tasks and then are maintained as subtasks of more complex operations or user-defined tasks (cf. [5]). Thus, hierarchical tasks record the user interaction at different levels of detail and abstraction and automatically determine versions to be kept. In Quilt, at any time users may send messages to other users, for

example to inform others about the state of their work. A notion of extended group awareness that is inherent to the system is not supported. Moreover, alternative versions in a single system state, parallel versions of the system, or joint editing of a single system state are not supported.

At the database level, Käfer [7] made a proposal to version-based cooperation control. The approach is to relax the atomicity and isolation properties of database transactions to support long duration teamwork in design environments. To do so, transactions work on object versions and additional attributes, called features, are attached to the object types. During a transaction, feature values are assigned by the applications. The actual feature value set determines the degree of consistency of the versions and triggers the sending of predefined notifications to other transactions. At a certain degree of consistency, versions may also be exchanged between transactions. Our notion of extended object identity providing the extended group awareness may be implemented on top of this database approach. Of course, Käfer does not discuss user interface issues and questions of asynchronous and synchronous work.

CONCLUSION

We have shown how the functionality and cooperation modes of cooperative systems can be enhanced by version support and how mechanisms to support the smooth transition between these modes allow a flexible use of a versioned cooperative system. We illustrated how the users of a versioned cooperative system may choose freely between various combinations of different modes of asynchronous and synchronous cooperation that best supports their actual job. In doing so, extended group awareness enables the users to decide to change their cooperation modes dynamically.

The approach presented in this paper is not limited to hypertext. Any application domain that can be modelled by interrelated objects can be mapped to hypertext. Vice versa, the version support mechanisms provided by CoVer can be applied to these data models. In addition, many application domains show similar patterns of individual and cooperative work we have identified for hypermedia authoring. In general, the transition between these cooperation modes depends on the operations and activities to be performed on the data. Having identified these transitions, CoVer may also be used to integrate the cooperation modes discussed in this paper into systems of these application domains.

The versioned cooperative SEPIA is now in its early use. The experience will guide further improvements for cooperative systems. Our next plan is to elaborate concepts to integrate pre-planned and scheduled tasks with project management tools. To make application programming easier, we investigate a notation to descriptively define the version behavior of application-defined tasks.

REFERENCES

1. Dart S. Concepts in Configuration Management. In *Proc. of the 3rd International Workshop on Software Configuration Management* (Trondheim, Norway, June 12-14, 1991), pp. 1-18.
2. Dourish P., Bly S. Portholes: Supporting Awareness in a distributed work group. In *Proc. of the CHI'92* (Monterey, California, May 3-7), pp. 541-547.
3. Ellis C.A., Gibbs S.A., Rein G.L. Groupware: Some issues and experiences. In *Communications of the ACM* 34, 1 (Jan. 1991), pp. 38-58.
4. Fish R.S., Kraut R.E., Leland M.D.P. Quilt: a collaborative tool for cooperative writing. In *Conference on Office Automation Systems* (Palo Alto, 1988), SIGOIS Bulletin, 9, 2&3, (April & July 1988).
5. Haake A. CoVer: A Contextual Version Server for Hypertext Applications. In *Proc. of the 4th ACM Conference on Hypertext* (Milano, Italy, Nov. 30 - Dec. 4, 1992), pp. 43-52.
6. Haake J.M., Wilson B. Supporting Collaborative Writing of Hyperdocuments in SEPIA. In *Proc. of the ACM 1992 Conference on Computer Supported Cooperative Work* (Toronto, Ontario, Nov. 1-4., 1992), pp. 138-146.
7. Käfer W.A. Framework for Version-based Cooperation Control. In *Proc. of the 2nd Int. Symposium on Database Systems for Advanced Applications (DASFAA)* (Tokyo, Japan, April 1991).
8. Katz R. Towards a Unified Framework for Version Modelling in Engineering Databases. *acm computing surveys* 22, 4 (Dec. 1990), pp. 375-408.
9. Rodden T.A. Survey of CSCW systems. *Interacting with Computers* 3, 3 (Dec. 1991), pp. 319-353.
10. Stefik M., Foster G., Bobrow D.G., Kahn K., Lanning S., Suchman L. Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings. In *Communications of the ACM* 30, 1 (Jan. 1987), pp. 32-47.
11. Streitz N.A., Hannemann J., Thüring M. From ideas and arguments to hyperdocuments: Travelling through activity spaces. In *Proc. of the ACM Conference on Hypertext* (Pittsburgh,USA, Nov. 5-8, 1989), pp. 343-364.
12. Streitz N.A., Haake J.M., Hannemann J., Lemke A., Schütt H., Schuler W., Thüring M. SEPIA: A Cooperative Hypermedia Authoring Environment. In *Proc. of the 4th ACM Conference on Hypertext*, (Milano, Italy, Nov. 30-Dec. 4, 1992), pp. 11-22.
13. Thüring M., Haake J. M., Hannemann J. What's ELIZA doing in the Chinese Room — Incoherent Hyperdocuments and how to Avoid them. In *Proc. of the 3rd ACM Conference on Hypertext* (San Antonio, Texas, Dec. 15-18, 1991), pp. 161-177.
14. Weber A. Publishing Tools Need Both: State-Oriented and Task-Oriented Version Support. In *Proc. of the 15th Annual International Computer Software and Applications Conference (COMPSAC '91)* (Tokyo, Japan, Sept. 11-13, 1991), pp. 633-639.