DocMan: A Document Management System for Cooperation Support

Andreas Bäcker, Uwe Busbach

GMD, Schloß Birlinghoven, 53754 Sankt Augustin, Germany

Abstract

This paper describes DocMan, an interactive document management system which supports cooperative preparation, exchange, and distribution of documents.

Document folders are shared among people, allowing for collaboration on the basis of a common information space. Access to document folders is provided transparently. Each folder contains a set of document revisions and a set of drafts currently in preparation. Meta-information concerning what is going on with folders, revisions, drafts, and users is distributed.

DocMan's revision concept is based on a soft-locking scheme, avoiding both loss of work done simultaneously and access restrictions, thereby allowing users to work mobile while being periodically disconnected from any network.

1. Motivation

Looking at today’s business world, decentralisation of organizations and business companies has become a major success factor. Outsourcing, joint ventures, and international consortia have led to work environments which are geographically distributed across organizational and national boundaries. Empirical studies [1] verify the need for coordination support to avoid expensive inefficiencies, errors, and delays, and emphasize the importance of common information spaces in decentralised working environments. The implications for everyday work are that 'high tech' applications like videoconferencing etc. are less important than systems which allow access to (shared) information at any time and any place on the basis of a minimal technical infrastructure.

Taking these issues into account, we have developed a shared document management system – DocMan – to support such collaboration. The aim has been to provide an easy to use application for the exchange and cooperative writing of documents based on standard technical infrastructure and minimal software requirements. This fundamental design principle enables simple integration and usage of DocMan within any organization, offering the possibility for effective support of intra- and inter-organizational work groups [2]. Interaction with DocMan is realized by an easy-to-use point-and-click graphical user interface.

A basic concept of DocMan is the need to be descriptive rather than prescriptive, following the idea that CSCW applications should inform rather than constrain [3], and be regarded as a medium rather than a mechanism [4]. In order to follow those ideas of building CSCW systems on social protocols which currently do and will continue to exist outside of applications, two principal concepts concerning collaboration support are used. First, the main focus of DocMan is to collect and distribute meta-information on what is happening with revisions and drafts, which allows user-controlled coordination [5] of activities. Secondly, the access control schemes on folders should not be restrictive and rigid, but soft.

Soft means that access to folders is not actually restricted by strict locking policies used in database or file systems, but that users are notified that someone else is working on a draft of a document revision. This allows for a certain kind of awareness within a cooperative process. The information 'Who does what and when and why?' is offered by soft-locks which are a part of DocMan's meta-information concept. It may cause users to exchange email messages for coordination purposes, or to delay their work for a certain time period, and provides a basis for negotiation if necessary. In combination with the soft access control scheme, DocMan provides a revision concept which prevents any loss of information caused by concurrent modifications. A document revision is never overwritten, but instead a new revision is created at the time users commit modifications by adding their draft. No work can be accidentally destroyed.

To support workers whose portable PCs are not hooked to the network permanently, a take along has been implemented on the basis of a Mobile Station Support Toolkit [6]. Document revisions and drafts can be downloaded onto the portable computer, and the take-along keeps track of modifications while disconnected and propagates new revisions added when hooked into the network again.
Related work

There are already some applications [7,8,9,10] which offer means to access documents in a cooperative manner within decentralized settings. Liveware [7] can be used to handle the distribution of information between dislocated people, but provides no means to describe and to monitor a common activity on a document. Meta-information on document processing is missing.

The principle functionality provided by Liveware is an automatic data update to maintain consistency between computers at various sites, and it does not matter if this is done between computers belonging to different persons or a single person. Information transmission occurs as a side effect of interpersonal communication. This generic concept implies that Liveware does not provide explicit means for collaboration.

In contrast to applications like Liveware, task management systems take coordination aspects into account. Documents can be attached as resources to shared to-do lists of the Task Manager [8]. The artefact of shared tasks implements the distribution of information among persons, and the exchange of messages allows sharing of meta-information on documents within a task context. The drawback with this approach is that document distribution cannot be done in isolation, but is bound to the existence of a shared task. This implies that users have to be familiar with the concept of shared to-do lists in order to distribute documents.

Lotus Notes [9,10], a well-known and widely used system to support collaboration, is designed for information sharing between dispersed offices using WANs. Although Lotus Notes seems to be very suitable for decentralized working environments, there are two major problems. First, using Lotus Notes requires establishment and maintenance of a common database infrastructure by all cooperation partners. Secondly, Lotus Notes does not provide sufficient means to distribute meta-information on documents. For instance, no information on simultaneous modifications is provided. There is always exactly one ‘official’ version depending on the hierarchy of databases, and the time of information retrieval. Whether two database servers holding different versions have just been synchronized or will be synchronized at a later time – since Lotus Notes uses a copying mechanism instead of a real replication approach – influences the retrieval result. Alternative versions are overwritten during the server’s synchronization process. This implies that work and information may be lost, and there are no means to distribute information about this.

Looking at the concepts of these related applications, DocMan has features which are better suited to support groups in decentralized working environments: support for mobile work, distribution of meta-information, and provision of both a revision concept and a soft locking scheme allowing use of social protocols.

2. Basic concepts

DocMan supports the organization of information associated with the cooperative preparation of documents. It ensures that all people participating in the collaboration will have access to all relevant information. In particular, DocMan has been designed for working mobibly and over wide-area distances.

Typically, DocMan will be used to support small to medium working groups with about two to ten people. Additionally, DocMan may be used for the organization of personal work or for information dissemination in the sense of a bulletin board.

The model of document preparation is that, starting with some initial draft of a document, the document will be revised several times, each time creating a new revision. Information about available revisions is distributed over local and wide area networks to all users, thereby giving each user the chance to access any intermediate state of the document as well as to prepare new revisions simultaneously with other participants. It is expected that coordination and negotiation of the document preparation process - the “Who does what and when?” - is performed by informal communication means, such as electronic mail, phone, or face-to-face.

Document folders, drafts, and revisions

Each document has an associated document folder holding the set of revisions produced so far by those people participating in the preparation of a document (Figure 1). In this paper, the term ‘document’ will commonly be associated with the set of its revisions, and when speaking about the contents of a document, the contents of the last revision are implied. Users participating in the preparation of a document are called participants, and may reside on any host in the Internet, identified by their user name and their Internet or network domain name.

The set of all document folders existing world-wide is managed by the distribution and replication service (or simply the distribution service). The distribution service is distributed over several network sites, with one server running in each different network domain where users will be participating in the cooperative preparation of documents. The distribution service can be viewed as an information store which stores, replicates, and distributes information about which document folders and revisions exist, which users have access to folders, and which drafts are currently in preparation. This information is provided to participants for coordination, negotiation and collaboration purposes.
The distribution service is also responsible for storing the revisions in the document store, which is implemented using plain files. The distribution service, also operating as a rudimentary WWW-Server, generates a Universal Resource Locator (URL) for each revision. This allows participants to grant people outside of the folder context access to a revision by simply sending them the URL. Those people may then access revisions with their WWW-browsers.

Drafts are always private to a single participant. They are stored on the participant’s local disk. When hooked into the net, the local document directory serves as a cache for copies of existing revisions in order to reduce access time. It is also used as a buffer for revisions that are about to be transferred to the document store, but cannot be transferred because the computer is not currently connected to the network.

A document may be any piece of information that is storable in a file, such as texts, spreadsheets, drawings, and pictures. A newly created document folder contains the initial revision of the document, together with a list of people who are allowed to access the already produced revisions and who may release new revisions. In order to revise a document, users first fetch an existing revision from the document store and produce an editable draft copy in a local document directory on their local disk. Producing a draft copy does not release a new revision, but distributes to the other participants a soft-lock object containing the information that a participant intends to prepare and release a new revision. After making the intended modifications to the draft, users release the revised document, thereby creating a new revision.

Existing revisions are always read-only and may never be changed. Modifications to a document may only be made by distributing new revisions. This scheme avoids the problem of information loss due to race conditions caused by concurrent work on the same revision of the document, since no information will ever be overwritten. Another advantage is that no strict locking mechanism is required for concurrency control, since revisions may safely be read simultaneously.

**User interface**

The main window of DocMan’s user interface (Figure 2) shows in three scrollable lists the document folders users have access to, the revisions in the selected document folder, and the soft-locks that are currently set on the selected revision.

Double-clicking on a document folder line fetches the latest revision from the document store, launches the revision’s application and shows the revision in a read-only window. Double-clicking on a revision fetches it and shows it in a read-only window. Double-clicking on a soft-lock opens the draft in a window ready for editing. Additional menu entries allow for printing of revisions, for making local copies of revisions, and for cancelling drafts.

Choosing the Create Folder menu entry pops up a dialog box that lets users create a new document folder. The initial revision will be copied to the document store, thereby making it available to the other participants. After the new folder has been created, further participants who share the new folder may be added by using the Participant Manager dialog and the Personal Address Book dialog described below.
Selecting a revision and then choosing the Make New Draft menu entry opens a dialog box that allows to create a draft copy of the selected revision on a local disk. Finishing the dialog box copies the revision from the document store to the local document directory, and launches the revision’s application. Additionally, a soft-lock object is placed on the draft and is distributed to all participants. After making changes and extensions to the draft copy, users release it as a new revision of the document through the Release Draft Dialog (Figure 3).

Identification of revisions

From the point of view of the distribution service, revisions (and other objects as well) are named using unique global identifiers. Although unique identifiers are an essential means for distribution and replication, they are not well suited to distinguish revisions at the user interface level. Therefore, revisions carry additional attributes that simplify their presentation at the user interface. From a user’s point of view, revisions are identified by their major and minor revision numbers, creator, creation time, and an optional textual comment giving an overview about what’s new (Figure 2).

This identification scheme allows different revisions created by different participants to carry the same revision.
numbers, thereby avoiding the need for a computer-supported scheme for generating unique global revision numbers. It is expected that participants negotiate revision numbers by informal communication. Users may even decide to create multiple revisions carrying the same revision numbers. For example, if users want to release several slightly differing revisions of a drawing or a picture, they might prefer to give them the same revision numbers, indicating differences through textual comments. Also, a new revision may be based on multiple existing revisions. In this case, the new revision may be created by using a merging mechanism (assuming there is one) built into the application the revisions were created with.

Revision management

The revision concept implies that the number of revisions may increase dramatically during a collaboration process. First, this may exceed the document store’s storage capacity. Secondly, orientation may be difficult due to information overflow.

To avoid those problems, a garbage collection mechanism is implemented based on an interested-in concept. Users can individually tailor the set of revisions being displayed on the main window. When users want to hide a revision, they are asked whether they are still interested in it. If they only want to hide the revision temporarily and guesses that they might need it again later, they indicate their interest. Otherwise, users are removed from the ‘interested-in’ list which has been initialized earlier with all document folder’s participants and is kept within the distribution service. A revision is removed from the document store as soon as nobody has any interests in it. This approach implies that a revision can be accessed as long as at least one participant is interested in it.

Participants may leave a document folder by removing themselves from the folder’s participant list. Leaving a document folder breaks any connection between users and the folder. Users may only be added to the participant list again by some other user on the participant list. When the last user leaves a folder, the folder and all its revisions are destroyed.

Since it may be necessary to keep a revision history available for documentation purposes, it is possible to configure whether versions are physically deleted from the document store or archived on to a backing store.

Automatic generation of revision numbers

Users may number revisions with arbitrary revision numbers. However, DocMan offers default revision numbers according to a scheme which normally frees users from keeping track of revision numbers: An initial revision will be numbered 1.0. When fetching a revision for drafting, the minor part of the highest revision number in use will be increased by one. The set of revision numbers in use is defined by the union of the set of revision numbers of all existing revisions and the set of reserved revision numbers from all drafts currently in preparation. This intention of preparing a new draft with this revision number and the participant’s name is distributed to all other participants as a hint that someone is preparing a new revision.

Due to network transportation delays, intentional part-time disconnection from the network, or network malfunction, the knowledge about what’s going on with a document folder may vary at different Internet sites, which might in turn cause DocMan to reserve the same revision number for different drafts. When releasing a draft as a new revision, DocMan suggests to assign the revision number that was determined when the draft was created. If additional revisions were released in the meantime, DocMan again suggests a different revision number, which is higher than all revision numbers given out so far.

User objects, user database and the personal address book

Establishing groups of participants requires the availability of a user database. At the system level, the user database is modelled using distributed and replicated user objects. At the user interface, the user database may be organized and manipulated through the Personal Address Book.

The user database is organized as a sequential list of all user entries and a list of groups, each containing subsets of user entries. Groups may be used in contexts where some operation should be applied to a number of users, such as adding a group of people to the participant list of a document folder. Groups are not distributed to other users, they are intended as a means for organizing a user’s personal working environment.

Interactive selection of users is supported by the Participant Manager dialog (Figure 4). The Participant Manager allows to select users from the user database and to add them for example as participants to a shared document folder.

Contacting other users

For each user of DocMan, a user object is distributed and replicated. Each user is uniquely identified by a pair (domain-name, login-name). Optional fields of user objects are the user’s full name, email address, postal address, phone, and fax numbers.

One problem is how to offer methods which support population and maintenance of the user database with as little effort as possible. Another problem is that due to the large number of potential users, user objects must be replicated exactly at those Internet sites that really need them. This problem immediately poses a few other questions, mainly related to the bootstrapping process:
In the average case, the solution described below requires users to enter a full description of themselves to the user database, and minimal descriptions containing the name and domain for half of the number of users they come into contact with. Replicates of a user’s user object are kept only at sites with other users sharing a folder with this user, allowing DocMan to operate with good performance even if a large number of people uses it.

**Primary and secondary user objects**

Whenever users run DocMan for the first time, they must provide the information necessary to create objects for themselves, which will then be marked as their primary user objects. Users must only supply the mandatory fields login name and domain. For the other fields, they may freely choose which fields to provide for other users and which fields to leave empty. Before being able to come into contact with other users, the other users must be added to the user database as well, thereby creating a secondary user object for them. We assume that a group of users have agreed through inter-personal communication on the cooperative preparation of a document, and that at least one member of the group knows enough about the rest of the group to be able to add them to the user database.

Usually, a secondary user object will have all its optional fields empty, as these fields will be automatically filled in later with data from the corresponding primary user object. DocMan also offers easy-to-use facilities for generating large numbers of secondary user entries from information sources such as user account databases, group account databases, or from directory services such as X.500.

Both primary and secondary user objects are distributed and replicated, but each time a replicate of a primary user object meets a replicate of its corresponding secondary user object, the fields from the primary user object are copied to the secondary user object and the secondary replicate becomes a primary replicate. As this process continues, all secondary user objects, including those in the user database of other users, will become primary user objects. Users’ primary user object also migrates into the user database of those people who share at least one folder with them, and whose address book do not yet contain an entry for them.

Users may change fields in their own user object, and propagate these changes to the user database of other users. This is achieved by a similar mechanism, which copies the fields of the most recently modified primary user object to the fields of older primary user objects.
Notification

Since changes to a document folder may be caused by different users, it is essential that all participants are informed about relevant things, such as the release of new revisions, or when other users leave or join a shared document folder. Notification about changes is of high importance after periods of disconnection from the network, for example when working while travelling.

Changes are always highlighted in a distinguishable colour in DocMan’s user interface. For example, the complete line for a revision is highlighted if that revision is new; the name of a document folder will be highlighted if someone has changed it; a user name is highlighted if this user has been recently added to the participant list. Changes indicated by highlighting may be confirmed explicitly by users after having noticed them, thereby unhighlighting them and making future notifications easier to spot.

Change histories

There are cases where users want to obtain more details about what has changed; for example who changed something, or when something was changed. Also, some changes can not be indicated by highlighting. For example the names of users who have left a document folder can not be highlighted because their names no longer appear in the participant list.

Explicit change histories have been modelled for these cases, providing a more detailed level of notification and information. For example, the change history for the participant list states in detail who introduced whom and when, and who left the folder at what time. If different users have made conflicting changes concurrently, change histories will serve as a very helpful mechanism to find out what really happened.

Email notification

Automated email notification is available as an optional mechanism for those users who merely act as observers rather than as active participants of the document preparation process. Those users may indicate that they want to be informed by email about really important changes and events, such as the release of new revisions or becoming a participant of some folder. Users with observer status will work with DocMan only occasionally, but are expected to check their mailboxes regularly.

3. Mobility

The notion of mobility [11] expresses that users can run an application on portable equipment while being disconnected from a network. For single-user applications, there is generally no difference between the mobile and stationary use of a computer. When users employ two or more computers for their work, they have to download the required data to work mobilly on their portable equipment. When returning to the office or home, they have to update the data on the stationary computer to preserve data consistency.

Within multi-user and CSCW applications, two new issues with respect to mobility arise

- Several persons may modify one and the same piece of data

Modification may not only occur sequentially, but also concurrently. Whereas simultaneous updates of data may be detected and subsequent conflicts may be resolved more or less immediately across LANs, time delays until detection of concurrent modifications may be long across WANs, and in particular within mobile working environments. This latency implies that conflict resolution processes are in general more complex due to the fact that more data modification operations may have been executed. An application which aims to run mobilly must provide specific means such as history information, version control, dependency detection etc. to support users in re-establishing a common consistent view on data.

DocMan’s concepts already consider this issue, allowing users to continue work mobilly. DocMan’s revision concept implements a simple, but effective revision control mechanism. Revision history and dependency information is distributed among participants. These features support the conflict resolution and negotiation processes when concurrent modifications occur. No modifications performed while disconnected are lost because no revision will be overwritten.

- Availability of revisions while being disconnected has to be ensured

Computers connected over a network are employed as a basic infrastructure to share information and to store common data persistently. While disconnected, this global network-based persistency store cannot be accessed. This means that an application which aims to support mobile work has to provide the means to up- and download data from the global persistency store onto the local disks. To allow for that, DocMan offers a take-along functionality.

The take-along keeps information on all revisions and drafts which are locally available, and provides means to put revisions into the local take-along directory. Information on the location of drafts are also kept in the take-along by default. Access to revisions is transparent, i.e. users cannot distinguish whether data is retrieved from the local take-along directory or from the global network store. When users release a draft as a new revision, this information is stored for future propagation. URLs and users’ rationale behind the new revisions are propagated by the same shared container mechanism generally used to
distribute meta-information about document folders among people. The contents of the take-along are displayed within DocMan’s main window when switching to the Take-Along view. The entries of the take-along are highlighted, providing users with a comprehensive overview.

Whereas references to drafts are automatically added to the take-along at creation time, users are responsible for copying revisions to the take-along. Users are also responsible for the removal of entries from the take-along when they are not longer needed. The direct involvement of users in the maintenance of the take-along addresses work done by [12] which indicated that no cache transparency for mobile working is possible as some reasonable pre-planning is required to download the data to be worked with while disconnected. In order to provide an easy set up of the take-along, the idea of establishing a ‘working set’ [6] in a smooth way during users’ regular work was adapted for DocMan’s take-along implementation. When a revision is retrieved from the document store for the first time, users can decide to copy it from the temporary local document directory to the persistent take-along. The local document directory cannot implement the take-along functionality since there is no guarantee that documents recently accessed are still kept in this directory while disconnected because of caching strategies and cache size restrictions. In contrast, the take-along guarantees that a selection of revisions is always stored on the portable computer. Users can add further revisions just before disconnection to work mobilly if required.

Taking a closer look at the take-along’s usage, it is quite obvious that users cannot expect to always have all revisions on their portables which are required for working mobilly. There are two main reasons for this:

- Users may simply forget a revision needed.
- While not hooked on the net, some new revisions are added to the take-along.

Two independent object structures are used for implementation purposes. The data layer stores and keeps all relevant information to be displayed at the user interface. Its objects are implemented in the object model of C++. The objects in the data layer reside on the client contains the revisions he needs currently. In the meantime, Uwe adds a new revision while working at his office. Since he knows that Andreas is staying at home today, and since he thinks that his recent revision tackles some very new aspects, he calls Andreas at home to discuss these issues. Of course, the take-along does not contain Uwe’s current revision since it was added after disconnection, but it would be helpful to have it available to support this collaboration.

The scenario shows quite obviously that – although the take-along supports mobility by making information available while disconnected – there may arise the need to connect to the server in order to retrieve revisions while working mobilly. Access to revisions not yet in the take-along can be implemented using either wireless radio links or ordinary telephone modem connections.

4. Implementation

DocMan is based on a three-layered client-server architecture (Figure 5). DocMan’s user interface displays data retrieved from the data layer. The object structure of the data layer is mapped onto objects of the shared container layer. The shared container layer implements the distribution service (Section 2) and consists of a client and a server. It provides means for storing information within a single domain as well as for distributing and propagating information amongst domains.

![Figure 5. DocMan software architecture](image-url)
side, and are not distributed or replicated. The data layer wraps the distribution and replication mechanisms of the shared container layer, and thereby eases the implementation and evolution of DocMan’s user interface.

The shared container layer is based on an object model implemented using a term notation similar to Prolog. The term notation is very suitable for the implementation of replication, distribution and synchronization mechanisms. For example, objects are referred to by unique names. The basic abstraction of the shared container layer is the shared container. Shared containers keep attribute instances implementing the specific object to be distributed, and they are replicated between both clients and network domains. Containers and their contents, respectively, are updated by sending operation request messages to be executed. Logical vector clocks [13] are employed to implement a causal order delivery protocol which takes possible dependencies between operations into account and is able to overcome periodic disconnection. Modification events performed on containers while not connected are spooled to allow for later propagation. The server part of the shared container layer stores and administers the containers within one domain whereas the client part contains only those containers modelling the folders users participate in.

An interface between the data layer and the client part of the shared container layer maps the two object models together. To do so, references and call-backs are set up at instantiation time allowing for propagation of events in both directions. Notification and consistency maintenance based on causality occurs between servers of different domains as well as between server and client. The latter allows for mobility since it handles periodic disconnection between portable computers and server.

This basic scheme for distribution and sharing of arbitrary data is generally suitable for asynchronous CSCW applications. The concepts have been generalized from work in the EuroCode 1 project which implements a Shared Container Toolkit (SCTK), and the implementation of DocMan’s shared container layer is based on the SCTK functionality.

The document store is based on a minimal HTTP protocol. When users add a new revision to a document folder, it is copied over to a file in the domain’s document store. A unique Universal Resource Locator (URL) is created, returned, and then distributed to other users. The usage of URLs enables access to the revisions throughout the network. In particular, people who are not using DocMan can access a document. In order to ensure that DocMan’s installation relies only on standard network infrastructure, a program module has been implemented which launches the revision’s application. This means that it is not necessary to install a WWW viewer to run DocMan. However, this module can be easily replaced by any other WWW viewer to provide richer functionality.

The DocMan server runs on Solaris 2.4 using TCP/IP as the network protocol. DocMan clients require a PC with at least 8 MB RAM and run on both Windows NT and Windows 3.11.

5. Outlook

In this paper we have described our interactive document management system DocMan which supports the cooperative preparation, exchange, and distribution of documents. It requires a minimal standard technical infrastructure and follows the idea that CSCW applications should support collaboration by providing a common meta-information space based on document folders. The basic paradigm is to describe what is going on rather than to prescribe what to do next.

Two different document types, revisions and drafts, can be put into document folders. The revision concept developed and implemented avoids the loss of information, and the notion of drafts and the soft-locking scheme employed allows for both simultaneous working and distribution of meta-information on drafts currently in preparation. The architecture of DocMan and its basic concepts allows users to work mobilily while being periodically disconnected from any network.

Future research activities will include empirical studies on the suitability of DocMan’s meta-information concept and its capability to support mobility adequately. A six months practical, 'mobile' use has recently taken place at Great Belt, the progress chasing company monitoring the construction of a bridge between Denmark and Sweden, and we are now in the process of evaluating the users’ comments.

In particular, further investigations have to be carried out into consistency issues. Our current approach utilizes social competence and (in addition to information provided by DocMan) informal information received from outside the application to keep the system in a consistent state from the user’s point of view. However, conflicts may arise since it might be ambiguous which revision has to be regarded as the official one within a collaboration context. For example, three people are working on a document, but they cannot decide which revision should be submitted to their manager when the time comes to

---


finish their task. To support users in this conflict resolution process it may be helpful to introduce notions of owner and possessor at the revision level and responsible at the document folder level. Along with these notions certain roles may be defined which can be employed to determine which revision is the approved one in case negotiation fails. The relationships of roles and their rights needs to be specified.

Further, a closer look at the possible notification schemes is necessary. Should there be passive notification - e.g. information about drafts in preparation is given only when someone either tries to access a revision or clicks explicitly on an respective icon to obtain the data - or is an active notification scheme - e.g. the colour of an icon changes when something has happened - more suitable. What influence has the collaboration or organization context on this issue? To allow for an even more individual tailored interface, an event filtering mechanism [14] may be implemented and evaluated.

DocMan provides a simple, but powerful revision control scheme which supports handling revisions created by arbitrary applications. The question whether it is adequate to implement enhanced revision control and administration schemes and, if so, which ones are most suitable, is a research challenge. Are there generic mechanisms supporting merge processes within a collaboration which do not restrict the users' choice to use an editor according to their individual preferences? A general, object-oriented framework was proposed in [15], however implementations of merge algorithms rely either on rudimentary document formats - e.g. only ASCII text can be handled or are application specific (e.g. [15], [16]).

6. Acknowledgements

We would like to thank all members of the EuroCODE project for helpful discussions. In particular, we wish to thank Jonathan Trevor and Richard Bentley for their comments on drafts of this paper.

7. References


5. Greif, I., Sarin, S. 'Data sharing in group work'. ACM Transactions on Office Information Systems, 5(2): 187-211.


