Aquanet: a hypertext tool to hold your knowledge in place

Catherine C. Marshall
Frank G. Halasz
Russell A. Rogers
William C. Janssen Jr.

Xerox Palo Alto Research Center
3333 Coyote Hill Rd.
Palo Alto, CA 94304
415-494-4740
marshall.parc@xerox.com
Fax: 415-494-4777

Keywords: representation, graphical knowledge structures, complex relations, collaboration

Abstract:

Hypertext systems have traditionally focused on information management and presentation. In contrast, the Aquanet hypertext system described in this paper is designed to support knowledge structuring tasks. Aquanet is a browser-based tool that allows users to graphically represent information in order to explore its structure. In this paper, we discuss our motivations for developing Aquanet. We then describe the basic concepts underlying the tool and give an overview of the user interface. We close with some brief comments about our initial experiences with the tool in use and some of the directions we see the Aquanet research moving in the near future.

from Hypertext '91 Proceedings, San Antonio, Texas, December 15-18, 1991

1. Introduction

The power of hypertext derives from its dual nature: hypertext is simultaneously a tool for managing and presenting information and a tool for representing the underlying structure of that information. Although this essential duality has been duly noted in the hypertext literature, by far the largest part of the effort in building, using, and researching hypertext remains on the information management and presentation side of the duality. From Memex [Bush45] and Augment [Engl68] through KMS [Aksc88] and Intermedia [Garr85], the main focus has been on hypertext as vehicle for managing large collections of non-linear information and for presenting this information to readers in a manner that does justice to its non-linear nature. The other side of this duality, information (or, if you prefer, knowledge) representation has until very recently been considered more the province of Artificial Intelligence than of Hypertext. With the exception of a few scattered projects such as gIBIS [Conk88] and IDE [Jord89], relatively little attention has been focused on exploring and further developing the underlying representational capabilities of the hypertext data model.

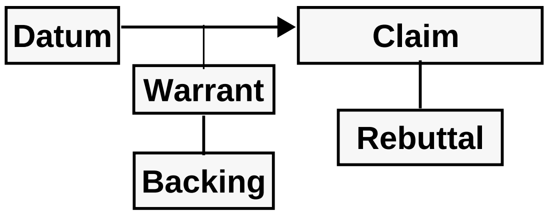
Aquanet is a hypertext tool for people trying to interpret information and organize their ideas, either individually or in groups; we have been calling such activities *knowledge structuring tasks*. Some knowledge structuring tasks - for example, information analysis - involve movement from unstructured fragments to coherent, organized structures [Brow85][Hala87]. Other knowledge structuring tasks such as design deliberations may use a known structure and the methodology it embodies to facilitate group discussions; Yakemovic's use of IBIS for software design is an excellent example [Yake90].

Our goal in developing Aquanet is to explore the utility of hypertext facilities in the realm of knowledge representation and, as a result, to broaden our understanding of hypertext's representational characteristics. In this paper, we will describe our motivations in developing Aquanet. We will then describe the basic concepts underlying the tool and give an overview of the user interface. We will close with some brief comments about our initial experiences with the tool in use and some of the directions we see the Aquanet research moving in the near future.

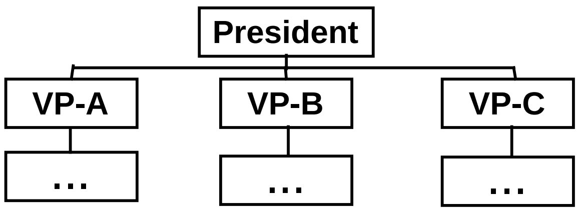
Throughout this paper, we will be using the terms "knowledge structure" and "graphical knowledge structure." We use *knowledge structure* to refer to an interconnected network of information-bearing nodes that are used to represent the primitive objects and their interrelationships in some domain of discourse. Technically, any hypertext network can be considered a knowledge structure under this definition. But we intend the term to connote hypertext networks (and other information structures) whose basic purpose is to represent or model the structure of some real-world domain. By *graphical knowledge structure*, we mean a knowledge structure whose primary presentation to the user is as a graphic display on a computer screen. Figure 1 shows three representative examples of the kind of graphical knowledge structures that drove our thinking in developing Aquanet.

2. Background: NoteCards meets gIBIS

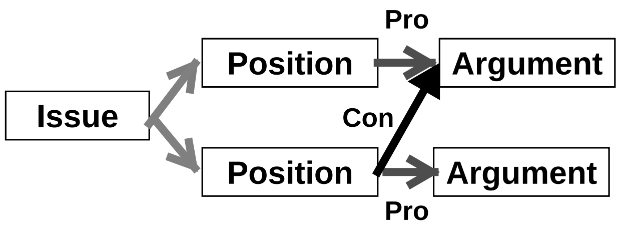
Aquanet brings together two separate but convergent lines of hypertext research. First, over the last five years we have been observing people using NoteCards [Hala87] for a variety of knowledge structuring tasks. Of particular interest are our investigations into representing the structure of argumentation in hypertext [Mars87][Mars89] and our use of the NoteCards-based IDE system for analyzing knowledge in instructional design tasks [Russ87][Jord89]. Second, we have been



Abstract Toulmin micro-argument structure [Toul58]



Abstract organizational chart



Abstract IBIS network after [Conk88]

Figure 1: Representative knowledge structures

influenced by the ideas introduced in gIBIS [Conk88] and its subsequent generalization, Germ [Brun88]. The gIBIS and Germ tools provide a valuable counterpoint to NoteCards since they were explicitly designed to support the construction and maintenance of constrained knowledge structures.

These two lines of research have highlighted the discrepancies between the needs of users engaged in knowledge structuring tasks and the functionality provided by information management and presentation hypertext systems like Intermedia and KMS. Two discrepancies stand out. First, information management hypertext systems focus on nodes and the local connections between them. But in knowledge structuring tasks it is important to see and manipulate a global view of the network. Second, information management systems provide only a simple node-link data model. Knowledge structuring tasks often require a richer language for expressing the interconnections among nodes.

2.1 Centering interaction on a network overview

One of the most important consequences of emphasizing the information management and presentation aspect of hypertext is that interaction tends to be centered on nodes or documents; authors and readers are expected to focus on the textual or graphic content of nodes, and to move from node to node in navigating through a network. In fact, systems like NLS/Augment, KMS, and Hypercard [Appl87] provide no structural overview of the network. By contrast, gIBIS interactions center around a graphic overview of the emerging hypertext network; new nodes and links are always created in this global context.

NoteCards allows us to examine the distinction between node-based and overview-based access to the hypertext since it provides users with both modes of interaction. Users generally regarded access using the graph-style overview (referred to as the browser) as clumsy and slow, and did not use it if they were engaged primarily in information management tasks. On the other hand, users faced with large structuring tasks frequently chose to work from the browser, in spite of its drawbacks. They would maintain browsers across sessions, and use them as a context-setting backdrop for their work. These browsers functioned as accelerators for accessing and referring to existing structures (see page 839 in [Hala88]). NoteCards users also developed strategies for using browsers to group nodes by means of spatial layout, especially in the very earliest brainstorming stages of a task (see page 96 in [Trig87]).

Because of the effectiveness of the overview in gIBIS and our experiences with the NoteCards browser, we decided to center Aquanet interactions around a graphical view of the knowledge structure.

2.2 Complex relations as a richer linking model

A second important artifact of emphasizing the information management and presentation aspects of hypertext is that links are designed primarily as navigational aids. Even though in systems like NoteCards and gIBIS, links are labelled or even typed, it is difficult to build coherent composite structures using links alone. The legal case cluster card described on page 843 of [Hala88] is a representative example. In this example, a NoteCards user attempted to represent the structure of a complex legal case, but failed because he could not adequately represent its overall structure using a simple collection of nodes and links.

We experienced analogous problems of reference and scope in representing Toulmin structures in NoteCards. As a result, we found it necessary to create extra mechanisms (specifically, a Toulmin card) to capture the semantics of the Toulmin relation [Mars87]. For similar reasons, the IDE developers built a structure library [Jord89] to facilitate their representation development. Even though in both of these cases the extra mechanism aided the task at hand, there were still significant limitations to its generality. For example, after structures were created through these mechanisms they were still maintained within the node-link model. Problems like the inability to ripple structural changes through all instances were endemic to both applications. Thus developing a richer linking model to express complex relations is an important goal of Aquanet.

3. The Aquanet Task: Collaborative knowledge structuring

Looking more closely at knowledge structuring tasks, we can distinguish between two kinds of activities. First, people *use* knowledge structures to organize and categorize content. Second, over the course of a task people *develop* knowledge structuring schemes; these schemes evolve through negotiation and use. In practice, the two activities aren’t that easy to pull apart, but they each suggest a different set of use situations and user requirements.

Our own experiences using Toulmin structures as a descriptive way of organizing the content of reasoned discourse [Newm91] provide some interesting insight into how knowledge structures evolve as they are used collaboratively. Originally, we chose a fixed knowledge structuring scheme, Toulmin’s model of argument [Toul58] (see Figure 1 on page 2), as the primary vehicle for performing a series of analyses. In the course of the analyses, we discovered that not only was

Toulmin's model insufficiently prescriptive, but it also failed to cover and highlight all of the phenomena of interest to us. To resolve the aspects of the model that were insufficiently prescriptive, we found ourselves negotiating what kinds of statements could be used as each of Toulmin's micro-argument elements (for example, we had to decide whether a datum could be a very general statement, or whether very general statements were always warrants) and how Toulmin structures could be hooked together (for example, we had to decide whether it was possible to use the backing of one argument as the claim in another). To cover the parts of the argument that were difficult to "Toulminize," we had to create new kinds of structures. Thus we found it necessary to both *constrain* and *extend* our original knowledge structuring scheme.

As the example above illustrates, knowledge structuring tasks are frequently collaborative; they can involve more than one person, and they usually take place over an extended period of time. The kind of knowledge structuring that we've observed is semi-synchronous rather than the tightly-coordinated synchronous collaboration that takes place in meetings. Much of the interaction among collaborators takes place through actions on the knowledge structures as well as through meta-comments attached to the knowledge structure.

The design and development of Aquanet was driven by these and other experiences using NoteCards for knowledge structuring tasks. Out of these experiences, we have derived the following requirements:

(1) To define a knowledge structuring scheme, a user must be able to specify what its elements are and how they are interconnected. For example, in an IBIS model, there are Issues, Positions, and Arguments, and the Arguments can support Positions, but they cannot respond to Issues.

(2) To develop a knowledge structure, a user must be able to modify and extend the knowledge structuring scheme as her understanding of the task changes. As we have suggested by our example, the evolution of structuring schemes is an important side-effect of using them.

(3) To build or use a knowledge structure in the collaborative settings described above, users must be able to see - within a reasonable interval - what other users have done. We call the kind of updating required for this style of interaction WYSIWID: What You See Is What I Did.

(4) To display multiple views onto a single knowledge structure, a user must be able to specify alternate graphic renderings of the same structure. For example, a user might want to see arguments about design options and criteria as a matrix at the same time as the information is shown as a dependency tree (see MacLean et al.'s QOC [MacL91] or Lee's SIBYL [Lee90]).

(5) To use combinations of methodologies in a single task (as in Streitz's activity spaces [Stre89]), a user must be able to compose knowledge structuring schemes. For example, a task that requires both issue-structuring and argumentation might combine an IBIS Issue-Position-Argument model and a Toulmin Data-Claim-Warrant argument model.

(6) To develop a knowledge structure collaboratively, users must be able to negotiate about its contents; they must be able to talk *about* the knowledge structure as well as *through* the knowledge structure (see also Conklin and Begeman's account of "going meta" in [Conk88]).

In designing Aquanet, we tried to address all six of these requirements. While the current implementation of the tool falls short of fully meeting all of them, it is our long term research goal to fully support the requirements of knowledge structuring tasks.

4. Knowledge Structuring Concepts in Aquanet

Aquanet is designed to support users in creating, storing, editing, and browsing large graphical knowledge structures.

4.1 The Aquanet data model: basic objects and relations

Aquanet knowledge structures conform to a data model that merges hypertext constructs (e.g., see [Hala90]) with frame-based representations (e.g., see [Bobr77]). In line with its hypertext roots, Aquanet makes a strong distinction between data-containing objects, typically called "nodes" in the hypertext model, and relational objects, typically called "links" in the hypertext model. The 'nodes' in Aquanet are called *basic objects*, the 'links' are called *relations*. In line with its knowledge representation roots, Aquanet objects (both basic objects and relations) are typed, structured, frame-like entities.

Every Aquanet object is made up of an unordered set of named, typed slots. The distinction between basic objects and relations lies in the nature of the allowable slot values. In basic objects, all slot values are restricted to be primitive datatypes (e.g., text, images, numbers, strings, dates, etc.). Thus basic objects are analogous to standard hypertext nodes (e.g., cards in NoteCards, frames in KMS, documents in Intermedia, etc.) except that instead of having a single content they have a set of named contents.

In contrast, the slots in relations may have either primitive datatypes or other Aquanet objects as values. Thus relations are analogous to hypertext links in that they are structural elements that serve to connect other entities in the structure. In hypertext terms, relations are "node-to-node" n-ary links that can be anchored to either nodes (basic objects) or other links (relations). Unlike typical links, however, relations have what amounts to named and typed endpoints.

Every Aquanet object is an instance of some type. A type's definition specifies its slots, the type(s) of objects that can fill each slot, and the graphical appearance of the object (see Section 4.2 below). Aquanet type definitions are organized into a multiple inheritance hierarchy. Objects of a given type include not only the slots defined in their type but also the slots that they inherit from their supertype(s). The inheritance rules in the Aquanet type hierarchy are taken directly from the CommonLisp Object System specification [Stee90].

4.2 Graphic appearances

Both the hypertext and frame data models focus on structure *per se* and largely ignore the issue of how that structure is to be graphically displayed to the user. In contrast, the appearance of the knowledge structure is a critical component of the Aquanet data model. Specifically, every type definition includes information about the type's *graphic appearance*.

An Aquanet object's graphic appearance specifies what the object and its slots should look like on the display. When rendering a knowledge structure, Aquanet reserves a rectangular region of the display for each object. The object's graphic appearance determines what is drawn into this rectangular region. Figure 2A (top) shows an example of the graphic appearance of a representative basic object (a Statement). Figure 2B shows a representative relation (an Argument). Statements have one important slot called Text. Arguments have three slots - the Conclusion, the Grounds, and the Rationale - each of which can be filled by a either a Statement or another Argument.

Graphical appearances can contain two types of items: *graphic elements* and *slot values*. The

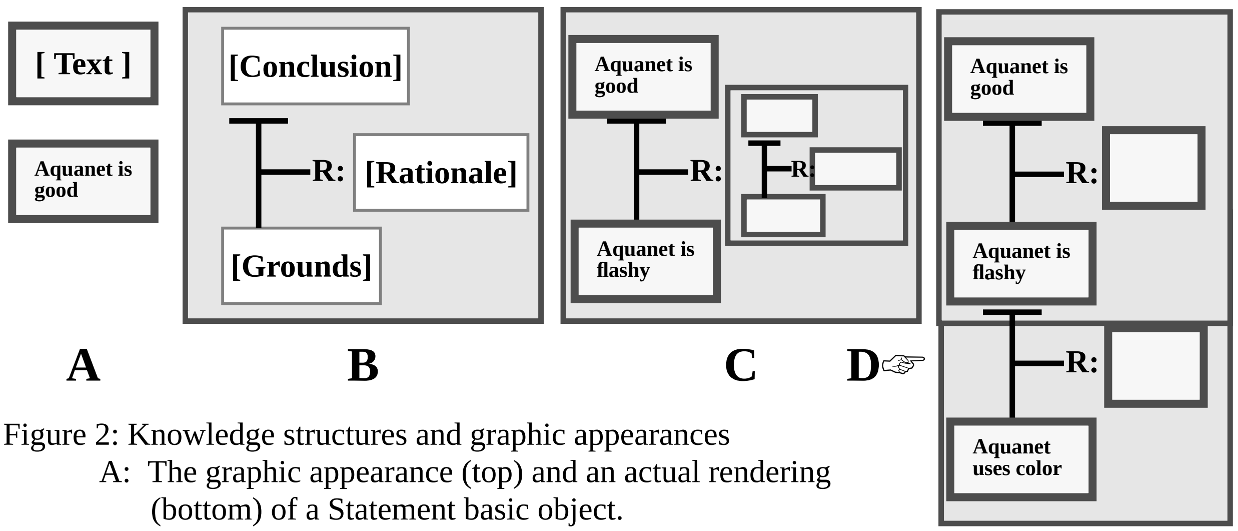


Figure 2: Knowledge structures and graphic appearances
A: The graphic appearance (top) and an actual rendering (bottom) of a Statement basic object.
B: The graphic appearance of an Argument relation.
C: An actual rendering of an Argument relation where two slots are filled by Statements and the third is filled by another Argument relation. This is an example of inclusion composition.
D: A rendering of two Argument relations that share a Statement object (“Aquanet is flashy”). This is an example of chaining composition.

graphic elements are items such as lines, circles, squares, text labels, background colors, etc. that are drawn onto the display (scaled to fit into the allotted region). In Figure 2B, the vertical and horizontal black lines and the characters “R:” are examples of graphic elements.

Slot value items reserve an area into which the value of a named slot will be rendered. For primitive valued slots, the value of the named slot is printed in this area (see the bottom illustration in Figure 2A). For entity-valued slots, the graphic appearance of the slot’s value is recursively rendered (after the necessary scaling) into the reserved area. In Figure 2B, the three dashed boxes containing the bracketed slot names are slot value items. Figure 2C shows the rendering of an instantiated Argument relation. In this instantiation, two of the Argument’s slots are filled with Statement basic objects and one is filled with another Argument relation.

The graphical appearance mechanism just described is most appropriate for relations in which the layout of the relation follows a specifiable convention. Many relations have no such convention. For these cases, Aquanet allows the user to arbitrarily position objects on the display. The region occupied by the relation that contains these objects is then adjusted to be the bounding box just big enough to hold all of the contained objects. Figure 3 illustrates the operation of these adjustable relations. Such relations are used to construct standard network diagrams such as the browsers found in NoteCards and gIBIS.

Composing a 2-dimensional graphical appearance for a complex knowledge structure out of the graphical appearances of its component objects raises many challenging design issues. To avoid some of the very difficult layout mechanisms that would be required for a general solution, we chose to employ two simplifying mechanisms in our first implementation. First, Aquanet structures are arranged in a $2^1/_2$ dimensional space which is then rendered onto the 2 dimensional space of the display screen. Where the graphical appearances of objects overlap, one of the objects is

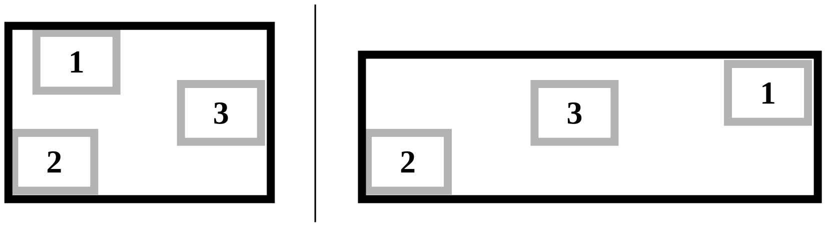


Figure 3: The rendering of an adjustable relation (black border) containing three basic objects (gray borders), both before (left side) and after (right side) the user moves the “1” basic object.

stacked on top of the others, thus partially or fully obscuring them. The user can manipulate the stacking order of Aquanet objects to ensure that the desired object is displayed unobscured.

The second simplifying mechanism involves the creation of multiple views of a single Aquanet object (called *virtual copies* in the Aquanet interface) that can be placed at disparate locations on the display. Using virtual copies, the layout of a complex structure is simplified because the structure can be split into pieces that can be arbitrarily placed on the display without the constraint that logically connected relations (i.e., relations that share a common included object) be spatially co-located on the display.

4.3 Building knowledge structures

As suggested in Figures 2C and 2D, there are two methods for constructing complex knowledge structures in Aquanet: *inclusion* and *chaining*. In inclusion, an entire relation is included as the value of a slot in some other relation. In Figure 2C, the Rationale slot of an Argument relation has been filled by another Argument relation. In chaining, two or more relations are connected because an Aquanet object fills a slot - not necessarily the same slot - in each of the chained relations. In Figure 2D, the two Argument relations are chained because the Statement object “Aquanet is flashy” fills the Grounds slot in the upper Argument and the Conclusion slot in the lower.

4.4 Aquanet schemas

One of our major goals in designing Aquanet was to provide users with the ability to customize knowledge structures for their specific task. Aquanet accomplishes this goal through the use of *schemas*. Every Aquanet session is controlled by a schema that defines a set of allowable basic object and relation types. Since the basic object and relation types specify their slots and constrain the slots’ values, the schema defines the nature and organization of the knowledge structure that the user can construct. For example, one can easily design a Toulmin schema that allows the user to create only one type of relation, the Toulmin relation, whose five slots (Datum, Claim, Warrant, Backing and Rebuttal) can only be filled by a Statement basic object. On the other hand, one could as easily design an IBIS schema that allows the user to create 3 types of basic objects (Issues, Positions, and Arguments) and connect them using the 9 types of relations (with appropriate type restrictions) that are described in [Conk88].

Aquanet includes a mechanism for composing schemas. Specifically, a schema can include by reference any other schema. The types in the included schema are added to the list of types in the including schema. The design of a more sophisticated composition mechanism that would, for example, provide for subtyping of schemas is a topic we are currently exploring.

The Aquanet schema language is somewhat limited in expressiveness. In particular, the schema determines the knowledge structure only on a local level. There is no way in Aquanet to express multi-object or global restrictions on the organization of the knowledge structure. For example, one cannot ensure that only one instance of a certain type exists in a knowledge structure, nor can one state that some object should not be related to itself through a series of connecting relations. A richer schema language is another area we are actively investigating.

Aquanet includes both a type editor and a schema editor. With the type editor, the user can easily access all of the properties of a type including its list of supertypes, its graphic appearance, its slots, and the slot value restrictions. With the schema editor, the user can add and remove types in a schema.

Allowing the user to edit types and schemas brings up a host of difficult issues about how to reconcile existing objects with schema changes. Although these issues of schema evolution are a critical focus for our work in the long term, we chose to simplify our initial implementation by restricting the possible changes that users can make when editing types and schemas. In particular, users can add and delete types to/from a schema and they can add/delete slots in a type. They can also arbitrarily change the graphical appearance of a type. Few other type or schema modifications are currently possible. For example, changing the type restriction on a slot value is not generally allowed. Since they are clearly at odds with our goal of supporting schema evolution during knowledge structuring tasks, such restrictions on schema editing will be removed in future releases of Aquanet.

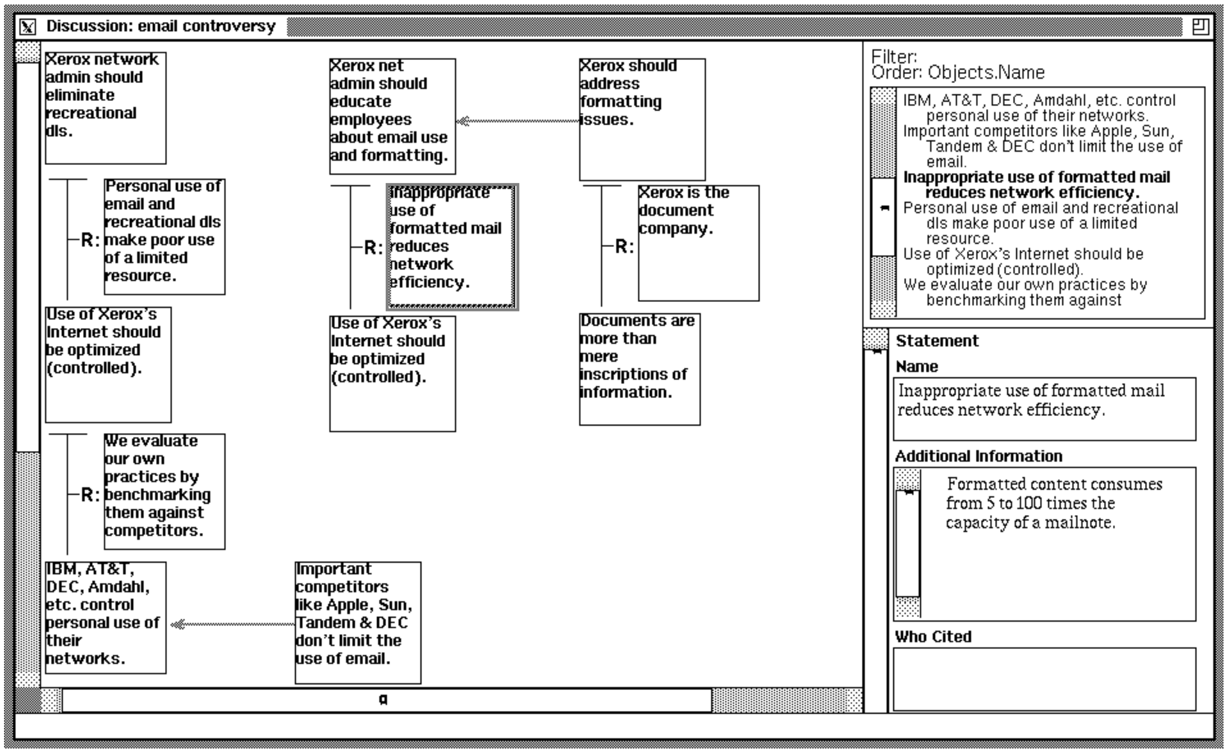
4.5 Aquanet discussions

Aquanet stores individual knowledge structures as well as schema information in a central, shared database. Individual knowledge structures are known as *discussion*s, following the terminology used in gIBIS. Each discussion uses a single schema (but many discussions may share a single schema). At any given time, each instance of the Aquanet tool can only have a single open discussion. Objects contained in one discussion cannot “reference” (i.e., use as a slot value) objects in another discussion. Discussions are also the level at which most access control is enforced.

5. Using Aquanet

Users create and browse the graphical knowledge structures in an Aquanet discussion through the multi-paned window shown in Figure 4. In this example, the discussion concerns a recent controversy over the appropriate use of electronic mail; the Argument schema described in Figure 2 provides the knowledge structure used in the discussion. The pane on the left side of the window is a scrollable view onto the full structure; this view can be scaled so the desired amount of structure is visible in the pane. The top right pane displays a filtered list of objects in the network. This pane will be used to contain other kinds of user-specifiable views in future implementations. Objects can be selected using either of these two panes. The lower right pane displays the content - the primitive slots and their values - of the selected object. In this case, the Statement basic object “Inappropriate use of formatted mail reduces network efficiency” has been selected, and its Name (a string-valued slot), Additional Information (a text-valued slot), and Who Cited (another string-valued slot) are displayed in the content pane. Objects displayed in this pane can be checked out from the database for editing.

Users can extend a structure in two different ways: they can use an existing object in a new role,



Discussion: email controversy

Xerox network admin should eliminate recreational dls.

R: Personal use of email and recreational dls make poor use of a limited resource.

Use of Xerox's Internet should be optimized (controlled).

R: We evaluate our own practices by benchmarking them against competitors.

IBM, AT&T, DEC, Amdahl, etc. control personal use of their networks.

Important competitors like Apple, Sun, Tandem & DEC don't limit the use of email.

Xerox net admin should educate employees about email use and formatting.

R: Inappropriate use of formatted mail reduces network efficiency.

Use of Xerox's Internet should be optimized (controlled).

Xerox should address formatting issues.

R: Xerox is the document company.

Documents are more than mere inscriptions of information.

Filter:
Order: Objects.Name

IBM, AT&T, DEC, Amdahl, etc. control personal use of their networks.
Important competitors like Apple, Sun, Tandem & DEC don't limit the use of email.
Inappropriate use of formatted mail reduces network efficiency.
Personal use of email and recreational dls make poor use of a limited resource.
Use of Xerox's Internet should be optimized (controlled).
We evaluate our own practices by benchmarking them against

Statement
Name
Inappropriate use of formatted mail reduces network efficiency.
Additional Information
Formatted content consumes from 5 to 100 times the capacity of a mailnote.
Who Cited

Figure 4: An Aquanet window.

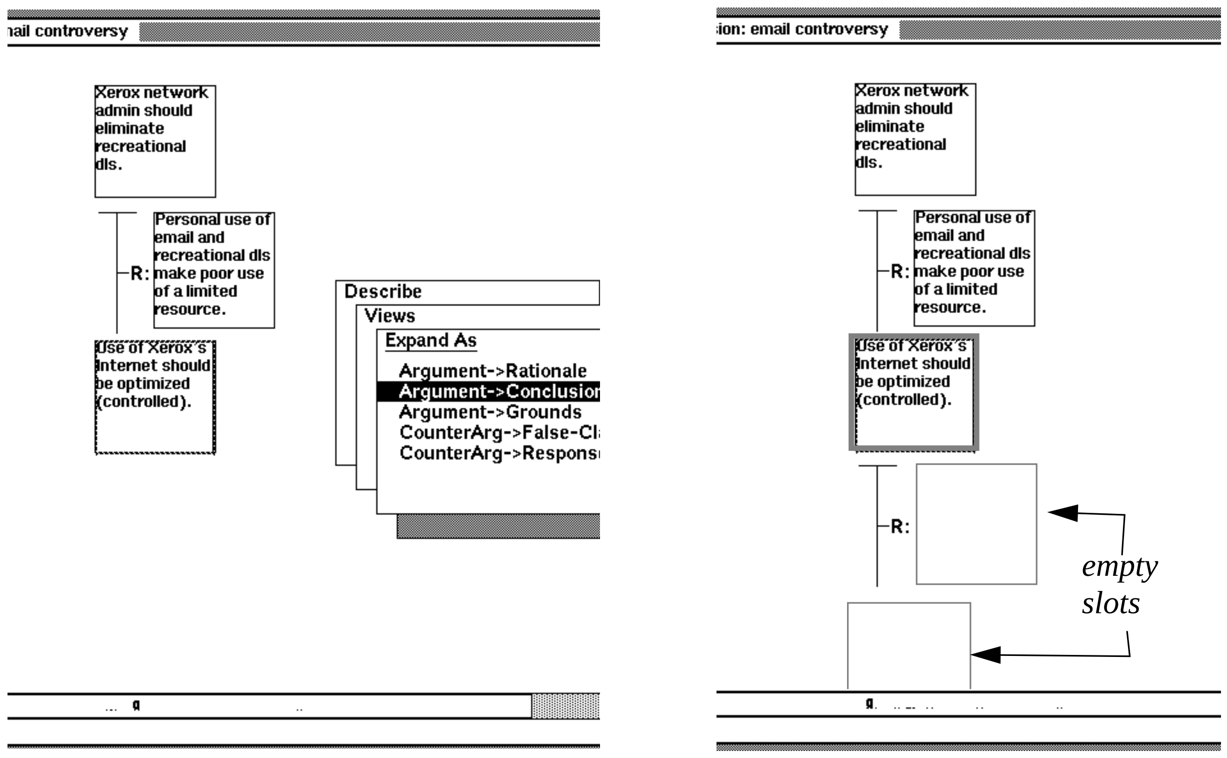
thus creating a new relation, or they can create a new relation and fill its slots with new or existing objects. Figure 5 shows an example of how a structure grows through chaining. In Figure 5a, the selected Statement “Use of Xerox’s Internet should be optimized,” which is already Grounds for one argument, is being used as the basis for creating a new Argument relation. In the new relation, the Statement will fill the Conclusion slot; the other two slots will be empty. Figure 5b shows the results of this operation.

Users develop and modify schemas with a schema editor as shown in Figure 6. The types that are included in the schema may be selected from the list on the left side of the window. The schema “Simple Arg” shown in the figure includes four types, a Statement basic object, an Argument relation, a CounterArg relation, and a Note basic object. The Argument relation has been selected for editing.

Users edit types with a separate type editor. The type editor allows users to name a type, list its supertypes, define its slots, and specify its graphic appearance. Figure 7 shows the type editor invoked on the Argument relation. It has no supertypes beyond the system types of Relation and Basic Object. It contains three entity-valued slots, Grounds, Conclusion, and Rationale. Its graphic appearance is shown in the editing pane on the right hand side of the window. In Figure 7, one of the lines has been selected; a user is changing its color to red and its width to 2.

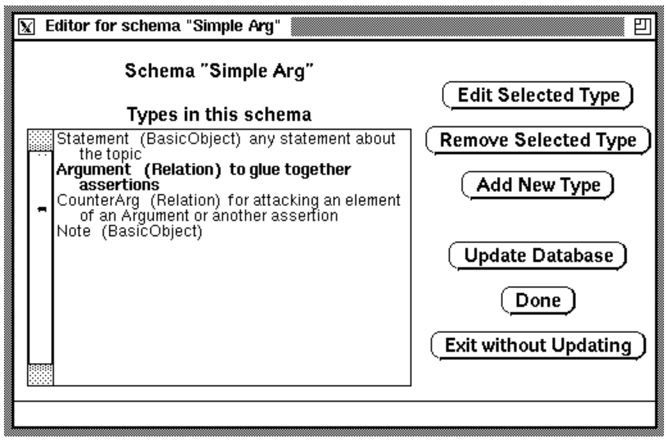
6. Implementation Notes

Aquanet runs on Unix workstations and can be used from any workstation or personal computer that supports X Windows. A color monitor is useful but not required. Aquanet is written in CLASS, a single inheritance object extension to “C”, and built on the Andrew Toolkit [Pala88]. A relational database server is used to store the knowledge structures and the user-defined schemas.


ail controversy
Xerox network admin should eliminate recreational dls.
R:
Personal use of email and recreational dls make poor use of a limited resource.
Use of Xerox's Internet should be optimized (controlled).
Describe
Views
Expand As
Argument->Rationale
Argument->Conclusion
Argument->Grounds
CounterArg->False-Cl
CounterArg->Respons
ion: email controversy
empty slots

a. Using a Grounds statement as a Conclusion b. The extended network

Figure 5: Using an object in a new role

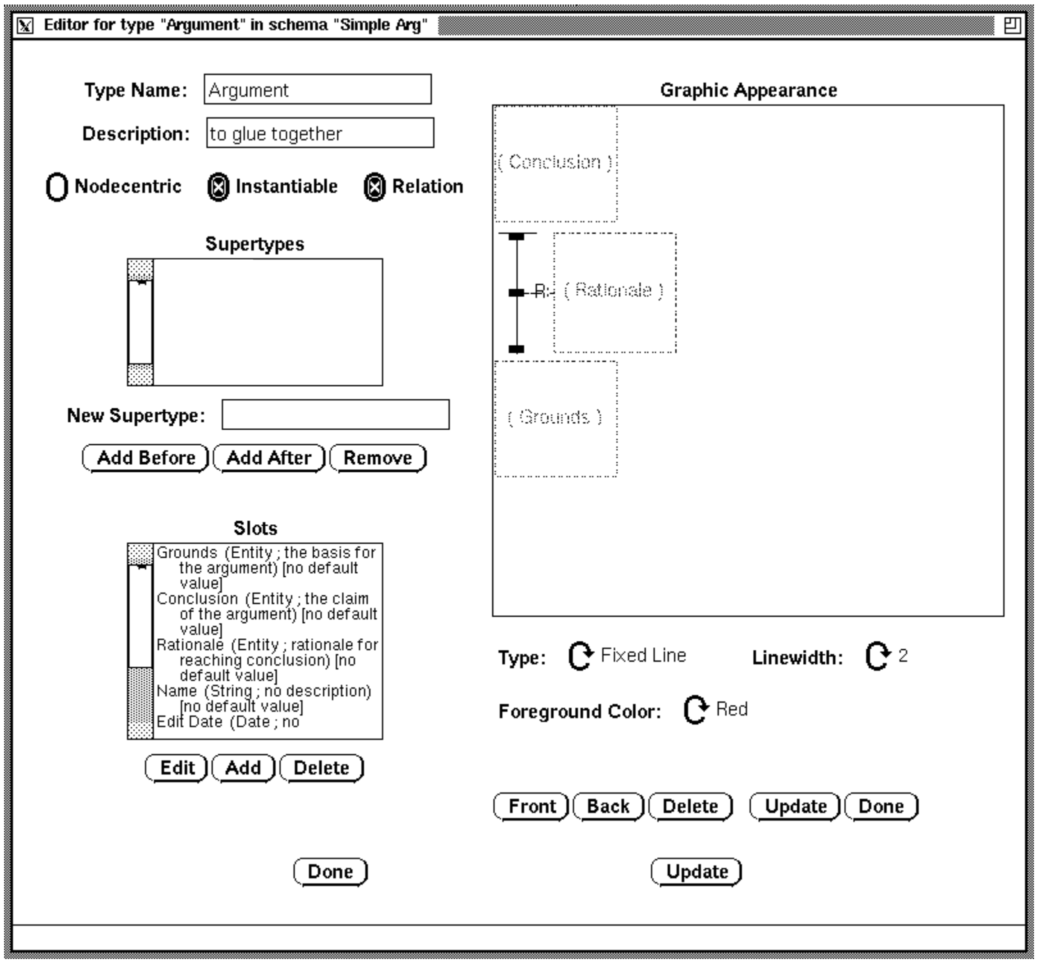


Statement (BasicObject) any statement about the topic
Argument (Relation) to glue together assertions
CounterArg (Relation) for attacking an element of an Argument or another assertion
Note (BasicObject)

Figure 6: The schema editor. The Argument relation has been selected for editing.

Aquanet's object-oriented model is mapped onto the relational model. Each type is stored as a separate database table. Another set of tables stores the schema information.

To support semi-synchronous collaboration, the database server mediates access and changes to each discussion. Before an object is changed, it is locked in the database and its latest state is recached locally. When the object is unlocked, the edit is noted in a change log. Each session periodically polls this log to see which elements have been changed. Any new or changed elements


Editor for type "Argument" in schema "Simple Arg"

Type Name: Argument

Description: to glue together

Nodecentric [✓ Instantiable] [✓ Relation]

Supertypes

New Supertype:

Slots

Grounds (Entity ; the basis for the argument) [no default value]
Conclusion (Entity ; the claim of the argument) [no default value]
Rationale (Entity ; rationale for reaching conclusion) [no default value]
Name (String ; no description) [no default value]
Edit Date (Date ; no

Graphic Appearance

Type: Fixed Line

Linewidth: 2

Foreground Color: Red

Figure 7: Editing a type

are recached, resulting in an up-to-date view of the discussion. The default polling interval is set, somewhat arbitrarily, at 30 seconds.

The implementation uses a data / view architecture; the discussion is stored in a data object and multiple views are created to display the knowledge structure. A user, by interacting with one of the views, edits the knowledge structure which broadcasts the change to all of its views. Each view responds, updating its appearance to match the new state of the discussion. This facilitates viewing and interacting with different representations of the discussion while preserving consistency across these views.

7. Early Experiences with Aquanet

The aerosol version of Aquanet was released for use within our laboratory during May, 1991. Our experiences are drawn both from our own collaborations and from the experiences of some early users. Already we have accumulated a diverse collection of schemas that reflect the varying representational needs of the initial set of tasks. We have also noticed some important patterns in these early experiences and have noted some additional requirements central to using the tool for its intended purposes.

To date, Aquanet discussions have been initiated in service of analytic tasks (one is a competitive analysis of current machine translation efforts), group design efforts (for example, designing a new drawing editor), organizing an existing collection of structured information (usage notes for a dictionary), and more general collaborative work (we use it to keep track of Aquanet bugs and feature requests and we've used it to organize this paper). While many of these discussions are still in early phases of organization, they have already caused us to reflect on some general issues. First, we have noted that people invent ways of creating lightweight structure to lessen the problems of premature organization. Second, we have looked at the kinds of schemas people have come up with, how they differ, and how they share certain characteristics. Finally, we have recognized the difficulties with bringing information into the tool and defining output when structuring must result in a linear document.

7.1 Using spatial organization

As noted in [Stef87], [Trig87], and elsewhere, spatial organization of nodes is an important lightweight way of creating structure. In Aquanet, users have created representational types like labels that help them partition and differentiate a large space. Furthermore, when simple grouping relations like the one shown in Figure 3 (page 7) have been available in addition to labels, users have shown a preference for labels; they just manipulate layout to indicate relationships among objects.

Some discussions use the spatial characteristics of a layout to partition a task among members of a work group. For example, in writing this paper, after initial brainstorming about topics, we each chose lists of nodes under a given topic, and created our own "work areas." Similarly, in our discussion about the system itself (an interchange about the current bugs and desired features), certain areas of the main view became the "property" of a particular author, while others tended to be group efforts.

Spatial layout also provides some useful cues for authors and readers navigating a large discussion, especially when it is coupled with a birds-eye view of the entire structure. Figure 8 shows an example of a large portion of the Aquanet bugs and features discussion. From this overview, it is easy to discern major object groupings and the basic structure of the discussion.

Given the prevalence of these spatial organization and layout strategies, it is clear that future versions of Aquanet should provide explicit support for them.

7.2 Supporting common representational underpinnings

Our early experiences with Aquanet have shown us that schemas will differ radically given different tasks and different people involved in a discussion. Most of the schemas to date are highly specific to the content of the discussion. For example, for our discussion of Aquanet bugs and features, we have developed a schema that includes types like "Bug," "Feature," and "Architectural Change," and relations like "Fix" and "Group" (a relation to group associated objects). Other schemas are more general, like the one expressing the Argument relation that we've used in our earlier examples; the types that it provides do not necessarily correspond to specific characteristics of the task at hand. Our intuition is that the more general schemas are either more difficult to use - it is harder to get people to agree on a content-representation mapping - or that they aren't as powerful in structuring content since useful distinctions aren't brought out.

Although we've discovered that schemas may differ radically, we've also found that some representational needs don't vary that much across applications. One of the first types that we created

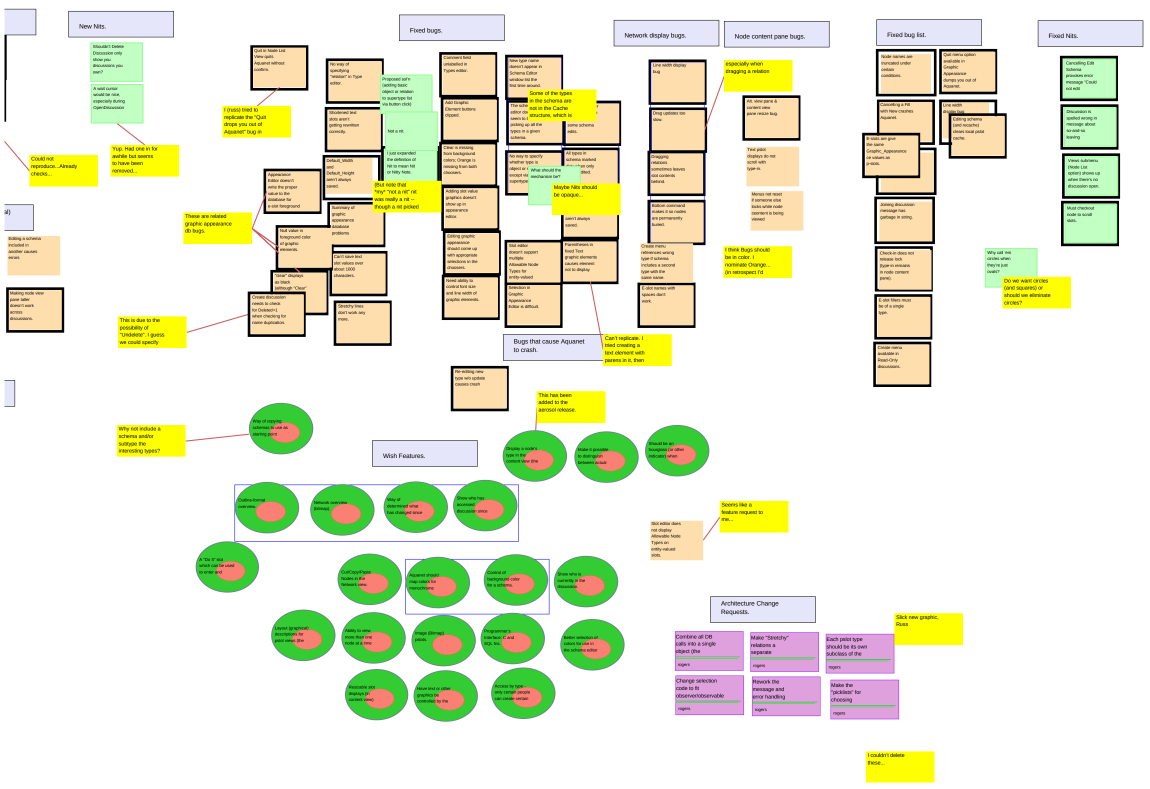


Figure 8. Using spatial layout to organize a discussion.

was modeled after a post-it note. This type is displayed as a bright yellow rectangular background with the contents of a note shown in the foreground in a large font. It proved to be so useful, especially for talking about the schema and for quick annotations, that we included it in many other schemas. We also noted several useful representational primitives like groups, ordered lists, and binary links; these types recur in variant forms in many schemas.

One way we have considered providing users with sets of standard types like lists, groups, outlines, labels, and post-its is as a schema library. Schema developers could then specialize these generally useful constructs for their applications, or include meta-discussion objects like post-its and labels in any discussion.

7.3 Information import and export

Users often come to knowledge structuring tasks with a collection of existing material and some preliminary ideas about how this material should be structured. For example, in the machine translation analysis, we have some on-line sources pulled from USENET and COMLINE; these on-line sources have well-defined internal structure and could easily be automatically integrated into the task's knowledge structure. Automatic import of external information would not only relieve the tedium of entering such information by hand, it would also promote consistent encoding of the information. We are currently designing general mechanisms for importing information into Aquanet.

The converse of the import issue is the externalization of knowledge structures from Aquanet into forms usable by other tools. Because many early Aquanet users requested such a facility, we have implemented a simple report generation program. A more sophisticated externalization capability is also being developed.

8. Dangling Links

With the aerosol release in use within our laboratory, we have begun to focus on building a user community around Aquanet. The feedback we are receiving from people using the tool in real tasks will help direct its future evolution. Based on early observations, we intend to pursue the following research themes:

(1) Extend methods of viewing and interacting with knowledge structures. Specifically, we intend to allow users to define schema-specific viewers. We also will design a generalized network description language.

(2) Provide better support for the schema evolution process and the exploration of alternative structures. This includes a versioning mechanism for both schemas and discussions.

(3) Provide mechanisms for integrating Aquanet with the computing environment. This includes a programmer’s interface.

(4) Provide a mechanism for knowledge structure computations, including associating behaviors with types, schemas, and objects. This mechanism would support for dynamic objects, computed role values, and intelligent discussion “agents” for example.

Acknowledgments

We would like to thank Tom Moran, Susan Newman, Jintae Lee, and Norbert Streitz for their helpful comments during the development of Aquanet and Mark Stefik for his comments on an earlier draft of this paper.

References

[Appl87] Apple Computer, Inc. *Hypercard User's Guide*, Cupertino, California (1987).

[Aksc88] Akscyn, R., McCracken, D., and Yoder, E. "KMS: A Distributed Hypermedia System for Managing Knowledge in Organizations." *CACM 31*, **7** (July 1988), 820-835.

[Bobr77] Bobrow, D.G. & Winograd, T. "An Overview of KRL,A Knowledge Representation Language." *Cognitive Science* 1, **1** (1977), 3-46.

[Brow85] Brown, J.S., & Newman, S.E. "Issues in Cognitive and Social Ergonomics: From Our House to Bauhaus." *Human-Computer Interaction* 1, **4** (1985), 359-391.

[Brow87] Brown, P.J., "Turning Ideas into Products: The Guide System." *Hypertext '87 Proceedings*, Chapel Hill, North Carolina (13-15 November, 1987), 33-40.

[Brun88] Bruns, G. "Germ: A Metasystem for Browsing and Editing." MCC Technical Report STP-122-88, Austin, Texas (1988).

[Bush45] Bush, V. "As We May Think." *Atlantic Monthly*, (August 1945), 101-108.

[Conk88] Conklin, J. and Begeman, M.L., "gIBIS: A Hypertext Tool for Exploratory Policy Discussion." MCC Technical Report Number STP-082-88, Austin, Texas (1988).

[Enge68] Engelbart, D.C., English, W.K. "A Research Center for Augmenting Human Intellect." *Proceedings of the 1968 Fall Joint Computer Conference*, 33 Part 1, Montvale, N.J., AFIPS Press (1968), 395-410.

[Garr86] Garrett, L.N., Smith, K.E., & Meyrowitz, N. "Intermedia: Issues, Strategies, and Tactics in the Design of a Hypermedia Document System." *CSCW '86 Proceedings*, Austin, Texas (3-5 December, 1986), 163-174.

[Hala87] Halasz, F.G., Moran, T.P., & Trigg, R.H. "NoteCards in a Nutshell." *Proceedings of the ACM CHI + GI Conference*, Toronto, Ontario, (5-9 April, 1987), 45-52.

[Hala88] Halasz, F.G., "Reflections on NoteCards: Seven Issues for the Next Generation of Hypermedia Systems." *CACM 31*, **7** (July 1988), 836-852.

[Hala90] Halasz, F.G. & Schwartz, M. "The Dexter Hypertext Reference Model." *Proceedings of the Hypertext Standardization Workshop*, National Institute of Standards and Technology, Gaithersburg, Maryland (16-18 January, 1990), 95-133. (Available as NIST Special Publication 500-178, March 1990.)

[Jord89] Jordan, D.S., Russell, D.M., Jensen, A.M., & Rogers, R.A. "Facilitating the Development of Representations in Hypertext with IDE." *Hypertext '89 Proceedings*, Pittsburgh, Pennsylvania (5-8 November, 1989), 93-104.

[Lee90] Lee, J. "SIBYL: A Qualitative Decision Management System," to appear in

Winston, P. and S. Shellard (Eds.) *Artificial Intelligence at MIT: Expanding Frontiers*, Chapter 5, The MIT Press: Cambridge, MA (1990).

[Mars87] Marshall, C.C. “Exploring Representation Problems using Hypertext.” *Hypertext ‘87 Proceedings*, Chapel Hill, North Carolina, (13-15 November, 1987), 253-268.

[Mars89] Marshall, C.C. “Representing the Structure of a Legal Argument,” *Proceedings of the Second International Conference on AI and Law*, Vancouver, British Columbia, (13-16 June, 1989), 121-127.

[MacL91] MacLean, A., Bellotti, V.M.E., & Moran, T.P. “Questions, Options, and Criteria: Elements of Design Space Analysis.” *Journal of Human-Computer Interaction* 6 (1991).

[Pala88] Palay, A., et al. “The Andrew Toolkit: An Overview.” *Proceedings of the USENIX Technical Conference* (February 1988).

[Newm91] Newman, S.E., and Marshall, C.C. “Pushing Toulmin Too Far: Learning From an Argument Representation Scheme.” Xerox PARC Technical Report, (1991).

[Russ87] Russell, D.M., Moran, T.P., & Jordan, D.S. The Instructional Design Environment. In *Intelligent Tutoring Systems: Lessons Learned*. J. Psotka, L. D. Massey, & S. A. Mutter (Eds). Lawerence Erlbaum Associates, Inc. Hillsdale, N.J. (1987).

[Stee90] Steele, G.L. *Common Lisp: The Language (Second Edition)*. Digital Press, Bedford, MA (1990).

[Stef87] Stefik, M., Foster, G., Bobrow, D., Kahn, K., Lanning, S., and Suchman, L., "Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings." *Communications of the ACM 30,* **1** (January, 1987), 32-47.

[Stre89] Streitz, N.A., Hannemann, J., and Thuring, M. "From Ideas and Arguments to Hyperdocuments: Travelling through Activity Spaces." *Hypertext ’89 Proceedings*, Pittsburgh, Pennsylvania (5-8 November, 1989), 343-364.

[Toul58] Toulmin, S. *The Uses of Argument*. Cambridge University Press, Cambridge, (1958).

[Trig87] Trigg, R.H., Irish, P.M. “Hypertext Habitats: Experiences of Writers in NoteCards.” *Hypertext ‘87 Proceedings*, Chapel Hill, North Carolina, (13-15 November, 1987), 89-108.

[Yake90] Yakemovic, K.C.B., Conklin, E.J. “Report on a Development Project Use of an Issue-based Information System.” *Proceedings of CSCW ‘90*, Los Angeles, California (7-10 October, 1990), 105-118.