13/1

AUTHORSHIP PROVISIONS IN AUGMENT

Douglas C. Engelbart

Tymshare, Inc.
Cupertino, California 95014
(OAD,2250,)

ABSTRACT

AUGMENT is a text processing system marketed by Tymshare for a multi-user, network environment. In AUGMENT's frontend is a User Interface System that facilitates flexible evolution of command languages and provides optional command recognition features. Exceptionally fast and flexible control of interactive operations is enabled by concurrent action of mouse and optional one-handed chord keyset. Files are hierarchically structured, and textual address expressions can flexibly specify any text entity in any file. The screen may be divided into arbitrary, rectangular windows, allowing cross-file editing between windows. Many options exist for controlling the "view" of a file's text in a window, e.g.: level clipping, paragraph truncation, and content filtering. Structural study and modification of on-line documents are especially facilitated. A Journal system and "Shared Screen Teleconferencing" support collaboration among authors and their colleagues. Graphic illustrations may be embedded in the same file with text.

INTRODUCTION

AUGMENT was designed for augmenting human intellectual capabilities. It was targeted particularly toward the core work of professionals engaged in "tough knowledge work" – e.g., planning, analyzing, and designing in complex problem domains. And special attention was paid to augmenting group collaboration among workers pursuing common goals.

Authorship has received a great deal of attention in AUGMENT's evolution, as one of the central human activities to be augmented. An important set of provisions within AUGMENT – in its architecture, design principles, and specific features – is directly aimed toward bringing high performance to the authorship activities of knowledge workers. For the purposes of this paper, we thus speak interchangeably of "knowledge worker" and "author."

We recognize explicitly that highly skilled workers in any field, and knowledge work is no exception, are those with good command of their tools. Our basic design goal was to provide a set of tools that would not themselves limit the capabilities of the people using them. A system designed to encourage more skilled workers will always enable higher human performance than one designed to support less skilled workers.

In this regard, our design goal was to provide as much capability as possible for each level of system usage skill, and a continuous evolution path between skill levels. We believe firmly that knowledge workers are motivated to grow in knowledge and skill and that provisions in system design should support this. As the rest of the paper reveals, this approach translates into a rich set of AUGMENT provisions, aimed at providing speed and flexibility for skilled workers in organizing and pursuing their core knowledge work – in which "authorship" is a primary activity.

An explicit sub-goal in AUGMENT's development was to "augment" the development, production and control of complex technical documentation – through the whole cycle of gathering information, planning, creating, collaborating, reviewing, editing, controlling versions, designing layout, and producing the final documents.

This paper concentrates upon the development phase of this cycle. AUGMENT has well-developed tools to support the later, production phase, but their discussion is not included here.

Studying another's work provides a well-recognized challenge, but one of the toughest jobs is to study one's own work during its development: to see what it really says about Issue X; to see if it does provide for Concept Y; to see if it is reasonably organized and structured – and to do these over a body of material before it is "polished", i.e., before it is well structured, coherently worded, non-redundant and consistently termed.

SOME BACKGROUND

HISTORY

AUGMENT is an integrated system of knowledge-worker tools that is marketed by Tymshare's Office Automation Division. The system was developed at SRI International over an extended period under the sponsorship of NASA, DARPA, and RADC. Commercial rights were transferred to Tymshare in 1978 (where the system has since been renamed from NLS to AUGMENT) and its evolution continued. A short history of AUGMENT's development may be found in <Ref-1>, along with a summary of system characteristics and features. The general R&D philosophy and the design principles behind AUGMENT'S development are laid out in <Ref-2>.

The system evolved on time-shared, mainframe computers, and in a packet-switched network environment.

CH2017-2/84/0000/0465$01.00 © 1984 IEEE

In 1970 our computer was the second to be attached to the ARPANET, and since 1978 we have also operated extensively in the TYMNET environment. We have benefited directly from both the time-sharing and the network environments in matters that are important to the authorship process – especially in dealing with large documents and multi-party documentation activities. In 1976-77 we conducted some applied studies for the Air Force, as reported in <Ref-3> and <Ref-4>, which concentrated upon this latter application.

RELEVANT ARCHITECTURAL FEATURES

Perhaps AUGMENT's most unique architectural feature is its User Interface System (UIS), a special software module, which handles the human/computer interfaces to all interactive programs. It takes care of all command-language dialog and connection protocols, and provides a framework for building a coherent and integrated user environment while supporting flexible evolution on both sides: on the user's side, with evolution of command function and terminology; and on the technology side, with evolving hardware and software. (Design details are outlined in <Ref-5>; rationale and utilization in <Ref-6>.)

The UIS provides a reach-through service to non-AUGMENT systems, and can optionally translate back and forth to a foreign program's command language. It also supports the shared-screen, remote collaboration capability discussed below.

AUGMENT's architecture provides for open-ended expansion and flexible evolution of system functionality and worker command languages.

It is assumed that for any class of knowledge workers, specialized application systems developed by other parties, perhaps running on other computers, will provide services worth integrating. The "author class" of worker should be no exception. Continuing evolution toward the "author workshop of the future" will certainly depend upon some such features in workshop architecture.

It provides adaptation for different terminal characteristics, enabling application programers to work as though with a virtual terminal.

FILE CHARACTERISTICS

AUGMENT employs explicitly structured files, with hierarchically organized nodes; each node can contain either or all of: up to 2,000 characters of text, a graphic structure, or other forms of useful data (e.g., digitized speech). The worker has a definite model in mind for the structuring of any file that he works with; in composing and modifying it he can organize and modify structure using the same verbs as for working with text strings (e.g. Insert, Replace, Move, Copy, Delete), with appropriate structural-entity nouns (e.g., Statement, Branch, Group, Plex). For any existing hierarchical structure, he has many flexible alternatives for addressing its entities, modifying its organization, jumping around within it, and viewing it in a most beneficial manner.

(Note: AUGMENT workers generally use the term "statement" to refer to a file node, which is natural enough since the terminology became established before we added the graphic capability. Now an AUGMENT "statement" can contain either or both a text statement and a graphic diagram.)

CONTROLLING THE TOOLS

Many of AUGMENT's unique author-support provisions address basic operations common to almost every task, things done over and over again. These operations, executed with speed and flexibility, provide for composing and modifying one's working material, and for studying what is there over a wide range of substantive levels – from a single text passage to a collection of end-product draft documents and their associated set of working notes, reference material, and recorded-message dialog (assuming all to be on line).

In the early stages of our program at SRI, we did a great deal of detailed work on what we called the "control interface" – how users control the functional application of their tools. These details can be very important to "low-level" interactions which are done hundreds of times during a working day. Some of these details are quite relevant to bringing high performance to the authorship process.

AUGMENT commands are expressed with verbs, nouns, and appropriate qualifier words; every command word is designated by entering one or more characters. The UIS recognizes the command word from these characters according to the command-recognition options designated in each individual's "profile file." Users seem to migrate fairly rapidly to "expert" recognition modes, where a minimum number of characters will elicit recognition of command words. The fully spelled-out command words are presented in the Command Feedback Window as soon as they are recognized. The Backspace Key will cause backup, one command word at a time.

Of the system requirements behind our choice of this noun-verb command form, two are particularly relevant here: (1) The "vocabulary" of the functions of the tools, and of the entities they operate upon, must be as extensible as is a natural language; (2) Textual lists of commands must conveniently lend themselves to writing, documenting, and executing as "macro" commands.

Screen selection is done with a mouse. If the command's noun is a single, defined text or structure entity, e.g., a "word", then there is only one selection needed (e.g., to pick any character in the designated word).

Besides using a standard keyboard for character entry, an AUGMENT user may optionally use a five-key, one-hand, chord keyset. Remarkably little practice is required in order to enter alphabetic characters, one hand-stroke per character. With less than five hours practice, a person can begin profitably working in a two-handed, concurrent mode – operating the mouse with one hand and simultaneously entering command characters and short literal strings with the other hand.

Here is an example of a low-level action which reveals some basic characteristics of high-performance execution. It is a very simple situation, but representative of what is met over and over and over again in doing hard knowledge work. The worker is composing or modifying something in one area of the screen, when his eye catches a one-character typo in another area. For a skilled AUGMENT worker, the typo could be corrected

in less time than it would take someone to point it out to him – with three quick strokes of the keyset hand during a casual flick of the mouse hand, and an absolute minimum of visual and mental attention taken from the other ongoing task.

Fast, flexible, graceful, low effort – these are important to all high-frequency, low-level, knowledge-work actions. This same kind of speed and flexibility are achieved by skilled AUGMENT workers in executing all of the other functional features described below. Description of mouse and keyset, and their concurrent employment, may be found in <Ref-7>.

ADDRESSING THE WORKING MATERIALS

There is a consistent set of addressing features that a worker may use in any command to designate a particular structural node or some element of text or graphics attached to that node. It adds appreciably to the power and flexibility of the system commands to have a rich, universally applicable vocabulary for directly addressing particular entities within the working files. Below are some examples.

EXPLICIT STATEMENT ADDRESSES

There are four "handles" by which a given statement may be directly addressed:

STRUCTURAL STATEMENT NUMBER. This designates the current "structural location" of the statement. It is assigned by the system, depending upon where the worker installs or moves a statement within an existing structure, or how that structure might have been re-organized subsequently. It is usually expressed as an alternating sequence of number-letter fields – e.g. "1", "1a", "1a1", "1a2", and "1b". At a worker's option, these same statement numbers could be shown as "1", "1.1", "1.1.1", "1.1.2", or "1.2", but this bulkier alternative is seldom chosen.

STATEMENT IDENTIFIER, or SID. This is a unique integer, assigned in sequential order by the system as each statement is first inserted, and which stays with a statement no matter how much its content may be altered or where it may be moved in its file structure. To make it uniquely recognizable for what it is, a SID is always displayed, printed, or designated with a prefixed "0" – e.g., "012", "0417", etc. SIDs are particularly useful for referencing passages in a document while it is evolving.

A WORKER-ASSIGNED STATEMENT NAME (or label). For any statement or part of the file structure, an author can designate as "name delimiters" a pair of characters that indicate to the system when the first word of a statement is to be treated as a name for that statement. For instance, if "(" and ")" are set by the author as name delimiters for a specified part of the file, any parenthesized first word in a statement would be recognized by the system as that statement's name.

(Note: It is optional whether to have any of the above three identifiers displayed or printed with the statements' text.)

A DIRECT SCREEN SELECTION. When a statement to be designated is displayed in a window, usually the best way to "address" it is to use the mouse to position the cursor anywhere on the statement and depress the mouse's "Select" key (indicated below by "<Select>"). This mode is generally used for text manipulation – selecting characters, words, numbers, visibles, invisibles, etc. (any of the text entities which have been made system recognizable).

MARKERS

As one "holds a place" in a book by leaving a temporary place marker in it, an author can place "markers" at arbitrary locations within an AUGMENT file. When placing a marker, he attaches it to a specific character in the text and gives it a name or label. Marker names are local to each file. Simple commands provide for displaying where one's markers are located and what their names are, for deleting or moving a marker, or for installing a new one.

A marker name may be included in an address expression, to provide another way of designating an address. A marker name can designate not only a particular statement, but a specific character within that statement. For example, "Copy Word #x (to follow word) <Select>" would designate that a word located somewhere in the file and marked with an "x" is to be copied to follow the cursor-selected word. There are many unique ways in which markers may be employed by an author who has integrated their artful use into her working methodology.

As a comparative example of some of the foregoing addressing forms, consider a statement whose SID is "069", whose statement number is "3b5", that has statement-name delimiters designated for it as "NULL" and ":", that starts with the text "Capacity: For every ...", and that has a marker named "x" positioned on one of its characters. A command to move this statement could optionally be expressed as:

"Move Statement <Select> ...",
"Move Statement 3b5 ...",
"Move Statement 069 ...",
"Move Statement Capacity ...", or
"Move Statement #x ...".

RELATIVE-ADDRESS EXTENSIONS

A sequence of characters may be appended to the address of a given statement to specify an address of a position "relative" to that statement. A major class of these designations deals with relative structural location, such as: Up a level, Down a level, Successor at same level, Predecessor at same level, Head at this level, Tail at this level, and End statement at last and lowest position in this branch. A period (".") in the address string indicates that relative addressing is beginning, and each of these relative-location designators is indicated with a directly mnemonic, one-letter designation.

For example, "Move Statement 0609 (to follow statement) 4b.dt" would move Statement 0609 to follow the tail statement of the substructure one level down from Statement 4b – or, to conceptualize the associated address-location pathway, "go to 4b, then Down a level and to the Tail".

EMBEDDED CITATION LINKS

A special use of address expressions is within an explicit text entity that we call a "Citation Link" (or "Link" for short). Links are used as textual citations

to some specific file item within the workshop domain. A link is delimited by parentheses or angle brackets and contains a valid address string whose path leads to the cited file entity. For example, "(0306)" or "(4b.dt)" are valid links. Also, the reference items at the end of this paper are statements named "Ref-1", "Ref-2", etc., and as such can be cited with links "<Ref-1>", "<Ref-2>", etc. An AUGMENT reader may travel via such a link directly to the referenced bibliographic citation.

A special feature in AUGMENT's link provisions is the use of "indirect link referencing". In path-following terms, including ".1" in an address string stipulates, "scan forward from this point to the next link, and follow that link to its target." For example, to follow the path prescribed by link "(4b.1)", one would "go to 4b, then find the first link in that statement and follow the path that it specifies." This latter path in turn could prescribe use of another link, etc. There is no intrinsic limit to the number of these indirect links that may be employed in a given path – only a natural caution against such a path looping back upon itself.

As an example, note that "<Ref-1>" is a link to the statement named "Ref-1", a bibliographic citation at the end of this paper. In that citation, there is a link to the original source document of the referenced publication, permanently stored in the AUGMENT Journal as item 71279 (the Journal is described below). The point to be made here is that with the link "<Ref-1.1>", I can reference the original source document – and a Jump Link command would "take me there."

TEXT AND CONTENT ADDRESSING

Other addressing options include scanning for a content match, and/or stepping backward and forward a given number of characters or words (or other text entities). For instance, the foregoing link could have involved a bit more smarts in designating which link to follow: e.g., the path for '(4b "*D" .1)' would be "to 4b, scan for first occurrence of "*D", then follow the next link found in that statement."

OTHER-FILE ADDRESSING

By preceding an in-file address string with a file address, and separating the two strings with a comma, one obtains a composite address designating a given entity within a given file. Extending this principle lets one prefix the file name with a directory name in which the file is to be found; and further, one can prefix this with a computer name.

For example, '(Office-5, Program-Documentation, Sequence-Doc, Specifications "Journal")' specifies the path: to the Office-5 host computer, to its Program-Documentation file directory, to its Sequence-Doc file, to its statement named "Specifications", and then scan to the location of the text "Journal".

If a person were working on the Office-5 host, he would only have to specify '(Program-Documentation, Sequence-Doc, Specifications "Journal")'. If he were already working within a file with its "link default" set to the Program-Documentation directory, he would only have to specify '(Sequence-Doc, Specifications "Journal")'. And if he were already working within the Sequence-Doc file, he would only have to specify '(Specifications "Journal")'. And if he were planning to reference items relative to the Statement named "Specifications" very often, he could affix a marker (e.g., named "s") to its front and would then only have to specify '(#s "Journal")'.

Or, suppose he were working in another file in a different directory on Office-5 and wanted to reference items relative to that same "far off" statement with special ease: in some temporary place in that file he could install a statement named "Ref" (for example) containing the textual link, "(Program-Documentation, Sequence-Doc, Specifications)". He could then cite the above reference with the link, '(Ref.1 "Journal")'. This path description is: go to the statement in this file named "Ref", take the first link that you find there (traveling across intervening directories and files and statements), and beginning in the statement on the other end of that link, scan forward to the string "Journal".

This is only a cursory treatment, but should illustrate well enough what is meant by "a rich and flexible addressing vocabulary." As with other high-performance features in AUGMENT, a beginner is not forced to become involved in the larger vocabulary in order to do useful work (with productivity on at least a par with some other, restricted-vocabulary system). But an AUGMENT worker interested in higher performance can steadily pick up more of the optional vocabulary and skills in a smooth, upward-compatible progression.

CONTROLLING THE VIEWS

A user of a book, or of most on-line text systems, is constrained to viewing the text as though he had a window through which he sees a fixed, formatted document. But as described below, our worker can view a section of text in many ways, depending upon his need of the moment.

MULTIPLE WINDOWS

For whatever total screen area is available to the worker, his general performance will be improved significantly if he can flexibly allocate that area into arbitrary-sized windows whose contents can be independently controlled. AUGMENT has long provided this basic capability, along with the provision that material from any accessible file may be shown in any window, and also that screen-select copying or moving can be done across the different windows.

(Note: Cross-file editing can be done at any time, between any two legally accessible files. If one or the other file's material or destination is not being displayed in any of the windows, one may always opt to employ a textual address expression instead of a <Select> within any editing command.)

User-adjustable parameters are used to control the view presented on the display. Adjusting one's view parameters is a constantly used AUGMENT feature that has solidly proved its value. To facilitate their quick and flexible use, the view-specification actions evolved into cryptic, single-character codes, called "viewspecs." The syntax of all Jump commands (used for traveling) includes the option of designating new viewspecs, and a special combination of mouse buttons enables quick, concurrent, keyset action to change the viewspecs for a given window. Here are a few of the

frequently used view controls:

WINDOW VIEWS

STRUCTURE CUTOFF. Show only the statements that lie "below" this statement in the structure (i.e., this "branch"); or show only those following statements that are at this level or deeper; or show all of the following statements that will fit in this window.

LEVEL CLIPPING. For the designated structure cutoff, show only the statements down to a specified level. Lower-level statements are "clipped" from the view; the worker can thus view just a selected number of the upper levels of his document/file.

STATEMENT TRUNCATION. For those statements brought into view (as selected by other view specifications), show only their first n lines. Truncation to one line is often used, along with level clipping, in order to get an effective overview.

INTER-STATEMENT SEPARATION. For viewing ease – blank lines can be optionally installed between statements.

(Note: The foregoing view controls are extremely helpful when studying and modifying a document's structural organization.)

STATEMENT NUMBERS AND NAMES. Optionally, for a given window, show the Statement Number (or the SID) of each statement – with an option for showing them at either the right or at the left margin. Independently, the showing of statement names may be turned on or off.

FROZEN STATEMENTS. A worker may select a number of statements, in random order, and designate them as "frozen." One of the view-specification options is to have the frozen statements appear at the top of the frame, with the rest of that window left for normal viewing and editing. The frozen statements may be edited, or even cross-edited between any other displayed (or addressable) statements.

USER-SPECIFIED CONTENT FILTERS. A simple content-analysis language may be used in a "Set Content Pattern" command, which compiles a little content-checking program. One of the view-specification options will cause the system to display only those statements which satisfy both the structure and level conditions imposed by other viewspecs, and which also pass the content-analysis test applied by this program. Where desired, very sophisticated content-analysis programs may be written, using a full-blown programming language, and placed on call for any user.

USER-SPECIFIED SEQUENCE GENERATORS

In the foregoing, a "view" is created by beginning at a designated location in a document (file) and selecting certain of the the "following" statements for display, according to the viewing parameters – possibly suppressing statements that don't pass the test of a content-analysis program. This is essentially a "parameterized sequence generator," and provides very useful options for selectively viewing statements within a document; however, it works only by selectively discarding statements from a sequence provided in standard order.

Application programmers can provide alternate sequence-generator programs, which any user can invoke in a straightforward manner. In such a case, the apparent structure being presented to the user could be generated from a sequence of candidate statements according to any rules one may invent – and the actual views could be further controlled by the above-described viewspecs for level clipping, truncation, content filtering, etc.

Perhaps the most commonly used, special sequence generator is one that provides an "Include" feature, where specially tagged links embedded in the text will cause their cited passages to be "included" in place of the Include-Link statements, as though they were part of this file. This provision enables arbitrary assemblage of text and formatting directives, from a wide collection of files, to represent a virtual, one-document, super file. For instance, the whole assemblage could be passed to the formatter, by means of a single user action, to generate a composite, photo-typeset document.

TRAVELING THROUGH THE WORKING FILES

An important provision in AUGMENT enables an author to freely "travel around" in his on-line file space to reach a particular "view point" of his choice – i.e., the position within a file from which the system develops the desired form of "view" according to the currently invoked view specifications.

Traveling from one view point to another is accomplished by Jump commands, of which the simplest perhaps is a direct Jump to a statement designated by a screen selection. Then, for a worker grown used to employing address strings, a next form would be a Jump on an embedded link, or to a statement designated by a typed-in address string – using any combination of the addressing elements and viewspecs described above. For example, the link "<4b:mI>" points to the Statement 4b, while invoking viewspecs "m" and "I" which cause the statements' SIDs to be displayed. The link "<Ref-1.l:i;LL>" points to the document referenced by the link in the statement named "Ref-1", invoking viewspec "i" for user content filtering, and sets the filter to "LL" to show only those statements beginning with a lower-case letter. The applications are effectively endless.

MODIFYING THE DOCUMENT STRUCTURES

Given the array of capabilities described above, it is very simple also to provide for very flexible manipulation of the file structure. For operating on a small, basic set of structure-entity nouns, essentially the same basic verbs may be used as for text manipulation – i.e. Insert, Delete, Move, Copy, Replace, and Transpose are quite sufficient for most cases. For instance, "Move Branch 2b (to follow) 3c" immediately moves Statement 2b and all of its substatements to follow Statement 3c – and their statement numbers are automatically changed from 2b, 2b1, etc., to 3d, 3d1, etc.

A few extra verbs are useful for structure manipulation. For instance, a "Break" command will break a given statement off at a designated point in its text string, and establish the rest of the text as a new, separate statement. And an "Append" command does the reverse – i.e., it appends the text of one or more existing statements to the end of a designated statement.

A major source of structure-modification capability derives from the associated "studying" capabilities. For example, if an author can view a file (document) with specifications that show him only one line each of just those statements in the top two levels, he gets an overview of the high-level organization that helps immensely to study his current structure or outline.

Concurrent use of mouse and keyset also provide considerable gains in speed and flexibility for studying and modifying document structure. For example, if when studying the overview described in the previous paragraph, the author perceives that Statement 2b really belongs in Section 3, following Statement 3c, he can execute the necessary move command in a very quick, deft manner:

> Keyset hand strikes "m" and "b" (for Move Branch), while the mouse hand is positioning the cursor anywhere in the text line of Statement 2b. [Two chord strokes.]
>
> The mouse hand depresses the <Select> button on the mouse while the cursor is on Statement 2b, then moves to Statement 3c and depresses it again, and then depresses it again to say, "OK, do it." [Three button pushes, synchronized with the mouse movement as it made two selections on easy, window-wide, whole-line targets.]

(Note: I just had myself timed for this above operation – an unhurried 2.5 seconds.)

In our view, interactive computer support offers an author a priceless opportunity to get away from the geometric bondage inflicted by pages, margins, and lines – things which have very little if any bearing upon the content and organization of one's text. In terms of value to the authoring process, we differ sharply from those who advocate a "What you see is what you get" working mode during the development of a document's content and organization. For this kind of work, experienced users of the foregoing kind of flexible facility for addressing, viewing, and manipulating structured documents, would consider a "What you see ..." mode as a relative handicap.

SUPPORTING MULTI-PARTY COLLABORATION

The support that advanced technology can provide for close collaboration among knowledge workers is a very important and much under-rated possibility. For principle-author activities, collaborative support is an important aspect of system capability. Some years ago, we introduced the following provisions into AUGMENT. (A more complete, overview treatment of these is given in <Ref-8>.)

ELECTRONIC MAIL. Its primary attributes of speed, automatic distribution, and computer-to-computer directness are well recognized – and are generally accepted now as important to the effectiveness of knowledge workers. AUGMENT Mail has features that are beyond what most electronic mail systems offer, and which provide unique benefit to the authorship process.

AUGMENT's mail system allows one to "send" complete, structured documents as well as small messages. In an authorship environment, an important role for "electronic mail" is for the control and distribution of documents -- where small, throw-away messages are considered to be but a special class of document. An author should be able to bundle up any combination of text and graphics, in the forms that he has been using for studying and manipulating them – and send the bundle to other workers. In AUGMENT, such a bundle is just like any other file structure, and can be studied and manipulated, incorporated into other files (documents), saved or deleted.

RECORDED MAIL – AUGMENT's Journal System. When mailing a document, an AUGMENT worker may optionally specify that it be installed as a "recorded" item. In this case, before distributing the item, the system will make a permanent record if it, as a file in a specified Journal collection. And, just as though it had been published, this recorded Journal item cannot later be changed. The system assigns a straightforward accession identifier (a simple number), and any authorized worker is henceforth guaranteed access to that Journal item by specifying the name of the Journal-collection and the Journal-item number – e.g., as specified in the link "<OAD,2237,>".

A given journal may be set up to serve multiple hosts and is much like a special library. It has its collection of documents, and AUGMENT provides associated support processes for entry, cataloging, retrieval, and access.

Together with the linking capability described above, a Journal system provides an extremely effective form of "recorded dialog." Cross-reference links between a succession of Journal items produces an inter-linked network of collaborative contributions – plans, outlines, document drafts, schedules, short comments, detailed critiques, reference material, etc. The on-line worker can follow these links very easily and, using multiple windows and flexible viewing options, can make very effective use of such records.

For instance, consider a detailed commentary directed toward a "preliminary design" document recorded in a given Journal collection. The author writing the commentary could view the design document in one window and his developing commentary document in another. He can easily establish links in his commentary to cite any passage in the design document – e.g., a statement, a term in the statement, or a diagram. Then this author would submit his commentary into the Journal, perhaps specifying a list of colleagues for "distribution." Each listed user would automatically receive a mail item announcing this new Journal entry, giving subject, author, date, etc., and the all-important link to the new Journal file containing the commentary. Any such recipient can subsequently study both the commentary and its cited planning document in a similar, multi-window, link-assisted manner.

Furthermore, this second reader could develop and submit his own recorded commentary, which because of the citation power of AUGMENT links could be as short and to the point as: "Frankly, John, I think your comment in (DDD,xxx,aa) is a mistake! Didn't you notice the earlier assumption in (DDD,xxx,bb)? Maybe you should go back to Tom's earlier requirements document 25-Oct-83 12:07-PDT OAD,2221,
– especially at (EEE,yy,cc)." (Here, "DDD" and "EEE" represent Journal names, "xxx", "yyy", and "zzz"

represent Journal item numbers, and "aa", "bb", and "cc" represent addresses pointing to specific passages in those Journal files.)

In official parlance, "retrieval" is the finding out about the existence of a relevant piece of information, whereas "access" is the subsequent process of gaining possession of the information. For users of AUGMENT's Journal system, retrieval is immensely facilitated by the widespread use of citation links. When one can follow them as easily as can a practiced AUGMENT worker, these links provide extremely effective retrieval support. We have supplemented this with some simple, automatically generated catalog files, which made a rather nice balance. Access is provided by direct Jump on a reference link if the file is on line; if it isn't, AUGMENT asks the worker if she wants it retrieved, and a simple affirmative response automatically launches a request for the system operator to retrieve the file from its archive tape, after which the worker is notified of its availability via electronic mail.

A private document can be submitted into a Journal. In this case, only those workers listed at Journal-entry time can get access to the central copy. Such a private item would not be listed or indexed in the "public" catalogs.

We have used the Journal system very heavily since 1970 to support AUGMENT's development activity; many customers have employed it heavily since 1975. There are about 100,000 entries recorded in the original Journal now (I don't know about other, newer AUGMENT Journal collections). We found that as workers became at home in this environment, they were increasingly free about submitting their items to the "public." It became evident that the scientific tradition of active and open interchange has some solid relevance to the collaborative processes in our smaller, "colleague communities." Time and again a worker would come across others' dialog and be able to contribute some valuable information (sometimes a one-sentence comment with a critical citation link). Often the payoff went the other way: the new party found immediate value in an old piece of recorded dialog.

SHARED-SCREEN TELECONFERENCING. Consider a case where two people sit down to work together at a terminal, where they can both see the screen(s), and where either one can take over the controls. This is being done countless times every day throughout the country, in different combinations of expert-expert, expert-novice, novice-coach, etc. When talking together on their telephones, two or more distantly separated AUGMENT users can collaborate in a manner very similar to this.

Suppose that two workers, Smith and Jones, want to set up and operate in a Shared-Screen Conferencing mode. Smith is in Princeton, working on host Office-4, and Jones is in San Francisco, working on host Office-12 – and both of these host computers are connected to the same network. Assumedly they are in telephone contact when they decide to work in this shared-screen mode to collaborate on Smith's current job.

Jones will enter the command "Share (display with user) SMITH! On host OF12! Viewing (other display)!!"

Smith will enter the command "Share (display with user) JONES! On host OF4! Showing (this display)!!"

To give these commands, each person only entered the characters shown in upper case (entry case actually irrelevant), plus the digits, plus an "OK Key" action where each exclamation point is shown.

Whatever tool that Jones is currently using will continue responding to his controlling actions, as evidenced by various feedback and portrayal actions in the windows on his screen. Smith's screen image will clear, and be replaced with a replica of Jones' screen image – multiple windows and all. For the duration of the shared-screen session, Smith's screen image will continue to replicate what is shown on Jones' screen.

There are provisions for passing control back and forth between workers. For instance, Jones can pass control to Smith so that Smith can show him some material or method of work. There are also provisions for the subsequent entry and departure of other conference participants.

EMBEDDING THE GRAPHIC ILLUSTRATIONS

For complete support of document development, it is important to provide integrated means for developing, viewing, and manipulating graphical portrayals. These portrayals should be part of the working files from the very start, to be studied, passed about in mail, shared in Conferencing mode, edited, captioned, labelled, and moved about within the document structure. Furthermore, active, relevant citation links pointing to these graphical constructs would be installed in and followed from textual passages throughout the associated set of documents (including Mail and Journal documents).

AUGMENT's architecture and file structure were designed for this end, and a good bit of the associated implementation is in place.

A graphical data structure can be attached to any given file node, and there are basic capabilities for composing, studying, and modifying graphical diagrams. When formatting for a suitably equipped phototypesetting device, there are formatting directives to designate the position and scale for placing these diagrams on a page. An AUGMENT file with integrated text and graphics can thus be mapped automatically onto a high-quality document whose pages contain both text and line drawings.

Our goal here was for what we call an "illustrative graphics" capability – basic to which is a command that, when directed toward any conventional "plotter" file, will translate it into a diagram attached to a designated node. In this way we can make use of graphic constructs developed within almost any applications system, most of which have provision for outputting "conventional" plotter files.

The most important next step is to adapt a bit-mapped display as an AUGMENT workstation, so the integrated text and graphics can be viewed and manipulated on the same screen. Heretofore, to do graphic work, an author has had to attach a Tektronix 4014 storage-tube display to the special printer/graphic port of her AUGMENT workstation. This has made use of AUGMENT graphics slow and expensive enough to limit

the number of user groups who have developed the integrated use of mixed text and graphics.

CONCLUSION

AUGMENT's unique provisions stemmed for the most part from the conceptual framework within which AUGMENT was developed. For instance, consider the pervasive and significant changes in the environment in which humans will be doing their knowledge work. Note that the habits, methods, conventions, intuitions, etc., that comprise the "ways" in which we think, work and collaborate, are for the most part products of many centuries of cultural evolution – in a radically different environment. With a radically different environment, this constant process of cultural evolution can be expected to take some radical turns.

The AUGMENT developmental framework assumed that many of these "ways" are candidates now for change in directions that heretofore would not have been beneficial. The AUGMENT system emerged as a first step in considering a few such changes, which perhaps can improve human capability for doing knowledge work because their new "ways" will enable us more effectively to harness the new tools toward more effective basic capability. (This is very different from trying to "automate" our old "ways" of doing things.)

As an example, consider the "What You See Is What You Get" (WYSIWYG) syndrome. It is a highly touted feature for many vendors. It provides a definite advantage for the final process of converting a computer-held document to a nicely formatted hard copy. But what does it do for authorship? Well, in our framework, it has a negative impact. We were happy to abandon those constraints of lines and pages and other formatting geometry which did not contribute to matters of content and structure. We have chosen instead to provide the authorship process with structured files, flexible addressing, flexible window-size viewing, level and truncation viewspecs, etc. – things that would be awkward or impossible to provide in a WYSIWYG environment. This provides the authorship phase with flexibility and power for studying and manipulating content and structure that we wouldn't consider trading off for WYSIWYG. Save it for the production phase.

Here is another bit of culture that deserves reexamination. Consider the dictum, "Easy to learn, and natural to use." Or, "User friendly." The question is, whom are you judging that things will be easy, or natural, or friendly? For designers of craft-work tool systems, very different perceptions of this issue are warranted between a system for the occasional, weekend do-it-yourself person and a system to be heavily used day after day by professionals. The AUGMENT User Interface System enables us easily to configure either kind of a tool collection.

This paper describes part of what is provided to professional knowledge workers who do a significant amount of authorship work. We observe no more difficulty in their learning how to employ this relatively large collection of tools than one would expect for professional woodworkers in their learning about the relatively large collection of chisels and other tools of their trade.

It is a basic part of our framework that, to augment human knowledge workers, attention must be given not only to tools, but to methods and skills as well. Because of space limitations, the scope of this paper was restricted to a summary of those tool provisions within AUGMENT that especially facilitate the authorship process. A full description of "How to use AUGMENT to ..." would definitely need to include methods of work that effectively harness these tool provisions, and the special kinds of skills that yield unique payoff in executing these methods. This is true for every tool system, of course, but it seems especially true in this case because many AUGMENT provisions do not fit into the general cultural background of our authorship process.

Perhaps the best way for very brief summarization of what AUGMENT's users feel about its unique features is simply to say that those who leave its working environment really miss them.

REFERENCES

Ref-1: Engelbart, D. C., "Toward Integrated, Evolutionary Office Automation Systems," Proceedings of the 26th Joint Engineering Management Conference, Denver, CO, Oct. 16-18, 1978, pp. 63-68. (AUGMENT,71279,)

Ref-2: Engelbart, D. C., R. W. Watson and J. C. Norton, "The Augmented Knowledge Workshop," AFIPS Conference Proceedings, Volume 42, pp. 9-21, National Computer Conference, June 4-8, 1973. (AUGMENT,14724,)

Ref-3: Michael, Elizabeth K., Dirk H. van Nouhuys, Beverly R. Boli, Raphael Rom, and Ann C. Weinberg, "Document Production and Control Systems," Phase One report of Document Production and Control Systems Design Study, by the Augmentation Research Center, SRI International, for AF Rome Air Development Center, Contract F30602-76-C-003, March 1, 1977. (AUGMENT,37730,)

Ref-4: Boli, Beverly R., Harvey G Lehtman, Elizabeth K. Michael, Raphael Rom, Dirk H. van Nouhuys, and Nina Zolotow, "A Model Document Production System," Phase Two report of Document Production and Control Systems Design Study, by the Augmentation Research Center, SRI International, for AF Rome Air Development Center, Contract F30602-76-C-003, July 30, 1977. (AUGMENT,29000,)

Ref-5: Engelbart, D. C., "Toward High-Performance Knowledge Workers," OAC '82 Digest (Proceedings of the 1982, AFIPS Office Automation Conference, San Francisco, Ca., April 5-7), pp. 279-290. (AUGMENT, 81010,)

Ref-6: Watson, Richard W., "User Interface Design Issues for a Large Interactive System," AFIPS Conference Proceedings, Volume 45, AFIPS Press, 1976, Montvale, NJ, pp. 357-364. (AUGMENT,27171,)

Ref-7: Engelbart, D. C., "Design Considerations for Knowledge Workshop Terminals," AFIPS Conference Proceedings, Volume 42, pp. 221-227, National Computer Conference, June 4-8, 1973. (AUGMENT,14851,)

Ref-8: Engelbart, D. C., "Collaboration Support Provisions in AUGMENT," OAC '84 Digest (Proceedings of the 1984, AFIPS Office Automation Conference, Los Angeles, Ca., February 20-22). (OAD,2221,)