The operation of a model self organising data management system

P. A. Dearnley

School of Computing Studies, University of East Anglia, Norwich NOR 88C

The concept of a self organising data management system is summarised and operational data from a model of such a system is presented.

(Received October 1973)

The self organising data management system

The self organising data management system is designed to serve the interests of a variety of users, possibly remote and unco-ordinated, whose usage of the database either cannot be completely determined at system design stage or will change as the database and users' interests develop. In such a system the structuring of files occurs at access time or as the result of a forecasting procedure predicting future accesses, but not usually at data load time. The system attempts to minimise overall cost and gives potential users cost quotations, prior to carrying out their tasks. A further feature of the system is that the duplication or partial duplication of the same data in files of different structures and generations is used to balance access costs against storage costs.

The system is based on the following principles:

1. The system has the capability to determine and implement suitable structures for the files held in the database. Structures are chosen with the object of minimising the total cost of known or predicted accesses.
2. The access strategy adapted is constructed by the system.
3. Any correctly specified task can be completed by adopting some access strategy and the user is given a cost quotation in advance.
4. The system can restructure or update files as a result of (*a*) an accepted cost quotation or (*b*) by observing patterns of usage and predicting that a different structure will be economically advantageous to the body of users.
5. The user is allowed to leave requests for tasks in the system in the hope that batching or structure changes will eventually reduce the cost of his task to an acceptable level.

The original concept of a self organising data management system is described in more detail in Stocker and Dearnley (1973).

The model system

To validate the ideas behind the self organising data management system a working model has been constructed. The model implements principles (1) to (4) above, in some form. It is used to demonstrate that a viable data management system can be built along self organising guidelines and to obtain operational data for such a system. The operational data obtained is described in this paper.

The model system supports simple file structures including serial, sequential, indexed sequential and random organisations with the appropriate access methods. The system is able to convert files from one structure to another, and is also able to create new files both by extracting sub-records containing selecting fields and updating existing files. The model system is described in more detail in Dearnley (1973).

The two mechanisms which are of main interest in the system are *route finding* and *folio management*. Route finding occurs in response to a user request for access to the database. The activity involves choosing the files, access methods and reorganisation algorithms required to satisfy the user request. Folio management is the creation of new files or the restructuring of old files for the overall benefit of the users rather than as a result of one particular user's request. Folio management is undertaken when the system would otherwise be idle. It involves reviewing the usage of a particular corpus of data or 'folio' and deciding if some particular file could be created which would lessen the overall cost of future accesses. The examples given in the following sections illustrate both route finding and folio management.

Example 1: *Usage leading to an extended route*

Routes involving the choice of a single file and the appropriate access method are referred to a 'simple' route. Routes involving the use of several files and the appropriate access methods may be cheaper in certain circumstances; such routes are referred to as 'extended' routes. This example shows the advantageous use of such a route.

A folio* containing eight fields was defined. The definition is given in **Table 1.** A series of requests for access were made of the system. The characteristics of the requests are given in **Table 2.** The requests were interspersed with periods of idle time. During the idle time folio managment was undertaken. The requests were satisfied with serial searches of the one original file in the folio. The folio management module predicted that the sorting and indexing of various versions of the original data would be advantageous. The cost of the sorting being met out of the saving of subsequent indexed searches over serial searches.

The result of the folio management was the creation of seven new versions tailored to the first seven request types given in Table 2. At this stage the folio was represented by the eight versions listed in **Table 3.** When requests of type 8 in Table 2

Table 1 Test Folio for Example 1

Folio definition—Forestry data of specimen trees

FIELD	TYPE	NOTES
1. Area	Alphanumeric	geographical area code
2. Date	Numeric	date tree planted
3. Height	Numeric	tree height in feet
4. Width	Numeric	tree width in inches
5. Location	Numeric	geographic location code within plantation
6. Address	Alphabetic	address of plantation
7. Type	Alphabetic	tree description
8. Reference	Numeric	dossier reference

*The term file is avoided since, as the example later shows, one body of data or 'folio' may be represented by a number of files or 'versions'.

Table 2 Request types for Example 1

1. Given	date	return	location
2. Given	area	return	height, width
3. Given	area	return	date, height
4. Given	date	return	height, width
5. Given	area	return	date, width
6. Given	location	return	reference
7. Given	date, height, reference	return	area, width, address, type
8. Given	area, date	return	height, width

Table 3 Versions created in Example 1

version 1	fields 1, 2, 3, 4, 5, 6, 7, 8	serial	
,, 6	,, 5, 8	sorted and index on field	5
,, 8	,, 2, 5	,, ,, ,, ,, ,,	2
,, 10	,, 1, 3, 4	,, ,, ,, ,, ,,	1
,, 12	,, 2, 3, 4	,, ,, ,, ,, ,,	2
,, 14	,, 1, 2, 3	,, ,, ,, ,, ,,	1
,, 16	,, 1, 2, 4	,, ,, ,, ,, ,,	1
,, 18	,, 1, 2, 3, 4, 6, 7, 8	,, ,, ,, ,, ,,	8

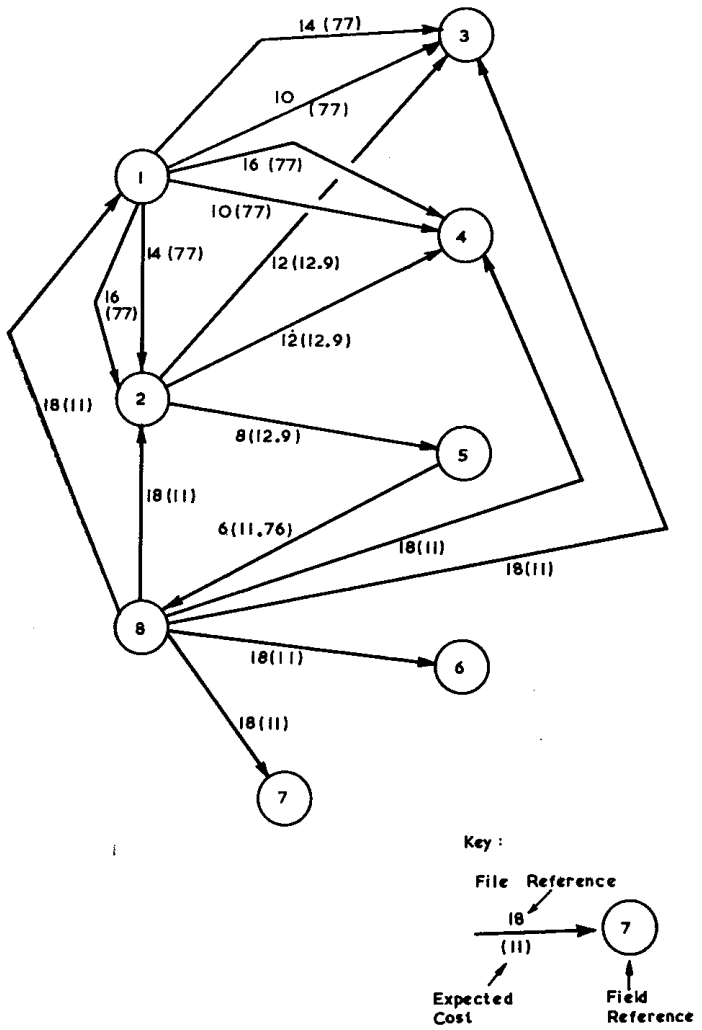

Fig. 1 Graph of versions in Example 1

occurred again they were satisfied by an extended route. The route finder had the choice of using versions 1 or 18 for either simple routes or for the terminal versions of extended routes. The versions 8, 10, 12, 14 and 18 were all potential starting places for an extended route, in that they contain one or more of the key fields to be matched. The graph in **Fig. 1** illustrates

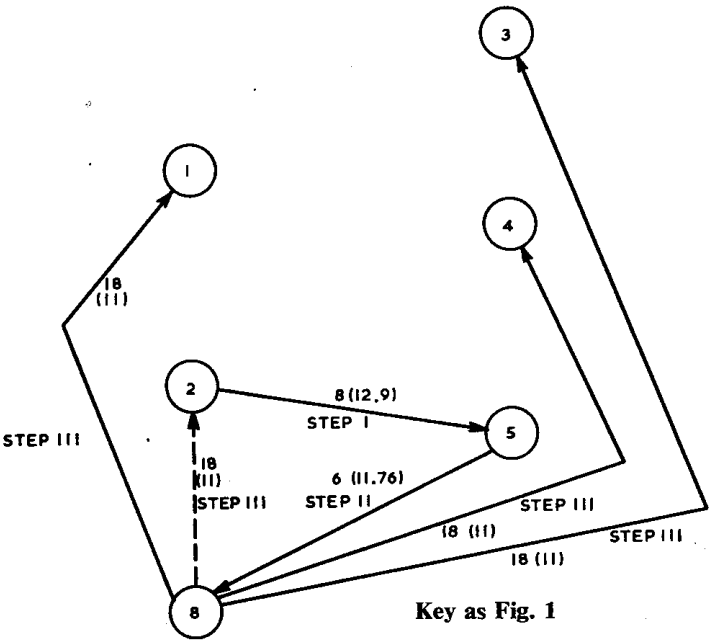

Fig. 2 Subgraph chosen by route finder in Example 1

the position. The nodes represent fields and the arcs represent access paths from key nodes to non-key nodes. The version facilitating a particular path is recorded on each arc. In addition, as the graph was used, each arc was weighted by the cost of using that arc. The costs cannot be determined exactly and were estimated at each stage. For example the cost of using an intermediate arc from node 2 to node 5 depended, in part, on the number of items found in version 8. This, in turn, depended on the number of items produced by the search going from node 1 to node 2 or node 8 to node 2. The route chosen by the route finder is shown on the graph in **Fig. 2.**

The route is as follows:

1. Define temporary files K, L, M.
2. Search version 8 for the user supplied values of field 2 producing values for field 5 in file K, carry forward values of both field 2 and field 1.
3. Search version 6 for the values of field 5 held in file K, producing values for field 8, in file L, carry forward values of both field 2 and field 1.
4. Search version 18 for values of fields 8, 1 and 2 held in file L producing values of fields 3 and 4 in file M, include the matched values of fields 1 and 2 in file M.
5. The required output is in file M, delete files K, L.

The route chosen was not an obvious choice when considering the range of alternative versions. The costs of using various arcs are shown in brackets in Figs. 1 and 2. If an arc is chosen then all other fields in the same file are available without further cost. For example choosing the arc from node 2 to node 4 given by version 12 makes node 3 available at no extra cost. Thus the cost of the chosen route is $12{\cdot}9 + 11{\cdot}76 + 11 \simeq 36$. Alternative simple routes would involve using version 1 or 18 at a cost of 154. The alternative of sorting version 1 or 18 and performing an indexed search was rejected because of the high initial cost of the sort.

The operation of the model has shown that machine costs are dominated by the time taken to access secondary storage, thus the unit of 'cost' used is a disc access. In this example the cost of occupying additional areas of secondary storage is ignored. (This factor is considered in examples 2 and 3.) Further, the cost of route finding was not included. The gross saving of the extended route over a simple route was $154 - 36 = 118$. The

Table 4 Folios for Example 2

Folio 1
1. CACM review category
2. Document reference number

Folio 2
1. Document reference number
2. Title of document
3. Author of document
4. Journal name, volume and number
5. Date published

route finding involved making disc accesses for 1 folio definition and 8 version definitions, a total of 9 disc accesses. Thus the net saving is 109.

Example 2: *Usage with a basic theme plus other random enquiries*
The purpose of the second example was to test the system on a large number of requests occurring in a limited system lifetime. In the previous example the lifetime of the system was assumed to be very long and thus new versions made by the folio management routine would eventually prove to be economic. In this test a short lifetime was specified and thus it was possible that the folio management would be unable to create new versions economically.

A main theme was assumed for the requests, and other requests with randomly selected characteristics were made. The folios used are described in **Table 4.** Folio 1 had one version only which was indexed on CACM category. Folio 2 started with one version indexed on document reference number. This initial position represented the file structures which a user might have employed if he had used a conventional data management system. The basic theme of requests assumed that most requests would be 'given CACM category recover document reference number' followed by 'given document reference number recover document details'. These main types of requests were satisfied by indexed sequential searches; other requests were performed by a serial search of the appropriate version.

Folio 1 was only used for searches on CACM category as the key field, thus the cost of using this folio was the same as that which would have been incurred by a conventional system. Thus the comparison of interest is that of folio 2 against the conventional equivalent.

Sixty sets of requests were produced. The appropriate key and other field references were chosen with the aid of tables of random numbers. The frequency of the key 'document reference number' was biased to reflect the main expected area of usage, i.e. retrieval by document reference number. Each set of requests used one key field and required the return of one, two, three or four other fields. The number of requests per time period was also determined by random number tables.

The model system was run with the sixty chosen sets of requests with a spell of idle time after each time period. The results were compared with the use of a static conventional system. The results are summarised in **Table 5.** Until the end of time period 5 the systems used the same combinations of serial and indexed searches. At the end of time period 5 a second version was made containing all 5 fields but ordered and indexed on field 3. The cost of this operation was 2657 thus the cumulative cost of the self organising system is increased at the end of time period 5. The use of this new version yielded lower costs in time periods 6 to 12 giving a gross saving of 6936. But some 105 extra blocks on disc were required for the second version by the self organising system for 7 time periods (from 6 to 12). The cost of this must be offset against the saving in cumulative operating cost. To mix operating cost in disc accesses with extra disc space it is necessary to have some conversion factor. It was assumed that an 8 million character disc pack cost £100 and that it had a life of five years. It was further assumed that the time period represented in the test covered a total of three months, thus the cost of an extra block of storage was approximately 0·003 pence per time period. For disc accesses a time of $\frac{1}{10}$th of a second per access was assumed and for machine time a cost of £20 per hour; thus the cost of an access was approximately 0·06 pence.

Hence the equivalent in 'disc accesses' of 105 blocks for 7 time periods was approximately 37. Thus the gross saving was reduced to 6899. An allowance must also be made for the cost of route-finding, for periods 1 to 5 there is only one version and one folio definition accessed for every search. For time periods 6 to 12 there are two versions definitions and one folio definition. Each review by the folio management routines requires accessing the search statistics for every version of the folio. Thus the total allowance required was 187. Hence the net saving was 6712. This represented a net saving of 35 per cent over the cumulative cost of the conventional system. The effect of the adjusted cost estimates is shown in **Fig. 3.**

Table 5 Results of Example 2

	CUMULATIVE OPERATING COST	
TIME PERIOD	CONVENTIONAL SYSTEM	SELF ORGANISING SYSTEM
1	746	746
2	1926	1926
3	3327	3327
4	3619	3619
5	6613	9270
6	7198	9535
7	9195	10412
8	10908	10445
9	12898	11322
10	15409	11513
11	18753	12217
12	19211	12275

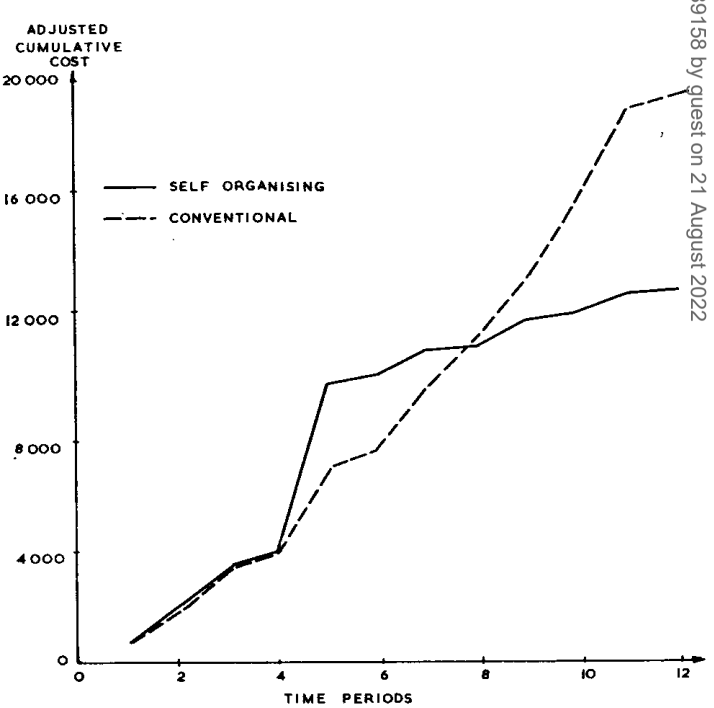

Fig. 3 Operating cost for Example 2

Example 3: Usage with seasonal variation in enquiries

The third example is based on the file used by universities to keep track of student applications. The file is created and maintained by a central authority and copies are sent to the universities at regular intervals through the year. The use of file changes as offers of places are made to students, as students react to the offers, when GCE 'A' level results are published and so on.

The actual file used consists of one record per student containing his personal and school details with a repeating group embedded in the record containing an entry for each application made for a place at university. The record length is 240 characters and the records are held in a sequential file keyed on a centrally allocated reference number. The file builds up very rapidly in the first two months, then is used for amendments (e.g. offers made, refusals, etc.) and interrogation (e.g. reply to offers). The vast majority of amendments are to the application repeating group entries; the personal and schooling details remaining largely unchanged.

For the test a reduced number of records and fields was used and the record stucture was normalised. It was also assumed that updates were sent to the university in place of a new master file. Folio 1 was used to hold the relationship between student reference number and personal/schooling details. Folio 2 was used to hold the relationship between student reference number and a single application. **Table 6** gives the folio definitions used.

Whilst the original file is designed to provide information about students it can also be used to get information about types of offers made by other universities and for reviewing the offers made for particular courses.

The various types of enquiries made are given in **Table 7.** The communication enquiries are used when writing to candidates to offer interviews, to offer places, to confirm offers, etc. The interview enquiries are used prior to selection interviews to give information about the current states of other applications. Strategy position enquiry types allow departments to choose a competitive offer strategy and to observe the results of offers. Clearing is used after GCE 'A' level results to try to allocate unplaced students to vacant places. The general school and personal profiles of candidates applying for particular universities and courses are given by review type enquiries. The frequency of the various enquiry types is given in **Fig. 4.1** and **4.2.** The various enquiries were performed by system with a period of idle time for folio management at the end of each time period. The cumulative costs of performing enquiries, creating new versions and occupying further disc storage were recorded.

Table 6 Folios used in Example 3

Folio 1 Students

FIELD	TYPE
Reference Number	Numeric
Name	Alphabetic
Address	Alphabetic
Sex	Alphabetic
School code	Numeric
School Type	Numeric
Age	Numeric

Folio 2 Applications

FIELD	TYPE	NOTES
Reference Number	Numeric	
Preference Rating	Numeric	Indicates the candidate's order of preference for applications
University reference number	Numeric	
Course Code	Numeric	
University Decision	Alphabetic	
Conditions on offer	Alphabetic	Indicates the GCE 'A' level grades required of the student by the university
Conditions expressed in 'points'	Numeric	As above but expressed numerically with each grade given a weighting
Candidate Reply	Alphabetic	

Table 7 Enquiry types for Example 3

ENQUIRY TYPE	SPECIFICATION
Communication	Given student reference number get name and address.
Interview	Given student reference number get all application details.
Offer Strategy	(*a*) Given university number and course number get university decision and conditions. (*b*) Given course number get university number, decision and conditions. (*c*) Given university number and course number get preference rating.
Recruitment Position	Given university number, course number and decision get candidate's reply.
Review	Given university number and course number get reference number. Then from reference number (*a*) Get school areas (*b*) Get school types (*c*) Get sex distribution (*d*) Get age distribution.
Clearing	Given course number and university decision = 'reject' get reference number.

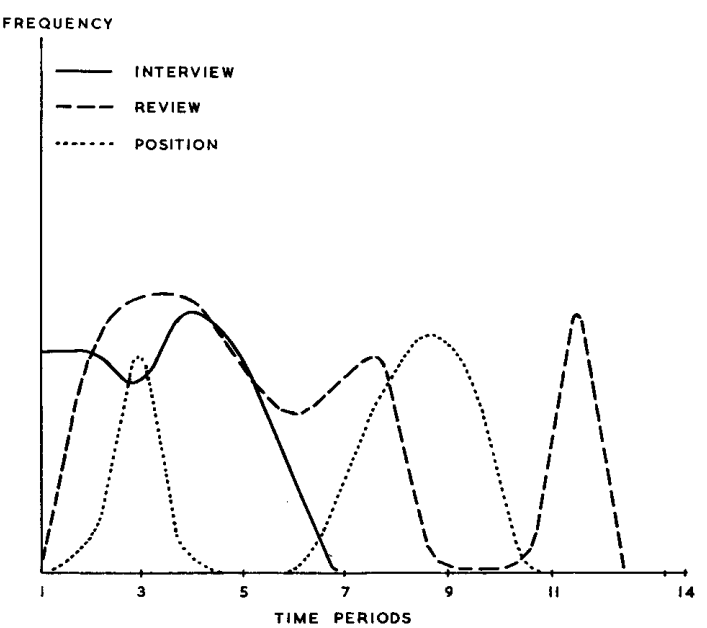

Fig. 4.1 Frequency of enquiry types in Example 3

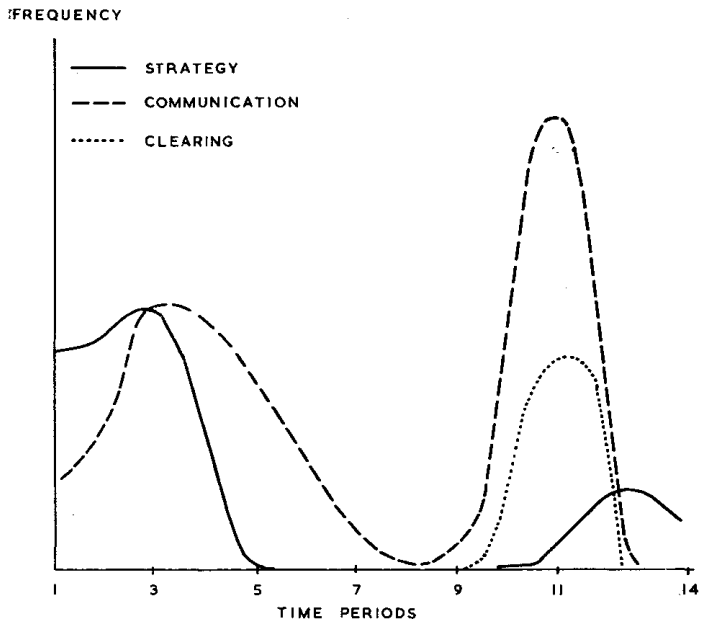

Fig. 4.2 Frequency of enquiry types in Example 3

These costs are shown in **Fig. 5.** The approximate cost of answering the same enquiry pattern with a conventional system from one file, sorted and indexed on student reference number, was calculated for each time period. Similarly the costs with a conventional system and two files (one of reference number with personal/school details, the other with reference number with application details) were calculated. The comparisons are given in **Fig. 6,** and the differences in cost between the self organising system and the two conventional equivalents considered is given in **Fig. 7.**

Enquiries using folio 1 were always satisfied by an indexed search of the one version of the personal/schooling data. Similarly interview type enquiries for folio 2 were satisfied by an indexed search of the original version of the application data which was keyed on student reference number. At the end of the first time period a version of folio 2 was constructed containing the fields, University number, Course number, University decision and conditions. This version was an indexed sequential file keyed on course number. This version was used in later time periods to meet strategy type enquiries. At the end of time period two a version of university number, course number, university decision and candidate's reply was constructed. Again this was an indexed sequential file keyed on course number. Further position type enquiries were met using this version. At the end of the third time period an indexed sequential file of course number, university number and student reference number was constructed. This was used to give access via the student reference number field to folio 1 for review type enquiries. During time periods four to eleven folio 2 was represented by four versions:

(*a*) the original application data used for interview enquiries,
(*b*) a version linking course and university to offer details,
(*c*) a version linking course and university to candidate's decisions,
(*d*) a version linking course and university to personal/schooling details.

In retrospect one may note that versions (*b*) and (*c*) above might usefully have been combined, indeed if folio management had not occurred at the end of period one then this would have been the system's decision also.

Finally at the end of period eleven a further version is made to handle clearing enquiries. The predictive algorithm in the folio management forecast that clearing would continue in periods thirteen and fourteen; in fact clearing finished in period twelve, thus this 'investment' was not worthwhile.

The cost comparisons cover only interrogation, file creation and storage costs; the processes of amending records and inserting new records are not covered.

The file grows quickly in the first two time periods. Insertion of new records keyed on student reference number occurs at the end of the files, thus this is not a problem in either conventional alternatives or in the self organising system. However in period two the self organising system has an additional version keyed on course number. Since the approximate final file size is known in advance this file can be constructed with partially packed blocks and insertions made *in situ.*

Amendments present more of a problem. Every application record is amended twice, once with university decision and

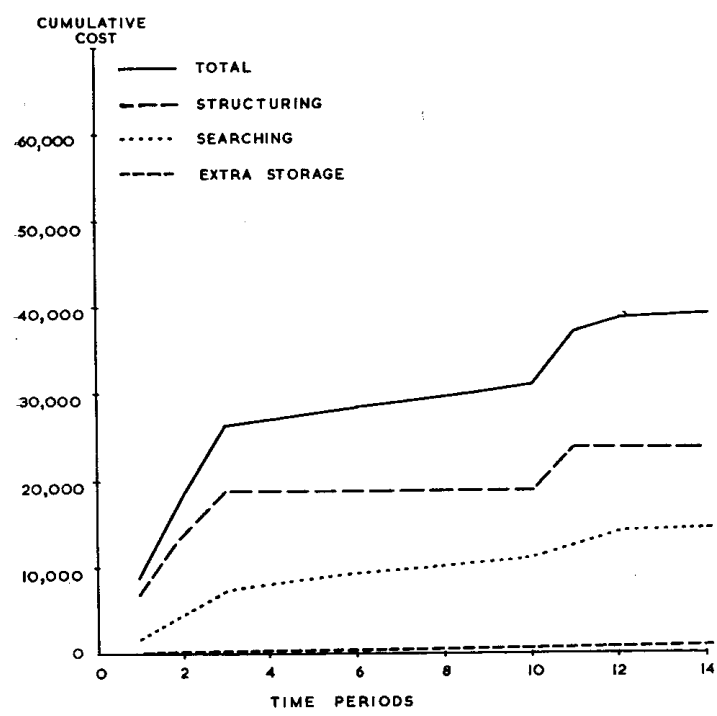

Fig. 5 Cumulative cost of self organising system

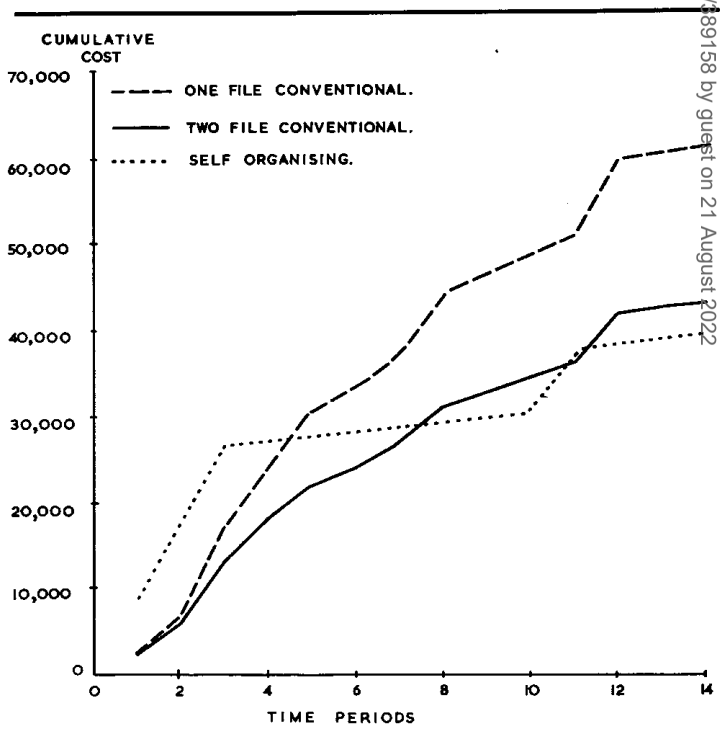

Fig. 6 Cost comparisons

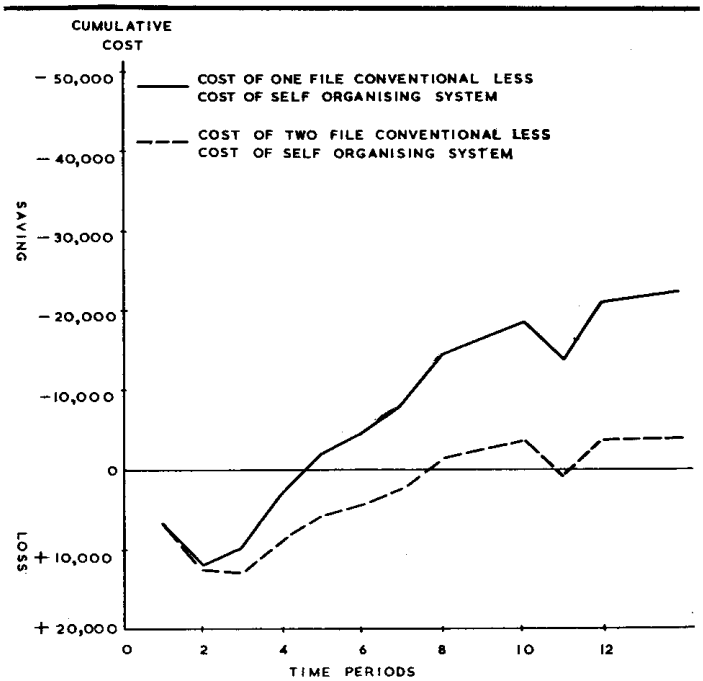

Fig. 7 Savings—negative costs indicate a saving, positive costs indicate a loss

conditions, once with the candidate's reply. Thus we assume that the overall amendment cost is equivalent to two passes through the entire file amending every record. The conventional system using one file has a large record length due to the personal/schooling details, thus the amendment cost will always be greater than that of amending only the application data in the two file conventional alternative.

The self organising system has one version similar to the application only data in the two file conventional alternative and four other versions. One of the other versions exists for time periods twelve to fourteen only, at this point the process of offering, and accepting places should be complete. Thus the additional cost of amending data in the self organising system is due to the versions made after time periods one, two and three. These three versions occupy 630 blocks on disc, thus the additional number of disc accesses required to amend every block twice is 2520. This reduces the saving over the two file conventional alternative from 3865 to 1345 and for the one file conventional alternative from 21,959 to 19,439.

Summary

The three examples quoted give some idea of the capabilities of the model of a self organising data management system. The existing model is crude and, as in Example 3, a human observer may, retrospectively, be able to suggest better reorganisation. However even in its existing form the self organising system can offer cost savings of 35 per cent in Example 2 and, in Example 3, 31 per cent against the usual alternative or 3 per cent against an alternative with data normalised into files holding the two basic relationships.

References

STOCKER, P. M., and DEARNLEY, P. A. (1973). Self organising Data Management Systems, *The Computer Journal*, Vol. 16, No. 2, pp. 100-105.
DEARNLEY, P. A. (1974). A Model of a Self organising Data Management System, *The Computer Journal*, Vol. 17, No. 1.

Book Review

Algebraic Coding Theory, edited by I. F. Blake, 1973; 413 pages. (*Dowden, Hutchinson and Ross, Penn*; *John Wiley, London*, £10·00)

The sub-title of this book is 'History and Development' and it is one of a series devoted to 'benchmark' papers in various subjects. It contains 35 papers by 33 authors (many of them well known names) which the editor has selected as having contributed substantially to coding theory. Twenty-five of the papers come from three source journals; *IEEE Transactions on Information Theory*, 13, *BSTJ*, 6 and *Information and Control*, 6; another seven are from three journals and a book all published in the USA. Of the remaining three, two are of Russian origin: Varshamov's 1957 paper on his coding bound has been specially translated into English for inclusion in this collection while a 1962 paper by Vasil'yev (On Nongroup Close-Packed Codes) is reproduced from the available English translation of the Russian journal *Problems of Cybernetics*. Hocquenghem's paper on his discovery independently of the BCH codes is reproduced in the original French. The papers are grouped into nine sections; each is preceded by a brief editorial introduction of length about one page. The pre-BCH era (up to about 1960) is given 144 pages; of this the fundamental work of Slepian on group codes is represented by three papers occupying 69 pages which makes him easily the most quoted author. The post-BCH era occupies the rest of the book except for a 1954 paper by Elias on a coding scheme aimed at the Shannon limit. The book is almost entirely concerned with systematic (n, k) block codes and there is one section devoted to three papers on the weight properties of these codes (Mrs. MacWilliams *et al*, and Pless). Convolution codes are given only a passing mention in the introduction with the comment that 'their future appears promising' but that is all and the name of Viterbi is absent. The editor notes some important concepts that unfortunately did not first appear as journal papers and which are therefore omitted; amongst these is Berlekamp's iterative scheme for BCH decoding though here it is fortunate that Massey's related paper on shift register synthesis is included. A minor point is that in some papers from journals with a large page size the print appears very small even though the linear dimension reduction factor is only about three-quarters.

Although there is no doubt that the selected papers are of prime significance in the historical development of the subject yet it is much more difficult to recommend it as a book to be acquired. Most of the papers in it are readily available and their substance, including that of the less accessible ones, is to be found in several textbooks. These give much more of the essential theoretical background (e.g. on finite fields), do not need to omit important but 'unpublished' developments and have extensive bibliographies which provide a means of follow-up just as effective as the lists of references ending the original papers. Had the papers been issued in cheap paperback form they might have found a market but at £10 anyone with a limited budget will have to consider very carefully whether to invest in this book rather than a textbook.

D. A. H. BROWN (Malvern)