Information Technology: Research and Development (1982), *1* (41-53)
© 1982 Butterworths

AN APPRAISAL OF FACTORS AFFECTING THE PERFORMANCE OF TEXT RETRIEVAL SYSTEMS

N. GOLDSMITH

Price Waterhouse Associates, York House, York Street, Manchester M2 4WS, UK

(*Received 14 November 1980, revised 29 April 1981*)

ABSTRACT

The paper examines the factors that influence the performance and limitations of full text retrieval systems, and attempts to provide sufficient background knowledge of design features to enable the reader to define selection criteria. In particular, the way in which the retrieval power of a system and the up-dating capability are affected by the database structure and file design is dealt with in some detail. It is suggested that, whilst not the most important criteria for selection, the technical features mentioned, in particular ease of data entry, are extremely important. Throughout the paper reference is made, for specific examples, to four currently available full text retrieval 'packages'. These are ASSASSIN, DOCU/MASTER, STAIRS and STATUS II. No attempt to provide a selection evaluation of these four is made, though the specific computer hardware requirements of each are detailed.

1. INTRODUCTION

The advent of computers with ever increasing power and large scale rapid access storage space together with the decline in costs of computer hardware, in cash terms as well as real terms, has aided the increase of computerised information management. Perhaps of more importance has been media coverage of the microchip and television's participation through video text systems. Most of the developed world is now aware of the computer's ability to handle information.

A topic of increasing interest has been the use of text retrieval systems. Computerised text retrieval has been a concept and a reality for some years but has not really been a workable proposition until recently, probably because of lack of sufficient computer power and because of a general lack of awareness, and hence commitment, by those most likely to benefit.

There are now many text retrieval systems available on the market. These come in all shapes and sizes: key word systems, in context and out of context; free text systems; complete 'bundled' units of retrieval system and hardware; and keyword or full text systems on computer service bureaux.

This paper will confine itself to discussing the available free or full text retrieval systems only. Full text retrieval systems are defined for the purposes of this paper as being systems that index, or are able to index, all words of the text database. Keyword systems are those that use only predefined keywords to index the text.

It is intended to discuss the design features that affect both performance and capability of the systems. This will be by reference to specific systems that may be of interest to potential purchasers of free text systems for their own (in-house) computer. The systems referred to are ASSASSIN, DOCU/MASTER, STAIRS/VS and STATUS II (version 80).

It is not intended to produce a 'blow by blow' evaluation of the systems leading to the conclusion that system X is the best. To evaluate the systems in such a way requires more space than this paper allows. Selection of a system ought to be made primarily from the user's viewpoint. In particular the command or enquiry system (the user interface) and its ease of use will be most important. Not unreasonably, technical considerations are often left to the last; yet they may be of crucial importance.

This paper will attempt to provide potential purchasers of free text retrieval systems and interested parties with sufficient technical background and knowledge of how the systems work to enable them to define more precisely their own selection criteria. No evaluation of the detail of the enquiry languages is undertaken. For the present readers will have to refer to the relevant user guides, in particular Croall (1980), the IBM STAIRS/VS program product information (1976) and that published by Turnkey Systems Inc. (1979).

The subject area is one of the most rapidly increasing areas of interest and development. It is only possible to state that the information provided in this paper is valid at the time of going to press! However, this paper should enable interested parties to evaluate whether any future developments are significant or only of minor importance.

2. FILE STRUCTURES

2.1 Overview

All the available full text retrieval systems use a fully inverted file structure. An inverted file is one where an index to the data is held in a separate file to the data. Reference will be made to the logical structure; the file design, and the physical structure; the effect of the logical structure on the hardware.

The prime requirement of a text retrieval system is that ultimately the source text can be retrieved and displayed. Thus all systems will hold a reference to the prime document that each word occurs in. The size and structure of the word reference differ from system to system. This has implications for the retrieval capability of the system. The method of mapping the reference list onto the physical storage medium, inevitably a disk, affects the performance of the system.

2.2 Database structures

It is possible with all the systems to have multiple unrelated databases. The number of these depends upon the method of loading the retrieval system onto the operating system, and this is outlined later in this paper. However, each system will allow an individual database (a grouping of text data that can be accessed by a single search) to be structured in some way.

Multiple databases. STAIRS allows up to 16 databases to be searched at once. Each database consists of documents which are in turn sectioned into paragraphs. STATUS only searches a single database but this can be sectioned into chapters which can be included or excluded from the search. DOCU/MASTER like STAIRS allows the data to be split into sub-databases, up to 198 in all. ASSASSIN works on a single database that consists of documents.

Pointer structure. If the reference structure is examined a slightly different picture emerges. The reference, or pointer, structure used by STAIRS is: document, paragraph, sentence, word. STATUS uses chapter, article, paragraph, word. DOCU/MASTER uses a sub-database and document and ASSASSIN uses document and word.

The pointer (reference) structure defines the logical access capability. Thus when STAIRS searches different databases it needs first to refer to a master control index, the Data Base Control Block, which points to the relevant first level index of each database. Each database index is held separately. Hence a search of *n* databases involves *n* times the number of logical file accesses that would be required if the indexes were held as one index. STATUS searches the whole index for each enquiry and removes references for chapters excluded from the search, whereas DOCU/MASTER accesses all its sub-databases through a single index.

Clearly at this level there are performance implications. If all the STAIRS databases are to be searched for an enquiry, then holding separate indexes will be slower than a single concatenated index. On the other hand searching single databases ought to be faster and more efficient than confining a search to a single STATUS chapter equivalent in size, though it does involve an extra logical access. As will be seen later, the physical structure will play a more important part in efficiency. It should be noted that the storage size will also be affected by multiple indexes.

Of more importance is the effect that the pointer structure has on retrieval capability and performance. Because both STAIRS and STATUS hold the position of each word within the document or article they can perform proximity, or collocation, searching of words. For example, the phrase 'Chancellor of the Exchequer' can be found and in addition documents with 'Chancellor' in the same paragraph or within so many words, before or after 'Exchequer' can be retrieved. It is not necessary to predefine phrases for searching.

DOCU/MASTER by the nature of its pointer structure must first have a retrieve (hit) list to operate on before it can perform proximity searches. This it then does by scanning the documents on the retrieve list. This requires the documents on the retrieve list to be accessed and searched sequentially for the appropriate word sequence. A sequential search is much slower and uses more resources than the operations performed on the four level indexes.

It is possible to search for phrases using document only pointers if the phrase is defined previously as a single word. This is possible with DOCU/MASTER. The differentiation of upper and lower case characters presents another point of interest. STAIRS treats 'CHANCELLOR' 'Chancellor' and 'chancellor' as three separate words. All of the other systems for searching treat them as the same. This may be an advantage or disadvantage depending upon the application and viewpoint of the user.

2.3 Index file structure

The accepted method of searching an inverted file structure for text is to use a matrix of first letter pairs pointing to a word list. Thus for a word beginning with the letters AB the system goes to the matrix and finds the relative address of all words beginning with AB. This technique is explained by Martin (1975), who refers to the matrix as a character pair matrix. The writers of the STATUS system call it a first level digraph.

This matrix is held as a fixed format fixed length field for any one database. Its size will vary with the implementation. The minimum size will be 26×26 multiplied by the pointer address length in bytes. If numbers are indexed using the same technique, as opposed to a separate index for numbers, then the first level matrix will be 36×36 multiplied by the address length in bytes. Obviously special characters will again increase the size of the matrix.

STATUS uses this first level digraph to point to a second level digraph. This second level digraph is another matrix, this time of second letter pairs. The second level digraph is created as words are added and is of variable length. It points to the word list, or concordance, as it is referred to by the STATUS authors.

The retrieval speed will depend upon whether the first level matrix is held in the real main memory of the computer or on disk. For most implementations of STATUS it is held on disk, as the core storage size of most of the computer it is implemented on is too small. However, on implementations on large IBM machines the matrix is held in core. STAIRS also holds the matrix on disk, reading it into core for each retrieval.

The index of words is held in the STATUS system in the order in which words have been loaded onto the database. The only predefined order in the index is the first level digraph. All of the other systems being examined sort the index into a predetermined sequence, usually alphabetic.

The size of the pointers obviously affects the size of the index file. In STAIRS it is fixed at eight bytes, or 64 bits. In STATUS it can be 16, 32 or 64 bits, the size being determined at database creation time and is, therefore, under the control of the user. The maximum number of pointers to an individual word in the database on both STATUS and STAIRS is $2^{64} - 1$ or 32 767, for any one database.

Text bases must be able to accommodate documents of variable length and unknown word frequencies. Therefore, the pointers lists will also be of unknown length. All the systems except STATUS use records of variable lengths to accommodate the word pointers. STATUS uses fixed length records, usually 256 bytes in length chained together.

3. FILE SIZES

3.1 Text:index ratio

In retrieval systems the penalty paid for the accessibility of data is storage. The more refined the system the larger the index file. Mapping more complex relationships requires more space. This truism is generally well known. What is not so well known is quite how much larger the index file can be. Work done by the A.E.R.E. Harwell shows that the index to text file ratios conform in general to a negative exponential function. For small databases, up to one million characters of indexed text, this ratio can be as high as 12 to 1. The ratio reaches unity at about 30 million characters of indexed test. This is shown in Figure 1.

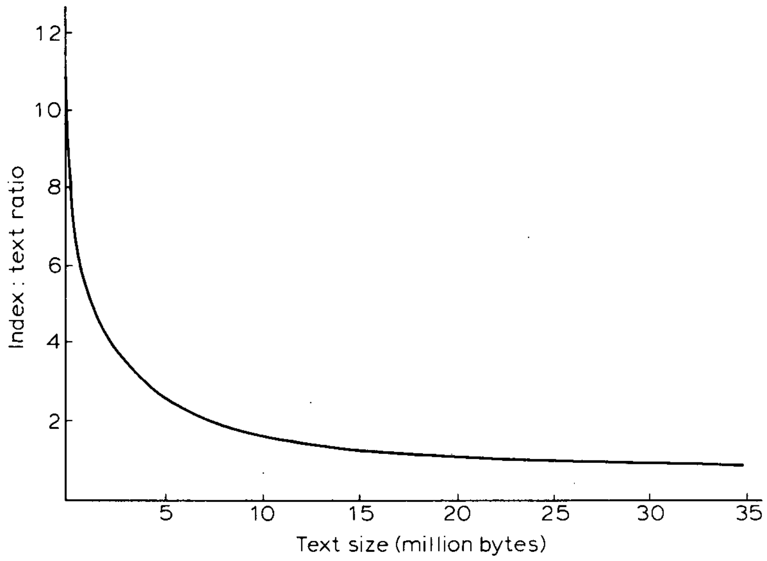

FIG. 1. Index to text ratio as a function of text size

The ratios given above relate to indexed text. It is possible, notably using STATUS, to choose not to index parts of a document. Thus in a database there may be sections of text within an article that it may be important to read on retrieval but it is not desirable to index. For example, the Hazardous Chemicals Database (HAZFILE) contains valuable information on first aid treatment; it is obviously undesirable to index this information as it may lead to false retrievals when enquiring to find out how to deal with a chemical.

If parts of documents are not indexed then the ratio of index to text is taken from the size of the indexed text. Thus to calculate the size of the index file of a database where only half the text is indexed, the ratio used is the one relevant to half the text size, i.e., it is a higher ratio than the size of the text file indicates.

Whereas the ratios for small volumes of text may appear at first sight to be alarming, they are an inescapable fact of word frequencies in language. The systems need first to set up each word of text in the index. For large databases the volume of new words added on each up-date will be small, the major addition will be the pointers, a maximum of eight bytes for each. Unlike numbers words can be considered to be a finite set, particularly in a defined application area. Needless to say the systems, STAIRS and STATUS, that index the word position within a document will use more index space. It is an act of faith, rapidly becoming justified, of these systems that the information value is worth the storage cost.

The Hazardous Chemicals database is an example that justifies this act of faith.

This system is used by basic grade control room staff in the Fire Brigades with no detailed chemical knowledge or expertise in information retrieval. It would not be possible to identify accurately and as quickly as they have done the nature of a chemical hazard without a system that identifies word position. Similarly, the EUROLEX System would not have gained acceptance so readily if the word position could not be indexed. As we are all well aware semantics and word position are a lawyer's stock in trade; 'car hits boy' is very different to 'boy hits car' though the same words are used.

3.2 Selective indexing

The obvious method of making the index to text ratio more favourable is to be more selective in indexing. So far, only STATUS allows for sections of text not to be indexed. The most usual method of being selective is by removing stop, or common, words from the index. This can be done safely without losing any information value. All of the systems discussed permit the use of stop or common words. Examples of such words are and, but, so, if, as, which do not have any information value *per se*; they act as joining words. They can also be words specific to a database which do not convey any information within that database. For example, 'LTD' in a database of companies may give no information.

Work done on word frequencies indicates that 40% of all text, and up to 60% in some cases, consists of words with no direct information value. Bishop and Smith (1980) found that in a database of 43000 documents containing 2591140 word references where all words in the database were indexed, 119605 references were to the word 'of'. Similar frequencies occur with 'the'. Selectivity of indexing in this way can be of prime importance to the size of the database. A table of common word frequencies from the same database attributable to Bishop and Smith (1980) is given in Table 1. Except for experimental purposes it is not advisable to include obvious stop or common words in a database. The author suggests that frequent careful thought is given to deleting words of no informational value.

3.3 Document storage

STAIRS and STATUS both use fixed block lengths to store text data, 1080 bytes and 256 bytes respectively (ASSASSIN and DOCU/MASTER use variable length records). A portion of each block is required for information defining the block; with STATUS, of the 256 bytes available, approximately 248 bytes are available for storing text. If for the sake of illustration an average document is 75 words long with an average word length of seven characters then the document length is 525 bytes (a small document). This will require three physical blocks with 243 bytes wasted. If the document size is reduced to 70 words it can be stored in two blocks with six bytes wasted. Wasted space is a remainder problem which becomes important when the document length is small. This can be of significance when a system is used as an index of documents, for example, where small abstracts of research reports are held and the system is used to find the relevant text which the user then obtains from the library.

Table 1. Frequency of words occurring more than 10000 times

Word	*Frequency*
Of	119 605
In	45 852
A	42 302
To	29 850
For	27 279
On	18 999
Is	17 938
Building	15 617
Research	15 318
Concrete	14 637
With	13 283
I	13 105
Are	13 078
From	12 446
By	10 909
Date	10 138

In this case care needs to be exercised by the database manager to ensure that the amount of wasted space does not become disproportionate.

4. PERFORMANCE

Designing a system, be it a straightforward batch system or a full text retrieval system, is a compromise. A balance needs to be made between machine performance and efficiency against power and facilities. This is not a clear balance and will depend upon the application. This is illustrated earlier by the sacrifice of storage space for power of retrieval. The illustration is even clearer when the requirements of up-dating capability and retrieval speed are considered.

The fewer the number of physical accesses needed to a file the faster the speed of retrieval. Thus systems which store pointers to a word next to each other on a disk will be faster in retrieval than systems that store pointers to the same word randomly over the disk.

Systems that store all the pointers to the same word next to each other on disc will have allocated a fixed amount of space to that reference list. When new references are added more space needs to be created. This is achieved by reading all of the index file merging in all the new references and writing the complete file back onto the disk. This is directly equivalent to a conventional data-processing sequential batch up-date. To do this is time consuming and uses a significant amount of computer resources. It also means that it is impossible to access the index to retrieve data during this up-date process. For this reason it is impossible to update inter-actively with these systems. (Up-dating is taken here to mean the addition and deletion of word references to the index.)

Most retrieval systems and particularly text retrieval systems opt for speed of retrieval; the most sensible side of the fence to fall. Of the systems being considered only STATUS allows on-line up-dating (though on some implementations of STATUS on-line up-dating is not allowed). STATUS achieves this capability by storing the word pointers in fixed length blocks, usually 256 bytes long and chaining new blocks to the existing blocks when the existing blocks are full. In doing this the system uses the next available unused block when more space is needed. Therefore, the blocks are not necessarily contiguous; in fact they seldom are. This chronological storing of pointers facilitates on-line up-dating because no re-ordering of the index is needed. For up-dating large volumes of text STATUS uses a batch operation that is similar to the other systems. In batch up-dating the normal method is to bring together all pointers to the same word and add them to the index at the same time. However, STATUS does not re-order the pointers and therefore if batch up-dating is used the pointers to any one word will consist of numbers of blocks distributed throughout the index in the order of their appearance in time.

With very large databases this method of up-dating can add to the retrieval time because to retrieve all pointers to a word with a high frequency the system has to make many physical accesses. On one large public database using STATUS as the retrieval mechanism on-line up-dating is not necessary, and retrieval speed and efficiency have been greatly enhanced by restructuring the index (concordance) so that all references to the same word are stored in contiguous blocks.

An application that is both large and requires on-line up-dating of small amounts of data must suffer the relative decline in retrieval speed for the sake of on-line up-dating.

5. DATA ENTRY FORMATS

To be worthwhile any information system must have enough data in it to provide a reasonable possibility of a successful enquiry. Insufficient data are the most common obstacle to the acceptance of information systems by would-be users. It is of paramount importance that any computerised information retrieval system provides an easy data entry mechanism so that the implementor can ensure that a database of sufficient size is tested for acceptability and approval.

Not surprisingly many potential applications fail at this stage. Usually the instigator is too involved with the prime task, or the organisation is not prepared to find the resources needed to start the project. It is often a massive task simply to collate the data, and, therefore, expensive in human resources. After having found the time, effort and resources to prepare and collate the data, the organisation is then presented with the further difficult exercise of entering the data.

The data formats required by a system can affect these tests, as can the required mode of data entry. Many companies now prepare research reports on word processing equipment; ideally they do not wish to have to re-enter the data into the text retrieval system. They want, instead, to be able to transfer the word processing files onto the text retrieval system.

The data entry formats required by the different systems serve several purposes. They discriminate between documents, define to the system the indexing markers such as paragraphs and they define sections of documents that can be output on retrieval, e.g., titles.

For example, STATUS provides several types of display and indexes the position of each word within paragraphs. To do this it requires each document to be prefixed with a ££T to mark the start of the document. Its end is also designated by a ££A. A truly basic STATUS document consists only of ££T to start and a ££T, ££A to end. However, within documents it is possible, though not necessary, to section the data further. In the basic case a document consists only of a title. It is possible to specify varied sections both for retrieval and display purposes. These are designated by ££N

'Name of Section' where the 'Name of Section' is up to eight characters long. Paragraphs within sections are designated by ££P.

The only other data entry requirements are that no data appears on lines used for text markers and that the data is input in 80 byte fixed length records (card image). Formatting of text needed for output necessitates the use of a line forcing character '<', a right chevron. In this way documents where the layout is important, for example poetry and addresses, can be accommodated. This is illustrated below:

Twinkle, twinkle, little bat!<
How I wonder what you're at!<
Up above the world you fly,<
Like a tea tray in the sky.<

Although the data have to be entered in 80 byte records the system will present the document in a wider format if the terminal, and terminal handling software, will allow it.

Both ASSASSIN and DOCU/MASTER place similar minimal data entry restrictions on the user. They also allow displaying of sections of documents. It is worth noting that DOCU/MASTER and STATUS allow on-line data entry.

The position with STAIRS is not so clear cut, though it is being revised. At present there are two methods of entering data. The user can either:

1. use the IBM program product — Advanced Text Management System (ATMS) which formats the data and puts them into a queue for up-dating, OR
2. format the data themselves and load them into the appropriate up-date queue.

If method (1) is adopted, the user still needs to designate paragraphs using 'Tags'. ATMS is basically a direct data entry system for text. It only runs under the IBM Customer Information Control System (CICS).

If the user adopts method (2) to enter data into STAIRS the user has a difficult task as the data entry formats are unusual. Each document is loaded into fixed length records of 1080 bytes. Text is input in lines no more than 72 bytes long; the first three bytes are reserved for paragraph markers. Paragraph markers can be numeric or alpha-numeric though the first character must be numeric. Paragraphs cannot be more than 54 lines of text. Each record has to have a document header and the length of the document in binary!

The implicit assumption of STAIRS is that the database manager will have the total commitment of all the necessary resources available on site to set the database up. This is seldom, if ever, true. The initial data preparation load usually represents a high proportion of the total amount of data loaded onto a database. Text databases rarely grow in a uniform manner; there is inevitably a large backlog of data to be loaded and additional data are usually added in steps. Therefore, it is not unusual for users to want to contract out the data preparation work. Indeed a well run computer installation is unlikely to have the excess capacity in terms of key punch staff. Moreover, the data processing department may not wish to use some specialist data entry software, which will use on-line resources and require support.

6. UTILITIES

Each system will have utilities for performing tasks outside of the day to day enquiry and retrieval activity. These will vary between systems in their ease of use, capability and availability. There are too many of these to discuss; the STATUS library contains approximately 20. Criteria to be considered are given below. They do not differ greatly from those for any database management system.

Updating. How easy or complex are the routines for batch operation? Are there error checking routines? Does the system need the interaction of several utilities?
Security and Recovery. Do special routines exist or is the system dependent upon those normally resident on the machine? Can security 'dumps' be taken? What happens if the machine fails during update?
Sizing. Is it possible to monitor the size of the database and the space utilization?
Extension. Can the database be increased in size or does the system require that space is allocated once and for all?
Error correction. At what level can error correction be made if at all?
Text source output. It may be necessary to add the text to an existing database on another machine; can the text be output with all the necessary markers?
Housekeeping. Word frequencies need to be monitored (this can also be part of the retrieval system); are utilities provided for this? It is desirable to be able to add new stop words (common words) to maintain the database efficiency. What is available, even in extreme cases when databases become corrupt? Can you dump the files in a readable form? Can you amend them?

7. HARDWARE CONSIDERATIONS

A paraphrase of this section could be how much does it cost and will it fit. The costs will need to be verified because they are prone to change particularly in the market conditions prevailing today.

The reader will also need to remember that in order to run any of the systems in addition to the potential purchase price of the system and the computer, the relevant operating system and terminal handler will be required. These extra costs will be significant. On mini-computers the operating system includes a system for handling the computer terminals. With large machines this is often a separate system and is

Table 2. Key factors

	ASSASSIN	*DOCU/MASTER*	*STAIRS*	*STATUS*
Cost	£6900 rental/year (inc. maintenance) £20 850 over 10 yr period + maintenance	£23–31 000 + maintenance	£137/month	£17–25 000 + maintenance
Minimum computer size	150 K bytes main memory + disc drive	IBM 370/115	IBM 370/135	96 K bytes + disc drive
Operating environment	Any supporting COBOL with inter-active capability (on-line version)	IBM DOS/VS or OS/VS with CICS or TASKMASTER	IBM DOS/VS or OS/VS supporting CICS or IMS	Any supporting FORTRAN IV with interactive capability
File type required	BDAM	BDAM	BDAM	BDAM

likely to exceed the rental costs of the text retrieval system. Equally on large computers the installation may already have the requisite terminal handler.

A summary of costs and requirements, but excluding additional costs such as operating system rental, is given in Table 2.

To some extent the questions of machine requirements are made irrelevant if the organisation does not have or does not intend to purchase IBM, or IBM compatible, computers. Both DOCU/MASTER and STAIRS at the time of writing are only available for large IBM machines. There is no reason why, in principle at least, the writers and vendors of these systems should not make them available for smaller IBM machines and in the case of DOCU/MASTER for non-IBM machines.

A restricted version of IMDOC, on which DOCU/MASTER is based, called FACTFINDER, is available on Data General mini-computers. It is part of an integrated word processing system. It only just classifies as full text and has not been discussed partly for this reason but mainly because it effectively 'disappeared' from the market place and at the start of writing this paper no detailed information was available. The system has re-emerged and the interested reader can get further information from the vendor: Mini Computer Systems Ltd.

7.1 ASSASSIN

This system has been implemented on IBM, ICL and Burroughs machines. It is written mainly in COBOL, and, in theory at least, there is no reason why it should not be implemented on any machine with a COBOL compiler. It needs the machine to make available approximately 150 K bytes of core storage. This can be real or virtual.

The IBM version works under CICS or TSO (time sharing operation). It, in common with the other systems, requires a disk drive on the system for the on-line version. It is available from ICI Agricultural Division who wrote and lease the system (the system is leased over a 15 year period).

7.2 DOCU/MASTER

The system is only available for IBM type machines running under the DOS/VS or OS operating systems. It also requires a communications handler, TASKMASTER or CICS. The minimum machine configuration will be decided by the smallest machine configuration needed to support the operating system. The system is provided by Turnkey Systems Inc. It is a packaged version of the IMDOC system developed in Sweden.

7.3 STAIRS

This is a program product of IBM and as such is only available for IBM machines running under DOS/VS or OS operating systems. It requires either CICS or the Information Management System (IMS) to act as a terminal handler.

If CICS is used an extra 200 K bytes needs to be added to the memory allocated for CICS, and each terminal will take 8 K bytes for its work space. (Note CICS normally occupies 1.5 million bytes of storage.)

The minimum machine configuration quoted by IBM is an IBM 370/135 control processor with a 2314 disc unit. The minimum size requirement is defined by the operating environment, CICS or IMS, as well as the STAIRS system.

7.4 STATUS

The system is written in FORTRAN IV and only the programs that interface directly with the operating systems, such as the disc control interface, are written in assembly code. The system was designed to be transportable and can be implemented on any machine with a FORTRAN IV compiler. Clearly some of the routines where the system interfaces with the operating system will be implementation dependent.

The proof of the portability is in the implementations. So far STATUS has been implemented on Burroughs, DEC, ICL, IBM, PRIME, UNIVAC and XEROX computers. The smallest machine used is an ICL 1904 (32 K words).

It must be noted that the ICL version is different in some respects to the other implementations of STATUS, and potential ICL users should make themselves aware of the differences; on-line updating is not available, interrupt trapping is.

STATUS was written by the Computer Science and Systems Division of the A.E.R.E. Harwell, who market the system through franchise holders.

8. GENERAL CONSIDERATIONS

The selection of a text retrieval system, or any system, will depend upon many factors not examined in this paper. Its ease of use, the support offered, and the development profile of the system are of greater importance than the factors considered here. Additionally, the experience base of the organisation may lend itself to the easy implementation and acceptance of one system over another.

In all cases the best method of selection is by trial. It is easy for the author to state glibly here that system X in his experience is the best. The reader can only verify that experience by repeating it.

The command/enquiry systems have not been discussed in this paper. They do differ markedly and the individual user's attitude towards the systems provoked by its dialogue will also differ. The author does intend to examine the systems *in toto* in a later work, though again these are best tested by demonstration.

In formulating an opinion of the qualities of each system the reader should not overlook the more usual 'data processing' selection criteria: the 'track record' of the system for resilience; its user base; the degree of support offered; error reporting systems; the response by the vendor to queries. In short is it a well 'packaged' product?

REFERENCES

Bishop, J. D. and Smith, G. J. (1980) Experiences with a STATUS database. *Software Practice and Experience.* (In press.)

Croall, I. F. (ed.) (1980) *STATUS Information Retrieval System User Manual.* Harwell: Computer Science and Systems Division A.E.R.E. Harwell.

I.B.M. (1976) *Storage and Information Retrieval System/Virtual Storage (STAIRS/VS) General Information. Program Product 5740-XR1.* London: IBM.

Martin, J. (1975) *Computer Database Organisation.* Englwood Cliffs: Prentice Hall.

Turnkey Systems Inc. (1979) *DOCU/MASTER System Description TM21-1.* Norwalk: Turnkey Systems Inc.