CMU-ITC-90-090
August 1990

Building Hypertext on a Multimedia Toolkit: An Overview of Andrew Toolkit Hypermedia Facilities

Mark SHERMAN, Wilfred J. HANSEN,
Michael McINERNY, and Tom NEUENDORFFER

Information Technology Center[*]
Carnegie Mellon University
4910 Forbes Ave
Pittsburgh, PA 15213, USA
Internet: mss+echt90@andrew.cmu.edu

ABSTRACT: This paper discusses several hypermedia facilities built on top of the Andrew Toolkit (ATK) and their use in ATK applications. As a general-purpose, multimedia, application-development system, ATK permits many kinds of links, references and other connections to be made within pieces of content and between pieces of content, regardless of the content's medium. We argue that starting with a multimedia architecture facilitates the construction of all forms of hypermedia. Four specific hypermedia facilities implemented with ATK are discussed: an integrated web and indexing system (Help), a simple multimedia link facility (Link), a cross reference (Textref) capability, and a link-supporting embedded object language (Ness). As a toolkit, ATK is used to build other applications which inherit ATK's hypermedia facilities. Therefore we consider briefly the way that hypermedia facilities are used in conventional applications, such as mail systems.

KEY WORDS: Hypertext, Implementation, Multimedia.

1 Introduction

Considerable effort has been devoted to cataloging dozens of different forms of links for hypertext [Meyrowitz 1989]. Unfortunately, if one sets out to build a system incorporating each as a special case, the implementation effort becomes unwieldy before the benefits become apparent. In this paper we argue that it is preferable to begin by building a general-purpose multimedia architecture; it is then trivial to integrate a variety of link types.

[*]This work was performed as a joint project of Carnegie Mellon University and the IBM Corporation. Some of the work was supported by the National Science Foundation under contract ASC-8617695. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies of the IBM Corporation, the National Science Foundation, or Carnegie Mellon University.

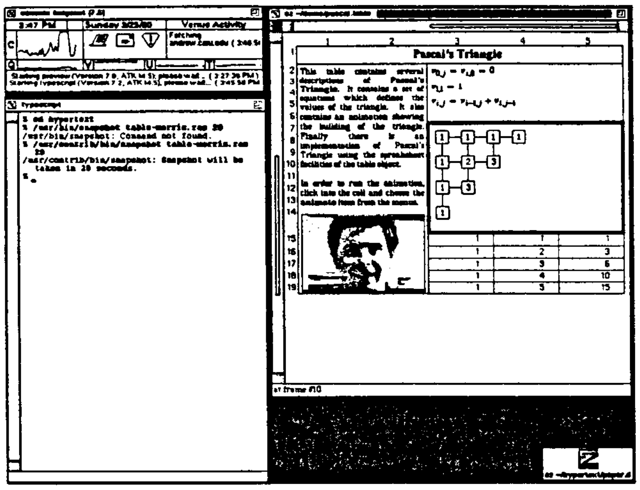
The Andrew Toolkit (ATK) [ITC 1990, Palay *et al.* 1988] is a multimedia application development system with which system builders can create applications quickly. Its applications are in daily use by thousands of people on the CMU campus, other universities, and several industrial research laboratories. In its current release, the system provides support for many media, including multifont text, raster images, structured graphics, animations, spreadsheets, equations and audio. The system also provides facilities for composing and connecting media, such as screen layout objects, buttons, sliders, knobs, and linking facilities of various kinds. The linking facilities provide the same service as hypertext links.

The Andrew Toolkit has three features that distinguish it from hypertext systems. First, ATK is a true multimedia architecture. The presence of media other than simple text is commonplace. Second, ATK is a toolkit for building other applications. Thus, it is not a hypertext system *per se.* Instead, ATK allows any application built with it to use linking facilities. Third, link facilities are optional and are implemented on top of the toolkit. Even in the absence of linking facilities, ATK is a fully operational multimedia authoring system, so both developers and users can ignore gratuitous linking facilities. People may choose other hypertext systems precisely because of their linking facilities, but people choose ATK for its editing and application construction capabilities and, as an added attraction, they can use linking facilities as well. Therefore, ATK provides one of the few test beds for empirical measurement of the utility of linking. In this paper, we provide some preliminary statistics of hypermedia use in one ATK application.

2 Multimedia Facilities

The primary composition mechanism in the Andrew Toolkit is inset nesting, as can be observed on the right side of figure 1. The entire figure shows the screen of an IBM RT running three ATK applications under the X Window System. The upper-left displays console, a system monitoring application; beneath it is typescript, a shell interface. To the right of both is the generic object editor, in this case editing a spreadsheet object. Within which, the left-side cells have been connected together to hold a multifont text object, which in turn contains a raster image. Some right-side cells have been grouped in two sections holding a collection of equations, and an animation. The rest of the right column is used as a spreadsheet.

The nesting of one object in another, such as a raster in the text in



```
% ez hypertext
% /usr/bin/snapshot table-morris.ras 20
/usr/bin/snapshot: Command not found.
% /usr/contrib/bin/snapshot table-morris.ras 20
/usr/contrib/bin/snapshot: Snapshot will be
   taken in 20 seconds.
%
```

Pascal's Triangle

This table contains several descriptions of Pascal's Triangle. It contains a set of equations which defines the values of the triangle. It also contains an animation showing the building of the triangle. Finally there is an implementation of Pascal's Triangle using the spreadsheet facilities of the table object.

In order to run the animation, click into the cell and choose the animate item from the menus.

$v_{0,j} = v_{i,0} = 0$

$v_{1,1} = 1$

$v_{i,j} = v_{i-1,j} + v_{i,j-1}$

1	1	1
1	2	3
1	3	6
1	4	10
1	5	15

Figure 1: Three ATK applications: console, typescript, and ez editor.

the spreadsheet, can continue as deeply as desired. This nesting is the central feature of ATK and is implemented as a set of protocols by which a surrounding object can completely control the environment perceived by an embedded object. In general terms, the protocols provide that events propagate inward so the surrounding object can determine, for instance, whether or not an inner object receives mouse hits.

The generic object editor does not understand the details of any object. If no object type is specified, a multifont text object is used by default—many casual users rely on this default and believe the object editor to be a word processing system. Since the object editor works on any object, the same program can be used to manipulate any medium, and thus forms the basis of multimedia applications.

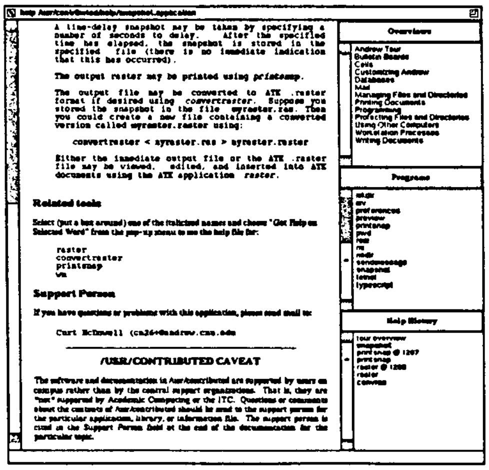
On top of these basic facilities, one can build many different, linking paradigms, such as those described in the next section.

3 Hypermedia Facilities

In this section, we discuss four kinds of linking facilities that have been implemented in ATK: an integrated web, simple links, cross references in text, and fully programmable links.

3.1 *Webs* (Help)

The concept of webs was made popular by Intermedia [Meyrowitz 1986]. Webs define a collection of documents connected by references. In ATK, the impetus for developing a web subsystem was to support the Help application, which organizes and cross references help from many sources, including locally written help files and Unix manual pages. Figure 2 shows Help with the main area of the window—on the left—displaying one of the help documents. The top two panes in the right column list predefined collections of articles, with general help on top, above help for commonly used programs. Both panes represent precompiled collections of references and act as starting points within a web. One can put together dynamically a collection of references, as shown in the bottom-most pane in the right column where we show the authors' attempts to learn how to make screen snapshots for this paper. As various help information is found, bookmarks to locations in specific documents may be added so that we can move from document to document quickly to collate the necessary information.[1] Users can add or remove the panes in the right-hand column as desired.


help /usr/contributed/help/snapshot.application

A time-delay snapshot may be taken by specifying a number of seconds to delay. After the specified time has elapsed, the snapshot is stored in the specified file (there is no immediate indication that this has occurred).

The output raster may be printed using *printsnap*.

The output file may be converted to ATK .raster format if desired using *convertraster*. Suppose you stored the snapshot in the file myraster.ras. Then you could create a new file containing a converted version called myraster.raster using:

```
convertraster < myraster.ras > myraster.raster
```

Either the immediate output file or the ATK .raster file may be viewed, edited, and inserted into ATK documents using the ATK application *raster*.

Related tools

Select (put a box around) one of the italicized names and choose "Get Help on Selected Word" from the pop-up menu to see the help file for:

```
raster
convertraster
printsnap
wm
```

Support Person

If you have questions or problems with this application, please send mail to:

Curt McDowell (cm26+@andrew.cmu.edu

/USR/CONTRIBUTED CAVEAT

The software and documentation in /usr/contributed are supported by users on campus rather than by the central support organizations. That is, they are "not" supported by Academic Computing or the ITC. Questions or comments about the contents of /usr/contributed should be sent to the support person for the particular application, library, or information file. The support person is cited in the Support Person field at the end of the documentation for the particular topic.

Figure 2: The Help application documenting the snapshot program.

The predefined lists provide starting points within the web; users navigate from document to document through the use of marked links. In figure 3, we see another help document with a link marked. When the link is followed, the new document is displayed in the main part of the Help window. The system uses auxiliary files to map link-names to help documents. Different

[1]Currently, Help does not remember a user's bookmarks from one session to the next.

auxiliary files can be used to provide different mappings and thus different webs. For example, the mapping files used by ATK developers contain references to local, obscure system programs while the mapping files used by the main campus provide information on stable, released systems. Thus, the use of new auxiliary files allows one to create webs of information for any application, not just help files.

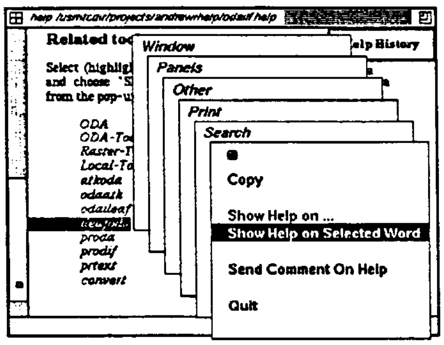


Figure 3: Following a link in the Help application.

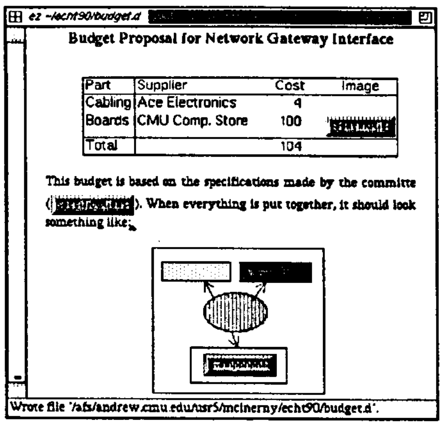
3.2 Simple Links (Link)

The most common form of linking is a reference from one document to another. The special object that contains a reference is called a link. When followed, the link causes the reference to be displayed in a new window employing the generic object editor, similar to the way a link works in NoteCards [Halasz 1987].

Figure 4 shows a document with three links in it. One link is contained in the spreadsheet, another in the text and the third is within the drawing. As an ATK object, a link can be placed anywhere other objects may be placed. The link is followed by clicking the left mouse button on the link symbol. Since it is a separate entity, the link object has no global knowledge about the containing document or further links in referenced documents. Consequently, there is no way to get global information about the web in which the link is embedded, although a history of visited documents could be built. Links are also unidirectional and contained only in the source document: the target document has no way of knowing that it is a target of a link.

The destination of a link is a filename.[2] Since our site uses a wide area

[2]We have several designs for pointing to a target within a file, but the simple solutions



Budget Proposal for Network Gateway Interface

Part	Supplier	Cost	Image
Cabling	Ace Electronics	4	
Boards	CMU Comp. Store	100	[illegible]
Total		104	

This budget is based on the specifications made by the committe ([illegible]). When everything is put together, it should look something like:

Wrote file '/afs/andrew.cmu.edu/usr5/mcinerny/echt90/budget.d'.

Figure 4: Simple links embedded in a spreadsheet, text, and drawing insets, which comprise one document.

network distributed file system [Howard 1988], links can refer to documents that exist across the country as easily as documents on the local workstation. However, moving a set of linked documents is difficult, since absolute pathnames are encoded in the document.[3]

3.3 Cross References (Textref)

While simple links provide a reference from one point in a document to another file, cross references are used to provide references from within a document to another point in the same document. Traditionally in printed text, a cross reference appears as a label at the target, with an indication of the target page and label at the source of the cross reference.

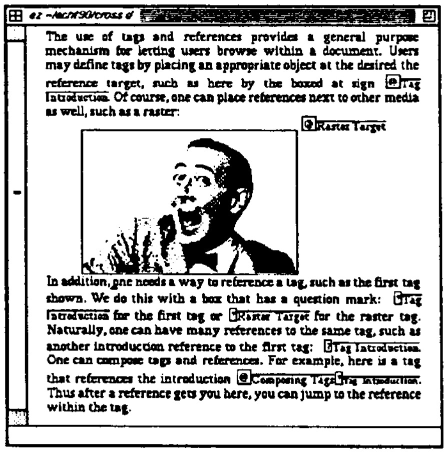
ATK provides support for the traditional use of cross references, but also allows the user to interact with a reference when the document is on the screen. Users define *tags* that denote destinations and *references* that denote links. Clicking the left mouse button on a reference will make the target (tag) visible and move the text caret to the target. In figure 5, we show a document with its tags and references exposed so that the names are visible. Normally, only icons indicating tags or references would be visible.[4]

While cross references work only in text, this is not a serious limitation. Since our text object can embed other objects arbitrarily, one simply places

are clumsy, and the better solutions would require modification of the toolkit, the target documents, or both.

[3]For example, we use sed to install the help documents describing the link inset!

[4]The visibility of the name for a tag or a reference is toggled by clicking on the icon with the right mouse button.



The use of tags and references provides a general purpose mechanism for letting users browse within a document. Users may define tags by placing an appropriate object at the desired the reference target, such as here by the boxed at sign @Tag Introduction. Of course, one can place references next to other media as well, such as a raster:

@Raster Target

In addition, one needs a way to reference a tag, such as the first tag shown. We do this with a box that has a question mark: ?Tag Introduction for the first tag or ?Raster Target for the raster tag. Naturally, one can have many references to the same tag, such as another introduction reference to the first tag: ?Tag Introduction. One can compose tags and references. For example, here is a tag that references the introduction @Composing Tags?Tag Introduction. Thus after a reference gets you here, you can jump to the reference within the tag.

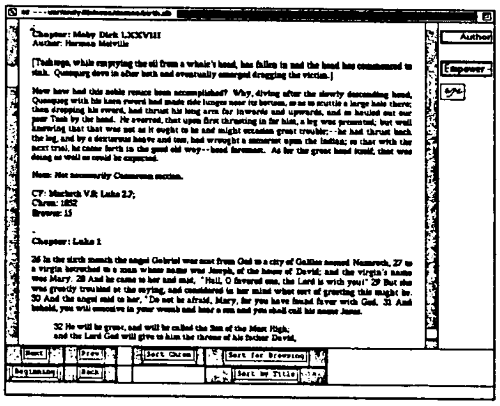
Figure 5: Active cross references within a document: "@" is for targets, "?" is for their references.

references adjacent to non-text objects.

3.4 Programmed Links (Ness)

The most sophisticated linking facilities in the Andrew Toolkit are provided by the embedded object language Ness [Hansen 1990]. This is essentially an advanced Hypertalk-like language [Atkinson 1987], where the elements of an application are laid out utilizing the ADEW application builder [Neuendorffer 1990]. The language can be used to extend objects in documents or to build entire applications, such as the quotation browser shown in figure 6. The main portion of the window is devoted to a scrollable document containing a number of quotations about birth. Across the bottom of the window are buttons which perform various operations such as moving to the next or previous quotation. These buttons are similar to the Next and Previous buttons often found in Hypercard applications. Unlike Hypercard stacks, the document is a normal text object and has all the behaviors that users have come to expect from text; they can scroll, search for strings, and copy text.

Within each entry, a mouse click on a reference in a "CF:" line scrolls the text to display the referenced message. For example, if the mouse is clicked on "Macbeth" in the middle of figure 6, the text scrolls to the entry for Macbeth shown in figure 7. The cross referencing is accomplished with a Ness routine which checks each mouse click to see if it is in a line beginning with "CF:".



Chapter: Moby Dick LXXVIII
Author: Herman Melville

[Tashtego, while emptying the oil from a whale's head, has fallen in and the head has commenced to sink. Queequeg dove in after both and eventually emerged dragging the victim.]

Now how had this noble rescue been accomplished? Why, diving after the slowly descending head, Queequeg with his keen sword had made side lunges near its bottom, so as to scuttle a large hole there; then dropping his sword, had thrust his long arm far inwards and upwards, and so hauled out our poor Tash by the head. He averred, that upon first thrusting in for him, a leg was presented; but well knowing that that was not as it ought to be and might occasion great trouble;--he had thrust back the leg, and by a dexterous heave and toss, had wrought a somerset upon the Indian; so that with the next trial, he came forth in the good old way--head foremost. As for the great head itself, that was doing as well as could be expected.

Note: Not necessarily Caesarean section.

CF: Macbeth V.8; Luke 2.7;
Chron: 1852
Browse: 15

Chapter: Luke 1

26 In the sixth month the angel Gabriel was sent from God to a city of Galilee named Nazareth, 27 to a virgin betrothed to a man whose name was Joseph, of the house of David; and the virgin's name was Mary. 28 And he came to her and said, "Hail, O favored one, the Lord is with you!" 29 But she was greatly troubled at the saying, and considered in her mind what sort of greeting this might be. 30 And the angel said to her, "Do not be afraid, Mary, for you have found favor with God. 31 And behold, you will conceive in your womb and bear a son and you shall call his name Jesus.

32 He will be great, and will be called the Son of the Most High;
and the Lord God will give to him the throne of his father David,

Figure 6: A hypertext database driven by a Ness script.

The linking performed by the embedded language is simple and quite general. If desired, one can use the full facility of an object-oriented, multi-media manipulation language. For example, one can move within a document, move to another document, move to multiple documents in multiple windows, and search and link based on media type and content. With a full programming language, many tricks are possible.

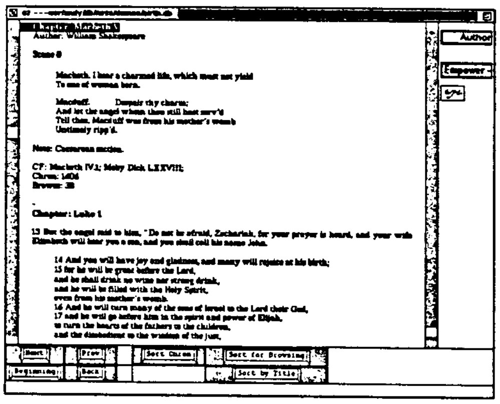
3.5 Documents with a Variety of Link Types

The linking facilities we have discussed are not mutually exclusive. For example, a document within the Help system can provide simple links to other documents. The link facility itself is documented this way. Figure 8 shows the Help system in the top window displaying the documentation for the link object, which contains a link to supplemental materials which have been brought up in the bottom window. Another example is shown in figure 9, which shows a database of articles on hypermedia and hypertext. There are programmed links that connect together pieces of information, supplemented by simple links that point to other collections of information.

4 Hypermedia Applications

All applications built with the Andrew Toolkit can utilize the above link facilities. To get a feeling for how these features are used, we consider briefly two applications now in general use: the Help and the Messages systems.

As used on the CMU campus, the Help system [Langston 1988, Ogura & Robertson 1989] provides help on all aspects of the computing environment, not just the Andrew system or programs on high-function



Chapter: Macbeth V
Author: William Shakespeare

Scene 8

Macbeth. I bear a charmed life, which must not yield
To one of woman born.

Macduff. Despair thy charm;
And let the angel whom thou still hast serv'd
Tell thee, Macduff was from his mother's womb
Untimely ripp'd.

Note: Caesarean section.

CF: Macbeth IV.1; Moby Dick LXXVIII;
Chron: 1606
Brown: 38

Chapter: Luke 1

13 But the angel said to him, "Do not be afraid, Zechariah, for your prayer is heard, and your wife Elizabeth will bear you a son, and you shall call his name John.

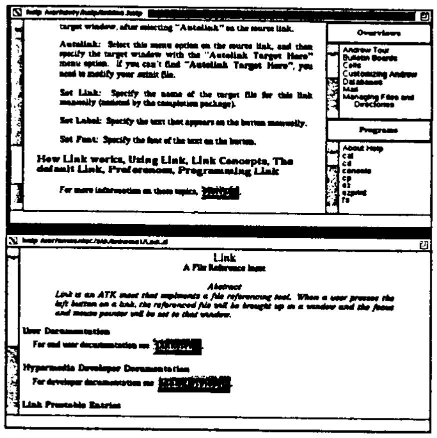
14 And you will have joy and gladness, and many will rejoice at his birth;
15 for he will be great before the Lord,
and he shall drink no wine nor strong drink,
and he will be filled with the Holy Spirit,
even from his mother's womb.
16 And he will turn many of the sons of Israel to the Lord their God,
17 and he will go before him in the spirit and power of Elijah,
to turn the hearts of the fathers to the children,
and the disobedient to the wisdom of the just,

Figure 7: The *Macbeth* quote referenced by the *Moby Dick* quote in figure 6.

workstations. Because users requesting help from low-function workstations (IBM PCs and Macintoshes) and terminals would not have access to the multimedia features of the Andrew Toolkit, the generally available help documentation does not contain multimedia information. The exceptions are documentation used only by the developers, such as the link documentation illustrated in figure 8. Of the approximately 200 documentation files written by our organization, there are over 3000 cross references and over 600 mappings between cross references and documentation files. Since these files occupy about 1.8 megabytes of space, there is a reference for every 300 characters, or about 5 lines of documentation. In addition, the system automatically links together general documentation and Unix manual pages. The Help system was designed, and the documentation was written, with links in mind, so the prevalence of references is not surprising.

However, the message system built atop ATK [Borenstein *et al.* 1988] had no *a priori* design for hypertext information. Indeed, the overwhelming majority of the 2500 bulletin boards it offers are generated by non-ATK sources, and hence do not exploit any multimedia or linking facilities. Fortunately, the private bulletin board used by our organization of 30 researchers consists of posts made almost entirely by an ATK program. Because members of our organization have both the knowledge of how to use the linking features of ATK and access to ATK-based message creation programs, we have an opportunity to examine the prevalence of linking.

Over a two month period, 369 messages were posted on our private bulletin board. Of those posted, 290 messages had some multimedia feature in them. The most popular feature was multifont text—258 or 88% of the


target window, after selecting "Autolink" on the source link.

Autolink: Select this menu option on the source link, and then specify the target window with the "Autolink Target Here" menu option. If you can't find "Autolink Target Here", you need to modify your atkinit file.

Set Link: Specify the name of the target file for this link manually (assisted by the completion package).

Set Label: Specify the text that appears on the button manually.

Set Font: Specify the font of the text on the button.

How Link works, Using Link, Link Concepts, The default Link, Preferences, Programming Link

For more information on these topics, [illegible].

Link
A File Reference Inset

Abstract
Link is an ATK inset that implements a file referencing tool. When a user presses the left button on a link, the referenced file will be brought up in a window and the focus and mouse pointer will be set to that window.

User Documentation
For end user documentation see [illegible].

Hypermedia Developer Documentation
For developer documentation see [illegible].

Link Printable Entries

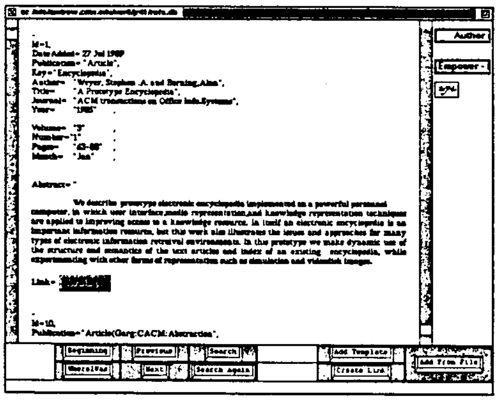
Figure 8: The Help application documenting the link inset, and the window brought up by following the link.

multimedia messages used only that feature. However, of the 32 multimedia messages that used more than multifont text, 22 contained some linking feature. While not an overwhelming endorsement of hypertext or hypermedia, a 6% casual usage rate indicates some appreciation for linking.

5 Conclusions

The Andrew Toolkit effort began not as a hypertext effort, but as a general architecture for multimedia documents and applications. We have shown in this paper that the resulting system is an excellent medium for hypertext and hypermedia. Several linking styles have already been utilized in ATK applications. Web-based linking was chosen for an application—the Help system—which is large, but changes slowly enough to permit a perdefinition of access paths. Simple links and cross references are used for casual references, as might appear on a bulletin board, while computed links are used for database-like applications. There does not appear to be a universal choice for a linking mechanism and, even within the Andrew Toolkit, other choices are possible. For example, a group at Olivetti has independently developed a hypermedia system on top of the Andrew Toolkit [Olivetti undated].

We believe that our investigations indicate that one should not concentrate one's efforts on a single linking system or on trying to extend a simple hypertext system into a hypermedia system. A better path is to investigate different linking mechanisms within a general-purpose multimedia system, such as ATK. This gives the hypertext implementor a clean architecture



```
Id=1,
DateAdded= 27 Jul 1989
Publication= "Article",
Key="Encyclopedia",
Author=   "Weyer, Stephen A. and Borning,Alan",
Title=    "A Prototype Encyclopedia",
Journal=  "ACM transactions on Office Info.Systems",
Year=     "1985"      ,

Volume=  "3"          ,
Number="1"            ,
Pages=   "63-88"      ,
Month=   "Jan"        ,


Abstract= "

        We describe prototype electronic encyclopedia implemented on a powerful personal
computer, in which user interface,media representation,and knowledge representation techniques
are applied to improving access to a knowledge resource. In itself an electronic encyclopedia is an
important information resource, but this work also illustrates the issues and approaches for many
types of electronic information retrieval environments. In this prototype we make dynamic use of
the structure and semantics of the text articles and index of an existing  encyclopedia, while
experimenting with other forms of representation such as simulation and videodisk images.

Link= [illegible]


Id=10,
Publication="Article(Garg:CACM:Abstraction",
```

Figure 9: A Ness-driven hypertext with embedded simple links.

for embedding hypertext within documents and applications while freeing the implementor from having to build a multimedia document authoring system. By investigating many different linking mechanisms, we may find the best ways that hypermedia can be used and add linking as another tool in our arsenal for organizing information.

References

[Atkinson 1987] Atkinson, B., *HyperCard*, Version 1.0.1, M0556 / 690-5181-A, Apple Computer, Cupertino, CA, 1987.

[Borenstein *et al.* 1988] Borenstein, Nathaniel, Craig Everhart, Jonathan Rosenberg, Adam Stoller, "A Multi-media Message System for Andrew", *Proceedings of the USENIX Winter Conference*, Dallas, February, 1988, pp. 37–42.

[Halasz 1987] Halasz, Frank G., "Reflections on Notecards: Seven Issues for the Next Generation of Hypermedia Systems," *Hypertext '87 Proceedings,* Nov. 13–15, 1987, Chapel Hill, NC, pp. 345–365.

[Hansen 1990] Hansen, Wilfred J., "Enhancing documents with embedded programs: How Ness extends insets in the Andrew ToolKit," *Proceedings of 1990 International Conference on Computer Languages*, March 12–15, 1990, New Orleans, IEEE Computer Society Press, Los Alamitos, CA, 1990, pp. 23-32.

[Howard 1988] Howard, John. H., "An Overview of the Andrew File System," *Proceedings of the USENIX Winter Conference*, Dallas, February, 1988, pp. 23–26.

[ITC 1990] *Andrew Toolkit Programmer's Manual,* Information Technology Center, Carnegie Mellon University, Pittsburgh, PA. 15213, USA, January 1990.

[Langston 1988] Langston, Diane, "Background and Initial Problems for the Andrew Help System," *Proceedings of the 35th ITCC,* Society for Technical Communications, 1988, pp. ATA-47–ATA-50.

[Meyrowitz 1986] Meyrowitz, N., "Intermedia: The Architecture and Construction of an Object-Oriented Hypermedia System and Applications Framework", *OOPSLA '86 Proceedings*, Portland, OR, 1986, pp. 186–201.

[Meyrowitz 1989] Meyrowitz, N., "Hypertext—Does it Reduce Cholesterol, too?", presented at Hypertext '89, Pittsburgh, PA, November, 1989, IRIS Technical Report 89-9, Brown University, Providence, RI, 1989.

[Neuendorffer 1990] Neuendorffer, Thomas P., "The Andrew Development Environment Workbench: An Overview", *The Andrew Project*, Technical Report, Information Technology Center, Carnegie Mellon University, 4910 Forbes Ave., Pittsburgh, PA 15213, 1990, pp. 65–72.

[Ogura & Robertson 1989] Ogura, Ayami, Jennifer Robertson, "Designing Hypermedia Help Systems: Problems and Issues", *Conference Proceedings, SIGDOC 89,* Pittsburgh, PA, November 8–10, 1989, pp. 5–12.

[Olivetti undated] "Hypermedia Help System", Olivetti Internal Memo, Olivetti S&N DOR, Advanced Software Laboratory, via Palestro 30, 56100 Pisa, Italy, undated.

[Palay *et al.* 1988] Palay, A. J., et al., "The Andrew Toolkit—An Overview", *Proceedings of the USENIX Winter Conference*, Dallas, February, 1988, pp. 9–21.