END USER ACCESS TO VERY LARGE DATABASES IN AN AUTOMATED OFFICE/WORKSTATION ENVIRONMENT

Roger M. Tagg

Independent Consultant, UK

There is today a large gap between the access facilities offered to users in a large corporate database system and the concept of "database" as offered in some of the integrated multi-service software packages now appearing on the newer microcomputers. Since these new micros may herald the arrival of more automated office environments with intelligent workstations, it is important that an improved concept of access to high-volume corporate data is developed. This paper examines likely user environments and needs, and the software facilities needed to support them. An architecture developed by members of the British Computer Society's End User Systems Group is introduced as a potential framework for a solution to the requirement.

INTRODUCTION

The classic model of end-user access to large databases has up to now been the Query Language, offered at a "dumb" terminal driven by the mainframe system on which the database is kept.

In the manner of all technical advances in computing, Query Languages received all the marketing razzamatazz - "English-like", "user-friendly", "user need know mothing about the data structure" were the sort of phrases used. It was quite as if the ultimate interface had already been achieved.

But in retrospect, Query Languages are just a first step - the computer man's answer to his initial understanding (or misunderstanding) of the user's needs and preferred ways of working. Already things have moved on - and many Query Languages have now been subsumed into "4th Generation Languages" no longer claiming to be appropriate for <u>end</u> users.

However the arrival of Query Languages did reveal a trend - away from formalised, predefined use of computer data towards a more ad hoc usage reflecting users' innovatory and constructive skills rather than their ability to adhere to strict routine.

The recognition of this trend also contributed to another idea - that of the Information Centre. This concept implies a computer service operated for a group of users in an organisation, outside the normal DP development service. It also has the connotation of being oriented to the use only of users' private data and data <u>extracted</u> from the corporate database. Though whether this limitation is the cause of - or the consequence of - IBM's decision not to offer Query Languages and 4th Generation Languages interfacing directly with VLDB's in the main DBMS, is open to some conjecture.

However it is the basic tenet of this paper that these earlier concepts are inadequate for the years ahead. When more end-users are familiar with micro software such as Word Processing, Spreadsheets and even the soi-disant Database packages than they are with Query Languages or Information Centre tools, a new model of the end-user's access to large databases is clearly required.

THE USER'S ENVIRONMENT
(see reference XEPH 83 for general background)

The right place to start when thinking about a new model is obviously with the users. What will be a better view of users' needs and preferred ways of working in 5-10 years' time? There is a great temptation of subjective crystal-gazing, but a knowledge of a number of organisations in a range of sectors, and an observation of the people who will form the user community in future years, leads to a number of observations.

The first observation is that users will be expecting to have access to a range of services or functions - a far cry from the previous situation where users were having to be coaxed to work with any computer system at all. Figure 1 shows some of the services that have been introduced to many people in organisations outside of the normal DP services, mainly through the medium of independently-acquired micro computers.

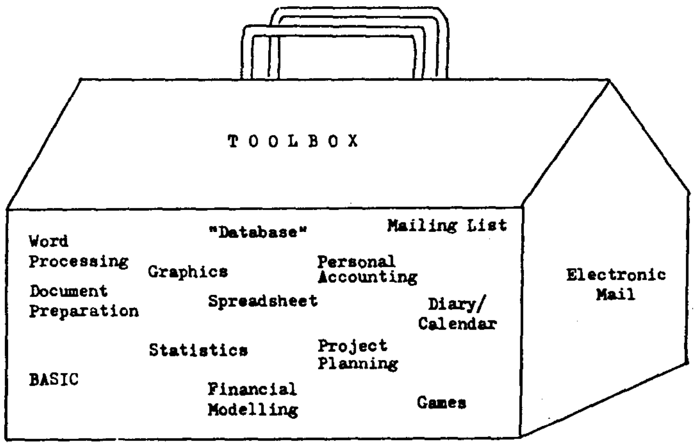

Figure 1: The user's expected toolbox of services

What is meant by "database" in such a toolkit may vary considerably. In many cases it is little more than an electronic version of a set of manual card boxes. Many of the packages describe themselves as Relational - presumably on the grounds that they leave <u>out</u> all the things that a Relational DBMS <u>should</u> leave out - like preset links between files. What they actually have <u>in</u> is less certain - they usually omit any efficient method of "relating" the different card boxes together.

The next observation is that users will not want to have to learn different languages, conversation patterns, special symbols and codes for each different service. Many vendors of micro software have already recognized this and are now selling "integrated" ranges of packages covering the different services - though sometimes the names of the packages suggest more integration than the actual software details.

Figure 2 shows how some of the vendors have covered this need for an integration of services.

Following on from this, it is clear that access to large, shared databases cannot be treated separately from these ranges of software.

A related observation is that users will certainly not want different terminals for different services unless absolutely necessary. This is the essence of the "workstation" concept - and in many go-ahead organisations, policies are being established as to what hardware should be used. Obviously there may be variations in user needs, so the most common policy is to nominate a range of workstations including such refinements as colour, multi-window, A4 page size screens, high-resolution graphics and hard copy devices of various qualities.

As soon as the user starts using multiple services, he will come across the need to shift data about between services. Even in some of the ranges in Figure 2, the only way of doing this is by converting in and out of "standard ASCII" files. Ideally, there should be a common "file" concept underlying all the services: but the opportunity for many vendors - and users - to adopt this idea may have passed.

This leads to the next observation, which is that users are beginning to acquire ideas "by default" on how the computer world behaves. Shakespeare may not have originated the line "All the world's a spreadsheet", but one or two vendors these days are going this way by allowing sharing of data between spreadsheet and database as long as the database is viewed as a matrix with rows and columns. Not that bad an idea, but surely a bit limiting. At least it is better than the view of data the user may acquire by learning BASIC programming!

Many current users of "professional" micros are also building up expectations of future developments. Mice, windows, icons and more advanced graphics are becoming available already, and voice and other interfaces are expected to be developed. The user also expects his workstation to be a communications device as well as a calculation and data processing device. Electronic mail, both inside and outside the organisation, and access to shared and public information services are all technically possible.

Talking about access to shared information services brings us back to our main point. The user's expectations and models of the computers he deals with are changing very quickly. How can we bring the best of past experience with querying large databases into the new environment - where it is just another service within a structure in which the user may have a very definite model?

Vendor	WP	Graphics	Doct. Prep.	Spread -sheet	Fin. Model	Statistics	"Data-base"	Project Planning	Diary /Cal.	Electronic Mail
ADR products		✓			✓	✓	✓			✓
All-in-1 (DEC)	✓	✓			✓		✓		✓	✓
CEO (Data Gen.)	✓	✓			✓	✓	✓		✓	✓
Easy xxx (Info. Unlimited)	✓			✓			✓			✓
FOCUS		✓			✓	✓	✓			
Goldengate (Cullinet)	✓	✓		✓			✓			✓
Knowledge-man+ (MDBS)	✓	✓		✓		✓	✓			
Lisa (Apple)	✓	✓	✓	✓			✓	✓		
Lotus 1-2-3		✓		✓			✓			
Manage (Cincom)	✓	✓		✓		✓	✓			
Orbis (Norsk Data)	✓	✓			✓		✓			
Perfect Software	✓			✓			✓			✓
PFS:xxx (SPC)		✓			✓		✓			
RAMIS		✓			✓	✓	✓			
Starburst (Micro-Pro)	✓			✓	✓		✓			
Super xxx (Sorcim)	✓	✓		✓	✓		✓			
Tomorrow's Office	✓						✓			
VisiOn (Visicorp)		✓		✓			✓			
Xerox Star 8010	✓	✓	✓				✓			

Figure 2: Some current ranges of "integrated" end-user software

REQUIRED COMPUTER FACILITIES

Any computer facilities - and we are talking chiefly about software here - that are developed to support access to "VLDB" within the new user environment, will obviously need to fit into the general pattern set by the facilities already present. This pattern may involve a hierarchical menu structure, or an overall command language, or possibly other approaches. However VLDB access brings in some considerations of its own. To a greater or lesser extent it is a "research" type activity; the user does not know at the outset exactly how much data he will retrieve and exactly what it will look like. He may even be vague as to exactly what data is available and in what form.

Figure 3 shows a hypothetical flow-chart of how a user might use VLDB access within the wider environment.

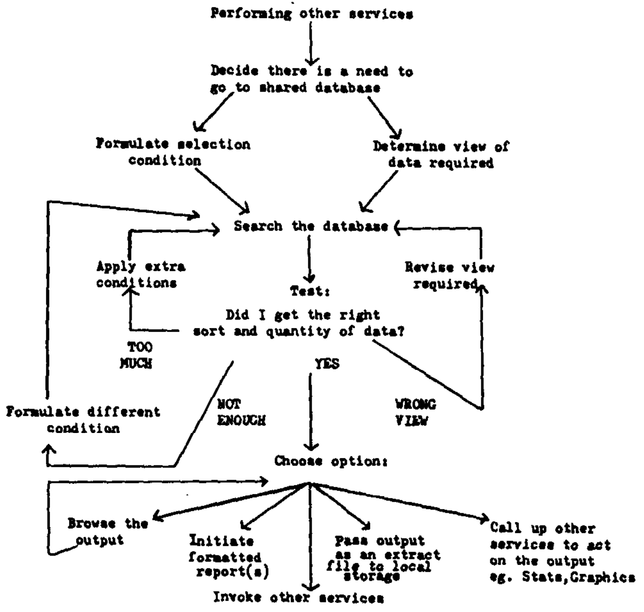
Performing other services
Decide there is a need to go to shared database
Formulate selection condition
Determine view of data required
Search the database
Apply extra conditions
Revise view required
Test:
Did I get the right sort and quantity of data?
TOO MUCH
YES
NOT ENOUGH
WRONG VIEW
Formulate different condition
Choose option:
Browse the output
Initiate formatted report(s)
Pass output as an extract file to local storage
Call up other services to act on the output eg. Stats,Graphics
Invoke other services

Figure 3: Flow chart of possible end-user access to VLDB.

This sort of flow is largely characteristic of VLDB access in this environment. But software to support the research-like, iterative, refinement orientation is not always the strong point of some current mainframe query packages.

Another feature not provided in any really consistent way in the Query Languages on the market - both mainframe and micro - is that of a good, powerful user view mechanism. This concept is perhaps worth discussing in a little more detail.

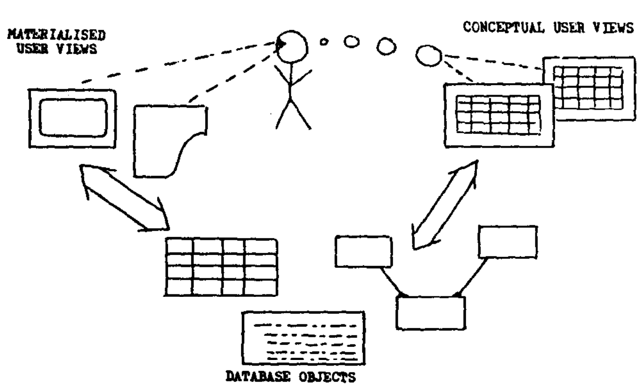

Figure 4: User Views of Data in a Database

Figure 4 shows how, when the user views data in a database, he normally has a view which is not identical to the structure of the data as stored. The most natural views are those which are materialised, ie. actually displayed to him (printed report or screen display format) or used as an input format (again probably on a screen). These views are natural because of their close approximation to the world of paper-based forms and files with which most users have some acquaintance.

However when a user is querying a database, he may also need conceptual views of what is in the database, against which he can put his queries. These conceptual views can correspond closely to materialised views: in this case they would be virtual files containing all the data for each page (or line) of the report or display. Conceptual views can also coincide with stored files, but this is not always a good idea, since:

- the user may, subsequently to database searching, have to apply explicit operations to link different files to achieve the required output
- the user may have to get involved with extra concepts which exist in the stored database, eg. predefined network links, or indexes.

A view mechanism along the lines described above is an essential aspect of Relational theory, yet we are still seeing many database query systems, including some claiming the adjective "Relational", that are still in the dark age of "file at a time" querying.

It is becoming evident that views may need to be rather more general than completely flat, normalised relations. A materialised form is often hierarchical (eg. Order Header data + repeated Order Line details). A matrix layout (eg. Spreadsheets or Market Research Statistics) may also imply a view with one or more repeats.

To "frig" these by such techniques (eg. in the order example) as treating the view on a line-by-line basis and taking special action over the duplicated fields is artificial, and means moving away from the user's natural view. This example is illustrated in Figure 5.

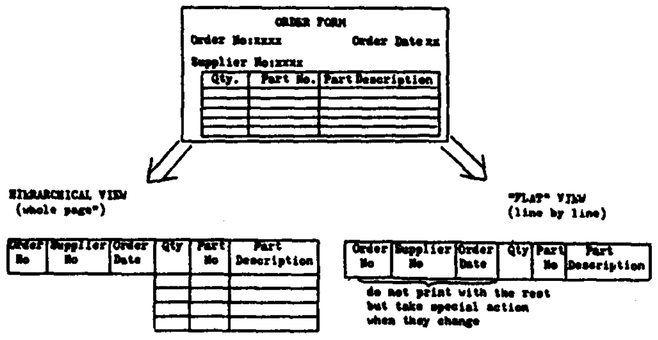

Figure 5: Two "views" supporting a user access requirement

Another area needing improvement in the facilities is the handling of more diverse types of data structure in the database. Relational theory is rather oriented to a data structure where all data is arranged neatly into "fields" and where every row of a table is conveniently and uniquely identified. Not all data available to the user is of this type, as the spectrum in figure 6 shows.

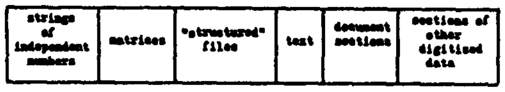
strings of independent numbers
matrices
"structured" files
text
document sections
sections of other digitised data

Figure 6: Spectrum of database structures

Starting from the left, the ultimate "structure" is a single number - and data in this form does exist! What we call "structured" files are really the normalised-down versions of the "paper form" model of data referred to early, but others would regard text as just as structured - document, section, paragraph, sentence, word is a fairly well understood model itself. Document sections are essentially units of digitised graphics, covering both genuine pictures and text in some font. These sections are the "cuttings" in a "cut and paste" operation. Finally, especially in the future, there will be other sorts of cuttings: photographs, voice or other audio clips, film clips etc.

This last consideration highlights the problem that arises over data which is passed between services in a multi-service user system. What is a matrix to one service could become a relational table to another, a piece of text to a third and a section of document to a fourth. Forcing all these transfers to pass through some standard format seems inappropriate. It would be better to have rules about the assumed transformations that take place when data created by one service is taken up by another.

To sum up on the matter of user views of data, there is still scope for considerable development work towards a truly general data view mechanism. The original CODASYL database groups did not tackle this question, while the Relational school has been too concerned with Third (and beyond) Normal Forms to provide a general enough view mechanism. All the systems obviously have to face the implementation problems - not inconsiderable - of how to optimise support for views while sitting on a very variable infrastructure of physical database technology, eg. chain navigation, index look up, bit matching, fast serial scanning devices etc. Perhaps the most consistent - and not sufficiently studied - proposals in the area have come from a different CODASYL school - the End User Facilities Committee. This group's work (see reference EUFC 83) is not slavishly tied to the use of CODASYL DBMS, and represents a genuine attempt to look at things from the office-based user's point of view, though perhaps before the big push with integrated micro software.

The final area of required facilities is that of "second-line" support to the user. This means giving assistance to the user when he cannot work out how next to use the language, menu and view facilities which are his primary tools.

Many comments have been made about this area, and this paper will concentrate on a number of particular issues only. One is the organisation of assistance facilities into early understandable groups. The following grouping seems sensible:

- "directory" assistance: this ranges from "what's on the database that I can look at" to "what values are allowed in this field". It will also cover what views are available. The software to support this will be standard but the data will be dependent on the particular databases open for access.
- function assistance: this normally covers tutorial explanations for each command or menu option. It should also include tutorials for macros - with facilities for the originators of macros to add their own explanations.
- general service assistance: this covers "everything else", such as a service overview tutorial, recap of a session, choice of dialogue modes and the "panic button" procedure.

To support these facilities and other aspects of the user interaction, a "User Support Database" is required. Some of the details of this are described in reference TLEE 82. Originally this was envisaged as being part of the "Data Dictionary", but this latter term seems too restrictive in meaning. Clearly the User Support Database is a concept applicable to all the services available at the user's workstation. It will also overlap to some extent with the System Management Database which supports the administrator of the whole network and its facilities. Both of these Databases may use existing Data Dictionaries and also Thesauri, Access Control Files, Macro Libraries etc.

SOME APPROACHES TO THE "DATABASE ACCESS" SERVICE

So far none of the micro-based integrated user systems offers an access route to large mainframe databases. As stated previously, the interfaces between their own local database services and other services are limited to methods like use of comma-delimited ASCII files or pretending that database tables are like spreadsheets or vice versa. So examination of current approaches has to be limited to what some of the mainframe software ranges can offer.

A good example is ADR's PC/DATACOM. As part of making the mainframe products available on and through IBM PC's, ADR has offered a route for the PC to access the mainframe database through a consistent subset of the same DBMS/Query system as on the mainframe. Other ADR services due to be available on the PC's this year include Electronic Mail and Decision Support (Statistics, Graphics and Financial Modelling). Data downloaded to the PC can also be stored in ASCII or DIF (format used by Visi On and Lotus) files, for subsequent processing by the commonly available packages.

Cullinet's approach is slightly different - they offer their own range of PC functions named "Goldengate", including access to mainframe data - but they do not cater for Visi On, Lotus etc.

Similar in many ways to the above are the facilities offered by PC FOCUS from Information Builders. This system also includes transaction processing using screens on the PC against a database on the mainframe. FOCUS, Cullinet and DATACOM are all limited to IBM-compatible mainframes.

Cincom have taken a different approach with their MANAGE range - they advertise "Personal Computing without Personal Computers", though a PC can always be used to emulate an IBM 3270 terminal. MANAGE, which was developed in Europe, includes MANTEXT, MANCALC and MANGRAPH. Access to the VLDB is achieved through MANTIS, the 4th generation language and application development tool. MANAGE provides an alternative for the organisation that wants to retain more centralised control over use of computer resources while fulfilling users' expectations for "personal" computing.

Also centrally controlled, but mini-based, is Data General's CEO (Comprehensive Electronic Office). The usual range of "personal" services are offered (see Figure 2), and there is both an "Electronic Filing" service with a Document/Folder/Drawer concept and access to databases (CODASYL or OS files) via the PRESENT language.

In a similar manner to the above, DEC can extend their ALL-IN-1 system to include the VAX DATATRIEVE system, including use of the VAX Common Data Dictionary. DEC personal computers can also be used as VAX terminals within the network.

An approach with an interesting solution of the "consistent data view" problem has been proposed by M.Zloof of IBM (reference ZLOO 80), who designed the Query-by-Example system. He has extended the QBE concept of example table layouts into office automation functions. He introduces the extra concept of Generalised Data Objects which include not only tables but also forms, reports, charts, facsimile and audio. These objects can be manipulated by "Explicit Program Objects" and can be subject to "Triggers" which may for example send messages built from the objects to another user's mail slot.

With QBE now included as an interface for IBM's DB2 DBMS it will be interesting to see if these extensions can allow DB2 to be accessed from a generalised office environment using PC's.

THE BCS END USER SYSTEMS GROUP ARCHITECTURE

The BCS End User Systems Group (EUSG) was formerly the Query Language Group and under that name produced reference BCSQ 81. Since that publication the group has widened its brief to explore the possibility of defining a unified view of general end-user systems as had been done for query languages.

The group soon realised the size of such a task, and has recently concentrated on proposing an architecture for a general end-user system such as would be offered through workstations in an automated office environment. Having a large proportion of members with a database background the group's work obviously takes particular interest in data aspects, but other aspects such as messages, knowledge bases and intelligent systems are beginning to receive more attention.

The group's architecture is essentially a series of models of the interaction between the user's world and the computer world. Over the whole set of models the group hopes to establish the relationships between the various components of an end-user system. The diagrams shown in this paper are based on reference SAND 84.

Figure 7 shows the simplest model - the total black box. The user is only aware that what he keys in may affect what appears on the screen. Playing "Space Invaders", using the computer as a calculator and one-off word processing all follow this model.

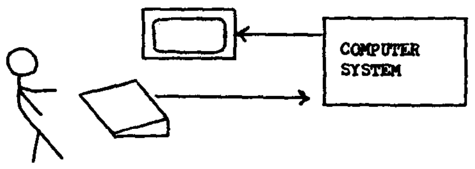

Figure 7 : The simplest user model: a total black box

Figure 8 introduces the idea that there is some storing of data. This model covers a large proportion of cases where the user works with a simple, independent package which involves storage of data - often seen by today's users as floppy disks. The user has to take a "leap of faith" that action on the keyboard will result in correct, re-usable data being stored on these "disks".

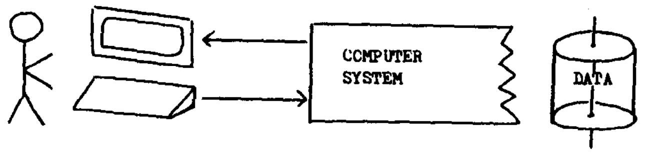

Figure 8: Black box with stored data

Once the user takes on more complex tasks or a number of different tasks, he becomes aware of the presence of programs. However he is not concerned with these other than their being the means of converting his requests to responses and stored data. He sees that he is required to express commands to the computer in the form of key words of menu options in order to trigger these programs. He may also be aware that the data is shared by other users similar to himself. Figure 9 shows this basic user **model for a general end-user system.**

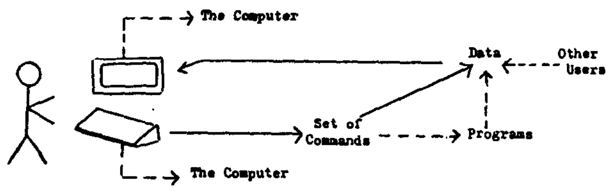

Figure 9: Basic user model for an EUS

As some of the services become more complex, and concepts such as electronic mail and linked computers are introduced, this basic model may need to be expanded. Figures 10 and 11 show two such extensions.

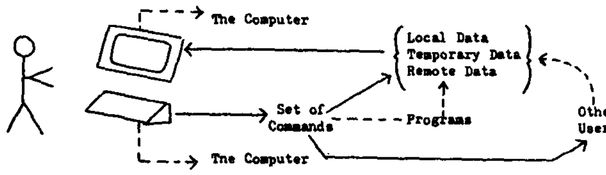

Figure 10: Extended user model: Remote Data & Electronic Mail

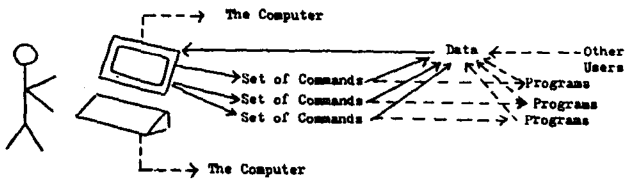

Figure 11: Extended user model: Multiple Activities

In Figure 11 the user has several tasks under way - they could be simultaneous but he may have suspended work on all but one and will resume the others later.

To develop the model further, EUSG proceeds to what it calls the Working Model. In this model the Set of Commands is divided into three functions:

- the Dialogue Manager, which controls the user's actual use of the keyboard and screen - this element might change if a different type of workstation were used

- the Global Service Model, which organises the user's choice of services available to him and handles the commands not specific to a particular service
- the Services themselves, with the commands specific to each service.

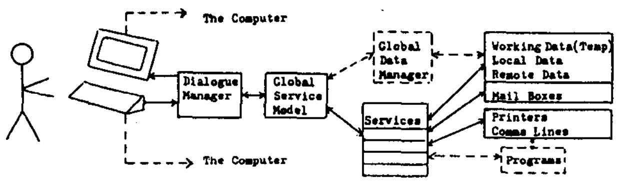

Figure 12: The Working Model

The model in Figure 12 is much more complex than the earlier models: it may be too complex for many users to be expected to understand, but it is useful at the level of design of an end-user system, since it separates those elements which should be separated from a design point of view even if users are unaware of any separation.

Programs are distinguished from services since the designer may wish to build user facilities which use a number of programs from different sources but yet appear as one to the users. Two different services could also make use of the same programe.

This mapping between services and programs is the subject of the final model, the Implementation Model, shown in Figure 13. One of the services may itself be a Service Building Tool, which will allow program modules to be imported, inserted or generated and then mapped into a new service within the Global Service Model.

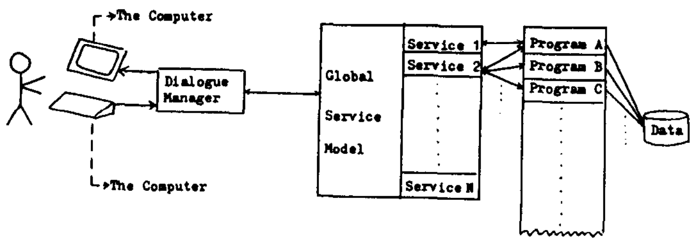

Figure 13: The Implementation Model

CONCLUSION

The whole subject of designing integrated end-user systems, rather than just watching them happen, has so far attracted little development effort and even less research interest. The provision of services to allow workstation-based users to access large corporate databases is not yet a common feature of most systems. This paper has highlighted **some of the needs and outlined some possible approaches, including an overall architecture. Hopefully it may trigger some more serious interest on what is an issue of immediate relevance rather than, as with certain research areas** on the 5th generation bandwaggon, of uncertain demand and uncertain usability.

BIBLIOGRAPHY AND REFERENCES

BCSQ 81 "Query Languages - a Unified Approach" - report by members of the British Computer Society Query Language Group, published by Heyden/Wiley, 1981

EUFC 83 "Codasyl End User Facilities Committee Journal of Development" - published by Canadian Government EDP Standards Committee on behalf of CODASYL, 1983

SAND 84 "End User Systems Architecture" - paper given by M. R. Sandford at BCS End User Systems Group Conference "Where to now the Mouse has arrived?" in London, February 1984

TLEE 82 "Data Dictionary Support for Query Languages" - paper by T. Lee, C. Chang, H. Hodgart and K. Meyer in Computer Bulletin, UK, September 1982

XEPH 83 "End User Computing" - a user survey by Xephon Technology Transfer Ltd, Maidenhead, UK, 1983

ZLOO 80 "A Language for Office and Business Automation" - paper by M.M. Zloof of IBM at Office Automation Conference, Atlanta, US. March 1980