CoAUTHOR

A HYPERMEDIA GROUP AUTHORING ENVIRONMENT

Udo Hahn, Matthias Jarke
Universität Passau, P.O. Box 2540, 8390 Passau, W. Germany

Klaus Kreplin
Triumph-Adler AG, Hundingstr. 11, 8500 Nürnberg, W. Germany

Marisa Farusi, Francesco Pimpinelli
Direzione Olivetti Ricerca, 56100 Pisa and 20090 Milano, Italy

ABSTRACT

The CoAUTHOR system provides a real-time-oriented environment for multiple authors who wish to collaborate on the production of hypermedia documents. In this report, we describe a model of hypermedia document authoring, consider the group aspects of co-authoring, and the technical communication and coordination tools we are using as a basis for the implementation of a CoAUTHOR prototype. The interactions among the members of the authoring team concerning idea processing, document design and generation as well as group-specific activities such as critiquing issues, negotiating divergent opinions, and treating inconsistent or incomplete specifications are shown to be fairly knowledge-intensive and thus require maintenance facilities provided by a sophisticated knowledge base management system underlying the hypermedia surface.

1 Introduction

The design and development of the CoAUTHOR* system approaches the problem of producing hypermedia documents (such as technical manuals, surgeon reports, etc.) by multiple authors. We shall refer to this computer-supported collaborative activity as *co-authoring*. *Hypermedia* documents integrate features of multimedia objects and of hypertext documents (see YANKELOVICH *et al.*. [1985], CONKLIN [1987], and HALASZ [1988] for surveys): they consist of multimedia (i.e., text, image, graphics, possibly sound and video) chunks interconnected by semantic-bearing links. CoAUTHOR documents can therefore be explored in a non-linear fashion, either following arbitrary link sequences or according to one of possibly several pre-designed "guided tours" [TRIGG 1988]. Additionally, however, CoAUTHOR documents also have a recorded *design history*, relating portions of the actual document to the ideas which they are intended to elaborate, to the formal and layout requirements of a particular document style, and to the authors of these objects. Thus, the CoAUTHOR system is fairly knowledge-intensive, and the above relationships must be maintained by a knowledge base management system underlying the hypermedia surface.

Our work in CoAUTHOR concentrates on three major facets of co-authoring, each of which will be elaborated in this paper:

- ***A Model of Hypermedia Document Authoring***
  A conceptual model of *idea processing* (incorporating idea generation and refinement based on critiquing content issues of a document), the transformation of ideas to a formal document outline in terms of *document design*, and the actual *generation* of documents within a hypermedia environment.

* CoAUTHOR is developed as part of MULTIWORKS (MULTimedia Integrated WORKStation), a recently begun ESPRIT-II Technology Integration Project supported in part by the Commission of the European Communities under ESPRIT contract 2105.

- ***A Group Model of Co-Authoring***
  A conceptual model of group activities involved in authoring documents which includes the individual *generation* of content issues, document outlines, and actual hypertext fragments, their critical *annotation* by other group members, and the final *configuration* of ideas, design decisions and hypertext portions in terms of a coherent hypermedia object. Group work of that kind requires consistency-checking routines and negotiation-based alignment strategies to cope with conflicting or alternative content issues, outline proposals, and hypertext chunks that may result from the annotation and previous synthesis cycles.
- ***Group Communication and Collaboration Tools***
  Electronic conferencing devices serve as tools which provide technical support to *idea exchange* and *discussion* of diverging opinions. Multi-user access facilities to remote data bases and text files additionally must be supplied for *retrieving* hypermedia background material (facts, graphics, etc.) while checking the validity of ideas, referring to document plan libraries, and actually writing parts of the document.

	GROUP LEVEL		
DOCUMENT LEVEL	**Generation**	**Annotation**	**Configuration**
Idea Processing	idea generation	idea annotation	idea combination, idea alignment
Document Design	outline generation	outline annotation	outline alignment
Document Generation	topic contracting, generation of text portions	text annotation	macro-level text assembly, text revision, text alignment

Table 1 Document Level and Group Level of Co-Authoring Activities

The interrelations between the document and the group level of co-authoring activities are summarized in table 1. The process of ***authoring a hypermedia document*** is based on a tri-partite model (cf. HIRANO [1988] for a similar organization of the hypermedia authoring process). During the first phase of *idea processing* the issues (major topics) are determined which have to be covered by the document. Ongoing conceptual refinement and alteration of content blocks leads to the incremental specification of the knowledge actually to be incorporated into the document. This first phase is entirely dedicated to the *conceptual* level of the authoring process (*what* actually should be communicated in the text). During the second phase of *document design* a *formal* document structure has to be set up and associated with the conceptual items from the idea processing phase. In this round the physical organization of ideas in terms of their localization in a hypermedia document graph is fixed (indicating *how* ideas are communicated in a text). Finally, during *document generation* ideas get implemented by appropriate hypermedia chunks. The members of the authoring team contribute natural language text portions, formulae, graphics material, pictures, etc. to fill in the document outline and produce the final hypermedia document.

The ***group model of co-authoring*** is also characterized by a partition into three major layers. Group work starts with the *generation* of individual contributions, such as ideas, outline proposals, and specific text portions. Each of the objects emerging from these phases is open for *annotation* by other group members, i.e. the attachment of critical comments, the augmentation by alternatives, etc. After collecting the group's conception of individual contributions these units have to be combined according to the thematical and formal requirements of the whole document. This is essentially a *configuration* task putting pieces of individual objects together -- relating single ideas to form a structured idea aggregate (contents of the whole text), structuring outline proposals in terms of a comprehensive text plan, assembling text fragments in order to build a coherent hypermedia document. During this phase, incomplete specifications, inconsistencies and multiple (often mutually exclusive) solutions are likely to evolve on different levels of the co-authoring process. Therefore, negotiation-based alignment strategies have to support the resolution of conceptual conflicts, selection/adaption of conceptual and thematic alternatives, etc. As a result, issues, outlines, and text passages can either be con-

firmed, modified, or withdrawn, thus leading to a global readjustment of the original contributions.

The co-authoring activity may alternate among these description levels rather freely; no rigid phase model is intended. However, the relationships between represented ideas, their association to document outlines, and the actual hypermedia document chunks should always be preserved for purposes of consistency and completeness control.

Co-authoring in a hypermedia environment can therefore also be viewed as the creation, continuous revision and maintenance of three distinct ***knowledge bases***: an idea knowledge base (*IdeaKB*) keeping the conceptual contents of the hypertext documents, a hypermedia document base (*HyperDocB*) which contains the formal characterization of document outlines, and a hypermedia object base (*HyperObjB*) which consists of literal hypermedia objects (text portions, pictures, images, etc.). IdeaKB, HyperDocB, and HyperObjB can be considered as different views of the same hypermedia knowledge base (*HyperKB*) that incorporates the relations holding among the various representation levels of issues, e.g., relating a set of ideas and its associated textual realization to a particular location in a document graph.

The knowledge base perspective on issues relieves them of their informal nature and assigns them the formal status of knowledge base objects. Accordingly, they can only be accessed by a limited set of operators which are described in the following sections. The internal structure of issues already distinguishes the different document levels of co-authoring. It consists of IDEAS, a layer used for conceptual specification and refinement, the OUTLINE layer, used for document design information, and the document's TEXT layer where actual hypermedia text is placed in (cf. the figures below).

The paper is organized as follows. Section 2 considers in detail the authoring activities of hypermedia document production, while section 3 focuses on the group procedures inherent to the co-authoring task. A sketch of the knowledge representation system we use is provided in section 4, together with an architecture for its integration with a multimedia object server and a real-time conferencing system. For space reasons, we do *not* discuss the models of general group conversation [HAHN/JARKE 1989] which structure our messaging mechanism, nor do we address the specific problems of large-scale multi-document creation management [THANOS 1989, ROSE/JARKE 1989].

2 Authoring of Hypermedia Documents

This section introduces the major operators provided by the CoAUTHOR system to create hypermedia documents. We shall concentrate on the most characteristic operators for idea processing, document design, and document generation. As an example, we illustrate the problem of developing a user manual for the CML knowledge representation system (which also happens to be the one used for the CoAUTHOR prototype). Such an application appears suitable for CoAUTHOR because it requires hypertext as a basis to provide on-line help, cross-referencing, and descriptions of hardware-specific system variants (e.g., running on SUN or VAX machines, requiring ASCII or bitmap terminals); similarly, multimedia is useful to give mock-up demos of interesting system features, and expertise of multiple specialists must contribute to its creation.

2.1 Idea Processing

Electronic idea processing shares a lot of similarity with brainstorming activities which have become an established group problem-solving instrument in face-to-face meetings [APPLEGATE *et al.* 1986, STEFIK *et al.* 1987]. We provide a basic set of *conceptual operators* to create and augment the set of ideas and relations holding among them; the reader may visualize these operators as choice buttons or shown on pop-up menus. The most basic operator for idea generation initializes a basic idea object in the IdeaKB:

`Create_Idea( Agent, IdeaLabel )`
creates an idea node in the IdeaKB whose internal structure is displayed below (assuming `Create_Idea( Ruth, CML#UserManual )` to be issued to the IdeaKB by the user '`Ruth`').

Each idea node is assigned a unique `IdeaLabel` (such as `CML#UserManual`) in order to reference ideas by name. The IDEAS area displays more specific ideas related to the

root node `IdeaLabel` in a hierarchical manner (see the figures below). The root node also characterizes the currently considered `ISSUE` (`Status` and `Score` are dealt with in a later section). Each concept node in `IDEAS` also contains the name of the originator of the idea (`Agent 'Ruth'`) and a time stamp (`Aug/89$09:21:10`) by default. This knowledge is required to dynamically configure debates about controversial issues (although it is hidden in the subsequent examples).

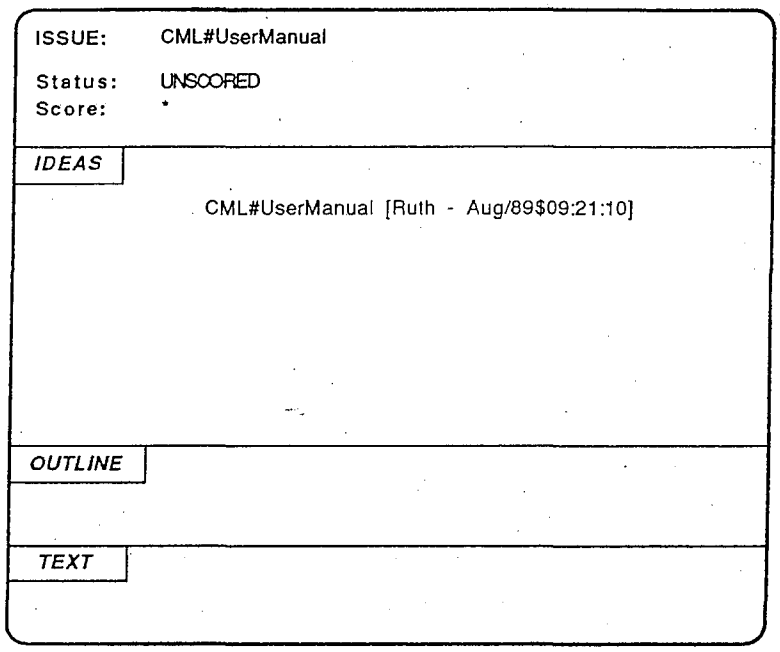

Figure 1 Creation of an Idea Node

Two basic operators are supplied by which a user can refine the IdeaKB with knowledge about conceptual hierarchies and concept aggregation:

`Refine_Idea_Hierarchy( Agent, GeneralIdea, SpecificIdea )`
introduces a conceptual specialization in the semantic net displayed in the `IDEAS` area -- given one of the concepts involved either a more `SpecificIdea` or a more `GeneralIdea` is inserted to expand the idea net in a hierarchical way.

`Refine_Idea_Attribute( Agent, Idea, IdeaAttribute )`
provides for the augmentation of simple concept nodes in terms of complex internal structure -- in this way, an already existing `Idea` node in the `IDEAS` area is assigned a conceptual `IdeaAttribute`:

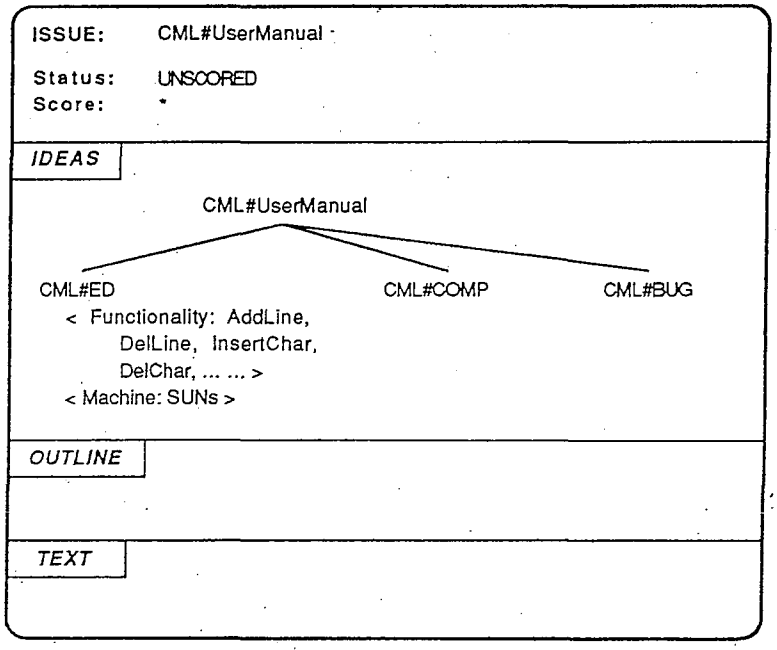

Figure 2 Hierarchic and Attribute Expansion of Idea Nodes

In Figure 2 the top level idea `CML#UserManual` is specialized by more specific ideas: `CML#EDitor`, `CML#COMPiler`, and `CML#DeBUGger`. Attribute expansion is illustrated with respect to the CML editor's functionality and an indication of its host machine.

The above operators introduce *taxonomic/terminological* knowledge into the IdeaKB. In order to account for *propositional* knowledge of the domain further relation types must be made available:

`Add_Prop_Relation( Agent, Rel_type, Idea1, Idea2 )`
serves as an operator which links node `Idea1` in the `IDEAS` area to an `Idea2` node via a pre-defined set of semantic relations -- `Rel_type` -- such as:

`Is_PreCondition_of`
`Idea2` logically, causally, etc. requires `Idea1`
`Is_PostCondition_of`
`Idea1` logically, causally, etc. results from `Idea2`
`Is_Smaller [_In_Space, _In_Time, ...]`
`Idea1` takes less space. time, etc. than `Idea2`
... ...

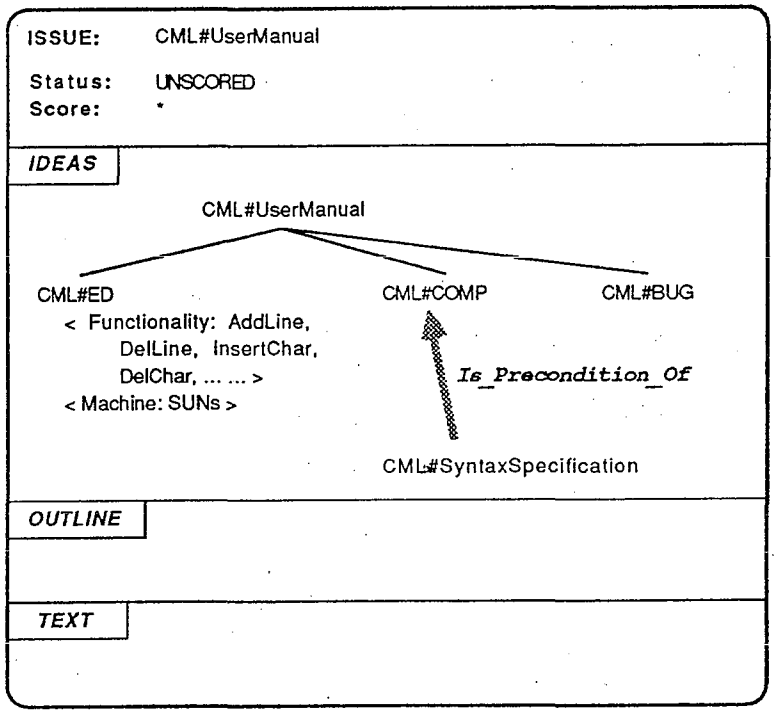

Figure 3 Propositional Expansion of an Idea Node

Figure 3 gives an example of the logical dependency between a language compiler and the availability of its syntax specification as a prerequisite.

Unless otherwise specified, ideas contained in the IdeaKB are displayed by a hierarchical idea browser. The following conventions underlie the graphical display mechanism of CoAUTHOR (see Figure 4): Each node in the `IDEAS` field is an idea itself. This is obviously true for the root node in the `IDEAS` field (`Editor`) of the upper window, since its label is identical to the name of the `ISSUE` it stands for. But it is also true for the more specific terms attached to the root, its siblings (`CML#ED-Line.1`, `CML#ED-Line.2`) as displayed in the lower two windows.

In the `IDEAS` field, subordinates of the root idea node are displayed up to two specialization levels below that of the root node. This seems sufficient for the conceptual orientation needed by the user. If more than two specialization levels are available for some node, a rounded box symbol attached to it indicates even more detailed specialization options. If that kind of more specific information has to be accessed a mouse click on the appropriate specialization node creates its associated issue object on the screen. Note that in this case the root node of the concept graph in `IDEAS` is labelled by the specialization object selected by the mouse

click and that the ISSUE is updated accordingly (illustrated in figure 4 by the selection of the more specific editor instances CML#ED-Line.1 and CML#ED-Line.2). This rearrangement of idea chunks serves to re-adjust the focus of idea generation.

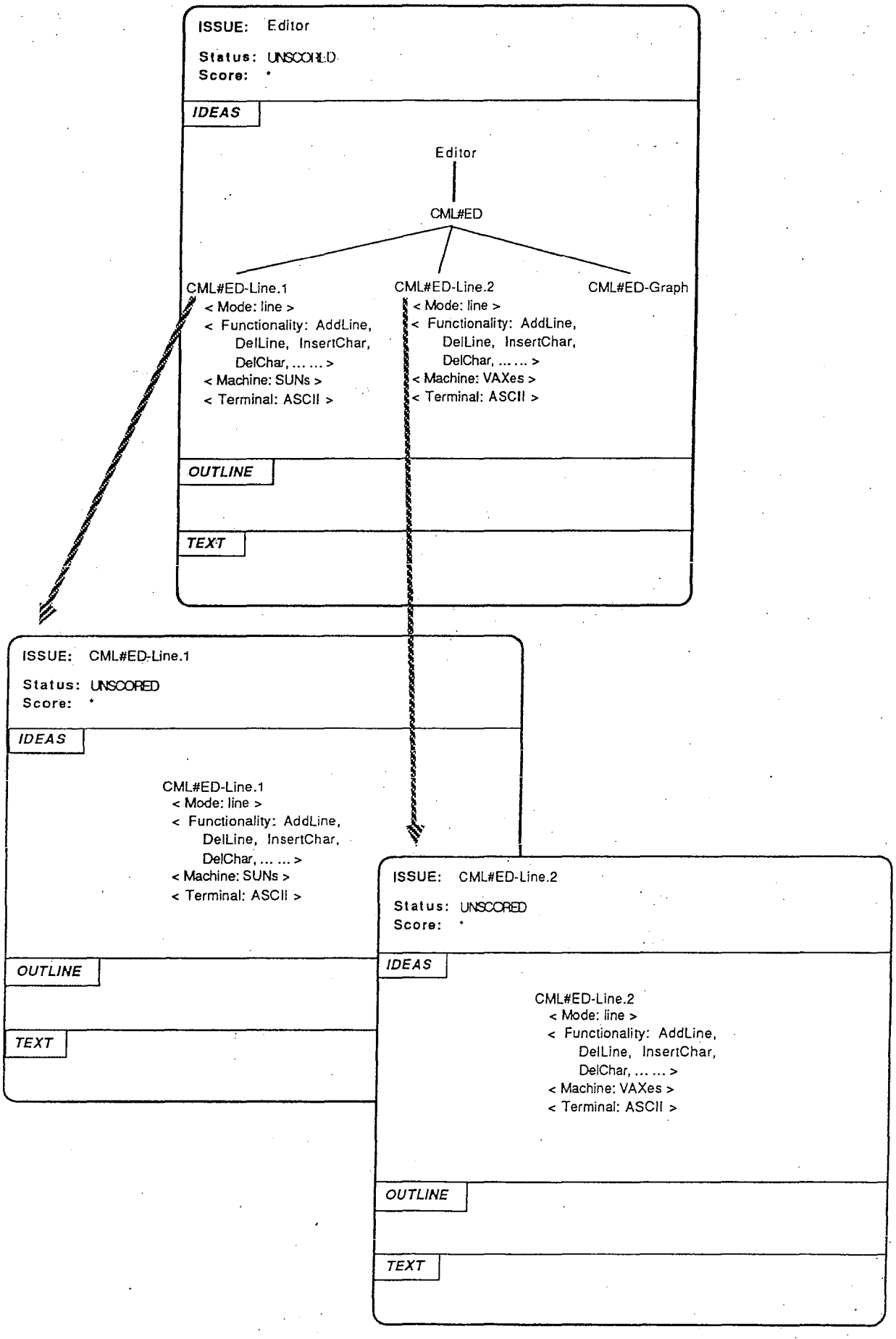

Figure 4 Hierarchical Browsing of Idea Nodes

2.2 Document Design

The IdeaKB consists of a complex conceptual graph. Its major structuring facility is hierarchical specialization. The conceptual relations do not account for any organizational requirements of communicating ideas to readers based on a comprehensive document outline (in hypermedia terms, some kind of standardized navigation script). This kind of information has to be supplied in the document design phase which, basically, consists of a two-step procedure:

- The specification of a formal document structure is achieved by *chunking and sequencing operators* which logically group and sequentialize content portions of a hypermedia document according to thematic, layout and message-specific requirements.
- The assignment of idea objects of the IdeaKB to that document outline is achieved by *outline operators.*

In the course of outline generation the formal structure of the hypermedia document is fixed. Formal structure relates to the *thematic dependencies* among and the *order* in which selected issues are dealt with. The basic object to be manipulated during outline generation is a topic chunk which represents a particular topic of the document and thus resembles the entries of a table of contents. Topic chunks are created by

`Create_Topic( Agent, TopicLabel )`
creates a topic node in the HyperDocB whose internal structure is displayed in the OUTLINE window:

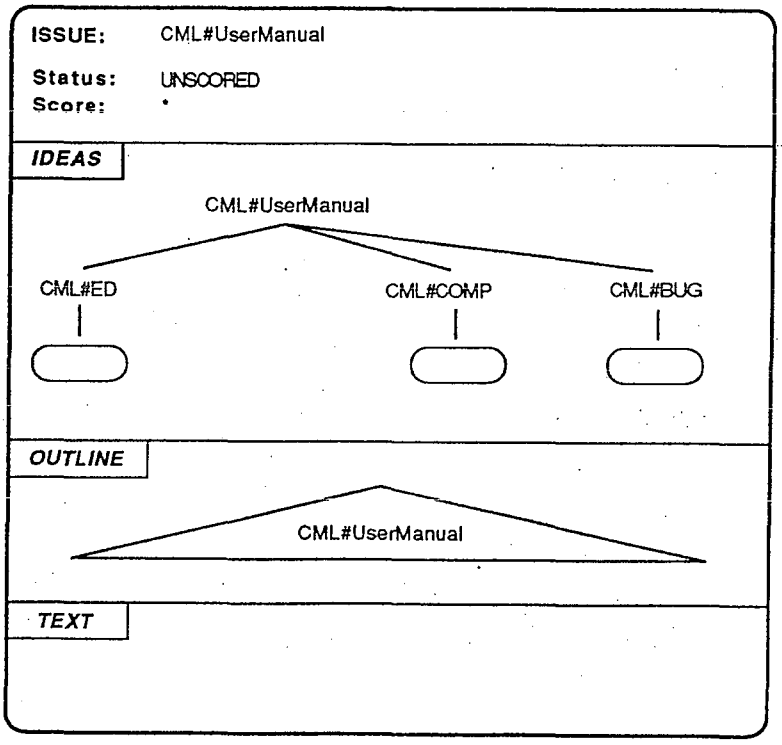

Figure 5 Creation of a Topic Node

Each topic chunk (iconized by triangles) is assigned a unique `TopicLabel` (`CML#User Manual`) in order to reference topics by name. The OUTLINE area displays the entire topic chunk graph in a hierarchical manner. Each topic node in OUTLINE also contains the name of the originator of the topic (`Agent`) and a time stamp by default (again, this information is hidden in Figure 5).

Thematic dependencies can be expressed following closely the schema provided by the conceptual relations in the IdeaKB, but they can also depart from it significantly. In such cases they mirror the influence of pragmatic considerations such as well-formed argumentation lines, the reconstruction of causal lines of reasoning, or tutorial matters of exposition, etc. We provide fairly general chunking operators to account for the thematical grouping of issues:

`Sub_Chunk( Agent, Topic1, Topic2 )`
subchunking indicates that `Topic1` covers a particular thematic aspect (is part of) `Topic2` and thus groups them together under document coherency aspects.

Application of the chunking operator to the conceptual ideas of the IdeaKB yields a hierarchic partition into thematically related issues. What still lacks is an indication of the sequential order of issue tokens when accessing the hypermedia document. This is supplied by

`Precede_Chunk( Agent, Topic1, Topic2 )`
the precedence relation indicates that `Topic1` physically precedes `Topic2` when running through a 'guided tour' of the hypermedia document.

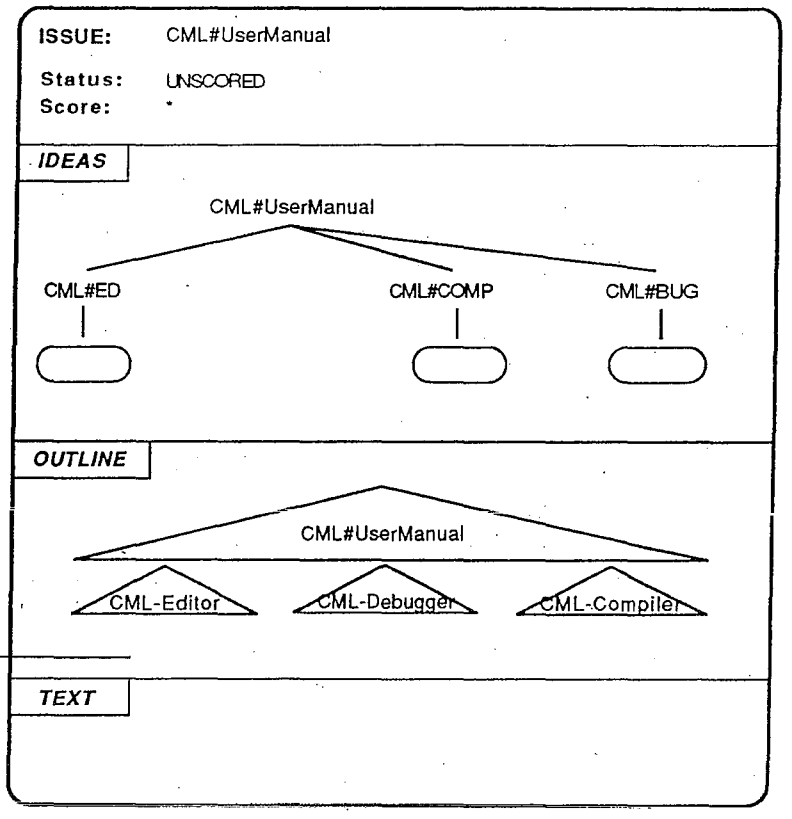

Figure 6 Chunking and Sequencing Topic Nodes

`CML-Editor`, `CML-Debugger`, and `CML-Compiler` are conceived as sub-chunks of a `CML#UserManual`. In addition, treatment of the `CML-Editor` will precede that of the `CML-Debugger`, while `CML-Compiler` will follow the exposition of `CML-Debugger`.

As a result of this phase of document design, a document structure specification graph is created with topics being arranged in hierarchical and sequential order. This is in line with cognitive findings related to the underlying principles of writing processes [SMITH *et al.* 1986]. It is also interesting to note that document specification graphs for text generation are a reverse interpretation of text condensation graphs which can be used for text summarization [HAHN/ REIMER 1988]. After agreeing upon the document outline (technical prospects of the negotiation procedures involved in that phase are treated in section 3) topic chunks have to be associated with ideas from the IdeaKB. The basic outline relationship is provided by

`IdeaToTopicMapping( Agent, Idea, Topic )`
an outline link connects a `Topic` from the document outline graph contained in HyperDocB (`OUTLINE` window) to an `Idea` from the IdeaKB (`IDEAS` window).

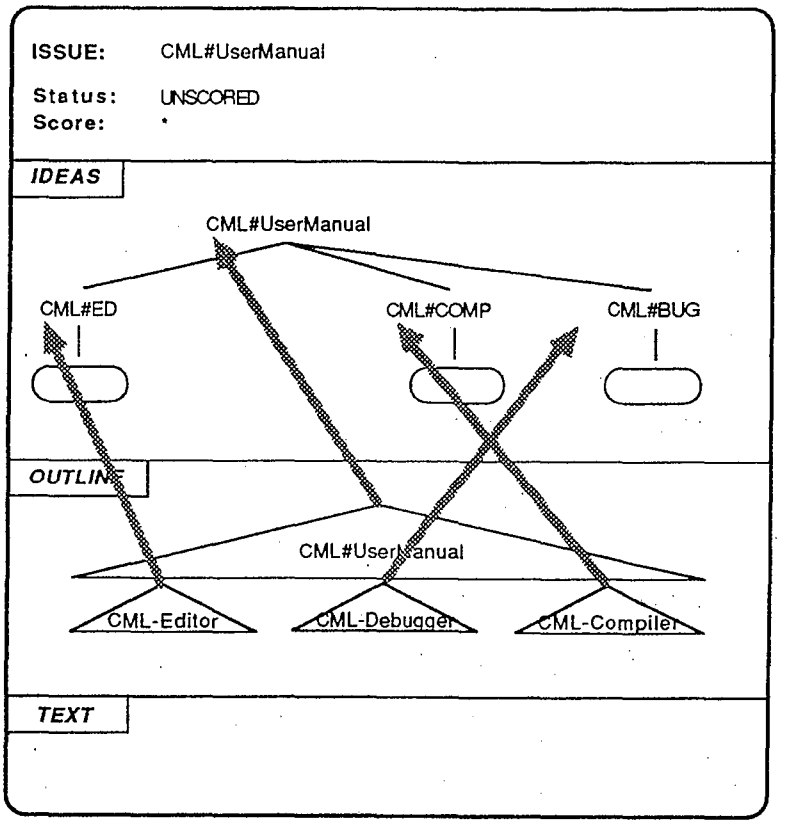

Figure 7 Outline Linking of Idea and Topic Nodes

Figure 7 provides a straightforward mapping of topic chunks onto idea identifiers; more complicated cases can be imagined.

2.3 Document Generation

After the document design phase the basic structure of the HyperDocB has been specified in terms of an outline description of the entire document. Furthermore, the objects of the HyperDocB are properly related to the objects of the IdeaKB through outline links between ideas and topic chunks. Relations to the HyperObjB, however, are set during document generation in that ideas related to some already established formal document structure are gradually transformed into actual hypermedia text. In order to achieve this goal,

1) agents have to take the responsibility for the realization of a topic in terms of literal text based on a specific commitment (topic contracting),
2) agents have to fill in topical outline specifications by hypermedia text, i.e. they implement a topic based on its idea specification.

Following the proposals of DAVIS/SMITH [1983] the process of *topic contracting* is preceded by a negotiation procedure where some topic is announced to be ready for implementation by text, a bidding phase where authors interested in elaborating on that issue are submitting an offer for realization, while the awarding procedure assigns the most capable bidder a contract in which a formal agreement is settled to produce literal text on the topic under negotiation. We shall not discuss this process here; its final result is established by:

`Topic_Contract( Agent, Topic )`
some `Agent` takes full responsibility in the textual realization of the `Topic`.

Given the contract of an author agent for a topic, the agent can choose how to realize the part of the document that is covered by the contracted topic. The formal document specification is now transformed, filling the document outline with multimedia objects (text, graphics, images, voice, etc.). We consider this step to be mainly carried out manually. Only limited semi-automatic devices will be supplied which provide technical assistance for creating the final document form.

Basically, the author may use existing material from the HyperObjB or may create text pieces on her/his own. This requires the provision of appropriate *textual* operators:

```
Use_Object( Agent, HyperMediaObject, Topic )
Create_Object( Agent, HyperMediaObject, Topic )
Edit_Object( Agent, HyperMediaObject, Topic )
```

The `Use` operator retrieves some `HyperMediaObject` (text, picture, diagram, voice recording, movie, etc.) from the HyperObjB and links that object as part of the hypermedia document portion related to `Topic`.

The `Create` operator initializes some `HyperMediaObject` (say, a particular text passage) that is physically located in the HyperObjB and links that object as part of the hypermedia document portion related to `Topic`.

The `Edit` operator is a macro for various common hypertext editing commands (such as add, delete, modify, paste) which manipulate some `HyperMediaObject` in the HyperObjB and link that object to the hypermedia document portion related to `Topic`.

The focus of our work in the model of hypermedia document authoring is particularly on the idea processing and to a lesser degree on the document design phase of co-authoring, while the document generation step (after the contracting procedure) requires only the application of sophisticated hypermedia editors and document management systems that are part of the system's tool environment (cf. section 4). With respect to output considerations the system provides full hypermedia functionality, but it shall also allow for the production of linearized documents, e.g., for printing.

3 Group Aspects of Co-Authoring

So far, we have concentrated on the *generation* part of the authoring process of a hypermedia document that could have equally well been developed for a single-author application; however, in that context, it may be less useful to make such knowledge explicit than in a multi-author context where the knowledge base takes on the role of a communication device.

In this section, we shall therefore address the specific group aspects involved in *co*-authoring. These are basically due to the management of *annotations* of individual contributions by other members of the authoring team and the *configuration* of single specification/text parts to a coherent hypermedia document. Three major types of operators can be distinguished for these tasks:

- *Commentary operators* are used for critiquing already introduced issues, and to indicate alternatives or specification gaps.
- *Maintenance operators* check for the consistency, completeness, and coherence of the specifications provided.
- *Alignment operators* deal with the resolution of conflicts, inconsistencies, and specification gaps.

3.1 Annotations

Annotations occur at every level of the document authoring process: idea processing, document design, and document generation. They are attached to structures created by conceptual operators (IdeaKB level), chunking operators (HyperDocB level), or editing operators (HyperObjB level) and *modify* already existing nodes and vertices in terms of critical or supporting comments, or by the indication of alternatives. Corresponding operators form part of the standard methodology of hypertext systems considered as collaborative tools (cf., e.g., TextNet [TRIGG 1983], or gIBIS [CONKLIN/BEGEMAN 1988] for similar models). However, in the CoAUTHOR system they directly refer to the underlying knowledge representation system which allows for various control features.

```
Pro_Object( Agent, KnowledgeBase, TargetObject, Support )
Contra_Object( Agent, KnowledgeBase, TargetObject, Counter )
```

this operator links the already existing object node `TargetObject` to a new `Support/Counter` node with a positive/negative comment; depending on the choice of the `KnowledgeBase` (either IdeaKB, HyperDocB, or HyperObjB) this comment is displayed in the `IDEAS`, `OUTLINE`, or `TEXT` area of the `ISSUE` object.

Comments may not only relate to complex objects (ideas, topic chunks, text portions) as a whole, but may also refer to particular attributes of them. An appropriate operator to express this very specific kind of comment is given by:

`Pro_Attribute( Agent, KnowledgeBase, TargetObject, AttrLabel, Support )`
`Contra_Attribute( Agent, KnowledgeBase, TargetObject, AttrLabel, Counter )`
this operator links the attribute `AttrLabel` of an already existing object node `Target-Object` in the `IDEAS`, `OUTLINE`, or `TEXT` area to a new `Support/Counter` node with a positive/negative comment.

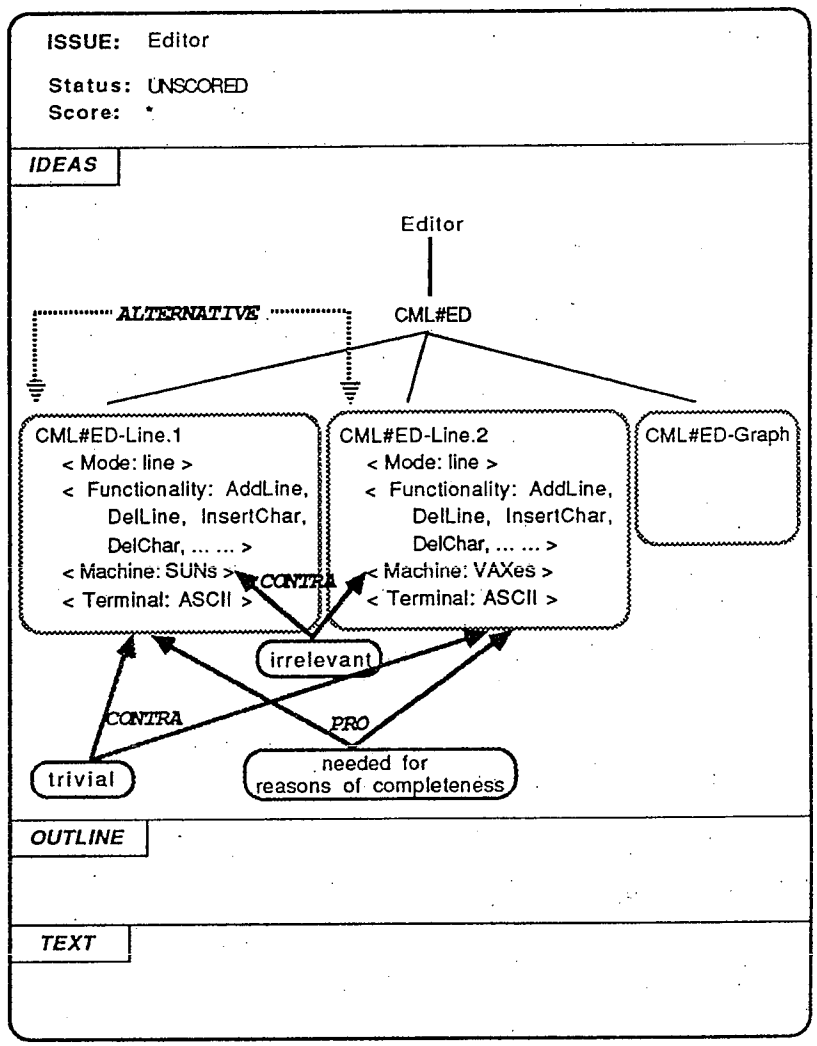

Figure 8 Commenting on and Alternative Linking among Issue Objects

In Figure 8 the description of both line mode editors (`CML#ED-Line.1`, `CML#ED-Line.2`) is criticized *in toto* as containing only trivial information (*CONTRA*), while supporting evidence (*PRO*) is based on considerations of providing complete descriptions in the manual (this may illustrate the effects of `Pro_Object/Contra_Object` operators). An unchallenged counter relates to the host machine attribute of both line mode editors which marks this information as being irrelevant.

To complete the collection of commentary operators provisions are made to comment on the relations linking the objects in different knowledge bases, too. Here we have:

`Pro_Rel( Agent, KnowledgeBase, TargetRel, Support )`
`Contra_Rel( Agent, KnowledgeBase, TargetRel, Counter )`
this operator links the already existing relation `TargetRel` to a new `Support/Counter` statement with a positive/negative comment; depending on the choice of the `KnowledgeBase` (either IdeaKB, HyperDocB, or HyperObjB), this comment is displayed in the `IDEAS`, `OUTLINE`, or `TEXT` area of the `ISSUE` objects involved.

The final operator intended for annotating already established objects relates pairs of objects as being alternatives:

`Alternatives( Agent, KnowledgeBase, Object1, Object2 )`
this operator links `Object1` and `Object2` that already exist in the chosen `KnowledgeBase` as alternatives on the conceptual/structural/text level (note that relations also are characterized by the name of their originator).

Figure 8 gives an appropriate example of the latter operator, since the descriptions provided by `CML#ED-Line.1` and `CML#ED-Line.2` are judged as alternatives.

3.2 Configuration and Alignment Strategies

Having established a common conceptual ground for the hypermedia document on the level of basic ideas, document outlines, and text portions, the members of the authoring team are called upon to reason and decide upon divergent opinions expressed by annotations. In addition, the group must synthesize individual contributions from the previous generation and annotation rounds to a combination of ideas/outlines/text confirmed by the group. The exchange, debate, and rearrangement of the differing views of single contributors must be supported by appropriate technical communication media in order to arrive at a 'group' conception of ideas and their transfer to a proper document format. This is the place of electronic conferencing devices which should not only passively record, but *actively* stimulate opinion exchange, the assimilation of individual or sub group opinions in order to arrive at some group opinion on debated issues. Besides dealing with various controversial issues, the focus of the members of the authoring team then gradually shifts from the *local* perspective of single ideas, topic chunks and text portions to the *global* perspective of the contents and structure of the entire hypermedia document.

At the end of the generation and annotation phases, application of conceptual, chunking, editing operators on the one hand and commentary operators on the other hand may have lead to a considerable growth of representation structures. Negotiable issues are likely to evolve on two different layers (see Figure 8):

1) on the conceptual/structural/textual level each object node and relation may have assigned to it bundles of pros and cons;
2) on the relational level pairs of concept nodes may have been marked as alternatives.

In order to resolve indeterminacies and potential inconsistencies the authoring team must take decisions which objects to include into the document based on current evidence and which to reject from it. In order to reach such a definite decision an alignment process is going to be started. This phase requires group-specific negotiation procedures for the filling of specification gaps, the selection of alternatives and the resolution of conflicts at each level of document production. Three major *maintenance operators* are provided to focus on incoherent and controversial issues:

`Show_Conflicts( KnowledgeBase )`
this operator returns all those object nodes and relations which have been assigned pro *as well as* contra links in the underlying `KnowledgeBase` indicating subjectively inconsistent specifications; note that *formal* inconsistency of the objects themselves is not considered here. In the examples above, e.g., `CML#ED-Line.1` and `CML#ED-Line.2` would be identified as conflicting ideas (see figure 8).

`Show_Alternatives( KnowledgeBase )`
this operator searches for those object nodes in the selected `KnowledgeBase` which are connected by links labelled as `Alternative` (cf. the corresponding insertion operator in section 3.1). Again, `CML#ED-Line.1` and `CML#ED-Line.2` would be identified as alternative ideas (see Figure 8). The object pairs identified by that operator indicate indeterminate specifications which obviously have to be made determinate by a subsequent negotiation phase. In particular, ideas linked by alternative vertices after the negotiation phase are required to have just one and only one idea with status IN and all other related ideas with status OUT (see below).

`Show_Islands( KnowledgeBase )`
this operator searches for incoherent specifications in the `KnowledgeBase` -- it returns isolated nodes/subgraphs, i.e. those which have no linkage to other nodes of the idea/ topic/text net; using conceptual or commentary operators establishes the required total graph connectivity.

After identifying debatable objects through these maintenance operators, negotiations start among the various conflict parties, e.g., pro agents vs. contra agents in the case of conflicting ideas. These negotiations may be conducted informally through conferencing communication channels, but may also use the formal level of commentary operators. As an alternative to a model of strongly typed argument exchange (cf. HAHN [1989]), a voting procedure may elicit an impression of the group's estimate of the issue under debate. It is based upon importance ratings given by the members of the authoring team using a special *alignment* operator:

`Score( Agent, KnowledgeBase, ObjectLabel, ScoringFactor )`
each `Agent` in the authoring group assigns an significance weight (indicated by the `ScoringFactor`) to the currently considered issue (`ObjectLabel`).

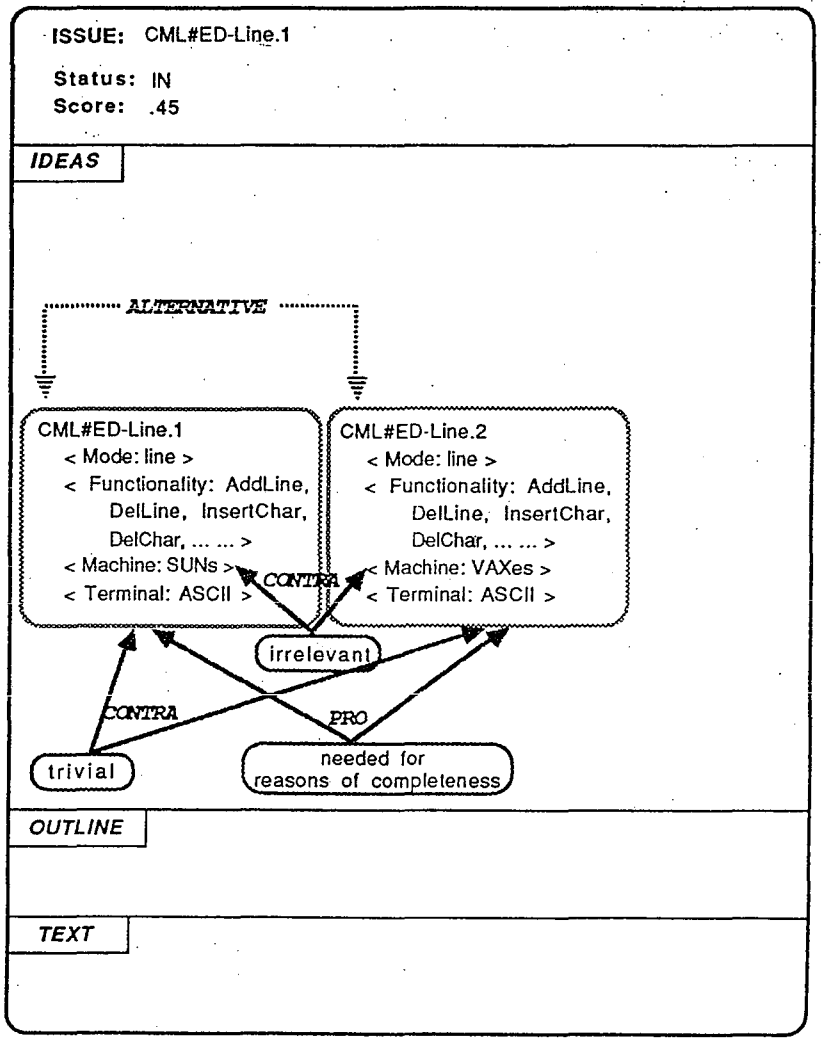

Figure 9 Scoring of an Object Node

Depending on the degree of deviance among the individual ratings the `Status` information in the issue node will be updated: If there is a strong positive/negative aggregate value without much scattering of individual opinions the `Status` of the issue will change from UNSCORED to either IN or OUT, i.e. the issue will (not) be elaborated in the document. In the example above (Figure 9) scoring on `CML#ED-Line.1` exhibits the group's rejection of the contra argument (which considers the description as being irrelevant) and lends support to the supporting argument (completeness of descriptions in the manual). The strength of the scoring factors may be considered as an additional indicator for choosing among both alternatives.

In case there is a lot of scattering among the individual ratings the issue under consideration is open for further manipulation by commentary operators (indicated by the `Status` information PENDING) and repeated scoring rounds. If, however, the controversy still persists after several negotiation cycles some appropriate termination policy must be applied. One of them is to provide a dogmatic solution, e.g., the chairman takes full responsibility of a decision that cannot be questioned any further by the members of the authoring team. Another one applies the majority rule based on the scoring procedure.

After a series of group discussions on open issues the authoring group must ensure that the combination phase can be terminated properly. This can be tested by calling up

`Show_Open_Issues( KnowledgeBase )`
an operator which returns all those ideas, topic chunks, text portions in `KnowledgeBase` whose `Status` is PENDING.

Finally, each idea/topic is either UNSCORED (confirmed without negotiations), IN (confirmed on the basis of negotiations), or OUT (canceled on the basis of negotiations or the unchallenged attachment of contra arguments to some issue object). Confirmed ideas/topics form part of the hypermedia document, while those which are OUT need no longer be considered.

To keep track with the assignment of confirmed ideas/topics with their textual realization, additional maintenance operators are available. One of them checks whether all ideas in `IdeaKB` which have been confirmed (`Status`= UNSCORED or `Status` =IN) are outline linked to objects in `HyperDocB`. Similarily, another operator tests whether contracted topics are already assigned text portions in `HyperObjB`.

Summing up, from the perspective of group work support, the development of CoAUTHOR is focused on procedures through which individual contributions are combined to form a composite group document on the levels of idea, outline, and text synthesis. Based on various maintenance operators inconsistent, incomplete, or alternative issues are identified and negotiated by the exchange of supporting and counter arguments, while definite selection decisions among competing alternatives are based on the quantitative scores of alignment operations. Thus, we emphasize in this paper the idea of MEDIATOR [JARKE/JELASSI/SHAKUN 1987] that views a group problem as a design object, rather than following a purely conversational approach.

As an alternative, the author group may also engage in qualitative modes of typed argument exchange (cf. HAHN [1989]). In this model, following the terminology of language action theory [WINOGRAD/FLORES 1986], there will then be frequent switching between open conversations for possibilities, more structured negotiations for agreement on ideas and outlines, and strictly monitored conversations for action to have these commitments realized by subcontracting agreements with individuals or subgroups. Conversations for action are also used to organize the authoring group.

It is worthwhile comparing this style of opinion elaboration and exchange by information system implementations with the SYNVIEW system [LOWE 1985]. Similar to our approach, its focus is on the representation and support of the course and results of group debates. It is not biased towards consensus enforcement (like Co-oP [BUI/JARKE 1986]), but provides a support environment for gathering, developing, evaluating different viewpoints related to a group problem. However, it lacks explicit mechanisms for formal control of group acitivies by a knowledge base management system covering document development as well as group-related activities for co-authoring.

4 Group Communication and Collaboration Tools

Our implementation concept for CoAUTHOR follows a three-way strategy. In a preliminary empirical study, several real-world applications are "rapidly prototyped" in the sense that user interfaces for supporting these applications (inventory control, design tasks, co-authoring, etc.) are built in a mock-up fashion to obtain some feedback from potential users. In a parallel effort, we build a full-scale prototype for more serious implementation from tools developed in earlier projects; this prototype is being constructed in a UNIX environment. Based on actual experience with one or more variants of this prototype, a second project phase will then construct a final, industrially oriented system on the MULTIWORKS multimedia workstation.

In this section, we outline the first prototype. This prototype will offer a client-server type architecture with a specialized client and the following three servers (figure 10):

- a knowledge base management server called *ConceptBase* [JARKE *et al.* 1988] for acquiring and maintaining the knowledge about ideas, designs, documents, and interrelationships among them (knowledge kept in the IdeaKB and HyperDocB)
- a multimedia database server called *MULTOS* [THANOS 1989] for organizing, storing, and retrieving typed multimedia chunks (managing the HyperObjB)
- a real-time conferencing facility, the *Conference Desk* [BONFIGLIO *et al.* 1989], for short-term communication management of group authoring.

Two kinds of clients of the CoAUTHOR system deserve further explanation: The *Interaction Toolbox* supports hypermedia editors and browsers for the graphical manipulation of multimedia objects and knowledge representation structures. The *Group Toolbox* contains methodologies for voting, structured argumentation, management of authoring roles, access to external programs, etc.

In the following subsections, we sketch how these components can support the functionality discussed above, and how they can be integrated with each other.

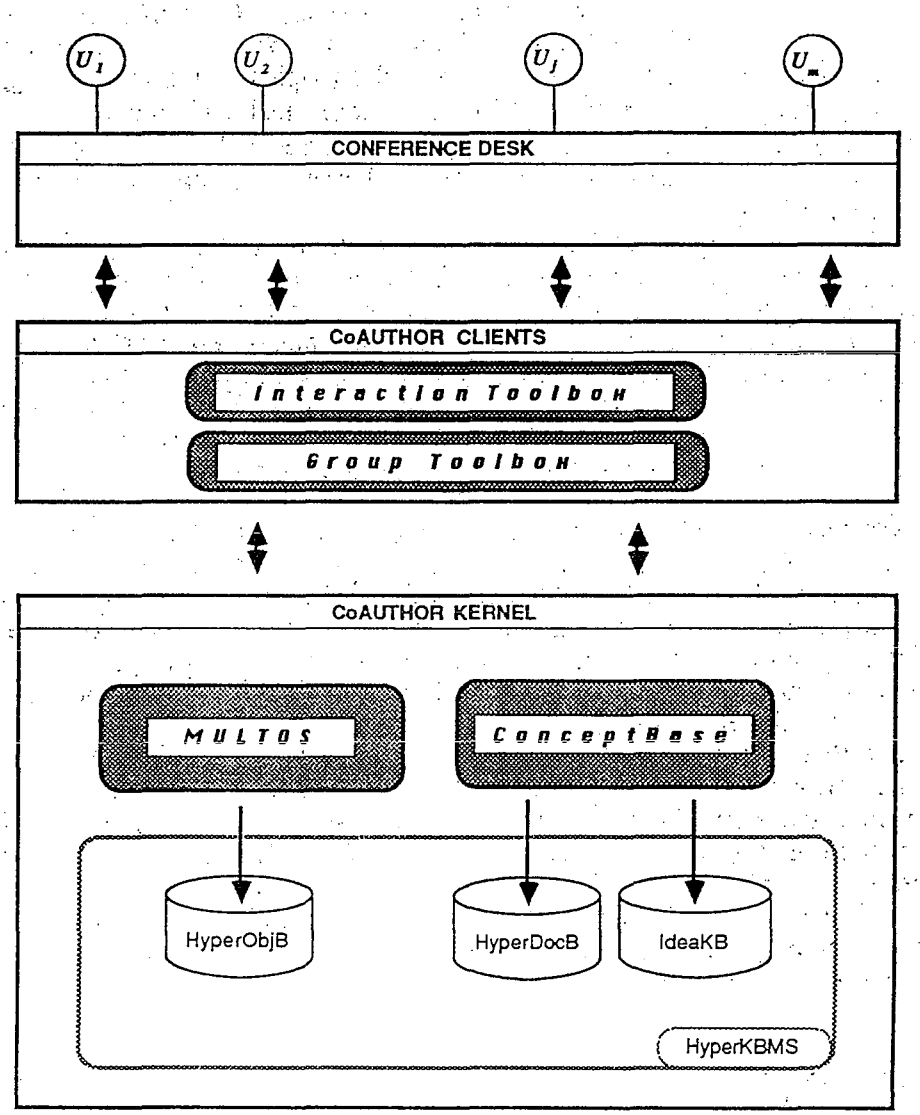

Figure 10 CoAUTHOR Software Architecture

4.1 Formalizing the Model in a Knowledge Representation System

Current efforts aiming at the specification of appropriate data models for collaborative hypertext applications [AKSCYN *et al.* 1988, SCHLICHTER/MILLER 1988]) support the formal structure of hypermedia documents only by simplification of node types or constraints on publication layout, but do neither cover the relations among the different document levels (see table 1), nor the particularities of group processes involved in collaborative authoring.

In our model, a knowledge representation scheme serves as the conceptual backbone on which the individual operators of the authoring and group collaboration models interact. Moreover, such a knowledge representation system must also enable the connection of these models to other models needed for a good collaborative environment, but not discussed in this paper; for instance, a general coordination and negotiation mechanism which addresses issues such as group formation, work organization, group decision support, etc.

These application demands place rather stringent constraints on the structure of suitable knowledge representation languages and systems. Specifically, from the models discussed in sections 2 and 3, we can derive the following requirements:

- The language must naturally accomodate a *hypermedia surface structure*, that is, offer a semantic network with typed nodes and links, with access to external references. Moreover, nodes and links must have internal structure. A semantic network schema that relates frames or similar complex objects appears suitable from this perspective.
- Links as well as nodes of the network have types (belong to classes), are subject to arguments, can be the starting point for further elaboration, can be specialized or attributed. Thus, nodes *and* links should be *first-class objects* of the language.
- Since we wish to experiment with different variants of co-authoring models, the set of node and link types must be *extensible*, using a *metaclassing* mechanism. For a formal understanding of what these new link types mean, the simple procedural method attachment offered in many object-oriented languages is not enough; additionally, a facility for declarative definition of *integrity constraints* on new classes is required.
- Searching in complex semantic networks requires recursion; *deduction rules* in the style of deductive databases are useful to declaratively represent such relationships. Moreover, they represent knowledge implicitly rather than by explicit storage of all facts, and therefore offer a more compact and maintainable representation.
- Since the knowledge base changes over time, *temporal* information and/or *versioning* information about all its objects and their interrelationships may be useful.
- Once such a knowledge base is available, the idea quickly emerges to *reuse recorded experience* for future tasks. As one consequence, the knowledge base now has to manage knowledge about more than one authoring task; an appropriate modularization facility should make available to each user only those chunks of knowledge relevant and permitted to him. Also, a group of users may wish to configure their personalized initial knowledge base of previously developed idea, structure, or document modules, as a starting point for a new project. *Version and configuration management* facilities as well as *coordination and security mechanisms* may then become important.

Initially designed for requirements modeling, the knowledge representation language CML/Telos [KOUBARAKIS *et al.* 1989] covers all but the last two requirements. CML integrates a predicative assertion language and an interval-based time calculus into a structurally object-oriented kernel that can be viewed as a tightly constrained semantic network. The ConceptBase server [JARKE *et al.* 1988] used for the CoAUTHOR prototype additionally includes a specialized metamodel of software processes which also covers the integration of external tools via triggers [JARKE *et al.* 1989]. Further metamodels for version and configuration management [ROSE/JARKE 1989] and for group communication and coordination [HAHN/JARKE 1989] are currently being added to the system kernel. A module concept which integrates these extensions with the rest is under design.

4.2 Managing Multimedia Structures in a Database

The ConceptBase client has a hypertext-like user interface where the text portions show the frame view of a piece of the knowledge base, and the links visualize a portion of the semantic network view. A nice property of the system is that the decision what is shown as a network and what is shown as frames (or not at all) can be dynamically changed, using a deductive query language.

However, ConceptBase has no facilities to deal with multimedia chunks, neither at the storage level nor at the interface level. In the CoAUTHOR prototype, the CML kernel will therefore be coupled with the server of the multimedia database system, MULTOS [THANOS 1989], and a new client will be built specifically for the co-authoring scenario.

MULTOS manages multimedia documents under a system of types which describe the layout, formal structure, and contents of documents. The MULTOS query language [BERTINO/RABITTI/GIBBS 1988] offers access by type structure and by content for text and im-

age chunks. A MULTOS client facilitates the formulation of queries and the presentation of results, using CMU's ANDREW toolkit. Moreover, there is a prototype automatic classification mechanism for new documents [EIRUND/KREPLIN 1988].

At the server level, the integration aims at making multimedia chunks available as ConceptBase nodes. This will simply be achieved by representing portions of the MULTOS type system in CML and then attaching the multimedia chunks to ConceptBase nodes as external references (similar to the triggers alluded to above). The CoAUTHOR client is then responsible for constructing the hypermedia representation on the screen; for the final system, a more integrated genuine hypermedia system will probably be preferred to avoid the expected performance problems of the prototype.

4.3 Interacting with CoAUTHOR Through a Real-Time Conferencing System

Some parts of the co-authoring process occur in an asynchronous mode of adding contributions to the various models, as sketched out in section 3. At other times, a more intense mode of interaction in real-time is needed, where people can contribute quasi-synchronously to the group model in a real-time setting. A *Conference Desk* prototype [BONFIGLIO *et al.* 1989] is currently being adapted so that it can multiplex the CoAUTHOR client to multiple participants in an on-line setting.

Each user sits in front of a workstation in which one window constitutes the shared group workspace (a view of the knowledge base state under discussion) while other read-only group windows can be used to look at supporting information which is currently not the subject of design. Under control of a pre-agreed protocol which is enforced by the Conference Desk, the users can then contribute to the objects in the group workspace. In principle, any CoAUTHOR tool could be multiplexed in this way but we expect that real-time discussion occurs only at certain negotiation-intensive phases of a co-authoring process, e.g., when arguments are exchanged, when a document structure must be agreed upon, when subtasks are contracted, or when individual contributions to a document must be integrated into a coherent structure.

5 Discussion and Outlook

Collaborative authoring has been a subject of hypermedia research from its inception. The mainstream of work, however, has been devoted to such issues as shared document files, interface design as well as group communication technology, and hypertext editing/browsing tools (cf. systems such as AUGMENT [ENGELBART 1984], CES [GREIF *et al.* 1986], or Shared Books [LEWIS/HODGES 1988]). Only limited and often ad-hoc facilities address the collaboration of multiple authors as a *social* activity, e.g., by modeling different roles authors can play in co-authoring environments such as expert, editor, etc. (as in the QUILT system [FISH *et al.* 1988]) or the support of substantive, annotative, and procedural activities and discussions for mutual intelligibility of collaborative work as exemplified in an extension of the NoteCards system [TRIGG *et al.* 1986].

These are approaches to co-authoring similar to ours in perspective. But they tend to keep the information about the hypermedia objects quite informal and thus inaccessible to all but the most simple reasoning mechanisms. In contradistinction, CoAUTHOR views all objects created during the document development and group interaction processes as parts of a carefully structured knowledge base. As a consequence, formal relationships between such objects can be analyzed and maintained in a much more flexible manner. In fact, the set of operators proposed in this paper just represents one of several possible views of such a knowledge base, and we have put some emphasis on making this set as easily extensible as possible. One reason for this adaptability is that there is little experience with what users of co-authoring systems really need, and that extensibility gives us the option of experimenting with several methods and tools. Besides technical progress with the system, the study of these requirements is a major motivating factor for our future work.

References

AKSCYN, R.M. / D.L. McCRACKEN / E.A. YODER [1988]: KMS - A Distributed Hypermedia System for Managing Knowledge in Organizations. *Communications of the ACM* Vol.31, No.7, pp.820-835.

APPLEGATE, L.M. / B.R. KONSYNSKI / J.F. NUNAMAKER [1986]: A Group Decision Support System for Idea Generation and Issue Analysis in Organization Planning. *Proc. of the Conf. on Computer-Supported Cooperative Work.* Austin, Tx, pp.16-34.

BERTINO, E. / F.RABITTI / S. GIBBS [1987]: Query Processing in a Multimedia Document System. *ACM Transactions on Office Information Systems,* Vol.6, No.1, pp.1-41.

BONFIGLIO, A. / G. MALATESTA / F. TISATO [1989]: Conference Toolkit - A Framework for Real-Time Conferencing. (*in this volume*)

BUI, X.T. / M. JARKE [1986]: Communications Design for Co-oP - A Group Decision Support System. *ACM Transactions on Office Information Systems* Vol.4. No. 2, pp.81-103.

CONKLIN, J. [1987]: Hypertext - An Introduction and Survey. *Computer* Vol.20, No.9, pp.17-41.

CONKLIN, J. / M.L. BEGEMAN [1988]: gIBIS - A Hypertext Tool for Exploratory Policy Discussion. *ACM Transactions on Office Information Systems* Vol.6, No.4, pp.303-331.

DAVIS, R. / R.G. SMITH [1983]: Negotiation as a Metaphor for Distributed Problem Solving. *Artificial Intelligence* Vol.20, No.1, pp.63-109.

EIRUND, H. / KREPLIN, K. [1988]: Knowledge-based document classification supporting integrated document handling. *Proc. Conf. Office Information Systems,* Palo Alto, Ca, pp.189-196.

ENGELBART, D.C. [1984]: Authorship Provisions in AUGMENT. *COMPCON 84: 28th IEEE Computer Society International Conf. Intellectual Leverage: The Driving Technologies.* Los Alamitos, Ca: IEEE Computer Soc. Pr., pp.465-472.

FISH, R.S. / R.E. KRAUT / M.D.P. LELAND / M. COHEN [1988]: Quilt - A Collaborative Tool for Cooperative Writing. *Conference on Office Information Systems.* Palo Alto, Ca, New York/NY: ACM, pp.30-37.

GREIF, I. / R. SELIGER / W. WEIHL [1986]: Atomic Data Abstractions in a Distributed Collaborative Editing System (Extended Abstract). *Proc. of the 13th Annual ACM Symposium on Principles of Programming Languages.* St. Petersburg Beach, New York/NY: ACM, pp.160-172.

HAHN, U. [1989]: Dialogstrukturen in Gruppendiskussionen - Ein Modell für argumentative Verhandlungen mehrerer Agenten. In: *GWAI-89 - Proc. of the 13th German Workshop on Artificial Intelligence.* Geseke, Sept. 18-22, 1989. Berlin: Springer.

HAHN, U., JARKE, M. [1989]: CoNeX: Collaboration and Negotiation Support for Expert Teams. *EC-CSCW'89: Paper Fair Collection of the 1st European Conf. on Computer Supported Co-operative Work.* Sept. 13-15, 1989, London, U.K.

HAHN, U. / U. REIMER [1988]: Automatic Generation of Hypertext Knowledge Bases. *Conference on Office Information Systems.* March 23-25, 1988, Palo Alto, Cal., New York/NY: ACM, pp.192-188.

HALASZ, G.G. [1988]: Reflections on NoteCards - Seven Issues for the Next Generation of Hypermedia Systems. *Communications of the ACM* Vol.31, No.7, pp.836-852.

HIRANO, F. [1988]: Hypermedia-Based Documentation System for the Office Environment. *RIAO 88. Proc. of the RIAO 88 Conference. User-Oriented Content-Based Text and Image Handling.* M.I.T., Cambridge, MA, March 21-24, 1988. Vol.1. [Paris:] Centre de Hautes Etudes Internationales d'Informatique Documentaire (C.I.D.), pp.535-546.

JARKE, M. / M.T.JELASSI / M.F.SHAKUN [1987]: MEDIATOR - Towards a Negotiation Support System. *European Journal of Operations Research* Vol.31, No.3, pp.314-334

JARKE, M. / JEUSFELD / M., ROSE, T. [1988]: *A Global Knowledge Base Management System for Database Software Evolution: Documentation of First ConceptBase Prototype.* Report MIP-8819, Universität Passau.

JARKE, M. / JEUSFELD / M., ROSE, T. [1989]: A Software Process Data Model for Knowledge Engineering in Information Systems. *Information Systems Vol.* 14, No. 3.

KOUBARAKIS , M. / MYLOPOULOS, J. / STANLEY, M. / BORGIDA, A. [1989]. *Telos: Features and Formalization.* Technical Report KRR-89-04, Dept. Computer Science, University of Toronto, Ont.

LEWIS, B.T. / J.D. HODGES [1988]: Shared Books - Collaborative Publication Management for an Office Information System. *Conference on Office Information Systems*. March 23-25, 1988, Palo Alto, Cal., New York/NY: ACM, pp.197-204.

LOWE, D. [1985]: Cooperative Structuring of Information: The Representation of Reasoning and Debate. *International Journal of Man-Machine Studies* Vol.23, pp.97-111.

ROSE, T. / JARKE, M. [1989]: *A Decision-Based Configuration Process Model.* Working Paper, Universität Passau (submitted for publication).

SCHLICHTER, J.H. / L.J. MILLER [1988]: FolioPub - A Publication Management System *Computer* Vol.21, No.1, pp.61-69.

SMITH, J.B. / S.F. WEISS / G.J. FERGUSON / J.D. BOLTER / M. LANSMAN / D.V. BEARD [1986]: *WE - A Writing Environment for Professionals.* Chapel Hill/NC: Univ. of North Carolina at Chapel Hill, Dept. of Computer Science (TR86-025).

STEFIK, M. et al. [1987]: Beyond the Chalkboard - Computer Support for Collaboration and Problem Solving in Meetings. *Communications of the ACM* Vol.30, No.1, pp.32-47.

THANOS, C. (ed) [1989]: *Multimedia Document Filing: The MULTOS Approach.* Amsterdam: North-Holland.

TRIGG, R.H. [1983]: *A Network-Based Approach to Text Handling for the Online Scientific Community.* College Park/MD: Dept. of Computer Science, Univ. of Maryland (Ph.D. Thesis).

TRIGG, R.H. [1988]: Guided Tours and Tabletops - Tools for Communicating in a Hypertext Enviornment. *ACM Transactions on Office Information Systems* Vol.6, No.4, pp.398-414.

TRIGG, R. / L.SUCHMAN / F. HALASZ [1986]: Supporting Collaboration in NoteCards. *CSCW 86 - Proc. Conf. on Computer-Supported Cooperative Work.* December 3-5, 1986, Austin, Texas, pp.153-162.

WINOGRAD, T., FLORES, F. (1986). *Understanding Computers and Cognition: A New Foundation for Design.* Norwood, NJ: Ablex.

YANKELOVICH, N. / N. MEYROWITZ / A. VAN DAM [1985]: Reading and Writing the Electronic Book. *Computer* Vol.18, No.10, pp.15-30.