UNIVERSITY OF WATERLOO COMPUTER SCIENCE DEPARTMENT

Personal Data Structuring in Videotex

Darrell R. Raymond

Data Structuring Group
CS-84-7

February 1984

UNIVERSITY OF WATERLOO COMPUTER SCIENCE DEPARTMENT

Personal Data Structuring in Videotex †

Darrell Ronald Raymond

Data Structuring Group
Department of Computer Science
University of Waterloo

ABSTRACT

Current videotex systems seem inadequate to provide useful and desirable computing services. Two key reasons for this lack are identified. First, videotex systems fail to separate data structure from data content. The integrated nature of videotex databases makes maintenance and information retrieval difficult. Second, videotex systems do not support a personalized approach to services. Users are treated as a captive audience in a monolithic timesharing enterprise, with little capability to alter their computing environment.

Two new ideas are introduced. *Multi-menus* modify current access strategies to allow more personalized database searching. Multi-menus exploit the distinction between content and structure, permitting many variants of one or more structures to be used for the same content. A *structure editor* provides facilities to interrelate videotex pages in any desired way. Such an editor can make videotex a more personal enterprise and at the same time integrate a large collection of services. The structure editor can become the central tool in a videotex knowledge workshop.

A prototype structure editor employing multi-menus was implemented. Experiences with the prototype are described.

---

† Originally a master's thesis presented to the Faculty of Mathematics at the University of Waterloo.

Acknowledgments.

I am indebted to Frank Tompa, who supervised my research. He was a valuable source of ideas and direction, and a most patient editor. I would not have finished this thesis without his continual support and encouragement.

My thanks also to Kelly Booth and Doug Dyment, the members of my thesis committee. They provided the objective view necessary to coalesce my thoughts into a coherent document.

My parents, Elmer and Rita, maintained a loving atmosphere and contributed in other essential ways too numerous to mention. To them I owe the most.

Last but not least, thanks to Lynn and Susan for always believing.

Introduction.

The emergence of generally available information services is a potent technological and social force. Applications are now less dependent on highly trained technicians; computers are more often directly manipulated by end users [Tsichritzis 1981]. Office information systems *(burotique)* meld word processing, electronic mail and database technology into a tool usable by secretaries, accountants and executives [Conrath 1980]. Personal computers and videogames offer increasingly powerful educational, business, and entertainment software to the novice and hobbyist [Malloy 1983]. Two-way television introduces interactive capability to a popular entertainment medium [Wicklein 1979]. 24-hour banking machines let people communicate directly with an electronic teller.

Videotex is one attempt to provide general information services [Bown 1978]. Videotex systems are identifiable by three main characteristics. First, there is an emphasis on *graphics.* Videotex enhances communication by presenting pictures and diagrams [Mills 1981]. A major area of videotex research has been the development of schemes to transmit and display graphics in a simple and inexpensive way [O'Brien 1982]. Second, videotex information is *page oriented.* A page is a collection of text and graphics that can be shown on a display device such as a television set. Pages typically convey a single block of information. Third, the data in a videotex system are structured in a *hierarchy.* Pages of information are typically accessed by selecting options from a hierarchical set of *menus.* Menus serve to partition the pages into pre-determined categories.

Videotex systems can be used to present pages about weather, news, vacations, products, farming, stock prices, or almost any information amenable to classification. Videotex has the potential to support retrieval of highly dynamic information such as financial reports. New interactive services such as home banking and shopping have already been implemented using videotex [Larratt 1980].

Despite its promise, videotex has not been spectacularly successful. Each new information processing technology must compete with the others, complicating attempts to secure a market share [Malloy 1983] [Mayer 1983]. Videotex is no exception. Demand for information processing services is unpredictable; although great possibilities were forseen, they have failed to materialize. On the other hand, new social problems as a result of videotex seem all too real and forboding: "brain clash" [McKay 1983], "invasion of privacy" [Wickelein 1981], and the new "electronic morals" [Mack 1983] [Wynn 1980].

This thesis is based on the conjecture that renewed interest and confidence in videotex will result from a more personalized approach to services. Current videotex systems operate with a "computer center" mentality [Tsichritzis 1981]. A very clear distinction is made between *information providers* and *information consumers.* Information providers create and structure pages. Information consumers can do little more than retrieve these pages from the pre-constructed index. A similar lack of user control occurs in page creation system design [Pickersgill 1981]. Part of the reason for centralization is the need to control costs, but the cheapest system will be overbudget if subscribers are unwilling to use it.

An alternative view was suggested by Douglas Engelbart, among others [Engelbart 1973]. Insofar as they use computers, people are *knowledge workers;* the system and its interface a *knowledge workshop.* The knowledge workshop is a place for managing messages, forms, memos, notes, tables, pictures, diagrams or any other form of data commonly associated with human knowledge. Engelbart's goal was to create a useful and powerful information system supporting the user's knowledge tasks – the *augmented knowledge workshop.* Although much of Engelbart's work is over ten years old, today's videotex systems could profit by attempting to provide *knowledge* rather than *retrieval* services. I suggest that emphasis should switch from homogeneous database systems to more personalized services.

> Once a significant user-oriented system becomes established, with a steady growth of user clientele, there will be natural forces steadily increasing the effectiveness of the system services and steadily decreasing the cost per unit of service. [Engelbart 1973, p. 10]

Part I.

Videotex provides information retrieval and processing for a general audience. Some commercial videotex systems exist, but videotex requires more effective tools for retrieval, structuring and processing to reach its full potential.

The nature of videotex and the problems currently encountered are explored in this section. It is suggested that a key problem is the failure to distinguish between structure and content. The importance of this distinction is emphasized by switching attention from system design to the human cognitive aspects of data retrieval.

1. What is videotex today?

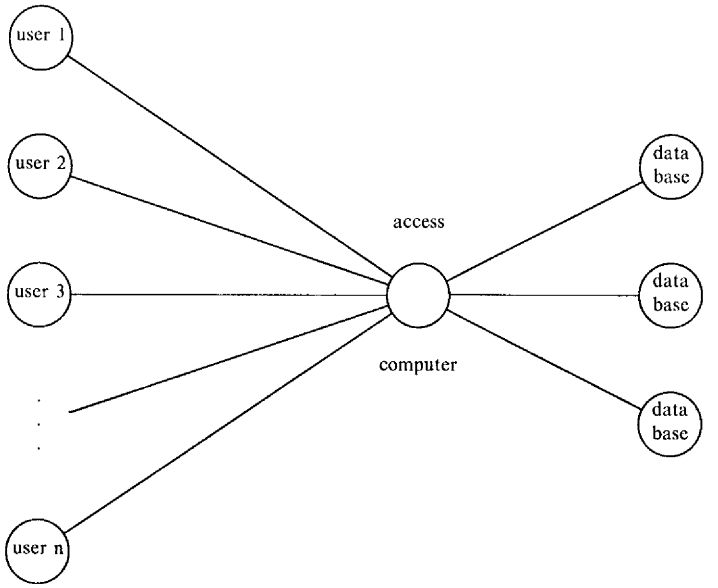

Figure 1.

Figure 1 shows a schematic view of a typical videotex system. A *videotex service* provides information and computing to its *users* or *subscribers*. The service maintains *databases* and *access computers*. Each subscriber has a *display device*, an *input device*, and a *decoder*.

The access computer processes user requests for pages, each of which is a set of display codes kept in a database (or produced dynamically when requested). The codes facilitate rapid and inexpensive transmission. When a page is requested, its codes are fetched from the database by the access computer and sent to the user's decoder. The decoder constructs the page from the codes and presents it on the display device.

Database pages are typically organized as a tree, as shown in Figure 2. Each *node* in the tree is a page. *Arcs* connect pages. A user can reach any page attached by an arc from the current node. Often a node will contain a list of accessible pages, known as a *menu*.

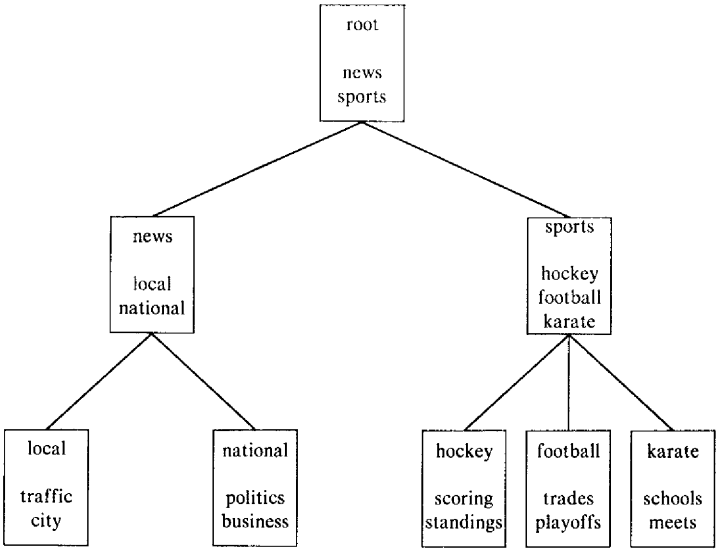

Figure 2.

A user begins a session by looking at the root page of the tree, which presents explanatory material and a menu to further pages. The root menu partitions the available information into several categories, one of which is selected by the user. The access computer determines the subtree the user wants to enter and sends the root page of this subtree for display. This process is repeated until the user finds an interesting page, backtracks towards the root, or terminates the session.

The combination of menus and tree structure is called a *tree index*. Tree indices are simple structures for both the system and user. Menus are simple to use; they require little learning and show the complete set of options [Robertson 1979] [Geller 1983]. Hierarchies are a common data organization, requiring the videotex service to keep only a record of each user's current node. Each user input is interpreted as a request to move to another node of the tree.

A collection of menus provides a *stepwise* approach to information retrieval. Each menu selection can be viewed as a small part of an overall query. As the user traverses the tree, the collection of query fragments constitutes a step-by-step construction of the overall query. This stepwise approach is often termed *navigational* [Lochovsky 1981]. Each step in the tree index is a step in some cognitive

geography.

Traditional record-oriented databases consist of two parts, the *schema* and the *data*. A schema is a pre-determined framework that restricts the data in order to ensure consistency, completeness, and compactness. The data are particular information values structured by the schema. Videotex commonly employs page-oriented databases, which do not require a schema. Information content in videotex is supplied by *information providers*. Although the videotex service may support the information provider's creation of a subtree, no schema or other formalization is enforced (other than ensuring that all pages are connected [Tompa 1981a]). As a result, videotex databases are *integrated;* information content and information structure are intermixed [Schabas 1983a]. For example, display pages containing a menu are almost always pre-constructed, and so are insensitive to the subtrees actually connected to the node. A hopelessly confusing tree index can be created while remaining within formal constraints.

Pages convey a limited amount of information. The typical amount of text shown in North American videotex systems is twenty lines of forty characters. This limitation is partly a result of the attempt to use standard television receivers as display devices.

There are two predominant graphics coding schemes. *Alphamosaic* pages are mapped into a 72 by 80 grid. Images are created by shading collections of squares in the grid. Alphamosaic pages have constant size, because the grid is fixed. Each grid element is shown directly on the display; this means little decoding is necessary, but resolution is limited. Improved display devices will not exhibit higher quality graphics from an existing alphamosaic database.

Alphageometric pages encode not grid elements, but graphic primitives. For example, a circle is represented by a code for "circle" and a set of codes describing its center and radius. An alphageometric page with n circles will have n times as many codes as a page having only one circle, so alphageometric pages vary in size. They require more decoding because the display device must be able to construct the primitives locally, but this makes resolution database-independent. Improved display devices can show a higher quality drawing. However, for either coding scheme the quality of the graphics is bounded by the level of precision chosen at picture creation time.

The most common transmission medium for videotex is the telephone line. Transmission rates are normally in the range of 1200-4800 baud. As a result, pages often take a few seconds to transmit. The amount of time necessary to fetch, transmit, and display a page can be frustrating for users.

Videotex provides services other than simple page retrieval.

Action pages are pages with an associated program. Retrieving an action page shows a new display page and begins execution of the program. Control of the interaction passes to the program; user input is now interpreted as program input rather than requests for new pages. Action pages can be used for playing games or performing other simple activities that can be placed under program control.

A common use of action pages is to carry out simple dialogues, which are often well-structured activities. For example, a banking transaction can be carried out automatically if the user provides a few pertinent facts. Structured dialogues are conducted with *form-filling* pages. A form is a collection of questions with which the system prompts the subscriber for information. Users attempt to fill in the form, and the system provides aid if data is entered incorrectly or incompletely. A completed form is a request for some activity to ensue, such as a financial transaction or a mail order.

More recently, *gateway* pages have been introduced [Fedida 1982] [Otto 1980] [Seguin 1982]. A gateway page is an interface to an external (usually non-videotex) service. The videotex service handles initialization of the external service, and supplies the user with a comfortable interface. Since external services often keep information in a record-oriented database, the interface must include a means to transform system responses into videotex-compatible pages. This can be costly.

2. Problems with videotex systems.

2.1. Size of the database.

Videotex services provide access to large amounts of information; some databases are 100,000 pages or more. The limitations of a single display page mean that many pages are necessary to describe complex topics properly.

Designers can manipulate two dimensions of the tree index to accommodate large databases: *depth* and *breadth* [Sisson 1983]. Breadth measures the number of choices presented on each menu. Depth measures the number of menus that must be looked at to access a leaf page.

The breadth of the tree is limited by two factors. First, the number of menu choices is limited by the amount of information that can be displayed at one time. Each menu choice requires an icon or descriptive phrase, so usually twenty or fewer items will be presented. Second, the number of choices that a user can comfortably compare is small [Schabas 1982]. In order to choose the best item, the user must rank all the choices given. Users more easily rank three choices than twenty. Very large menus are comprehensible only in special environments [Herot 1980].

Since the breadth of the tree is bounded, a growing database must become deeper, exacerbating most other videotex problems. For example, paths to leaf pages become longer. Since user errors become more likely in longer paths, a deeper tree is more likely to frustrate users. Furthermore, the lack of formalism to constrain the tree makes construction errors more likely in larger databases.

Normally, searches terminate when desired information is located. However, not all information needs can be anticipated, and users are often unsure if desired information exists. Rather than search large databases exhaustively, users terminate a search when the *search threshold* has been reached. A search threshold is a limit on the amount of cognitive effort exerted [Telidon 1981]. The threshold may be a certain number of pages or a fixed length of time. If a useful page is not found before threshold is reached, the search is abandoned.

Search threshold places an upper bound on the number of pages accessed. Large databases make it less likely that desired pages will be found, simply because the average path length exceeds the typical search threshold.

2.2. Response time.

Response time is the delay between user selection of a menu item and the display of the corresponding page. The importance of response time is illustrated by the ZOG experience [Newell 1977] [Robertson 1977] [Robertson 1979]. ZOG is an information retrieval system used at Carnegie-Mellon University. It bears a strong similarity to current videotex systems because of its use of a tree index.

ZOG has a shorter response time than most videotex systems: pages (or "frames" as they are known in ZOG) are usually displayed in less than one quarter of a second. An experimental version of ZOG provided responses within one twentieth of a second [Robertson 1977, p 31]. Response is consistently good throughout

a ZOG session. With touch screens, selecting menu items is easy and fast. The combination of these factors makes for extremely rapid navigation. Experienced ZOG users can find their way to known frames almost at touch typing speed.

Instantaneous response is highly desirable. ZOG's creators claim "The unique property of ZOG-like interfaces is the rate of change of visual textual (and pictorial) information under user control" [Robertson 1979, p 38]. A response time of approximately 50ms enables users to respond at a reflexive or *lexical* level [Foley 1974].

Speed is necessary when a tree index is large. Large tree indices mean long paths to data pages; the longer the response time, the more delay in reaching leaf pages. Videotex response time can be degraded by any system component. Access computers and databases are controlled by the videotex service, so their performance can be optimized. However, videotex services hope to deal with thousands of information providers and millions of subscribers. Sheer volume may frustrate efforts to provide rapid page access. Communications services have much potential, but for various reasons may provide only limited performance [Akgun 1981]. The subscriber's own equipment is often the weakest link in the chain. High speed in ZOG requires a powerful microcomputer dedicated to a bit-mapped display [Robertson 1979, p. 29].

A more subtle factor is gateway page response. If external facilities provide significant service, the performance of the videotex system will depend largely on gateway capabilities. Guaranteeing good response time is difficult, as gateway computers and databases are not controlled by the videotex service.

Page complexity can also affect response time. Complex alphageometric pages take longer to transmit than simple ones. Alphamosaic pages are constant in transmission time, but new features such as dynamically re-definable character sets and animation require higher transmission rates [Gecsei 1983].

The difficulty of ensuring high performance from all components simultaneously makes it unlikely that videotex will attain the unique speed of ZOG-like interfaces. Robertson suggests that slow response times degrade a tree index sufficiently to make it undesirable [Robertson 1979]. Rate of information transfer of keyword interfaces is superior to low speed tree indices.

Despite the system's inability to reach ZOG-like speeds, improvements in response time are desirable. ZOG speeds can be approached in a *distributed* browsing environment. The index is broken into sections that are downloaded into the user's workstation. The user traverses quickly to the bottom of the local subtree, and another section is downloaded. High performance is attained during local processing, and only subtree downloading need be delayed. Average response time is improved and communication traffic reduced, but at the expense of more powerful user stations and higher variance in response time.

2.3. Searching.

To *search* a tree index is to look for a specific topic. The "best" item is selected at each menu, and the search terminates when desired information is located. To *browse* a tree is to traverse menus without a specific destination. Menu choices are made on impulse, rather than for optimality. Browsing may be undertaken as part of the searching activity, in order to locate new paths, or just for entertainment.

Searching in a tree index is highly error-prone [Schabas 1981] [Sutherland 1980] [Telidon 1981] [Vrins 1982]. In a strictly hierarchic structure, a single mistake diverts the user to the wrong subtree, and an error-correcting technique is required. Several factors contribute to search error.

Best-fit strategy – traversing a tree index involves choosing the best item from a set [Engel 1983]. A menu is a set of alternatives, only one of which may be chosen. This task must be repeated at each stage of the search. The difficulty of the choice depends on how closely the items in a set are related.

The relatedness of a set of items is not an independent property of the set, but is dependent on the knowledge and experience of the user. Users with expertise in a particular domain usually have a better grasp of the terminology. Terms used in a fuzzy, imprecise way by non-experts have specific and precise meanings to experts. Hence, an index satisfactory for non-experts may be useless for experts, or vice versa [Furnas 1982].

The relatedness of menu items is also dependent on the distance of the menu from the root. Typically, the tree contains abstract categories at the root, and more specific categories near the leaves. At the beginning of a session, general categories can appear equally non-optimal: the user may not think any choice is worthwhile. As the search progresses towards more specific areas, items may appear equally optimal. Users want to select more than one item because several seem useful, but are only allowed one selection at each menu.

An important factor in selecting the best option is the structure of the items on the menu [Young 1982]. For example, all categories may be *disjoint*. Users can confidently select the first class that appears reasonable. More commonly, categories suffer some *overlap*. Extra evaluation is necessary when selecting from overlapping categories. *Multiple biases* organize information in more than one way on the same menu; for example, there may be an alphabetic and a geographic index presented. Users choose not an item, but a method of continuing the search. Finally, there is the "*n-plus-other*" approach: n disjoint options are presented, with an extra category *other* to contain any options not explicitly shown. Each of these structures requires a different search strategy on the part of the user; errors result when users employ an inappropriate strategy.

Faulty expectations – choice of a menu item is not modulated solely by the set presented on a single page. Users often appear not to examine or notice information that seems quite clearly presented [Robertson 1979] [Young 1982]. The explanation is that users treat menu selection not as an isolated problem, but as part of an overall search task. The task includes the history of previous menus,

choices, structures, and optimization strategies. The user's history creates expectations which often include spurious or incorrect data as a byproduct [Owens 1979]. In effect, users fail to read pages information they do not expect to see.

Search paths in videotex databases are likely to reinforce some expectations and repress others. During traversal of a hierarchy, categories are increasingly specialized. The history of previous categories may lead users to infer conclusions that were not intended. For example, a user in a subtree labelled "motorcycles" does not expect an entry "Honda" to lead to information about automobiles. However, the information provider may have included this item as a *cross link* (discussed in Section 2.4) to the automotive pages. Users expecting a refinement of the superordinate category will become confused.

Incorrect construction – information is difficult to locate if the tree is constructed incorrectly. For example, pages with information about canonized saints might be erroneously stored in a subtree accessible only via the menu item *criminals*. Users are unlikely to select this topic when looking for information about canonized saints.

The problem of incorrect construction is difficult because there is no formal technique for avoiding such errors. As noted earlier, the tree is unschematized. Correctness is difficult to guarantee, as each block of information is added without restriction by a formal model. The categories in a videotex tree index are a result of the designer's concept of the information content. The soundness of this reasoning determines the robustness of the index. Nothing about the form of the tree can indicate cognitive errors, because of their subjective nature.

Incorrect user comprehension – the dependence of tree index structure on content leads to user errors as well as construction errors [Telidon 1981]. Category labels may be correct, but misunderstood. A user might choose *entertainment* when looking for sports events, whereas the information provider chose to keep sports events in the subtree *attractions*. Labels may be correct when considered independently, but ambiguous in a menu context. The effect of such an error is the same as if the information provider had incorrectly constructed the tree.

User comprehension of menus can be improved by collecting statistics on term usage [Sutherland 1980]. Experience shows that the amount of improvement is useful, but search performance is still poor. For many categories, there is no term that will satisfy a majority of users [Furnas 1982].

Incorrect construction and incorrect user comprehension are two facets of a communication breakdown to be explored further in Section 4.

Dispersed categories – the user's task is more involved when desired information is dispersed in the tree. Dispersion commonly occurs when the index is split into information provider subtrees [Sutherland 1980]. Figure 3 shows a hypothetical tree index. A user looking for *audio tapes* finds only an index to audio companies. The desired information is available, but dispersed throughout the tree. If the user is unfamiliar with the companies, retrieval of all information will be difficult. Each company may use a different indexing technique within its subtree, compounding the problem.

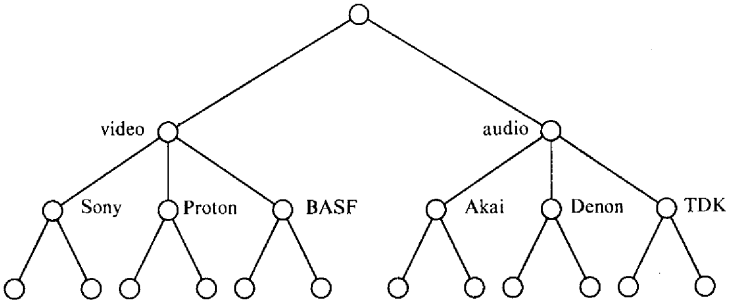

Figure 3.

Dispersion can occur whenever the number of categories provided is much less than the number of combinations of pages. Given n records, there are 2^n possible combinations. Each combination can be considered a different category. A strictly hierarchical tree index with k branches at each node has as many categories as leaves and nodes, or $\frac{kn-1}{k-1}$ categories. The number of categories is greatest for a branching factor of $k=2$, but is never greater than $2n$. At least half of these categories contain only a single record. No doubt many of the 2^n combinations are non-intuitive or even useless; yet $2n$ or fewer categories seems insufficient. A tradeoff must be found.

Dispersion becomes even more difficult when gateway databases are taken into account. The videotex service cannot be expected to provide comprehensive indices into the gateways, and so much useful information may never be examined.

Non-hierarchic information – dispersion reflects the fact that more than one hierarchy may be suitable for a set of pages. Some information is not properly represented by any hierarchy [Furnas 1981]. Non-hierarchic information "forced" into a hierarchic structure is difficult to retrieve. For example, colour is more properly dealt with as a function of three parameters than a hierarchy of possible values [Landauer 1982]. This is partly due to the physiology of colour perception. Sensory data generally seem to be complex, high-level concepts that do not lend themselves well to hierarchic partitioning. Art is also difficult to structure hierarchically. Works of art may be classified according to artist or time period, but only if these facts are known for most of the information. Such an index classifies external phenomena rather than the work itself, which might be better represented by compositional forces [Arnheim 1974].

2.4. Error recovery.

What happens when the user has made a mistake and knows it? Current systems provide one of two options.

Restart – the user can start again. Psychologically, this involves an admission of defeat. Restarting also enforces a repeat of the path from the root to the page with the suspected error. Recall that the number of pages examined by a user is limited by search threshold. Since restarting involves looking at several pages more than once, the number of distinct pages seen by the user (and thus the amount of *effective* searching) is reduced.

Restart is a cheap error recovery technique to implement, frequently achieved by providing a special button. It is most useful when the user believes the error occurred early in the search path.

Back up – if an error is suspected shortly after having been made, a possible solution is to retrace the steps taken since the error. After retreating, the user can attempt to make the correct choice. A record of the path may be made automatically [Engel 1983] [Feiner 1982]. When the user requires error correction, a menu of previous positions is presented. Alternatively, the user can determine search milestones and interactively record them. The MUPID system permits the user to save relevant menu pages on a stack [Maurer 1982]. At error correction time, the stack pages are popped off until a useful position is recovered.

A related problem is that of travelling backwards as part of the search task. Backwards travel involves a switch in context. A tree index is semantically designed to be travelled from root to leaves. Although the logical design of a strict hierarchy causes no problems when backing up (there is only one path to the root) videotex tree indices are rarely strictly hierarchic. *Cross links* are non-hierarchic links included to facilitate searching. When travelling backwards, should the user be forced to retrace the forward path? If the user followed a cross link, can the normal hierarchic route be suggested? Some evidence of possible conceptual problems can be found in a system maintaining a history of the user's activity [Engel 1983]. Initially, all moves were recorded, including the error-correcting ones. This proved unsuccessful. Users distinguished between making choices (moving forward) and correcting errors (moving backward). The conceptual split was so strong that the history mechanism had to be disconnected during error correction.

2.5. The navigational model.

Navigation is a favourite model of index designers [Lochovksy 1981] [Engel 1983]. In many ways the model is inevitable (for example, travel is implied by the phrase "going down the tree"), and is often exploited in an attempt to make videotex easier to use.

Successful implementations of the navigational model include Spatial Management of Data [Herot 1980] and Dataland [Negroponte 1979]. Users are presented with a pictorial *geography* representing the database. Important features in the geography are *landmarks*. Searching is done by *travelling*, with the system as *vehicle*.

SMD and Dataland possess several features whose absence may invalidate the use of a navigational model in videotex.

SMD and Dataland both rely highly on direct manipulation, a technique which provides the user with simulations of familiar data objects [Schneiderman 1983]. Operations are specified in a language that closely resembles how the physical object would be treated. For example, a direct manipulation mail system might require a user to drop a letter icon into a simulated mailbox in order to send mail.

In SMD and Dataland, users view high resolution graphics screens, controlling input with joysticks and special function controls. They are encouraged to think of the screen as an "electronic desk", about which they can move to examine information. In Dataland, motion produces a blurred screen, and even page flipping is faithfully simulated (these details not only maintain verisimilitude, but conceal system delay during preparation of the next screen). Multiple screens are used to present several levels of detail. Objects are simulated physical entities, such as pieces of paper and files. Each object is represented by a small iconic replica. Users can "zoom" in on an object to obtain more detail. Much of the system's appeal lies in the realism of its feedback mechanisms.

Simulation of reality implies a strict hierarchy of data objects. A desk may contain a piece of paper, but the opposite cannot occur. Nor can objects contain one another simultaneously. The hierarchy is small. A desk contains a file, and the file contains forms, but this is likely to be the extent of the nesting. In part, a deep hierarchy is unnecessary because the breadth of the index is large. The desk is a menu with tens or hundreds of items, organized and remembered by their location. The data's geographic position serves as its index.

Videotex does not have the capabilities necessary to exploit the navigational model fully. Simple graphics can be provided, but not the sophisticated feedback of SMD or Dataland. Multiple screens are too expensive for most users, and the realism of the system is affected by the lack of zooming, motion, and resolution.

A physical geography may not be the best representation for videotex tree indices. SMD and Dataland implement specific databases in which a strict and small hierarchy can be observed. Videotex tree indices, as already noted, are large, deal with many diverse environments, and often include cross links and other non-hierarchic features. Restricting the tree index to a strict hierarchy increases the dispersion problem.

The use of a navigational model is a convention more suited to abstract thinking than a technique useful for casual users. Users *can* visualize themselves "moving in information space", but how helpful is this analogue if environment feedback is poor? Perhaps the poverty of the navigational model is best illustrated by the frequency with which users get lost.

2.6. Getting lost.

Users who become lost are an expected consequence of a navigational model. People become lost in their own city; it is not surprising that they should become lost in an unknown information net. ZOG's designers call this "probably the most important phenomenon to understand in psychological terms." [Robertson 1979, p 34].

Users are lost when they are surrounded by unfamiliar landmarks. Physical landmarks have permanence and are recognized by a large community, but conceptual landmarks are rarely so well-defined.. Categories such as *entertainment, attractions, events, shows, what-to-do,* and *activities* seem precise until all of them are placed on the same menu. None of these has a permanent quality; consensus about meaning is hard to obtain. All are recognized as slightly different, but what constitutes the difference is a matter of opinion. Even less well-defined are the catchall categories *miscellaneous* and *other.*

Compensating for this loss of consensus is the designer's power to create a geography. Index structure will be based on the information to be catalogued and system access techniques, but much is left to the designer's landscaping ability. Landscaping involves graphical display, but more importantly the choice of categories. This capability is often ignored. Index designers like to make indices symmetric and bland. Menus have uniform, unimaginative entries. The number of items per menu is chosen to make the tree as complete and bushy as possible. A single technique (usually typing a number) is used to select items. All items are represented by a single type of descriptor, typically key words or phrases.

Symmetry is understandable. Index designers want to maximize use of the screen and make indices neat and orderly. A regular technique gives an impression of useful structure, but this is illusory. By choosing an antiseptic interface, the designer gives the user a sterile, featureless geography. Users get lost as quickly as they would in a desert.

Maps have been suggested as an aid to the lost in a navigational index [Lochovsky 1981] [Mills 1981]. A map displays important index landmarks in one of two ways. A *mimetic* map reproduces the environment as accurately as possible, using sophisticated feedback to reinforce relationships [Mills 1981]. Aircraft simulators implement mimetic maps. The need for complex interface tools makes mimetic maps unlikely for videotex, as discussed in Section 2.5. A *schematic* map distorts the "real" environment in order to emphasize functional detail, much as a person's internal representation does [Loftus 1983] [Moar 1983]. Schematic maps use simple icons and line drawings, and are quite possible in videotex.

Effective use of maps requires proper learning processes by the user [Thorndyke 1980]. Videotex maps can be highly interactive, pointing out a user's location, and perhaps suggesting a direction to proceed. However, thinking is the prerequisite of the user, not the system. Location finding is a skill. Good woodsmen are much better at keeping track of their location than determining what it is when disoriented [Fry 1981]. This requires effort before becoming lost; users must mentally orient and direct themselves.

2.7. Information dynamics.

A dynamic information system is one in which information content and structure change substantially. The problems discussed so far are compounded when information is highly dynamic.

Initially, there is the question of correctness. Information must be highly reliable, or else the system will not be trusted. No one wants an incorrect stock quotation. Second, information often has to be specially formatted for videotex pages. For example, news wire stories may require extensive real-time editing in order to preserve the dynamics of the information flow.

A more difficult problem is classification. Information collections should be standardised, relevant, and timely, but these goals are difficult to achieve when information flow is high. Standardisation implies that information classifications should remain as constant as possible. Timeliness suggests that classification should occur as soon as possible. On the other hand, relevance of the information is often best judged in hindsight, indicating that classification should be put off as long as possible.

In videotex, the responsibility for information classification is divided. Highly dynamic information may be provided by several gateway systems as well as videotex information providers. The extra levels of bureaucracy make effective response difficult.

A related problem is making new information widely known to those who are interested — *advertising* new pages. Users should not need to browse a familiar part of the tree to look for new information, especially if costs are on a "per page retrieved" basis. Current systems provide a *what's new* category on the root menu, but this is insufficient for large databases. Most people are interested in only a small fraction of new information; some way must be found to make them aware of pertinent pages.

2.8. Interface restrictions.

Improvements in interface design are constrained by several factors. Characterization of the user group is the main problem [Schabas 1983a]. The needs and capabilities of this group are largely unknown, but some conjectures can be made. It is unreasonable to expect a computer-literate population, previous experience, or even a learning curve. Any videotex tool has as audience a substantial number of casual (although not necessarily naive) people. Tools will be largely self-taught and are likely to be easily forgotten. However, experienced and capable users will have access to the tools, so their needs must be supported as well.

Interface design is complicated by the evolutionary nature of videotex systems. Record-oriented database designs require much preliminary work but emerge as complete systems. Videotex has yet to find a role or user group; the competition with other public information technologies results in a tendency to encompass many services and facilities. Since people's desires are not "designed" but evolutionary, new services become system add-ons. Users are subjected to different protocols for mail systems, gateway pages, page editors, and various retrieval tools. The result is

a cluttered interface, which becomes worse with each new facility. A particularly annoying example of this problem is the species of videotex keypad with *two* sets of numeric keys; one set is valid only for page selection, one for form-filling.

Advanced input devices often improve interfaces. Trackballs, lightpens, joysticks, touch screens, and pucks have been used in the past to enhance interfaces [Jones 1983] [Manuel 1982] [Smith 1982]. However, the population using these items has been small and select. Even videogames are patronized largely by the young. In widespread use, pointing devices can cause problems. Touch screens are erratic when used in poor conditions; mice require a desktop; joysticks can malfunction. System designers must accommodate users with little eye-hand coordination. Potential users of videotex include the elderly and disabled [Staisey 1982], who encounter problems even with current alphanumeric interfaces [Rex 1983].

3. Enhancing the tree index.

Several attempts have been made to alleviate the problems of the basic tree index.

3.1. Labels and lists.

The tedium of using a tree index to retrieve known pages has encouraged new access techniques. A common method is that of *marking* pages for later retrieval.

Some videotex systems provide *labels*, alphanumeric names that can be attached to pages [Orsnaes 1982] [Tompa 1982]. Instead of using the tree index, the user can enter a label and display the appropriate page directly. Labels can be *universal* or *personal*. Universal labels are available to any user, and are attached to pages of general interest. For example, a page describing tomorrow's weather might be labelled *forecast*. Personal labels are available only to the user who defines them. These labels provide direct access to pages of personal interest.

The CBS database provides universal labels and an alternative to personal labels, known as FAST TRACK℠ [Santo 1983]. FAST TRACK℠ maintains two lists of pages for each user. The user can append pages to either list during a browsing session. Each list can be displayed without accessing the tree index. In effect, the lists are sets of labelled pages with a defined order of presentation.

Lists and labels are attempts to provide some user structuring of pages in a videotex environment. In both cases, pages must be located initially with the tree index.

3.2. Keyword search.

A related tactic to improve page access is the use of *keywords* [Bochmann 1982]. Keywords are page descriptors that provide associative retrieval. Rather than following a pre-arranged sequence of menus, the user enters a set of words describing the desired pages. Pages labelled with the keywords are then retrieved and presented.

Keyword systems can be pre- or post-coordinate [Ball 1981]. Pre-coordinate systems are so named because pages are categorized before retrieval. Each category is described with a short label; the set of labels is the set of valid keywords. Post-coordinate systems perform their categorization at retrieval time. Pages are individually described by a set of single words. These words enter the index, and categories are dynamically determined using set operations. For example, a user might enter *French cars*. In a pre-coordinate system, pages in this category would be retrieved. In a post-coordinate system, the intersection of the two sets of pages (*French* and *cars*) is presented. There is an implicit Boolean "and" operation implied in this post-coordinate query; other set operations can be specified.

Keywords provide a direct path to known pages, as can any labelling system. Keywords have been suggested as a alternative to menus when accessing known data [Geller 1983], and simple experiments seem to confirm this view

[Stewart 1980] [Sutherland 1980]. Typically, though, keyword experiments are performed on a system that also has a tree index. The keyword facility only provides quick access to a few critical points in the tree.

Would a complete keyword system be preferable to an conventional tree index? Information theory states that selecting a single item from n options requires $log\ n$ bits of information. The most compact way to encode such a selection is with a complete binary tree containing the n options at the leaves. Each branch of the tree is encoded as "0" or "1"; the encoded option is a binary number. Use of keywords usually requires more input; the redundant information (above the information theoretic bound) is an aid to the user's memory. Keyword testing usually sidesteps this issue by simply enhancing the basic tree; in effect, the index encoded by the keywords provides less than n options.

Keywords, labels and menus are variants of a single technique that might be referred to as *selection*. Selection is the process of choosing from a finite, predefined subset of items. Conventional menus are explicit, restricted lists of short tokens. Keywords and labels are longer tokens on a single implicit menu [Tompa 1982]. The virtue of keywords and labels is that the redundant information helps the user to relate data and query. Users are more likely to remember the weather page as *forecast* than as the set of menu choices 1, 4, and 9. Whether such simple examples can be safely generalized is unclear, as discussed in Section 4.

Reduction of semantic error entails some increase in syntactic error. Users are more likely to make lexical errors when entering keywords than when entering menu choices, so a successful keyword system would extract meaning from constructions with minor syntactic errors. Techniques to extract meaning from lexically incorrect constructs are described elsewhere [Johnson 1983].

3.3. Cross links.

Cross links were an early attempt at extending the rigid hierarchy. A cross link is a menu item not leading to a direct subordinate. Figure 4 shows a tree index with three cross links.

A cross link is a relaxation of the requirement that a single path exist from the root to any node. Typically cross links occur at leaves, to keep users from entering a dead end. If information providers suspect that users will err on a menu, cross links can be added as an "escape hatch". Recall the integrated nature of videotex systems. The lack of a formalism for the tree means that cross links are included on an ad hoc basis. Structured tree creation can formalize the process, but it is still dependent on human categorization decisions [Schabas 1983a]. Information providers place cross links based on the content of the pages. As with any menu selection, the destination of a cross link should be conceptually close to its source.

Cross links reduce the hierarchic nature of the tree and make maintenance more difficult. In order to update a page, users must be prevented from accessing it (or they may receive incorrect or incomplete information). Access is prevented

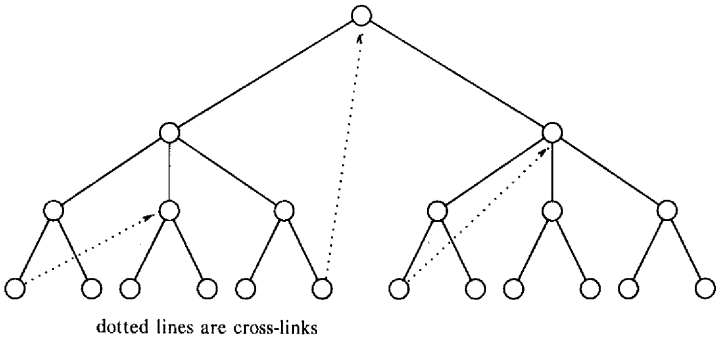

Figure 4.

by *locking* the access paths. In a conventional tree index, only the parent of an updated page need be locked. When cross links are present, each cross link source page that has the update page as destination must be located and locked. It may be necessary to lock subtrees of the page, as users could enter them via other cross links and find it difficult to retreat hierarchically. Locking and unlocking is expensive not only in computation, but also in terms of the reduction of pages available to users.

Traversing a cross link can involve context problems for the user. Traversal along the hierarchic links of the database is reinforced by the repeated refinement of the search category. Hierarchic search paths have continuity, but when a user traverses a cross-link this orderly progression is altered. A cross link destination need not remain in the same context. As described in Section 2.4, this may cause problems when trying to recover from errors.

3.4. Multiple contexts.

Adding cross links individually is tedious. Since no cross link is related to any other, nor required formally by the tree, an inconsistent or incomplete structure can be created. *Multiple contexts* are an advanced form of cross link that reduce these problems [Schabas 1983a] [Yhap 1983]. A context is a set of cross links collected into a complete system. Contexts were originally designed as a way of imposing multiple hierarchies on a single set of data pages.

Figure 5 shows an index with two contexts. Each tree is a context representing the same set of data pages. Thus, one tree represents *Nationality*. To find French pastry, one would look under *French*, then *Food*, then *Desserts*. Another

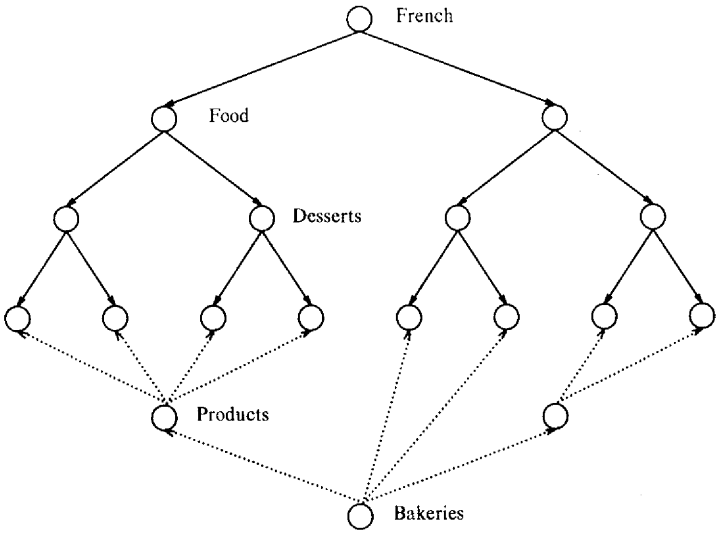

Figure 5.

tree represents *Business* — the same page is obtained by looking for *Bakeries*, then *Products*. The semantic structure helps both users and information providers organize information. Each context is represented by a tree, helping to ensure completeness and consistency.

Multiple contexts dis-integrate data and structure. By providing several paths to the data pages, the designers tacitly state that no single index is preferable. Multiple paths to data can be exploited when several context trees interact [Yhap 1982]. Multiple contexts are less confusing than an equivalent magnitude of cross links. Each context contains a restricted set of interconnections. This may provide useful landscaping but can re-introduce the problem in a more general form. Users may forget their current context. A *loss of context* is temporarily paralyzing, causing extra steps to be taken in the search path [Schabas 1983b].

Perhaps a more serious problem is that addition of cross links or contexts converts the original tree into a network. Networks demand more flexible searching tools than standard menus for at least two reasons. First, searching several paths in parallel is important. A network may not subdivide categories hierarchically, and so a hierarchical search strategy may not lead to the desired page. Second, the user is

more dependent on context for location, and so a more informative view of the network's immediate geography is mandatory. A current multiple context system requires advanced menus to handle searching [Schabas 1983a]. These menus include a description of the surrounding area and a means of changing context, but no parallel search capability.

Multiple contexts have encouraged experiments with menus. Tree index menus are restricted in several ways that might be profitably exploited. Horizontal movement is suggested by [Schabas 1983a]; movement towards the root is described in [Furnas 1981]. Other restrictions can also be relaxed, and these will be examined in Part II.

4. The problem in general.

Several attempts to improve or replace tree indices have been described, but no clearly desirable alternative has emerged. This suggests that the corrective is not as simple as careful index construction or improved response time [Schabas 1981]. Rather, the problems resemble those endemic to other devices used for information retrieval. Videotex systems, insofar as they are used for page access, function as *retrieval machines.* Relational databases, library card catalogues, the *Yellow Pages,* and maps are also examples of retrieval machines. The job of a retrieval machine is to provide information that a user needs to solve a problem. Successful retrieval machines must communicate effectively with people. Retrieval machines are *cognitive prostheses:* assistants capable of enhancing cognitive activity [Landauer 1982] [Robertson 1979]. Their value is dependent on the ease with which we can direct them.

The characteristics of a good cognitive prosthesis are not yet completely understood, as human factors research in computer science is still in its infancy. Some simple observations can be made, however, about inherent limitations in current retrieval machines, and how these cause difficulty. These limitations affect videotex as well.

4.1. The need for multiple paths to data.

A *path* is a query that results in a subset of the data. "Path" is not a notion restricted to navigational machines. A path in a tree index is a sequence of menu choices. A path in a relational database is a set of attributes and attribute values. A path in a post-coordinate keyword system is a set of keywords and Boolean operators.

Most retrieval machines provide few paths to data. Often, much of the data can only be reached with one path. Multiple paths are expensive both in storage and maintenance, but these efficiency considerations conflict with required functionality.

As retrieval machines evolve, designers include multiple paths in response to user demands. An increase in the number of users and diversity of their needs forces inclusion of multiple paths. Libraries provide author, subject, and title classification, as well as cross-references. Full text searching permits retrieval by any phrase that occurs in the document. Multiple lexical paths make documents easier to obtain [Gonnet 1983]. Cross links and multiple contexts increase the number of paths without changing the amount of data. Lists and labelling systems provide additional, shorter paths. Tests comparing various techniques often suggest that a combination of several is best, ensuring multiple paths [Stewart 1980] [Sutherland 1980].

Multiple paths are useful in large, public databases where many needs must be served. But multiple paths are also valuable in a single user system [Bush 1945]. Multiple paths reflect the view that data has multiple interconnections, a view not supported by most retrieval machines [Goffman 1968]. Most machines partition the database in a binary fashion: records either satisfy a given

query or do not. There is no connection between documents separated by this partition, and often little between documents within each subset of the partition. Goffman suggests that a query should produce a *document ranking* which describes the relevance of each document to the query. All documents are relevant to a query, and this relevance can be quantified. In order to produce such a ranking, the relevance of each document to all queries must be known. This is equivalent to knowing the relationship between documents. Further, Goffman suggests that the value of a document depends on a user's previous knowledge. Some documents are useless unless they are preceeded by others; then they become quite relevant. Part of the function of a citation list for a journal article is to explain the relationships to other articles.

Not only should multiple paths be provided, they must be easy to find and use. A single index has been found insufficient in library use to locate all information on a given topic [Albright 1979]. *Intellectual effort* is the energy expended to find a new path in a retrieval machine. Albright notes that the amount of intellectual effort necessary to locate a substantial subset of available information with a single index is much higher than users would typically invest. Albright's result is related to threshold as discovered in videotex search.

The human mind maintains a vast collection of information that is highly interconnected [Landauer 1982]. This is partly a result of the learning process, which depends on integrating new information with old [Clifton 1981] [deBono 1977] [Loftus 1979] [Rubin 1980]. Integration is the forging of links between old and new concepts, increasing the number of paths. A useful model of human cognition is the *spreading activation* model [Collins 1975]. In this model, human semantic memory is seen as a network in which nodes are concepts and links signify relatedness. Input causes activation to spread out from central excited nodes. Processing occurs as activation spreads along multiple paths in the network. Phrases are remembered or understood when activation streams intersect. In this model, comprehension and retrieval is impossible without multiple paths.

Videotex systems provide very few paths to information. A hierarchic structure has but a single path to data. In addition, the notion of multiple paths requires a recognition of the distinction between content and structure, one that videotex services fail to make. The result is a data organization that ignores the human cognitive need for multiple paths; a highly impersonalized structure.

4.2. The difficulty of telling a machine what you want.

Probably the most discussed problem in human factors studies is the gap in communications between people and machines [Carroll 1982] [Fitter 1979] [Mack 1983] [Nickerson 1982]. Some see the solution in a more adequate demonstration of the retrieval machine's rigid needs. "Office-by-Example" provides a template of the machine's categories: a user constructs a query by selecting attributes [Zloof 1982]. RABBIT displays a prototype of the answer set when queried: users modify this example to improve the query [Williams 1982]. SMDS uses graphics techniques to demonstrate the data dictionary pictorially: items in the dictionary

are selected by pointing [Herot 1980]. Full text searching accepts literally any input and produces a result [Gonnet 1983]. All these techniques improve the user's chance of producing a *valid* query. In order to produce a *useful* query, the user's needs must be properly anticipated.

Communicating need can be seen as a problem in two parts. The first difficulty is in stating precisely what one wants. Given a well-defined need, can it be effectively verbalised? For example, Furnas has studied the ambiguity of keyword specification of objects and operations [Furnas 1982]. The *vocabulary problem* is

> a severe and fundamental lack of consensus in the language community on what to call things. [Furnas 1982, p. 20]

and

> For a given user population, with a given degree of training, a certain amount of disagreement in word usage will occur. Even with perfect knowledge of the probabilities describing this usage, we could not predict individual referents perfectly. [Furnas 1982, p. 80]

The popular television gameshow *Family Feud* is based on uncertainty in category specification. Contestants are required to guess the most likely members of a category described by a brief phrase ("Name something one keeps under one's pillow"). The fun of the game revolves around the fact that several members may be considered equally likely representatives of the category in question.

There is an uncertainty principle in keyword use. People talking about well-known objects use different words and phrases to describe them. The set of words used by a large group may be completely known, but the system will probably fail when attempting to interpret an individual user's input. In part, this is a result of the large number of terms used to describe things:

> The unfortunate truth is that the majority of words used are 'rare' words, and so it is a mistake not to reckon with them. [Furnas 1982, p. 88]

and

> Designers are likely to think of names that few other people think of, not because they are perverse or stupid, but because everyone thinks of names that few others think of. [Furnas 1982, p. 85]

Even if common words for objects do exist, the advantages gained by using them may be insignificant. At least two studies suggest that the use of random or unfamiliar descriptors can have some value [De Leon 1983] [Landauer 1983].

A second difficulty of communicating a user's need is that very often users do not know what they want. Belkin describes the plight of the user in an "anomalous state of knowledge" [Belkin 1980]. Far from being able to elucidate a request, the user has only a vague apprehension that information is necessary. The user may not know what information is needed, or how to go about obtaining it. For Belkin, a more important part of information retrieval than document storage is *need representation.* Need can often be satisfied with suggestion, but this area has had little attention. Database research traditionally concentrates on the processing of well-defined queries. Few systems can answer the question, "what do you suggest?" even if this is the most important piece of data that could be supplied. Suggestion need not be complex; even ridiculous paths can be helpful [deBono 1977]. *Brainstorming* is a process in which "nonsense" suggestions lead to a relaxation of implicit and unnecessary problem constraints. One toy currently available aids lateral thinking by "randomly" choosing a word from among thousands. The user attempts to integrate this word with the problem situation, and in so doing, explores new solution paths.

4.3. Categorization.

Categorization is the process of defining things. Machine categories are almost always reducible to a *featural* definition [Kent 1981]. An object belongs in category X if and only if it possesses a set of characteristics, each necessary to objects in X. The advantage of the featural technique is that it is succinct and easy to learn and to communicate. Its disadvantage is the rigidity of the specification. Some categories may not be specifiable in a featural way (try to give a useful featural definition of humanity).

People categorize differently than machines. An example is *polymorphous* categorization [Hampton 1981]. Polymorphous categories contain objects possessing a sufficient number of features, none of which is necessary for membership. Fruit is sweet and has a seed inside (among other things), but lemons are sour and strawberries have seeds on the outside.

Another categorization technique is that of *prototyping.* An object belongs to a prototypical category if it is "close" to some distinguished member of the category. Prototypes are commonly used to identify superordinate categories [Landauer 1982]. For example, given the task of communicating the category "motorcycle" to someone else, a person might use a description such as "Harley-Davidson, Suzuki, and Honda".

Polymorphous and prototypical categories are more flexible and general than featural categories. They are also difficult to formalize, and so constitute a fundamental barrier in retrieval machine use. Machines deal only with signs; a different sign means a different object, unless the machine has been explicitly instructed otherwise. People also deal in signs, but more importantly, in real objects. A person's names for categories need be neither consistent nor correct, so long as the referent is plain. Virtually everyone recognizes "motorcycle" when given the prototypes above — except retrieval machines.

A path is a query that results in some subset of the data. A category is a subset of data, and so a path results in a category. People have many categories for the same information; and thus many paths. On the other hand, machines have very few categories, and these are rigid and inflexible. The difficulty of telling a machine what you want arises when it does not have a category similar to yours. Koll refers to this problem as the "mismatch in concept spaces" [Koll 1979].

Categorization is also a problem when attempting to define a framework for information. People have highly interconnected cognitive organizations with few superordinates or subordinates [Landauer 1982]. The fluidity of human categorization techniques means that almost any concept can be considered a subordinate of more than one category. The value of any public index is not an objective quantity independent of commonly applied structuring methods. For example, The New Encyclopedia Britannica chose to divide its articles into ten broad categories:

1. Matter and energy
2. The earth
3. Life on earth
4. Human life
5. Human society
6. Art
7. Technology
8. Religion
9. The history of mankind
10. The branches of knowledge

The ten categories are ordered so that the questions proposed in each category suggest its successor. This ordering is certainly questionable; does "Religion" suggest "Technology"? The problem of creating general categories is tightly connected to the problem of defining or explaining experience. The Britannica's editors decided that some framework was better than none:

> All that was hoped for, and what was achieved, was the construction of a workable and defensible outline, one that, without contentiousness, would set forth in some orderly way the major topical rubrics that must be dealt with in a general encyclopedia.
> [Britannica 1983, Preface to the Propaedia, *xvi*]

4.4. Implications.

Improvements to existing retrieval machines are usually based on empirical tests. Several machines are proposed and implemented. Then a sample group of users performs a task using both machines, and some measure of the work performed is taken. The number of pages looked at or the time taken to execute a search task are commonly used measures in videotex experiments [Telidon 1981].

Empirical measurements are useful and illuminating, but it is also valuable to know *why* a certain test produced the results it did. This requires a model in which retrieval machines can be compared. The observations suggested in this section are the beginning of such a model.

The separation of content and structure has pragmatic value for system maintenance and database consistency. This separation would also allow designers to support the important cognitive need for multiple paths. Existing information can be organized into many categories: the value of the database depends heavily on the choice of structure. Hierarchic structure is more convenient for the videotex service than for the users [Tompa 1981b]. The restrictions enforced by any one particular structure may not be compatible with a user's needs or opinions, so multiple structures are necessary. Such a conclusion cannot be drawn from purely formal considerations.

Problems in communication and category mismatch are not completely solved by lucid explanation of the machine's categories. The diversity of the user population, the size of the database, and the dynamics of information all contribute to the need for flexible and responsive categorization. User surveys are needed to create useful categories, but the videotex population is neither static nor well-known. System-wide categorization decisions are inevitably poor compromises. A possible solution is to permit user definition and creation of categories — a personalized approach to videotex data structures.

Personalized tools are considered more closely in Part II.

Part II.

The problems discussed in Part I lead to two suggestions.

Videotex has need of access techniques that provide more paths and more flexible categorization. Improvements should retain videotex's unique browsing capability or supply an alternative.

Information retrieval is not the sole function of a videotex system. Information services include processing, structuring, and communication. Videotex currently supplies these services in a disjoint fashion. A more integrated approach should be taken.

Part II discusses two ideas. The first is a menu enhancement that supports multiple paths and a separation of structure and content. The second is an editor for personal information structuring. The combination of these is a tool to manage personal and public information collections — a first step toward the augmented knowledge workshop in videotex.

5. Multi-menus for personalized searching.

Conventional videotex menus are non-personalized searching tools that permit the integration of database content and structure. In particular, videotex menus are usually static display pages which must be changed manually when the content of the database changes. Content independent menus have been suggested (e.g. [Yhap 1983]) but more personalized menus can be considered. These are called *multi-menus*. Before considering the relative advantages of multi-menus, an example of a browsing session using multi-menus will be helpful. Consider the following set of menus.

Videotex Information

Choose one of the following:

1. Cancer Heart Diabetes Campaign
2. Toronto information
3. Ontario information

This is the root menu of a conventional videotex database. Some of the submenus look like this:

Cancer/Heart	*Toronto information*	*Ontario information*
1. Cancer	*1. Attractions*	*1. Ministry of Agriculture*
2. Heart	*2. Entertainment*	*2. Bookstore*
3. Diabetes	*3. Retail*	*3. Ministry of Transport*
	4. Accommodation	*4. Ministry of Tourism*
	5. Nightlife	
	6. Sports	

Entertainment	*Attractions*	*Music and concerts*
1. Art galleries	*1. Special sites*	*1. Kingswood Music Theatre*
2. Music and concerts	*2. Historic sites*	*2. Massey Hall*
3. Movie theatres	*3. Museums*	*3. O'Keefe Centre*
		4. Roy Thompson Hall

What would an index using multi-menus look like for the same information? The root page might be as follows:

Videotex Information

Choose one or more of the following:

Cancer Heart Diabetes Campaign
Cancer - Heart - Diabetes

Toronto information
Attractions - Entertainment - Retail - (more)

Ontario information
Ministry of Agriculture - Bookstore - (more)

Refresh

With conventional menus, one item is selected and (implicitly) the rest are discarded. With multi-menus, several items can be selected. The process of selecting menu items and getting new menus is split into two steps. Suppose the task is to find information about films. In the conventional tree index, users would likely select *Toronto information* as the first item. At the second menu, several options are possible: *Attractions, Entertainment,* and *Nightlife.* The user must select one at random and explore.

With a multi-menu index, the user is presented immediately with the second level of menus. Thus the choice *Toronto information* need not even be made. The user can select both *Attractions* and *Entertainment,* as both are shown.

What about *Nightlife*? This topic is not displayed directly on the multi-menu, but is available if *(more)* is used. *(more)* is a special category included whenever there are too many sub-topics to be listed. When a user selects *(more),* extra sub-topics are displayed. Each list of sub-topics is cycled, so that continual selection of *(more)* will return the user to the initial subset. *(more)* provides "window scrolling" within a given menu.

If the user selects *(more),* the item *Nightlife* is displayed, and the user may choose this item as well as *Attractions* and *Entertainment.* Suppose that only *Attractions* and *Entertainment* are selected. The next task is to obtain a new multi-menu, which is done by selecting *Refresh. Refresh* "commits" all user choices and constructs a new menu.

Special Sites
Ontario Place - CN Tower - (more)

Historic Sites
Fort York - Casa Loma - (more)

Museums
Royal Ontario Museum - Neville Gallery - (more)

Art galleries
Art Gallery of Ontario - McMichael Collection - (more)

Music and concerts
Kingswood Theatre - Massey Hall - (more)

Movie theatres
Metro Theatre - Toronto Cinema - (more)

Refresh

The new multi-menu shows as major categories the sub-topics of *Attractions* and *Entertainment* (as can be seen from the conventional examples). Many more choices are presented on the second multi-menu, including the desired option — *Movie theatres*. The user can select a particular movie theatre by cycling through the list via *(more)*.

The navigational model depicts the user as travelling in a tree or network. A travel analogue means that the user is necessarily in one place at one time. Conventional menus provide a list of possible moves or steps. In order to reach the item *Movie theatres*, four steps are necessary. The correct path is *Root – Toronto information – Entertainment – Movie theatres*. If the user had selected *Attractions*, *Nightlife*, or any other reasonable option, the search would have failed.

In contrast, a multi-menu permits the user to be in many places at one time. Selection of *Attractions* and *Entertainment* is parallel travel of two subtrees simultaneously. Selections found to be "incorrect" (in this case, *Attractions*) can be disregarded at a later stage. Parallel search need be employed only when the user is uncertain about options. In addition, fewer menus are necessary to complete the search. Only two multi-menus were needed to find the list of movie theatres.

Selection of items in the multi-menu is dependent on the display device. *Pointing* with touch screen, mouse, puck, or light pen is possible with advanced display devices. Often such devices include a pointer; generally they are powerful enough to update screens quickly. Less capable devices are restricted to keyword entry, a number index, or possibly function keys to direct a cursor. Selected items can be highlighted with reverse video or colour. Users gain extra control if "unselecting" is allowed. In the first pass through the multi-menu, all desirable

options are selected. Subsequent passes reduce the number of selections to a comfortable searching set. Highlighting helps users distinguish between selected and non-selected items.

5.1. Improvements.

The most obvious difference between multi-menus and conventional menus is that more than one choice is allowed at each menu. Multiple selection converts a *best fit* problem into a *sufficient fit* problem. Rather than comparing each item against all the others, the user only needs to compare each item against some fixed internal yardstick of sufficiency. Any item that looks probable can be selected.

The second difference is the organization of the displayed information. Items on the menu are seen not as disjoint units, but in relation to other items in their grouping. Major categories are exemplified by their sub-topics; sub-topics are given context by their major categories. Users are less likely to choose inappropriate search strategies, and miscategorization and incorrect construction can be more easily noticed.

The third difference is the provision of *descriptors*. Descriptive phrases can help reduce menu selection error [Snowberry 1983] [Telidon 1981]. In multi-menus, these descriptors are selectable sub-topics. As discussed in Section 4, people often describe categories with a set of conspicuous subordinates [Dumais 1983] [Landauer 1982] [Williams 1982]. Further, some subordinates are better representatives than others [Mills 1981]. Since the order of sub-topics is under system control, it can be optimized. In particular, the system may maintain a profile of selections for each user, and present the most commonly selected item as a sub-topic.

A fourth difference is *range*. The range of a menu is the set of pages which it can access [Tompa 1982]. Conventional menus range only over the set of immediate descendants. In a multi-menu, users may select immediate descendants or the children of those descendants. The rate of descent through the tree is bounded by the user's step size. Conventional menus provide only one step size, that from parent to child. Multi-menus provide this step size and also the step from parent to grandchild. Small steps are useful when the tree and its categories are unfamiliar; large steps are useful when the tree is well known and traversal effort should be reduced.

5.2. Structure dependence.

Multi-menu search explores a tree index where data structure is explicitly separated from data content. Rather than pre-configured pages at each node, multi-menus are dynamically derived from the structure of the tree. Many multi-menus can be configured since the menu is a response to a set of user selections. Figure 6 shows how a multi-menu is derived for a given selection "1".

Arcs in the tree are assumed to be named with the page label (or can be used to access the page label). Multi-menus are not pages, but collections of these arcs. Accessing a node is a request for a display of two levels of arcs of the subtree rooted at the node.

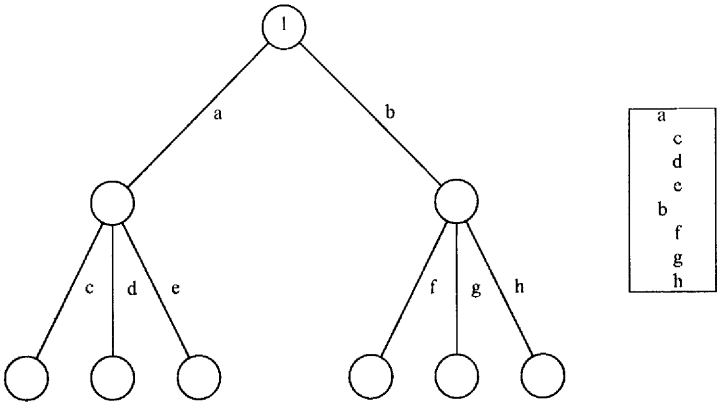

Figure 6.

Conventional menus are insensitive to database updates. Labels on conventional menus may or may not identify the subtrees accurately (the incorrect categorization problem). When an update is made to a conventional tree index, the pre-configured menu and any cross links must be manually altered. Descriptive phrases or exemplars used in conventional menus are as insensitive to database changes as are the selectable items.

Since multi-menus are generated from database structure, a change in the structure is automatically reflected in the associated multi-menus. The descriptive items provided by multi-menus are also structure dependent. Descriptors are merely the next lower level of arcs, and so are automatically generated.

5.3. Personal searching.

Multi-menus are a personal searching tool.

(more) is a personal feature. A user selects *(more)* to cycle through the list of sub-topics by accessing links one level down a sub-tree. Thus *(more)* fulfills two functions. First, it helps reduce uncertainty about a category by allowing users to look at all its descriptors. Second, it keeps the size of the display small to reduce confusion. Users need select *(more)* only for unknown or fuzzy categories. *(more)* allows local browsing without complete replacement of the multi-menu.

A second personal feature is the set of dimensions along which the multi-menu can be varied. Users can vary the breadth of the search: few or many choices are possible at each stage. The more choices made, the broader the search pattern.

Users can vary the depth of the search. Known categories can be examined quickly by taking large steps, and unknown sections of the tree can be searched with small steps.

An extra dimension is the number of levels displayed. The examples show multi-menus of two levels only, but this should be variable during a session. Experienced users could reach deeply into the tree very quickly if more levels were shown. Novices may start with a more conventional single level menu, still retaining the advantages of multiple selection.

One technique for introducing personalization is to let users manually adjust interface parameters. In order to be effective, users must acquire a knowledge of parameter values and the procedures for adjusting them. When possible, a preferable approach is the *self-adjusting* system. Multi-menu depth and breadth are self-adjusted. Confused users select fewer items, in order to reduce the number of options. Experienced users select many options for parallel search. There is no distinction between menus for novices and menus for experts. A continuum of menus from simple to complex is under user control.

Self-adjustment is more difficult for the number of levels displayed. This dimension is highly dependent both on the breadth of the tree and the capability of the display device. In addition, there seems to be no good way for the system to infer that a change in the number of levels is desired. It may be best to leave this parameter to manual adjustment.

5.4. Other factors.

Most errors are made in the first two levels of the tree [Telidon 1981] [Vrins 1982]. By compressing these levels into one, the user has a better chance to avoid errors when moving from the root to the first level.

Upward database compatibility should be little problem, if new databases are designed so that structure is separable from content. Existing databases can be converted to multi-menu use by replacing conventional menus and providing page labels. This may be an involved problem, as often the distinction between index pages and data pages is unclear.

Construction and display of multi-menus is easily carried out by intelligent terminals. A multi-menu construction program and index are downloaded to the terminal, and the user left to experiment locally until a new index is needed or a set of page requests has been determined. As described earlier, this distributed environment can improve response time.

Multi-menus provide a simple way to search multiple contexts. The root of each context tree can be placed on the initial multi-menu. Users search in the normal multi-menu manner. Rather than switching contexts in the middle of a search, the user searches all contexts from the beginning of a session.

Multiple contexts are an intermediate step to general information networks. Three factors distinguish network from hierarchic structures. First, networks need not provide context for the searcher by recursively nesting categories. Second,

parallel search techniques are more useful in a network when data is not partitioned according to user needs. Third, cycles are possible because there is less restriction on interconnections. Multi-menus are preferable to conventional menus in each of these areas. Context is provided by the organization of items in a multi-menu; parallel search is possible; cycling within the range of the multi-menu can be eliminated.

The capabilities of *(more)* can be expanded. Rather than constraining the user to cycle through the list of sub-topics in only one direction, *(more)* could provide cycling in two directions. This is especially useful when the number of sub-topics is large. A "home" position could also be defined, in order to return the display to its default state.

Although the examples have shown text menus, *iconic* multi-menus are also possible. The use of icons to describe objects has been much promoted, but the problems are considerable [Lodding 1983] [Mills 1981]. A key problem is the lack of context provided by a single icon. Multi-menus might help to remove this by structuring the display of icons. A completely incomprehensible icon may gain much significance when seen in context with a supporting set of icons (or even text phrases).

5.5. Problems.

The most obvious disadvantage of multi-menus is their size. The limitations of a videotex screen and the need for visual spacing to communicate semantic structure imply small multi-menus. The capability of the display device places an upper bound on the number of choices a user can select. It may also suggest a restriction on the number of selections provided at each level of the tree. If users select an average of k items at each multi-menu, and each node has an average of n branches, then kn major categories will be shown on the average multi-menu. k and kn can be estimated, fixing n. The number of selections is also restricted for cognitive reasons. Users may become confused about the parent nodes of the sub-topics on a new multi-menu. This might be avoided by imposing a new level of structure; sub-topics of a given choice are isolated from the others by grouping or colour. However, it remains to be seen how many choices are commonly used. Two or three selections may be sufficient; in this case, less effort needs to be spent on layout of large numbers of options.

A second problem with multi-menus is a result of the distinction between data structure and data content. Multiple selection is useful while the user is in the index, but at some point, data pages will be selected. More than one data page may be selected, and a combination of menu choices and data pages can also occur. One solution is to provide a *display index* on the next multi-menu. Users can continue searching normally or choose a page from the display index. The index may be implcit; for example, selected display pages may be highlighted in red, selected menu choices in blue. After each display page is shown, the user is returned to the previous multi-menu for the next choice.

Multi-menus are not significantly better than conventional menus during error

recovery. Since the menus are generated dynamically, users may be confused about previous positions. However, parallel search capabilities and a better view of the environment substantially reduce the need for error recovery.

6. An editor for personalized structuring.

Multi-menus add a personal dimension to searches of existing databases, but are not a complete solution. An alternative to explaining existing categories more thoroughly is permitting users to create their own categories. People are used to re-structuring existing information: books contain bookmarks, dog-ears, underlining, and margin notes. A similar capacity in videotex would provide the tools to manipulate personal databases.

In addition, videotex must become more than a page retrieval system. Information structuring, communication, and processing are features of an augmented knowledge workshop as important as storage and retrieval. Some of these facilities are provided in videotex, but in a disjointed fashion. There is a need for integration of general information services to eliminate redundancy, provide a comprehensive user protocol, and support tool development.

This section investigates *structure manipulation* as a model for the integrated interface.

6.1. Personal information structuring.

Structuring tools in videotex should support common information structuring habits. People use two basic tools to structure collections of information: *piles* and *files* [Malone 1983].

A pile is an unorganized collection of documents. Piles are usually comprehensible only by their creators, so they are most useful in a personal information system. Piles represent categories that are not explicitly named, but indexed and accessed by location in space and time [Engel 1983]. Space-time indices are created with little conscious effort. People use piles when they want to avoid the effort of categorization. Certain sets of information seem most useful, obvious or practical when physically or conceptually contiguous, but the source of the relationship is unknown, so a pile is created. Piles serve both finding and *reminding* functions. A document is recalled by its location in a pile; while searching, other documents remind one of tasks uncompleted or forgotten. In this sense, a collection of piles aids browsing.

Space-time is not always an effective index. Each library has a space-time index to its collection, but this is not easily accessed by new users, or those attempting to locate borrowed materials. *Files* are organized collections of information that can be located in a cognitive (as well as space-time) index. A file is a pile with a name.

Computer tools to structure information are of two basic types. Traditional database tools are highly developed [Date 1981], but the disadvantages noted in Section 4 (rigidity, complexity, lack of multiple paths) make these unlikely candidates for videotex. An augmented knowledge workshop is personalized and its information can be structured by unsophisticated users. Information collections are small but highly dynamic. Efficient processing is less important than clear specification, so some other technique should be chosen.

More flexible information structuring techniques can be found in editors. In traditional record-oriented databases, the schema or form of the database is treated independently of the content. In an editing environment, the data's structure is moulded to reflect or support the content. Informal, extemporized structures are created and modified in a series of interactive sessions.

Text editors vary in their capability to structure documents. *Ed*, a typical line-oriented editor, can only modify a linear sequence of characters [Kernighan 1978]. A more powerful technique is the hierarchical structuring of statements possible in the NLS editor [Engelbart 1973]. Text networks or *hypertext* can be manipulated in HES and FRESS [Rice 1972] [Meyerowitz 1982]. A text network with enhanced indexing and interface tools constitutes an electronic book [Weyer 1982]. When graphics and colour are added, the result can be a highly informative and interactive document [Feiner 1982].

As the structuring abilities of text editors increase, the type of structure manipulated becomes more general:

> Our experience with FRESS and the success of network databases in such applications as computer-aided instruction (e.g. PLATO) have convinced us of the need for such generality. [Feiner 1982, p. 60]

and

> It was important to us that the natural way of expressing graph structure – as a directed graph – become the actual means by which our authors structured their documents [Feiner 1982, p. 60]

Graph structures have been suggested by many implementors of personal databases [Cattell 1979] [Sauvin 1983]. A structure editor for videotex could employ directed graphs in a useful way. The basic atoms in videotex are pages and menu items, which map naturally to nodes and arcs in a directed graph. Graph structures are relatively easy to understand as there are only two simple components. In addition, graphs are flexible, so more restricted structures can be comfortably created. For example, a tree index is a restricted graph.

The simplicity of graph components can be disadvantageous. Moderately large structures have many interconnections and so require more work to specify. This problem can be avoided if tools are provided to reduce specification effort. For example, a more sophisticated node might represent a collection of zero or more pages. A second problem is display. Simple graphs are intuitive and easy to follow, but highly interconnected information can appear cluttered. Displaying graph structures is a difficult, unsolved problem [Baecker 1983] [Feiner 1982].

Graphs are a simple, natural way to encode small collections of information, but enhanced tools are necessary for larger collections. Some suggested enhancements (and more detail on editor design) can be found in Section 7; a prototype editor is described in the Appendix.

6.2. Structuring.

The goal of the structure editor should be to make "simple things simple and complex things possible."† While simple page collections should be no more difficult to manipulate than FAST TRACK® lists, complete videotex tree indices or networks should also be possible.

The editor can be used as a local structuring tool. Each user has a personal environment, containing structures, categories, collections, or other useful bits and pieces. Piles are structures referenced by their location on the display screen; files are structures that have been given an explicit name. Users treat the editing environment as an electronic desk: structures are arranged and designed in order to inform, remind, aid retrieval, or just as a place to keep odd information.

The editor can be used as a global structuring tool. Information providers employ the editor's enhanced capabilities to create and maintain complete videotex tree indices. A new videotex service is possible: *structure providers.* Information providers generate information and structure only these pages. Structure providers are "free lance"; they provide access to collections of information not restricted to a single information provider. Each structure provider serves a different need and maintains a different set of categories (although the base set of display pages may be the same). Structure providers compete to produce effective and popular indices. Competition to draw subscribers is a mechanism that generates multiple paths to data. A good structure provider can increase the number of users that view a given page, so information providers are wise to permit access. Some contemporary examples of structure providers include *Consumer's Reports, The Yellow Pages,* and the *Schwann's* catalogue of recorded music. Note that the roles of structure provider and information provider overlaps somewhat in these examples; they are likely to do so in a videotex environment as well.

6.3. Communication.

How can structure manipulation enhance communication?

Presently, videotex communication is restricted to simple mail systems. Rudimentary pages of text can be passed between users. A new emphasis on communication results from breaking down the distinction between users and information providers [Godfrey 1980]. If every user becomes a potential information provider, simple mail systems are not adequate.

Even if this egalitarian dream does not materialize, more advanced communication techniques are necessary. Users want to send information to other users. Complaints about system service and structuring must be handled. Gateway services need to communicate with users outside of the gateway page. Clubs and organizations want to exchange material, some of which may be sensitive and require secure communication techniques.

A related problem is information dynamics. New pages are constantly being

† Alan Kay, quoted in [Lipkie 1982, p. 116].

added to any videotex service, and users must be kept informed about relevant additions. Users should not have to use the tree index to look for new information in old locations.

Communication can be enhanced with *inboxes*. An inbox is a structure declaring its owner's interest in some category of information. For example, all users will have an inbox to receive personal mail. Other inboxes can be created for club or organization newsletters. System-named inboxes might contain updates for a particular subtree of the index. When new pages are added to this subtree, they are also appended to the inbox. Users wishing to be kept informed about new information in a subtree place the subtree's inbox in their own environment.

Inboxes can fundamentally change the function of the videotex service. Rather than an intermediary between the disjoint set of information providers and users, the system becomes a clearinghouse for videotex page communication. Any user is a potential information provider, and all may submit pages to the service. Submitted pages are inserted in the tree index, and appended to a set of inboxes as described by a distribution list. Such a facility resembles the USENET network news system available on UNIX®†.

How are users located for communication? A *subscriber list* provides information about all users on the system. This list may be a subtree of the main index searched with the usual techniques. More complex structures could be searched using the editor. An interface with approximate string-matching capabilities would help users find other users when spelling is uncertain.

The subscriber list introduces the notion of users as "data". An inbox becomes a "keyword" by which the service can access the user. Each inbox represents a different category of information in which the user is interested. Informing users about page updates can be mapped to the retrieval process.

Protective measures are required. Global distribution may be restricted to responsible information providers, or a fee charged based on the number of recipients. Inbox names may be appended to a directory or other system-defined name that prevents personal mail from being re-directed to other users. On a more social level, people must develop codes of behaviour for electronic communication much as they have for other media [Brotz 1983].

Communication of data structures has received little attention, as have many aspects of structuring. The structure editor enables creation of personal indices to videotex information. Structure providers want to transmit their services to other users. Structure information is transmittable when standardization of editor representations and transmission formats is realized. *Packets* are transmittable collections of pages and structure. Packets can contain any structure that the structure editor creates, in a form that can be easily sent to a user.

Packets, the subscriber list, and inboxes are all be created and maintained by the structure editor. Structure manipulation can become a tool to enhance

† UNIX is a trademark of Bell Laboratories.

communication.

6.4. Processing.

The need for information processing in videotex is uncertain. More thorough evolution is necessary for concrete predictions, but it is inevitable that people will want some simple tools. For example, a user might want to sort pages according to the date of creation or time of retrieval. Another possibility is filtering of information collected in gateways.

Tools in videotex are much more likely to evolve than emerge completely developed. The set of tools must respond to the demands of the users, and they are an unknown quantity. Each tool will be a "black box" utility, whose internals need not be understood by the average user.

Videotex might successfully employ a UNIX-like processing facility [Kernighan 1981]. UNIX is a popular operating system with a large collection of utilities. *Pipes* connect utilities to filter an information stream in some desired way. Useful combinations of low-level utilities become new utilities. UNIX supports its own evolution with the capability to create new utilities from old, and by providing a common medium for utility communication – the UNIX character stream.

A similar mechanism in videotex might use packets as the medium of communication. A utility exists as an action page; a collection of action pages with useful general functions is used to filter a page stream. For example, a *sort* page filters output from gateway pages as in Figure 7.

UNIX pipes and utilities are sequential in nature, so the "editing" facility provided is primitive. This restriction can be avoided if a general structure editor is available. *Sort* should accept input from several gateway pages concurrently. An arbitrary page structure could be passed through an arbitrary filter structure by using the structure editor to connect data and utilities.

Some UNIX users have complained of the difficulty of finding and understanding utilities [Norman 1981]. To avoid this problem in videotex, utilities may be collected in the *utility* list, an index to the system's pre-defined utilities. As for the subscriber list, this may be a sub-tree of the main index. Users access the utility list to read documentation and select a useful process. Then, the appropriate packet containing action pages and structure can be sent to the user. This packet may also contain action pages that educate the user about the process, or even a syntax directed editor to aid in using the utility.

The utility list, filters, and the process structures are all be created and maintained with the structure editor. Structure manipulation can become a tool to enhance information processing.

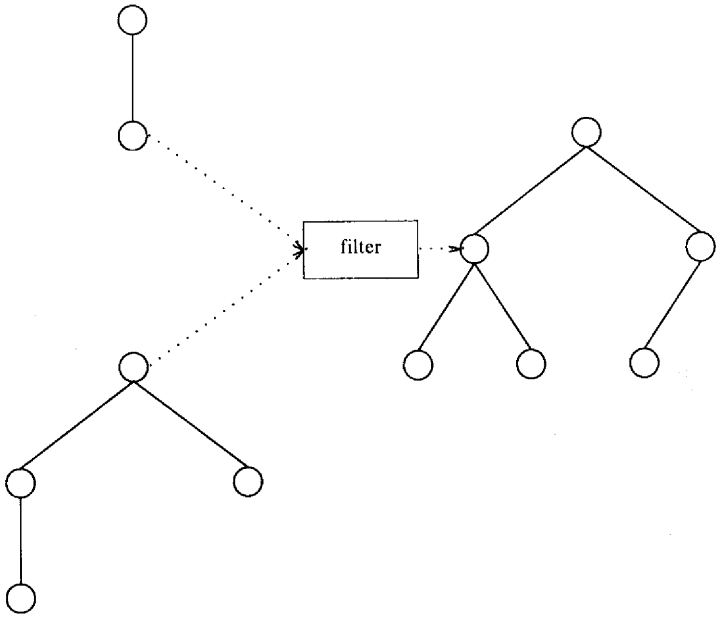

Figure 7.

7. Features of a structure editor.

What factors result in good editor design? There is considerable experience in design and use of editors for both textual and non-textual material [Fraser 1981] [Maxemchuk 1982] [Myers 1983] [Singh 1982]. Like any cognitive prosthesis, an editor is successful when it supports the eccentricities of its user. Given editor *A* and editor *B*, deciding which is better can be difficult. At most we can decide which editor is preferred by a group of users, and this test can only be applied after implementation.

A simple methodology for outlining an editor design is adapted from [Meyerowitz 1982]. Four steps are involved. First, the *objects of interest* are defined. The characteristics of the editor's targets are examined. Second, the *operations of interest* are defined. Given a set of objects, what are the modifications users are likely to want? Third, a *specification language* is created. This language succinctly communicates the tasks involved in an editing session. Each

task is a sequence of operations on objects as previously defined. Finally, *application details* are considered. Considering the design in a working environment suggests new capabilities and constraints. As a result, previous steps in the design process may be repeated.

7.1. Objects.

The editor should provide tools to manipulate a general graph. As discussed previously, graph components are primitive, so tools must be provided to simplify creation and maintenance of larger structures. This has been done by enhancing the primitive components.

Placeholders and *n-arcs* are the basic components of enhanced graphs. Placeholders are nodes; n-arcs are links connecting nodes.

Placeholders contain zero or more items, each of which may be a page, a placeholder or a template (to be defined shortly). Placeholders can be named, in which case they are analogous to files; unnamed placeholders are analogous to piles. Placeholders provide a way to designate a node without the need to instantiate it. Placeholders can be left empty, added to or deleted from, and removed. Loosely speaking, a placeholder represents a category or variable.

n-arcs connect the placeholders in the graph. An n-arc is a directed arc from one placeholder to another, and is to be interpreted as the set of arcs consisting of a primitive arc from each item in the source node to each item in the destination placeholder, as in Figure 8.

Placeholders and n-arcs can be manipulated as simple nodes and arcs would be. Placeholders can contain exactly one item; if this is the case, then all n-arcs reduce to primitive arcs. The use of n-arcs and placeholders can reduce specification time for a large graph. The display will be less confusing, as fewer arcs are shown. The hierarchic tree structure shown in Figure 9 uses n-arcs to reduce the number of arcs specified.

Placeholders and n-arcs reduce specification effort and complexity of the display. Users should have the capability to reduce specification effort for commonly used collections of placeholders and n-arcs. *Templates* are similar to typing facilities in modern programming languages. If a particular organization of n-arcs and placeholders is common, users can create a template of the organization. A new instantiation of the template is reduced to a single operation: specifying the template name. Users may also have the option of substituting a simple icon for the display of the template structure, further reducing the complexity of the display.

The editing session is contained in an *environment*, which is both restorable and transferable. Restorability means that users can terminate an edit session at any point, and later re-start without loss of work. This is achieved by maintaining a set of *environment variables* that capture the essential aspects of the session. Transferability implies that environments can be passed between users. This is achieved by providing a standard format. Environments serve as packets.

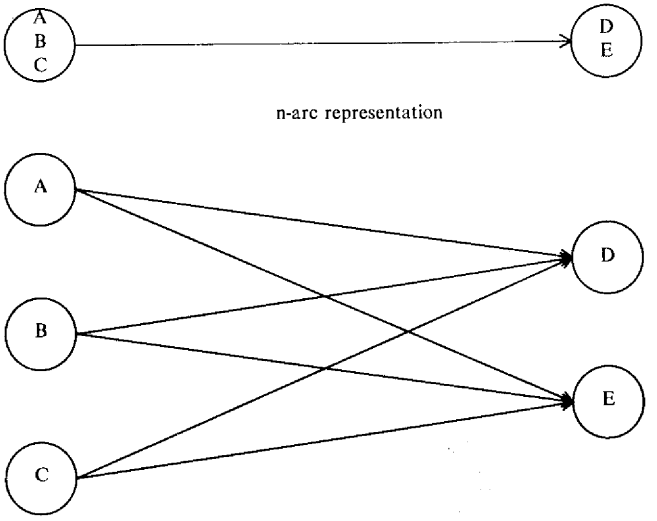

Figure 8.

It may be possible to edit page contents with the structure editor as well. The editor would then become a *virtual editor;* editing operations and interface are applicable to several targets [Maxemchuk 1982].

7.2. Operations.

Part of the difficulty of editor design is defining a minimal set of operations [Buxton 1983]. Several operations seem intuitive for the structure editor:

copy – makes a copy of an object at a specified location in the environment. This may be a location within some other object (such as a placeholder).

delete – removes an object from the environment.

name – names or renames an object in the environment.

move – moves an object from one position in the environment to another. *move* is a combination of copy and delete.

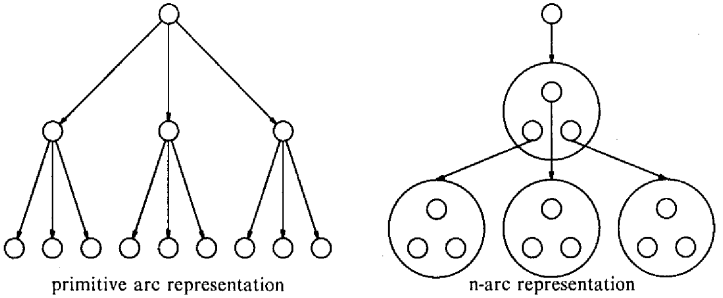

Figure 9.

execute – "executes" an object. If the object is a display page, it is shown. If the object is an action page, the page's program is initiated. If the object is a gateway page, the external service is initiated.

template – define a template containing a set of specified items.

undo – reverses the effect of the previous operation.

exit – leave the editor.

The full set of operations depends upon the specification language and application details. For example, a command line interface might require operations for displaying information about the current state of the environment (the number and names of structures). A graphics display presents this information without the need for explicit operations. In some environments, operations are subsumed as special cases of more general activities. For example, deletion may be obtained by "moving" an object to a garbage can, or off the screen [Buxton 1983]. As well, certain *object specific* operations may need definition. For example, a *forms* object might have special operations connecting various form entries to an internal data structure.

7.3. Specification language.

The specification language is governed by three principles. *Direct manipulation* is specification via manipulation of data objects analogous to physical objects [Schneiderman 1983]. Graphics tools are used to present and manipulate icons of objects and operations. The targets of the structure editor have no actual physical analogue; users are simply manipulating pictures of abstract concepts. The principles of direct manipulation still apply, as the pictures represent "real" structure

adequately.

Selecting is a basic principle. To select is to choose operations and operands from a menu using a pointing device, such as a mouse [Engelbart 1973] [Lipkie 1982]. The mouse directs a cursor on the screen to point at a desired icon. A button on the mouse is pressed to indicate that the icon is being selected. Multiple button presses step through an *object nesting*. The first press indicates the smallest icon directly underneath the cursor, another press expands the selection to the next including level, and so on. The nesting can be cycled so that continual selection returns the user to the lowest level.

Information hiding reduces the amount of presented information by keeping low-level details invisible. Such information is usually important for a pictorial representation (for example, details about colour, size, and shape of the icon). Another aspect of information hiding is to supress display of invalid options. A selection-based environment displays menus of operations and operands; if only valid operations and operands are shown, incorrect items cannot be selected.

Figure 10 shows a possible layout for the editor operand and operation menus. Each operation and operand is a selectable *button*.

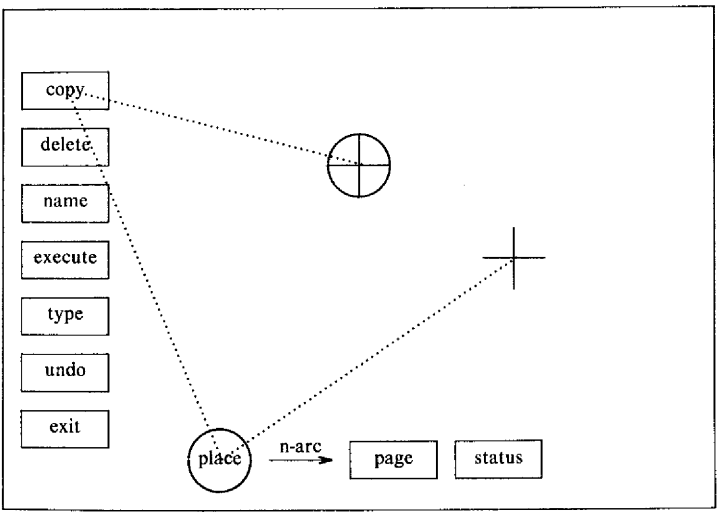

Figure 10.

The cross hair represents the cursor, which is controlled by the position of the

mouse on the desk. Operation buttons are on the left side of the environment. At the bottom right is a status block; this displays the name of the environment being edited and an indicator for errors. Object icons (placeholder, n-arc, page) are along the bottom. Underneath is a space for text entry, which is used when objects are named. Icons representing new objects declared with the *template* facility are presented along the right hand side of the environment.

Structure can be created by using the mouse to select various buttons on the screen. Figure 10 shows how a user can create a category by first selecting the placeholder icon, then selecting the *copy* operation, then selecting a point in the environment where the placeholder is to reside.

The specification language may contain several features to aid in structure modification. *Grids* are constructs that can be displayed in the environment to help users position placeholders and n-arcs. Concentric and rectangular grids are useful; as well, grids of basic data structures (such as trees) may also be desirable.

Property sheets are for the more experienced user [Lipkie 1982]. A property sheet describes each object in the environment, each type of object, and the environment itself (describing properties of environment variables). Property sheets contain information about size, shape, colour and other useful information. Property sheets require operations such as *display* and *copy* properties. The property sheets themselves can be modified with the editor's operations. Display of a property sheet temporarily obscures the environment; when the sheet has been edited the environment display is restored.

7.4. Application details.

Use of the editor in a videotex environment suggests several extra operations and possibilities.

Typically, users structure collections of pages which they do not own, such as pages belonging to the videotex service. However, some users will wish to maintain a personal copy of information that is accessed often, or which is subject to change by the service. The operation *local* creates a copy of the service page in the user's own storage (which may be maintained by the system, or be kept at the user's site). User-created pages will initially be local, until control is passed to the videotex service. Some pages are not capable of being made local; for example, a gateway page is only a surrogate for a port to an external database. Other pages will have little use as local pages: a service page that displays the current time is in need of constant update to be useful.

Users who search in the tree index and locate pages of interest will want to transfer them to the editor for later structuring. A *current page* stack contains the last pages selected in a browsing session. Users initialize the editor, create a few categories, and then transfer to the tree index to browse for relevant pages. Each time a useful set of pages is located, the user toggles back into the editor and copies from the current page stack into a placeholder. Pages in the current page stack can be viewed using *execute* and re-arranged with the other editor commands.

The limited display space of the videotex screen may necessitate *directories*.

For example, if a large number of templates exist, their icons might clutter the screen. A template directory is a page displaying the list of defined templates. This page can be temporarily laid over the environment display, and a few templates may be transferred to the environment. Similar directories might exist for grids and page primitives.

Extra operations are needed for directory display and storage. If many directories exist, they in turn may need to be kept in a directory. An effective solution is to let the directories be structures in the environment; then the editor can be used to access them. The virtual editor has several targets: structure, directories, property sheets, the current page stack, and possibly page graphics.

7.5. Conclusions.

Structuring tools are an important component of a personal approach to videotex databases. Multi-menus and a structure editor have been described; a prototype combining these ideas is reported in the Appendix. Further tools will result from intensive testing. The most important lesson of cognitive psychology is that designers are much more likely to miss important problems than they are to anticipate them. The only solution is exhaustive (and expensive) testing. Such testing must be carried out in concert with a useful videotex system being accessed by large numbers of users. Only then can the effectiveness or desirability of particular interface proposals be evaluated.

References.

[Abell 1981]
Implementation of a Telidon System Using UNIX File Structures
Richard Abell
in *The Telidon Book*, edited by David Godfrey and Ernest Chang, Press Porcépic Ltd., Victoria, British Columbia, 1981, pp. 203-209

[Akgun 1981]
Delivery of Telidon Services with a Fibre Optic Integrated Network
M.B. Akgun, Y.F. Lum, K.Y. Chang, C. Harwood and G. Tough
International Symposium on Graphics and Text Communication, Paris, November 16-18, 1981

[Albright 1979]
Some Limits to Subject Retrieval from A Large Published Index
John Burton Albright
Ph.D. dissertation, University of Illinois at Urbana-Champaign, 1979

[Arnheim 1974]
Art and Visual Perception: A Psychology of the Creative Eye
Rudolph Arnheim
University of California Press, Berkeley, 1974

[Baecker 1983]
Program Visualization
Ron Baecker
Colloquium delivered at the University of Waterloo, November 9, 1983

[Ball 1981]
User Interfaces for Future Videotex Systems
A.J.S. Ball and J. Gecsei
Publication #412, Départment d'informatique et de recherche opérationnelle, Université de Montréal, May 1981

[Belkin 1980]
Anomalous States of Knowledge as a Basis for Information Retrieval
Nicholas J. Belkin
The Canadian Journal of Information Science, Vol. 5, 1980, pp. 133-143

[Bochmann 1982]
Keyword Access in Telidon: An Experiment
G.V. Bochmann, J.Gecsei and E. Lin
Videotex '82, New York, N.Y. June 28-30, 1982, pp. 321-332

[Bown 1978]
A General Description of Telidon: – A Canadian Proposal for Videotex Systems
H.G. Bown, C.D. O'Brien, W. Sawchuk and J.R. Storey
Government of Canada, Department of Communications, Technology and Systems Branch, CRC Technical Note 697-E, Ottawa, 1978

[Britannica 1983]
The New Encyclopedia Britannica in 30 Volumes
Encyclopedia Britannica, Inc. Toronto, 1983

[Brotz 1983]
Message System Mores: Etiquette in Laurel
Douglas K. Brotz
ACM Transactions on Office Information Systems, Vol. 1, No. 2, April 1983, pp. 179-192

[Bush 1945]
As We May Think
Vannevar Bush
Atlantic Monthly, July 1945, pp. 101-109

[Buxton 1983]
Towards a Comprehensive User Interface Management System
W. Buxton, M.R. Lamb, D. Sherman and K.C. Smith
SIGGRAPH '83, Computer Graphics Vol. 17, No. 3, July 1983, pp. 35-42

[Cargill 1979]
A View of Source Text for Diversely Configurable Software
Thomas A. Cargill
Ph.D. dissertation, Department of Computer Science, University of Waterloo, CS-79-28, 1979

[Carroll 1982]
The Adventure of Getting to Know a Computer
John M. Carroll
IEEE Computer, November 1982, pp. 48-58

[Cattell 1979]
An Entity-Based Database Interface
R.G.G. Cattell
Xerox CSL-79-9, Palo Alto Research Center, August 1979

[Clifton 1981]
Integrating New Information With Old Knowledge
Charles Clifton Jr. and Maria L. Slowiaczek
Memory & Cognition, Vol. 9(2), 1981, pp. 142-148

[Collins 1975]
A Spreading Activation Theory of Semantic Processing
Allan M. Collins and Elizabeth F. Loftus
Psychological Review, Vol. 82, No. 6, 1975, pp. 407-428

[Conrath 1980]
Evaluating The Need For Burotique: Some Taxonomic and Methodology Issues
David W. Conrath
Proceedings of IFIPS TC-6 International Workshop on Integrated Office Systems – Burotics, Versailles, North-Holland, 1980, pp. 199-208

[Date 1981]
An Introduction to Database Systems. 3rd Ed.
C.J. Date
Addison-Wesley, Reading, Massachusetts, 1981

[De Leon 1983]
Is There Really Trouble with UNIX?
Lorenzo De Leon, William G. Harris and Martha Evens
CHI '83 Conference Proceedings, Boston, December 12-15, pp. 125-129

[deBono 1977]
The Mechanism of Mind
Edward deBono
Penguin Books, 1977

[Dumais 1983]
Using Examples to Describe Categories
Susan T. Dumais and Thomas K. Landauer
CHI '83, Conference Proceedings, Boston, December 12-15, pp. 112-115

[Engel 1983]
What, Where and Whence: Means for Improving Electronic Data Access
F.L. Engel, J.J. Andriessen and H.J.R. Schnitz
International Journal of Man-Machine Studies, (1983) 18, pp. 145-160

[Engelbart 1973]
The Augmented Knowledge Workshop
Douglas C. Engelbart, Richard W. Watson and James C. Norton
AFIPS Conference Vol. 42, 1973, pp. 9-21

[Fedida 1982]
Viewdata Gateways: Commercial Significance and Economic Criteria
Sam Fedida
Computer Communications, Vol. 5 No. 4, August 1982, pp. 181-185

[Feiner 1982]
An Experimental System for Creating and Presenting Interactive Graphical Documents
Steven Feiner, Sandor Nagy and Andries van Dam
ACM Transactions on Graphics, Vol. 1, No. 1, January 1982, pp. 59-77

[Fitter 1979]
Towards More Natural Interactive Systems
M. Fitter
International Journal of Man-Machine Studies (1979) 11, pp. 339-350

[Foley 1974]
The Art of Natural Graphic Man-Machine Communication
James D. Foley and Victor L. Wallace
Proceedings of the IEEE, Vol. 62, No. 4, April 1974, pp. 462-470

[Fraser 1979]
A Compact, Portable CRT-based Text Editor
Christopher W. Fraser
Software – Practice and Experience, Vol. 9, 1979, pp. 121-125

[Fraser 1981]
Editing Data Structures
Christopher W. Fraser and A.A. Lopez
ACM Transactions of Programming Languages and Systems, Vol. 3, No. 2, April 1981, pp. 115-125

[Fry 1981]
Survival in the Wilderness
Alan Fry
Macmillan of Canada, Toronto, 1981

[Furnas 1981]
Psychological Structure in Information Organization and Retrieval: Arguments for More Considered Approaches and Work in Progress
G.W. Furnas
Workshop/Symposium on Human Computer Interaction, Atlanta, Georgia, 1981

[Furnas 1982]
Statistical Semantics: Analysis of the Potential Performance of Keyword Information Systems
G.W. Furnas, T.K. Landauer, L.M. Gomez and S.T. Dumais
1982

[Gecsei 1983]
The Architecture of Videotex Systems
Jan Gecsei
Prentice-Hall, 1983

[Geller 1983]
User Interfaces to Information Systems: Choices vs. Commands
V.J. Geller and M.E. Lesk
Sixth Annual International ACM SIGIR Conference, SIGIR Vol. 17, No. 4, Summer 1983

[Godfrey 1980]
Gutenberg Two: The New Electronics and Social Change
David Godfrey, Douglas Parkhill, John Madden and Andre Ouimet
Press Porcepic Ltd., Toronto, 1980

[Goffman 1969]
An Indirect Method of Information Retrieval
William Goffman
Information Storage and Retrieval, Vol. 4, 1969, pp. 361-373

[Gonnet 1983]
Unstructured Data Bases or Very Efficient Text Searching
Gaston H. Gonnet
Proceedings of the ACM Principles of Database Systems Conference, Atlanta, Georgia, March 1983, pp. 117-124

[Grande 1980]
Aspects of Pre-Literate Culture Shared by On-Line Searching and Videotex
Sally Grande
The Canadian Journal of Information Science, Vol. 5, 1980, pp. 125-131

[Hampton 1981]
An Investigation of the Nature of Abstract Concepts
James A. Hampton
Memory & Cognition, Vol. 9(2), 1981, pp. 149-156

[Herot 1980]
Spatial Management of Data
Christopher F. Herot
ACM Transactions on Database Systems, Vol. 5, No. 4, December 1980, pp. 493-514

[Janes 1983]
Eliminate the Keyboard with Touch-Sensitive CRT Screens
C.C. Janes and D.A. Brodzik
IEEE 2nd Annual Phoenix Conference on Computers and Communications, Phoenix, Arizona, March 14-16, 1983, pp. 575-579

[Johnson 1983]
Formal Models for String Similarity
J. Howard Johnson
Ph.D. dissertation, University of Waterloo, Department of Computer Science, Waterloo, Ontario, Canada, Research Report CS-83-32, November 1983

[Kent 1981]
Data and Reality: Basic Assumptions in Data Processing Reconsidered
William Kent
North-Holland, New York, 1981

[Kernighan 1978]
A Tutorial Introduction to the UNIX Text Editor
Brian W. Kernighan
UNIX Programmer's Manual, Version 7, Bell Laboratories, Murray Hill, New Jersey, September 1978

[Kernighan 1981]
The UNIX Programming Environment
Brian W. Kernighan and John R. Maseby
IEEE Computer, April 1981, pp. 12-24

[Koll 1979]
The Concept Space in Information Retrieval Systems as a Model of Human Concept Relations
Matthew B. Koll
Ph.D. dissertation, Syracuse University, 1979

[Landauer 1982]
Human Factors in Data Access
T.K. Landauer, S.T. Dumais, L.M. Gomez and G.W. Furnas
The Bell System Technical Journal, Vol. 61, No. 9, November 1982, pp. 2487-2509

[Landauer 1983]
Natural Command Names and Initial Learning: A Study of Text editing Terms
T.K. Landuaer, K.M. Calotti, and S. Hartwell
Communications of the ACM, Vol. 26, No. 7, July 1983, pp. 495-503

[Larratt 1980]
Inside Videotex: The Future...Now
edited by Richard Larratt, Infomart Seminar, March 1980

[Lipkie 1982]
Star Graphics: An Object-Oriented Implementation
Daniel E. Lipkie, Steven R. Evans, John K. Newlin, Robert L. Weissman
SIGGRAPH '82, Computer Graphics Vol. 16, No. 3, July 1982, pp. 115-124

[Lochovsky 1981]
Interactive Query Languages for External Databases
F.H. Lochovsky and D.C. Tsichritzis
Telidon Behavioural Research 5, Government of Canada, Department of Communications, Behavioural Research and Evaluation, November 1981

[Lodding 1983]
Iconic Interfacing
Kenneth N. Lodding
IEEE Computer Graphics and Applications, March/April 1983, pp. 11-20

[Loftus 1979]
Reactions to Blatantly Contradictory Information
Elizabeth F. Loftus
Memory & Cognition, Vol 7(5), 1979, pp. 368-374

[Loftus 1983]
Since the Eruption of Mt. St. Helen's, Has Anyone Beaten You Up? Improving the Accuracy of Retrospective Reports with Landmark Events
Elizabeth F. Loftus and Wesley Marburger
Memory & Cognition, Vol. 11(2), 1983, pp. 114-120

[Mack 1983]
Learning to Use Word Processors: Problems and Prospects
Robert L. Mack, Clayton H. Lewis and John M. Carroll
ACM Transactions on Office Information Systems, Vol. 1, No. 3, July 1983, pp. 254-271

[Malloy 1983]
Commentary: Personal Computers and Videotex
Rich Malloy
BYTE, July 1983, pp. 114-129

[Malone 1983]
How Do People Organize Their Desks? Implications for the Design of Office Information Systems
Thomas W. Malone
ACM Transactions on Office Information Systems, Vol. 1, No. 1, January 1983, pp. 99-112

[Manuel 1982]
Disney's EPCOT Computer Festival
Tom Manuel
Electronics, November 3, 1982

[Maurer 1982]
Will Mupid Revolutionize Austria's Videotex?
H.A. Maurer
Videotex '82, New York, N.Y., June 28-30, 1982, pp. 187-197

[Maxemchuk 1982]
Virtual Editing: 1. The Concept
N.F. Maxemchuk and H.A. Wilder
Proceedings of the Second International Workshop on Office Information Systems, Couvent Royal de Saint-Maximiny, France, 13-15 October 1981, North-Holland, New York, 1982, pp. 161-172

[Mayer 1983]
Coming Fast: Services Through the TV Set
Martin Mayer
Fortune, November 14, 1983, pp. 50-56

[McKay 1983]
VDTs and the Brain
Shona McKay
Maclean's, November 21, 1983, p. 50

[Meyerowitz 1982]
Interactive Editing Systems: Part I & II
Norman Meyerowitz and Andries van Dam
ACM Computing Surveys, Vol. 14, No. 3, September 1982, pp. 321-416

[Mills 1981]
A Study of the Human Response to Pictorial Representations on Telidon
Michael L. Mills
Telidon Behavioural Research 3, Government of Canada, Department of Communications, Behavioural Research and Evaluation, August 1981

[Moar 1983]
Inconsistency in Spatial Knowledge
Ian Moar and Gordon H. Bower
Memory & Cognition, Vol. 11(2), 1983, pp. 107-113

[Myers 1983]
Incense: A System for Displaying Data Structures
Brad A. Meyers
SIGGRAPH '83, Computer Graphics, Vol. 17, No. 3, July 1983, pp. 115-125

[Negroponte 1979]
Books Without Pages
Nicholas Negroponte
IEEE International Conference on Communication, Boston, Massachusetts, June 10-14, 1979, Vol. 1, pp. 56.1.1 - 56.1.8

[Newell 1977]
Notes for a Model of Human Performance in ZOG
Allan Newell
Department of Computer Science, Carnegie-Mellon, August 5, 1977

[Nickerson 1981]
Why Interactive Computer Systems are Sometimes Not Used by People Who Might Benefit from Them
Raymond S. Nickerson
International Journal of Man-Machine Studies, (1981) 15, pp. 469-483

[Norman 1981]
The Trouble With Unix
Donald A. Norman
Datamation, November 1981, pp. 139-150

[O'Brien 1982]
Telidon Videotex Presentation Level Protocol: Augmented Picture Description Instructions
C.D. O'Brien, H.G. Bown, J.C Smirle, Y.F. Lum, J.Z. Kukulka
Government of Canada, Department of Communications, CRC Technical Note 709-E, Ottawa, February 1982

[Orsnaes 1982]
User Reactions to Keyword Access Videotex
Jorn Orsnaes
Videotex '82, New York, N.Y., June 28-30, 1982, pp. 315-320

[Otto 1980]
Access to Private Data Bases: An Essential Supplement to the Interactive Videotex System Concept
Jens Otto
ICCC, Atlanta, October 1980

[Owens 1979]
The "Soap Opera" Effect in Story Recall
Justine Ownes and Gordon H. Bower
Memory & Cognition, 1979, 7(3), pp. 185-191

[Pickersgill 1981]
An Assessment of the Production of Graphics on Telidon IPS
Peter Pickersgill
August 1981

[Rex 1983]
Telidon Info Centre Fails to Woo Seniors Into Computer Age
Kathleen Rex
The Globe and Mail, Toronto, Thursday, August 18, 1983

[Rice 1972]
An Introduction to Information Structures and Paging Considerations for On-Line Text Editing Systems
David E. Rice and Andries van Dam
Advances in Information and Systems Science, edited by Julius T. Tou, Vol. 4, Plenum Press, New York, 1972, pp. 93-159

[Robertson 1977]
ZOG: A Man-Machine Communication Philosophy
G. Robertson, A. Newell and K. Ramakrishna
Department of Computer Science, Carnegie-Mellon, August 4, 1977

[Robertson 1979]
The ZOG Approach to Man-Machine Communication
G. Robertson, D. McCracken and A. Newell
Department of Computer Science, Carnegie-Mellon, CMU-CS-79-148, October 23, 1979

[Rubin 1980]
Recall of Semantic Domains
David C. Rubin and Martha J. Olson
Memory & Cognition, Vol. 8(4), 1980, pp. 354-366

[Santo 1983]
An Eye on the CBS Database: A Case Study of Information Access Methods
Sherril L. Santo
Videotex '83, New York, June 27-29, 1983

[Sauvain 1979]
Computer-Based Personal Retrieval Systems
Richard W. Sauvain
SIGIR Vol. XIII, No. 4, Spring 1979, pp. 8-31

[Schabas 1981]
Optimizing Telidon Tree Structures
Ann H. Schabas
Proceedings of the Ninth Canadian Conference on Information Science, Pointe-Au-Pic, Quebec, Canada, May 27-30, 1981, pp. 175-182

[Schabas 1982]
Videotex Information Systems: Complements to the Tree Structure
Ann H. Schabas
Proceedings of the Fourth International Study Conference on Classification Research, Augsburg, June 28-July 2, 1982, Vol. 1, pp. 285 -291

[Schabas 1983a]
Trees and Forests: User Reactions to Two Page Access Structures
Ann H. Schabas and Frank Wm. Tompa
Videotex '83, New York, June 27-29, 1983, pp. 197-209

[Schabas 1983b]
Trees and Forests: Experimental Results
Ann H. Schabas and Frank Wm. Tompa
Supplement to Videotex '83, New York, June 27-29, 1983

[Seguin 1982]
A Study of Gateway Pages for Videotex
Marc Seguin
M.Math Essay, Department of Computer Science, University of Waterloo, December, 1982

[Shneiderman 1983]
Direct Manipulation: A Step Beyond Programming Languages
Ben Shneiderman
IEEE Computer, August 1983, pp. 57-69

[Singh 1982]
A Graphics Editor for Benesh Movement Notation
Baldev Singh, John C. Beatty, Kellogg S. Booth and Rhonda Ryman
Technical Report CS-82-41, Department of Computer Science, University of Waterloo, December 1982

[Sisson 1983]
Design Methodology for Menu Structures
Norwood Sisson, Stanley Parkinson and Kathleen Snowberry
IEEE 2nd Annual Phoenix Conference on Computers and Communications, Phoenix, Arizona, March 14-16, 1983, pp. 557-560

[Smith 1982]
A New Information Retrieval System: Got a Question? Just Touch the Screen!
George W. Smith
Bell Laboratories Record, October 1982, pp. 207-213

[Snowberry 1983]
Effects of Help Fields on Hierarchical Menu Search
Kathleen Snowberry, Stanley Parkinson and Norwood Sisson
IEEE 2nd Annual Phoenix Conference on Computers and Communications, Phoenix, Arizona, March 14-16, 1983, pp. 552-556

[Staisey 1982]
Videotex and the Disabled
Nancy L. Staisey, Jo W. Tombaugh and Richard F. Dillon
International Journal of Man-Machine Studies, (1982) 17, pp. 35-50

[Stewart 1980]
Prestel - How Usable Is It?
J. Stewart
Proceedings of the Conference in Human Aspects of Telecommunication: Individual and Social Consequences, October 1979, Munich, Springer-Verlag, 1980

[Sutherland 1980]
Prestel and the User: A Survey of Psychological and Ergonomic Research
Stuart Sutherland
Centre for Research on Perception and Cognition, University of Sussex, Brighton, May 1980

[Telidon 1981]
The Design of Videotex Tree Indexes
Telidon Behavioural Research 2, Government of Canada, Department of Communications, Behavioural Research and Evaluation, May 1981

[Thorndyke 1980]
Individual Differences in Procedures for Knowledge Acquisition from Maps
P.W. Thorndyke and C. Stasz
Cognitive Psychology 12, 1980, pp. 137-175

[Tompa 1981a]
Alternative Database Strategies for Videotex
Frank Wm. Tompa, Jan Gecsei and Gregor V. Bochmann
in *The Telidon Book*, edited by David Godfrey and Ernest Chang, Press Porcépic Ltd., Toronto, Ontario, 1981, pp. 226-247

[Tompa 1981b]
Data Structuring Facilities for Interactive Videotex Systems
Frank Wm. Tompa, Jan Gecsei and Gregor V. Bochmann
IEEE Computer, August 1981, pp. 72-81

[Tompa 1982]
Retrieving Data Through Telidon
Frank Wm. Tompa
CIPS 1982, Saskatoon, Saskatchewan

[Tsichritzis 1981]
The Death of the Computer Centre
Dennis Tsichritzis
Omega Alpha, CSRG-127, Computer Systems Research Group, University of Toronto, March 1981, pp. 235-240

[Vrins 1982]
Search Method Evaluation in the Dutch Videotex System
A.G.M. Vrins, R.H. Van Velthoven and J.L. Frankhuizen
Displays, April 1982, pp. 101-105

[Weyer 1982]
The Design of a Dynamic Book for Information Search
Stephen A. Weyer
International Journal of Man-Machine Studies, 1982 (17), pp. 87-107

[Wicklein 1979]
Wired City USA: The Charms and Dangers of Two-Way TV
John Wicklein
Atlantic Monthly, February 1979, pp. 35-42

[Wicklein 1981]
Electronic Nightmare
John Wicklein
Viking Press, New York, 1981

[Williams 1982]
RABBIT: An Interface for Database Access
Michael D. Williams and Frederich N. Tou
Proceedings of the ACM '82 Conference, Dallas, Texas, October 25-27, 1982, pp. 83-87

[Wynn 1980]
Computer Message Systems as a Unique Medium of Communication
E.H. Wynn
Proceedings of the 5th International Conference on Computer Communications, Atlanta, October 1980

[Yhap 1983]
Path Access Structure for Videotex
Christine B. Yhap
M.Math Essay, Department of Computer Science, University of Waterloo, 1983

[Young 1982]
Cognitive Aspects of the Selection of Viewdata Options by Casual Users
R.M. Young and A. Hull
Proceedings of the Sixth International Conference on Computer Communication, London, September 7-10, 1982, pp. 571-576

[Zloof 1982]
Office-by-Example: A business language that unifies data and word processing and electronic mail
M.M. Zloof
IBM Systems Journal, Vol. 21, No. 3, 1982, pp. 272-304

Appendix.

This appendix describes the implementation of a prototype data structure editor. The intent was not to create an editor with full capabilities, but rather to explore some of the facets of structure editing.

A.1 Programming environment.

The editor was written in C on a VAX 11/780 running UNIX. This environment was chosen primarily because it supports the local videotex system, UW/Telidon. Another example of a UNIX videotex implementation can be found elsewhere [Abell 1981].

The UNIX operating system is particularly suited to videotex databases. Files in UNIX are kept in a hierarchy similar to a videotex tree index. The ease of creation, modification, and manipulation of files makes a basic videotex system easy to implement. Each Telidon page is a file. A videotex tree index can be constructed merely by placing files in a UNIX hierarchy.

The editor structures collections of pages, and hence files. The user supplies the editor with a workspace, which is a UNIX file. The workspace contains a description of data structures and environment parameters, which is machine and user-readable, although terse. Workspace files can be modified with a text editor.

The terminals used for development and testing of the editor include the Volker-Craig VC 404, Ann Arbor Ambassador, and an Electrohome monitor for display of Telidon pages. These terminals provide only keyboard input, so the editor was designed with a command-line interface. This is a major disadvantage for any user, but sufficient for initial investigation. Full-screen control is possible with a system utility known as *curses*, but this capability was not explored.

Each page is a file known to the editor by its pathname. The editor manipulates collections known as page *stacks*. A stack is a LIFO structure; the most recently added item is the first to be removed. Each stack can have zero or more pages.

Users can also create pointers to stacks. A collection of pointers is known as a *menu*, and is in fact itself a stack. Each pointer is represented by an external label which is displayed on the menu.

A sample session will demonstrate the editor's capabilities. All editor responses are shown in boldface; user input is in italic.

An edit session begins with a request to specify a workspace:

workspace?
? *demonstration*

All previously defined stacks and environment parameters in the file *demonstration* will be initialized. If no parameters are specified, the defaults will be used. An example parameter is the depth of a multi-menu, modified with the command *level*. When *level* is specified with no parameters, it returns the current value.

? *level*
level 2

The default level is two. The structuring task demonstrated is that of organizing this thesis. A menu *contents* will point to various sections of the thesis.

? *make contents menu*

make contents menu creates a menu with the name "contents".

? *make part_one menu*
? *make part_two menu*

Three null menus have been defined. The next part of the structuring task is to organize them in a hierarchy. The two parts will be attached to the list of contents. *Save* is the procedure by which this is done. The first menu specified after the command verb is considered the superordinate.

? *save contents part_one*
Menu label?
? *Part 1.*

The editor needs an explicit label when an item is added to a menu. The label can be any string of characters terminated by a carriage return. The other part is now added.

? *save contents part_two*
Menu label?
? *Part 2.*

The menu *contents* is no longer null. It now contains two items, each of which is a null menu. The contents of each part can now be defined as page stacks:

? *make section_one*
? *make section_two*
? *save part_one section_one*
Menu label?
? *What is videotex today?*
? *save part_one section_two*
Menu label?
? *Problems with current videotex systems.*

A menu (or a stack) can be displayed by entering its name:

? *contents*

Select from the following:

1 : Part 1.
5 : What is videotex today?
6 : Problems with current videotex systems.

2 : Part 2.

A multi-menu of level 2 is displayed. Suppose there are two pages to be entered into section 1 (so far). They are the files *page1* and *page2*, located in the directory */u/drraymond/research/telidon/thesis*. These are now added to the index structure:

? *save section_one /u/drraymond/research/telidon/thesis/page1*

? *save section_one /u/drraymond/research/telidon/thesis/page2*

The stack *section_one* now contains pathnames for two files.

Obviously, specifying pathnames in full can be tedious. The editor provides a shortcut. The *prefix* command establishes a directory from which all files are taken. This is done by concatenating the prefix string with the specified filenames. So *prefix /u/drraymond/research/telidon/thesis/* would have simplified the previous commands to:

? *save section_one page1*
? *save section_one page2*

contents remains the same:

Select from the following:

1 : Part 1.
5 : What is videotex today?
6 : Problems with current videotex systems.

2 : Part 2.

If '5' is selected, the file */u/drraymond/research/telidon/thesis/page1* will be displayed. Hitting carriage return <cr> causes *page2* to be displayed. Repeated <cr> will continue displaying *page2*, as it is the last page in the stack.

A.2 Data structures.

The objects manipulated have a representation external to the editor and a representation internal to the editor. For example, a page is represented externally as a set of graphics codes kept in a file, but internally as a unique integer. Three tables are used to maintain the mapping between internal and external

representations.

The first table is the *stack name table.* Each stack and menu name is a distinct object resulting from a *make* command. Internally these are represented as integers, and externally as strings of characters. Each entry in the stack name table is a pair *(internal id, external id).* The *internal id* is a unique integer identifying the stack name; the *external id* is the string supplied to the *make* command.

The second table is the *stack contents table.* The contents of stacks and menus are kept in this table. A stack is represented as a linked list of items, each a page or a pointer to a page. Entries in the stack contents table have four fields: *(internal id, type, content, next).* The *internal id* is a unique integer identifying the member of the stack. *Type* specifies whether the member is a page or a pointer to a page. *Content* is the internal id of the page or pointer referenced by this member. *Next* is the next item in the list (if any).

The third table is the *filenames table.* Each entry in the filenames table is a pair *(internal id, external id).* Each page is represented here with its internal id and its pathname; each menu item is represented with its internal id and its label as used in a menu.

An example will illustrate the use of the tables. Assume the user enters *show forecast* where *forecast* is a stack of pages describing the weather. The stack name table is searched for the name *forecast* to find the stack's internal id. This number is used to access the first page in the stack, found in the stack contents table. The content of the first page is used to access the page filename in the filenames table, and this is shown on the user's display.

Each table is ordered by the internal id, and binary search is used to locate desired entries.

A workspace contains several environment parameters.

header page – a header page pathname is maintained. The header page is available to tailor initial output. When a workspace is initialized, the editor displays the header page (if specified).

level – this controls the depth of tree index displayed in a multi-menu. The default is two; users can set this to a finite positive integer value. With no parameter, the command returns the current value; when specified, the level is set to the parameter.

maximum internal id – for system use only. The largest internal id is kept. The next time the workspace is accessed, item numbering will begin from the next value.

current id – for system use only. The user may have been displaying a large stack before termination. The internal id of the last item displayed is maintained. When the session is restarted, display can continue from this item.

prefix – the current prefix is maintained. This is a directory from which files may be added to stacks. With no parameter, the command returns the current directory; when specified, the prefix is set to the parameter.

A.3 Commands.

The following command set is supported.

make name [menu] – a stack *name* is created. The editor ensures there is no previously defined stack *name*. If *menu* is specified, a menu *name* is created.

save name1 name2 – *save* is equivalent to "push *name2* onto *name1*". Menus can contain other menus or stacks; if *name1* is a menu, *name2* must be a stack or menu. Stacks are restricted to files; if *name1* is a stack, *name2* must be a file found in directory *prefix*.

delete name1 – deletes the last entry from *name1*. If *name1* is empty, it is deleted from the workspace. *Delete* is equivalent to "pop *name1*".

items – displays all stack and menu names.

prefix [pathname] – with no *pathname* specified, *prefix* returns the current prefix. If specified, the prefix is set to *pathname*.

level [integer] – with no *integer* specified, *level* returns the current level. If specified, the current level is changed to *integer*.

join name1 name2 – join exists because stacks can only contain files. Join is used to attach a menu *name2* to the bottom of stack *name1*. This is an ugly command and should be discarded.

display [name] – with no *name* specified, display prints the filename table. If *name* is specified, display prints stack content table entries for *name*. Used for debugging only.

As well as the set of explicit commands, there is a set of *implicit* commands. Each name is an implicit command. Entering the name of a stack causes display of the first page. <cr> causes display of successive pages. Entering the name of a menu causes display to the depth prescribed by *level*.

The editor as described has been in use for about four months. It has structured collections of Telidon pages, created its own *help* facility, and been used in preparation of this thesis.

A.4 Problems with the implementation.

The editor was written to uncover problems in interactive data structuring. Several were encountered.

type – trying to decide whether an atom of structure should be a stack or a menu was tiresome. In particular, the difficulty of rescinding such a decision made structuring difficult.

names – Malone contends that people often avoid the work of explicit categorization [Malone 1983]. Use of the editor provides further evidence in support of that view. Naming was always required, as objects could not be referenced by pointing. Short names were chosen whenever possible, in order to reduce specification effort. Further, there is a tendency to choose very general, non-specific names, simply because the final configuration is unknown. The combination of these factors resulted in an opaque set of names.

only half an editor – structure editing is only part of information processing. Typically, a user has partial knowledge of structure and content, but not a complete knowledge of either. Creating content in this environment required exiting the structure editor, starting up a text editor, creating a file, then re-accessing the structure editor. The structure editor should provide a transparent interface to an existing text editor [Fraser 1979].

stacks are not good enough – stacks were chosen because of ease of implementation. The poverty of stack-based structuring lies in the difficulty of correcting mistakes. If the stack is incorrectly ordered, all elements must be popped off to delete the offending item.

no display – other than an opaque *display* command which was designed for debugging, there was no way to look at the stack contents. Pictures would relieve a lot of short term memory load.

A.5 Good features.

Some good things can be said about the implementation.

simplicity – the editor took about six weeks to create. It deals with simple constructs in a simple way, and is unlikely to get much more involved. The editor is quite small, being made up of thirty-six simple modules (more than half of which are devoted to I/O).

easy to modify – prototypes are heavily modified. This editor was designed for ease of modification rather than efficient processing. It has undergone several revisions, and currently a graphics interface is being considered.

generally applicable – the editor's capabilities need not be confined to videotex.

UNIX provides a set of file utilities with general appeal. These are manipulated by the use of command lines to a processor known as the *shell*, which is in effect a rudimentary "structure editor". For example, a list of files can be specified in a command line:

```
lpr file1 file2 file3 ... filen
```

and output can be *piped* between utilities:

```
nroff -ms file1 | lpr
```

Each "structure" is linear, because that is the extent of the shell's capability. If a general structure editor were provided, more complex constructions could be imagined [Cargill 1979].

A.6 Conclusions.

No novice or casual user would ever want to use this tool as it stands. Two characteristics would make the editor more useful.

The first missing feature is a direct manipulation technique. Being able to look and point at structures, instead of trying to visualize and remember names would improve the interface enormously. If a mouse or puck were provided, operation of the editor would be much faster and more positive. The conceptual landscape would be augmented with a visual landscape.

The second quality the editor needs is to be *forgiving*. Structuring does not happen in a straightforward way. Users learn by making mistakes; they must have the flexibility to do so in the editor. Structures should be simple to modify, and errors must be tolerated. An important (but unimplemented) feature is *undo;* users should be able to produce a change and then rescind it if it is deemed undesirable.