Presented to the Staple Inn Actuarial Society

on 31st October 1989

THE CONSTRUCTION OF COMPUTER SYSTEMS AND THE USE OF FOURTH GENERATION LANGUAGES

by

John Harsant

STAPLE INN ACTUARIAL SOCIETY

THE CONSTRUCTION OF COMPUTER SYSTEMS AND THE USE OF FOURTH GENERATION LANGUAGES

(Presented by John Harsant to the Staple Inn Actuarial Society,
31st October 1989)

INTRODUCTION

In February this year a Paper was presented to the Institute by C. G. Lewin and others, entitled "Calculating Devices and Actuarial Work". This was the report of a Research Group which had been examining developments in computing devices, the effects on actuarial work and the future outlook. The Paper considered among other topics "Future Prospects" which was a subject of particular interest to me as my Office has been writing, under my direction, computer systems ab-initio to support my actuarial work using current computer technology. Since only brief reference was made by Lewin to this technology it seemed to me that a further discussion both of hardware and software was merited, hence this Paper. I concentrate on the use of Desk Top computers, as it is in this area where the current rapid expansion is taking place.

I start by relating my own experiences in computing in order to place the main substance of the Paper in perspective; I then discuss hardware, for it is developments in hardware which have made the remarkable developments in software possible. It is through the current software available, the 4th Generation Computer Languages (4GL), i.e. the generation after Cobol and Fortran, that the advances in the development of application have become possible.

Once I had become familiar with two significant 4GLs and the methods required to develop systems in them, it became apparent that there is a further area which is highly relevant to the work of anyone using computers but which is not yet universally adopted within the Country. This area relates to the structured analysis of systems which, if followed, can put a certainty to the successful completion of computer systems that has been conspicuously absent on many occasions in the past. I consider it is of importance to the profession that these developments should be publicised within it and to do so is my purpose in the second part of this Paper. It is not my purpose to go into detailed techniques of writing systems to do actuarial work, these are the techniques which we all know and are essentially as firmly established as the tablets which came down from Mount Araat. Some who read this Paper will know much about computers, and perhaps I will bore them. Others will know little and I may lose them, I have tried to do neither, I hope I have succeeded.

We live in a community and all we do relates to that community. I acknowledge here my indebtedness, not only to the Institute but to my erstwhile colleagues elsewhere and to those at the University of Liverpool, particularly Dr. J. L. Schonfelder, director of the Computer Laboratory whose wide experience and knowledge of computing has been invaluable to me. My secretaries, Jean Williams and Nina Veevers have everlasting patience and I am most grateful to them for their help.

ACTUARIAL COMPUTING 1970 - 1988

In this section I describe briefly my own experiences in Actuarial Computing in order to provide a framework for my later comments. My involvement with actuarial computing started essentially in 1970 at the time when Batch Machines were the only ones available, disk technology had just become widely available and Cobol was the latest language. In that year I specified a parameter based system for the valuation of Pension Funds which formed the basis of all the valuations in my former Partnership to at least 1984. This system was significant in that it :-

(a) showed that actuarial computing was just like any other computing,

(b) a good system specification produces a good working system,

(c) much more time and effort is spent in obtaining accurate data than in actually carrying out the valuation.

In 1973 we bought an ICL 2903, a small Batch Machine. This served us well but was in retrospect a mistake, for at that time we were seeing the beginning of inter-active computing and by continuing with batch processing we did in fact retard the progress of the business. We were inevitably saved however, because many others took that same decision.

By 1980 the requirement of Users for information from their computer systems was growing year by year at a much greater rate than the ability of their computer departments to deliver. Because of this Actuaries, amongst others, started to turn to micro-computers in which they saw the opportunity of achieving the results they required without having to depend upon the uncertain performance of others. Certainly I did this and it was inevitable that many should go that way. In hindsight to do so was unfortunate. It divided the responsibilities for computing within an Organisation, in many cases it divided the data, it demoralised the data processing staff and fragmented effort.

Fortunately this situation is now being resolved and more about this will be said later in this Paper. The resolution of the problem, however, is largely coming from the fact that Desk Top computers as they are now are so powerful that they are more than capable of taking over many of the roles which have until now been carried out by physically larger, but possibly in computing terms, actually smaller and certainly much more expensive machines.

A prime example of the fragmentation of effort can be found in the emergence in the mid 1980s of systems for the administration of pension schemes on micro-computers. Quite significant systems have been developed for these, but on a stand-alone basis, a basis not immediately capable of integration into the general systems of the company in which they are installed. They are written in languages which fit on to the small micro-computers of the mid-eighties and so inevitably compare unfavourably with systems which can be written in today's Desk Top computers.

I differentiate between the micro-computers of the mid 1980s and the Desk Top computers of today, for although they may be similar in external appearances, today's Desk Top computers have all the power and many but not all the facilities of the mainframe and so are totally different in their final application from their predecessors of five years ago.

Set out below is an attempt to illustrate the difference between the machines of, for example, 1984 and 1989.

MACHINE CHARACTERISTICS	1984	1989
Chip	8086	80386
Clock	4.7 Mhz	20/33 Mhz
Memory Usual	250 k	1-4 Mbyte
Maximum ?	640 k	16 Mbyte
Disk Usual	10 Mbyte	40 Mbyte
Maximum ?	20 Mbyte	300 Mbyte
Speed (million instructions per second)	.25 mips	2.5-5 mips
Cost	£4,500 (1984 money)	£2-4,000

The big difference is that the 1989 machines will run much of the software previously only available on mainframes or the bigger minis.

There is reason to suppose that this astonishing progress will continue, and some readers will already be aware of significant developments either not yet marketed or of only limited availability.

NETWORKING — Multiple User Machines

Desk top computers started as single user machines, in comparison to mainframes and minis which are multi-user. It is obvious that advantage may be gained by linking Desk Top computers so that a group of users may use common data, say, within the Actuarial Department of a life office. The technology of Networking exists and is used but it is still under development and my advisors have told me to wait at least until the end of the year before spending any money on networking.

The purpose in waiting is to try to spot which system will come out as the leader and then to buy that.

There are at least three areas to consider :-

(a) the operating system,

(b) the network itself,

(c) the software to run on a networked system.

I expect to use a Unix derivative operating system, perhaps Xenix and, of course, the software I am using, being developed in the first instance on mainframes will be ideal for multi-user systems. The big difference will be that each work station will do its own processing rather than being a dumb terminal waiting on the mainframe for its favours. Dumb terminals are out. Thus I see the actuarial office of the future with each Actuary having his own computer complete with processor, but with a local file common to all the Actuaries and with access to the office's mainframe. To do this requires management control and this will be referred to later in the Paper.

Finally, whilst on the subject of Machines, I cannot help but comment on how fortunate we are to have Desk Top machines available from a wide variety of different manufacturers with all the advantages of competition which follow. This compares with the situation which has hitherto existed of having to go to a particular manufacturer whose equipment is incompatible with that of any other, and once having bought his equipment being tied to him.

FOURTH GENERATION LANGUAGES

Fourth Generation Language (4GL) is a generic term for the many efforts made to simplify the labour of development of computer systems required when using the Third Generation Languages, such as Cobol, Fortran etc. Within this grouping there are at least three, and I expect there are more, different ways in which people have approached the problem depending upon their immediate circumstances and their perception of the future. The approaches vary from, as it were, mechanising the processes needed in the Third Generation Language area to a complete rethink of what computing is, where it is going and how we will develop systems in the future. I list the three approaches in order :-

1. Cobol Code Generators
2. Computer Languages as such with automatic file handling facilities.
3. Relational Data Base :-

   Data Bases themselves
   The tools to access the Data Base
   The tools for system design

1. COBOL CODE GENERATORS

I have no direct experience of Cobol Code Generators, because I am not familiar with Cobol and have decided not to spend time on it. This is because I see that although Cobol has been of immense value to computing for perhaps 20 years, and will continue to be the basis of many systems for a long time yet, its time is over. I little doubt that there will be those who disagree with this rather bare statement and I shall look forward to any comments with interest.

2. PROGRAMMING LANGUAGES WITH THE FILE HANDLING REMOVED

I have experience with one well established language in this area — PRO-1V — and it delivers everything which it claims, although it has some features which require careful learning. PRO-1V is by and large relatively simple to learn, relatively simple to use and is not over-demanding on machine capacity. It will run on a 1984 micro. Its logic is expressed in terms which will be immediately familiar to anybody who has used the Third Generation Languages and systems can be established relatively quickly in it.

Having said that, however, it looks to me as if the development along that particular line of approach has been taken as far as it can, written as it is for machines of limited size, and I do not see that it can go a great deal further. Again there may be those better informed than I on the future of these products. I have been out of touch with it for a little while and if there are developments of which I am unaware I would be very glad to hear of them.

3. RELATIONAL DATA BASES

To me these are the most significant development and the remainder of this section of the Paper is devoted to them.

Relational Data Bases, to me (and again this is my own opinion), represent a significant step ahead in data processing in that they forget about computer languages of the past and ask "what do we expect of a computer system?" It is from this simple question that all else follows.

In broad outline the three items which contain the answer to the question just raised are as follows :-

1. that we can find records which meet our requirements quickly and easily and that such records can be updated or manipulated as required. — *Data Maintenance*

2. that to assist in the programming and the running of the system there are tools associated with the data base which enable all necessary functions to be carried out easily. — *Application Tools*

3. that in addition there exists a set of computer tools which aid the systematic analysis of the application and once that analysis has been carried out do a large amount to write the system for you. Computer Aided System Engineering (CASE) — *Design Tools.*

At this point it is difficult to describe the general concept without referring to specific details of a particular product. I shall describe three features of a Relational Data Base with which I am familiar, "ORACLE", whilst intending that my comments shall be as general as possible. There are competitors to ORACLE — Ingres, Ashton-Tate and DB2 amongst others. I would welcome comments from those with experience of these languages.

1. Data Maintenance

The three features to which I refer are as follows :-

(i) keeping data on a computer file,

(ii) finding data on a computer file — SQL,

(iii) performing calculations.

(i) Keeping data on a computer file

All data is held in arrays known as tables. The reason for calling these tables and not files is that there is no attempt by the system to define a key on any one or more items or to index any of the items (fields) by the system. It is left to the user to create any indices which he requires, whether on one or more fields, and additionally to define any particular index as a unique index. Any number of indices can be defined although of course you have to be very careful about defining more than one unique index. The effect of this is that you can search (query) a record on any given field or on any given group of fields and this gives great flexibility and assistance to the User.

This contrasts with PRO-1V which belongs to the second category of 4GL which I referred to earlier in this Paper where each file has its keys and the order of those keys is initially defined. Those keys are indexed and are the only fields which are indexed. If you wish to search on other fields you have to then create a separate file with the data you require as keys described as such. The overhead of maintaining such files is considerable.

(ii) Finding records on a computer file — SQL

SQL is the definition of the means whereby data is accessed. The importance of SQL arises from two things as follows :-

SQL was originally defined by what amounted to an academic study of the requirement of Users, and

SQL is an ANSI standard and thus a de-facto international standard, independent of any commercial organisation.

In the preceding section I described how the data was actually held. SQL defines the commands which are necessary in order to access and process the data.

SQL — Examples

To demonstrate the power of SQL, let us consider the following requirement :-

To read a file containing details of a person, selected by being born before a given date and listed on the screen in date of birth order.

The file called "MEMBER" will contain :-

Initials
Name
Date_of_Birth
Salary

In Basic we would have to :-

Open the primary file.
Open a secondary file.
Sort the data from the primary file and write to the secondary file.
Set up a heading.
Do a selective read of the secondary file.
Send it to screen or print-out.

In SQL the only statements are as follows :-

```
Select    Name, Initials, Date_of_Birth
From      MEMBER
Where     Date_of_Birth <= '01-Jan-20'
Order by  Date_of_Birth
```

That is all, and will produce the listing on the screen. This can be printed out in one of several ways, by using the print screen facility which is crude, by saving it in ORACLE under SQL and printing it out or again, if in ORACLE, by using their Report Writer.

To use the Report Writer all that is needed is to put the SQL Statement, mentioned above, into the Report Writer and it produces the report without any extra work.

Of course this is a simple example in SQL. Let us consider another example, let us update one table from another.

```
UPDATE VALFAC
SET FACTOR1          =  (SELECT FACTOR1 FROM FACTORS
                        WHERE FACTORS. AGE = VALFAC.AGE
                        AND BASIS = :VAL.BASIS)
WHERE VALUATION      =  :VAL.VALUATION
```

This extracts a factor from a table and enters it as a valuation table. The references :VAL.BASIS and :VAL.VALUATION are two items entered on the screen representing the References to the Factors and the Valuation itself.

You will note that while these commands are very powerful, they are also demanding and a thorough knowledge of SQL is needed before you can do much else. How you achieve one before the other is not obvious.

(iii) Calculations
When data is accessed via SQL all calculations have to be carried out within the syntax of the SQL Language. This I found initially unattractive and very unlike the sets of commands which are available in say Fortran or Basic. The latter commands are essentially derived from the commands normally used within conventional mathematics and are easily understood by anybody with a reasonable background in modern mathematics.

SQL on the other hand looks very much as if it were written by someone who was not necessarily numerate, who recognised the need for the straightforward operators + - / *, and that was all. It looks as if at a later date the other necessary operators such as Exponentiation were brought in and it all has rather a strange appearance. A further feature of ORACLE, I know not about elsewhere, is that when carrying out calculations local variables are not left to float, but they have to be written to a screen before they can be recaptured. This, I think, is a deliberate feature of design in order to introduce a high degree of structure into the language. The overall effect of this was that it took me some time to come to terms with the layout of calculations in this environment, but once it is understood it is no more difficult than any other mode of calculation, just different. Although I do not completely understand the rationale it is quite clear that those responsible for designing SQL had very clear objectives in mind as to how systems should be structured and they designed it in such a way that those ideas should be carried through. SQL is widely used, and this must be regarded as indicative of its success (it is not a proprietary product or sold as such) and, therefore, I take its use extremely seriously.

Lewin's Paper, section 4, Future Prospects, mentions that it is still necessary to ensure that the entry of data is accurate. The importance of documentation is emphasised, and the need to write systems which give adequate information, so as to enable checks to be made all along the line as to the validity of the results which emerge. It is my thesis that the use of the appropriate 4GL is a great aid to ensuring that these very necessary objectives are met.

Acturial Calculations

The structure and methods by which actuarial calculations are carried out are well known in computing and it is not my intention to discuss them as such here. However, to illustrate the points made in the preceding section and, indeed, to set out some of the ideas in case they are of interest, I set out now the structure of computer systems as I know them for two particular applications, and demonstrate how the accuracy of those calculations can be checked all the way along the line. This does not deal with the structure of the programs or the documentation, which will come in the last part of this Paper.

The starting point of any calculation is the standing record and the existence of this is assumed and that it exists within the 4GL used for the actuarial calculation. This enables immediate access to be possible. The routines are defined by the requirement and fall into two groups:

One Offs,
Group Calculations.

One Offs

The stages are as follows :-

Enter a description of the calculation on a calculation file — Set up.
Carry out the calculation and write to file — Calculation.
Print out the results.

To illustrate this, consider the calculation of a deferred pension from a pension scheme as follows :-

Set Up Procedure — this will be entered from the first screen or part of a screen and will comprise:

(i) a unique internally sequential generated integer, as a reference number — to that particular calculation.

(ii) the name and reference number of the member.

(iii) other relevant data relating to the member.

(iv) data relating to the leaving of the member (technically it may be necessary to split this first screen, if it is required to search for the member, this will be a function of the User's Office system).

Thus all the external data needed to perform the calculations is made available to the computer and may be referenced in the future.

The Calculation

In carrying out individual pension calculations it is generally required to verify and possibly override the data held. Thus it is usual for there to be one or more stages of the calculations to be updated with the figures or estimates of the figures to the actual date of leaving. Pensionable Salary will be displayed with the earnings record and will be used to calculate the average salary, however in my experience of pension schemes it is quite often that another figure is used other than the figure calculated from the records, and in these circumstances the figure calculated by the system will have to be overwritten.

At each stage once the data has been validated more calculations are carried out and finally the calculations contain the full details needed to produce the letters to the individual.

Print Out

Further calculations may be undertaken for other members, or the results of a particular calculation or group of calculations printed out.

The 4GL will contribute to the style of the preparatory screens, in the ease of maintaining supporting files, and in the general ease of maintenance of the system. These all contribute to the accuracy of the calculation.

The calculations themselves will be just as demanding whether a 3GL or a 4GL is used, however, as I said above the appearance will be different.

Group Calculations

This of course is illustrated by a Pension Fund valuation and here I assume a small scheme and the use of factors.

Such group calculations differ from the individual calculations in two respects; firstly that we have a control file to control the whole process, and secondly a detail file which contains details of the calculations for each member.

Also, since the actual calculations take place at a given point in time, we have to extract the data from the standing data at a time when it represents the position at the valuation date. Then we work on that extracted data rather than take the data straight off the standing file and examine it in detail.

The routine is thus :-

Set up control file

Extract data on the appropriate date

Examine the data extracted for accuracy and amend it if necessary

Run the valuation on a given basis.

Check the calculations at random for accuracy.

Summarise the valuation results

Run the valuation on an alternative basis if required.

Confirm the valuation as final.

Inhibit further calculations on alternative basis.

Print all the reports needed to produce statutory returns.

These routines are the same whatever system is used — manual, 3GL or 4GL. The ease with which the data can be displayed, and the added accuracy which this imparts, is the value of the 4GL.

4GLs will make the writing of the systems quicker and in many ways more accurate. 4GLs come with many Tools to assist in the development of applications and one or two of them are widely used and if adopted for actuarial work enable the User to gain ready access to the files of his Office, if it is appropriate for him to do so. 4GLs are nearly always written for multi-user systems and have been recently made available for Desk Top machines, thus they are very well established for multi-user systems and contain all the checks and controls necessary for use in that environment.

2. Application Tools

It is all very well to have a data base, and to have a means of accessing data within that data base, but these facilities in themselves do not answer the requirements of the User. The two principal requirements as I see them are as follows :-

To create screens on which data can easily be added or updated,

To produce reports either on the screen or printed.

To find the records which will be displayed on a screen or report, one depends upon SQL, but to create the screens or the substance of a report, additional support is needed and this is provided by the Application Tools just mentioned.

In addition to these, as it were, prime Application Tools, ORACLE provides a number of others such as an integrated spread sheet, a system for writing menus and an ability whilst still in ORACLE to use other languages such as Cobol or C for specific calculations.

This latter facility I have not used as I wish to limit the range of skills within my Office (there is enough to learn already) and to introduce a Third Generation

Language on top of ORACLE is really too much. I have discussed this with those who know more about these matters than I do and the thought is that this facility is fine if you really need to go down into the depths of a computer. My general feeling is that with the rapidly increasing power of computers, resource to such techniques is likely to be very limited in the future.

3. Design Tools

Design Tools assist in a systems design and in my view are possibly the most important of the lot. They are, however, the latest developed and as such need quite a commitment before starting to use them. I do think that they will be a vital component in the future and the whole question of design is discussed in detail in the next section of this Paper.

ORACLE — THE DOWNSIDE

The advantages given by ORACLE are not achieved without a cost. ORACLE absolutely eats memory, both RAM and Disk. Here I think the ORACLE literature could do better. It gives the minimum capacity for ORACLE to work, not the capacity needed to be comfortable in it. I started off with 2 Mgbytes of RAM, 20 Mgbytes of Disk and I am now feeling comfortable with 4 Mgbytes of RAM and 80 Mgbytes of Disk.

This does not matter too much on a Desk Top where a few hundred pounds will buy the extra capacity, but it can be very expensive in other environments.

Calculations are slow and I would never use ORACLE for scientific work, whereas in pensions and life assurance the data maintained is paramount, with infrequent calculations required, and I am happy to live with it. I know that some look to the power of computers to perform even more complex calculations in attempts to refine the results of their work. This is fine, provided the priorities in terms of money reserves are properly controlled by well informed management.

The configuration of ORACLE on a Desk Top machine as opposed to any other machine is complex compared with PRO-1V and work is needed to master the complexity. I think it is worthwhile doing so. I can imagine that some may think otherwise.

It is a bore if one has gone to the trouble to buy a machine and set it up and then have to put in new disks and re-configure when halfway through.

STRUCTURED SYSTEM ANALYSIS

When I started out to use 4GLs I did not expect that I would move onto Structured System Analysis — I did not even know what it was! I fancy that some readers of this Paper will wonder what place Structured System Analysis has for actuaries. To develop the theme I will illustrate the problems I have faced and the resolution of those problems.

I was involved in the development of a system for which purpose the services of an outside computer-house were sought. The System Analyst was given considerable detail as to our requirements and asked to produce a specification for consideration by the User. His instructions included some broad standards to be adopted for the presentations of screens etc. In the event we were presented with pictorial representations of a number of screens, together with some general statements as to the functionality behind those screens, and told that this was his specification. The screens in fact departed in many respects from the broad standards which had been laid down and the file structure was not even presented for discussion. At an early stage the analyst had been given the first draft of a menu structure which he had ignored and, when asked why, he said that he always worked from the inside out. As a final gesture he ignored the requirement that people should be accessed via their name and we received a homily on the virtues of national insurance numbers! This put me in a state of some dismay as I did not feel that I had the technical competence to specify the system and yet it was quite clear that the System Analyst was not going to produce what was wanted.

Some time later I came across the approach to system analysis promoted by ORACLE, which is in fact the ORACLE version of the work carried out by many others for the last 10 or 15 years. This approach embodies all the features which I was looking for from the System Analyst mentioned in the last paragraph, as well as much more.

I was left with a sense of amazement that a System Analyst from a well known software house should be allowed to proceed in the way he had, and particularly that there was no management control over his activities. One is left wondering whether the management knew what was wanted or merely neglected to control their staff. This experience and further requirements led me to examine further how my ideas could be passed to a programmer, and I became familiar with the whole structure of Structured System Analysis and its extension, Computer Aided System Engineering.

I am committed to using Structured System Analysis in my work, for the convenience and for the discipline it imposes and, when it is possible, it is my intention to use Computer Aided System Engineering (CASE). Those interested in this work should refer to *Structured Technology — The basis for CASE* by James Martin and Carma McClare, published by Prentice Hall. I believe that systems should be developed in this way and my systems will be developed along this line.

4GLs come into this as they are part of the scenario for the use of CASE. CASE is used to analyse the system and then itself does a substantial part of the writing of the computer system using the 4GL of your choice. Thus we have moved to a position where the prime requirement is the definition of the system and the programming is almost incidental. In other words a great amount of thinking is needed before you ever start on the programming.

It may be that actuaries who have enquiring minds and wish to develop things would not like to be constrained by the use of such formal systems, and I can quite imagine this as I am one of those myself. However, to run an efficient office the use of these Tools is, in my view, vital.

It may be wondered why the use of these Tools is not more extensive and while I am sure that there are organisations where they are used extensively it is not yet universal. The answer seems to be that in the first place management is not always aware of these and, therefore, is unable to direct itself accordingly. As far as the staff is concerned a large proportion of the data processing personnel in this country is self taught and feels secure in the disciplines which are known, but does not feel too inclined to go outside those disciplines. Thus relatively new concepts are not always taken on board as quickly as they should be.

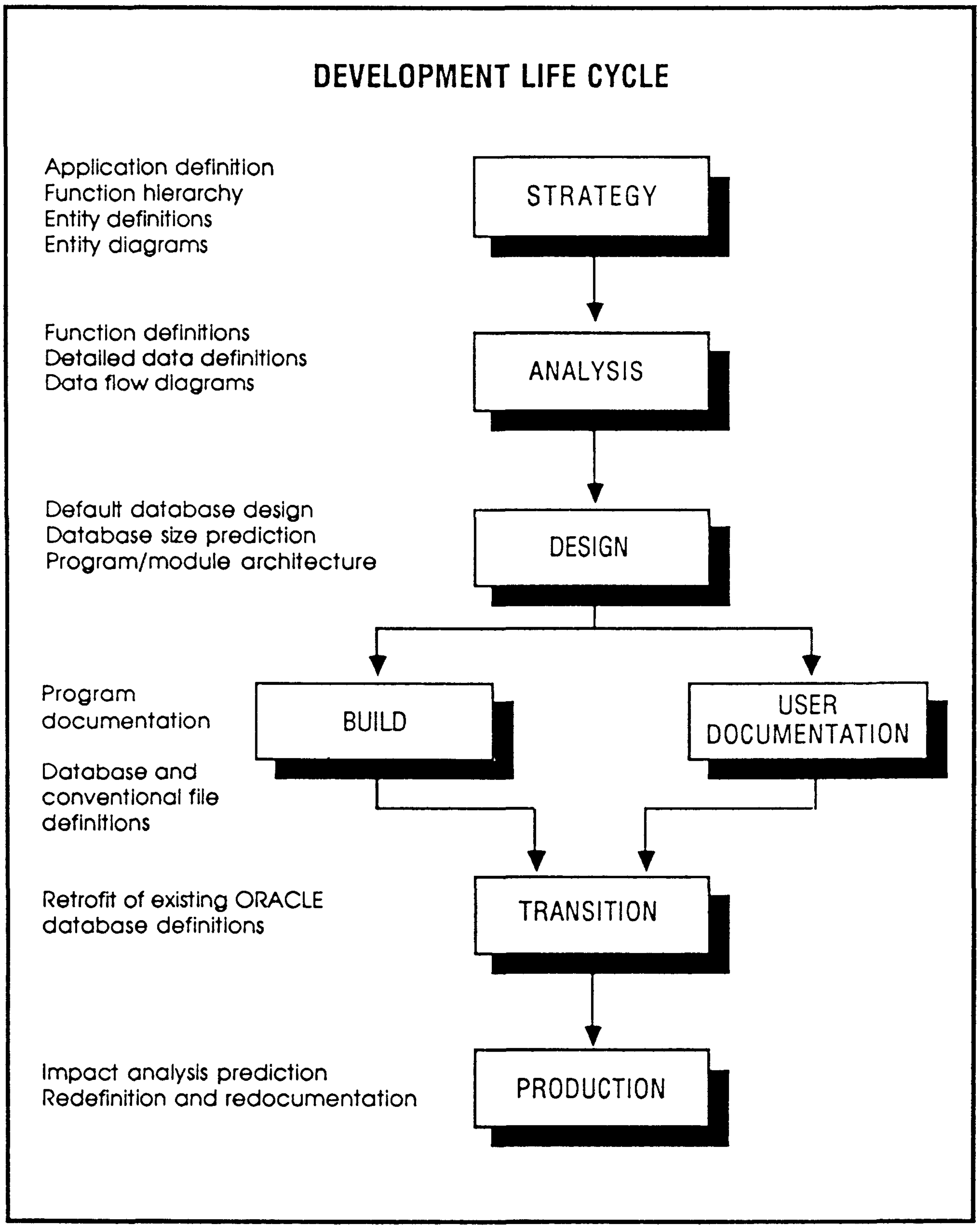
I spend the rest of this Paper discussing the implications of Structured System Analysis and CASE.

STRUCTURED SYSTEM ANALYSIS

Computer Aided System Engineering (CASE)

Structured System Analysis is simply a way of organising System Analysis so that a methodical approach is possible. This seems common sense but unfortunately, as stated before, the work done in this area is often ignored.

CASE is simply Structured System Analysis put on to a computer. This ensures that the methods of Structured System Analysis are fully carried out!


DEVELOPMENT LIFE CYCLE
Application definition
Function hierarchy
Entity definitions
Entity diagrams
STRATEGY
Function definitions
Detailed data definitions
Data flow diagrams
ANALYSIS
Default database design
Database size prediction
Program/module architecture
DESIGN
Program documentation
Database and conventional file definitions
BUILD
USER DOCUMENTATION
Retrofit of existing ORACLE database definitions
TRANSITION
Impact analysis prediction
Redefinition and redocumentation
PRODUCTION

Elements of Structured System Analysis

In this I am indebted to publications by ORACLE and I acknowledge here the help which they have given me. I show a diagrammatic representation of the "The Development Life Cycle". This contains all the elements, if you can understand the jargon. I did not understand it without a lot of effort and so I will comment on various features as follows:-

Strategy

This has nothing to do with computers but is an analysis of the Application which will be equally valid if a manual system is to be constructed.

Application Definition

This is simply writing down what you are trying to do. The mere fact of carrying out the work is an excellent discipline and does a great deal to clarify your approach.

Function Hierarchy

This describes the various functions to be carried out to meet the purpose of the Application. Typically it would take each prime function and break it down into a number of tasks.

For example if the Application is Pension Administration the prime functions may be :-

Maintain Members' Records
Process Leavers
Print Benefit Statements
Extract Valuation data
Maintain supporting files

These in turn will be further broken down.

Entity Definition and Entity Relationship

These form the heart of the analysis. An entity is a body of data, the items of which relate to each other according to certain rules.

In practical terms an entity giving details of a pension scheme member might be called "Member". It will contain such items as :-

Name
Initials
Title
Sex
Date of Birth
Address
A uniquely determined internally generated reference number.

Another entity might be "Earnings" and comprise :-

The Reference number above
The year in which the Earnings were received
The Earnings themselves.

Earnings cannot be included in the Member Entity as they are received in each year. In computer jargon, entities have to be "Normalized to the third degree."

The entity relationship defines how each entity relates to the other. Once this analysis is complete it enables the computer system to be built with ease. If it is not done, or it is done in the wrong way it is easy to run into trouble. I suspect many systems which gave trouble in the past did so because this analysis had not been carried out before the programming started.

Function Definitions
The functions identified in the function hierarchy are defined in detail.

Detailed Data Definition
Here the data defined with the entities is defined in detail, its length, its other characteristics.

Data Flow Diagrams
These show how the data goes from the input documents through the system to the output.

The Design Stage
This is undertaken by those with a knowledge of the language to be used to structure the whole computer system, but without going into the detail of screen design on other logic modules etc. Reports are specified but the layouts etc are not.

Transition Stage

Whilst I know very well that reference is made by ORACLE to the importance of this elsewhere I would emphasise that at this time there is a requirement to train Users, which is indeed a very important feature.

General

It will be seen that we start with an overall view of the application and gradually descend into the detail. By doing this we :-

(a) ensure that the total application is provided for even though it is not programmed immediately.

(b) obtain the full involvement of the potential users from the outset, before any computer work is carried out.

(c) can break down the work into small parcels which lead to more accurate estimation.

(d) obtain other advantages which relate to the detail of the work which is not my intention to discuss here.

Post Implementation Changes

Post implementation changes as opposed to additions, cause chaos. The cost of making them is many times the cost of including the detail from outset. Thus they are to be avoided if at all possible. Systematic system analysis is a great help in this.

CASE

CASE — (C)omputer (A)ided (S)ystem (E)ngineering is simply the computer implementation of Structured System Analysis. Each stage is written into the system on the computer, need not be done all at once, can be amended, and printed out for discussion with the User.

Thus we are achieving :-

(a) some certainty that the results will meet the need of User.

(b) some certainty that the system will work within the specified time.

Now, as an Actuary I do not want to use computers personally, they are great fun but only a tool for my work. I do need to communicate my requirements to my Computer staff; CASE is the way in which it can be done.

CASE: THE DOWN-SIDE

Whilst Structured System Analysis is well defined, the implementation in the various forms of CASE, seems to me to be slow and expensive. Of course this is inevitable and I do not complain about it but I do not wish to sing the praises of CASE without pointing out the problems.

CASE is implemented by many bodies each with its own area of application. Some lead to Cobol code generation, some to Relational Data Bases. Some run on one particular computer, some on many. The range and scope is increasing all the time.

The ORACLE products are not complete and are not available on all machines under all widely used operating systems.

For example the current release of the CASE*Dictionary V.3 runs on Desk Tops under MS-DOS, but is subject to one considerable restriction. Version 4 will not be subject to those restrictions but is not yet available on Desk Tops.

Other implementations of CASE are available on the mainframes and minis, but can be very expensive.

If you use CASE you will be at the leading edge of technology and you may not find it straightforward.

In my view CASE, in one form or another, will be the major area of advance in computing in the 1990s and this has been made possible by 4GLs.

CONCLUSIONS

I started with a wish to ease the burden of programming. I have concluded with the view that in future programming is a relatively subordinate part of the whole process of system development. I have found that the whole subject of system development has been studied in depth, but the results of that study are ignored by many.

I must ask, is this the way forward for the profession? Am I misleading myself, are other methods better? Should I have listened more to the System Analyst who ignored my instructions? But I do know that several directors of one of the biggest Merchant Banks spent days locked away with leading computer professionals, learning about CASE and how to manage its concepts within their business.

I have put my money on CASE.