Diploma

in

Business Administration

Study Manual

SYSTEMS ANALYSIS

The Association of Business Executives

William House • 14 Worple Road • Wimbledon • London • SW19 4DD • United Kingdom
Tel: + 44(0)20 8879 1973 • Fax: + 44(0)20 8946 7153
E-mail: info@abeuk.com • www.abeuk.com

ABE Diploma in Business Administration

Study Manual

SYSTEMS ANALYSIS

Contents

Diploma in Business Administration – Part 2

Systems Analysis

Syllabus

Aims

1. To give students an understanding of the tasks involved in designing a new information system to meet an organisation's specific requirements.
2. To enable students to create the major models used in the analysis and design of information systems.
3. To enable students to contribute towards the design of an organisation's information system taking into consideration the technological and human aspects of the design.

Programme Content and Learning Objectives

After completing the programme, the student should be able to:

1. Information systems: background and theory

- Understand the development of systems analysis over the past century.
- Understand how a system may be defined and its attributes identified.
- Understand how information may be defined and its attributes identified.

2. The systems analysis and design framework

- Understand the need for and the logic of the Systems Development Life Cycle (SDLC).

3. Structures systems analysis and design

- Understand the need for a more thorough analysis of systems by the use of various structured approaches known under the generic term of Structured Systems Analysis (SSA).

4. Systems analysis tools and techniques: the process model

- Understand the reason for, and creation of, Data Flow Diagrams (DFDs).

5. Systems analysis tools and techniques: the entity model

- Understand the reason for, and creation of, entity relationship diagrams/models (ERDs/ERMs).

6. Systems analysis tools and techniques: the behavioural model.

- Understand the reason for, and creation of, Entity Life Histories (ELHs).

7. **Systems analysis tools and techniques: CASE tools**

   - Understand the need for Computer Aided Systems Engineering (CASE) tools in the work of a systems analyst.
   - Understand the use of prototyping techniques in the reduction of a system's development time.

8. **Cross life cycle tasks**

   - Understand the need for the tools and techniques of project management, fact finding, documentation, quality management, interpersonal and presentation skills.

Method of Assessment

By written examination. The pass mark is 40%. Time allowed 3 hours.

The question paper will contain:

Eight questions from which the candidate will be required to answer any four. All questions carry 25 marks.

Reading List

Essential Reading

- Yeates, D., Shields, M. and Helmy, D. *Systems Analysis and Design*; Pitman Publishing
- Robinson, B. and Prior, M. *System Analysis Techniques*; International Thomson Computer Press

Additional Reading

- Lucey, T. *Management Information Systems*, 8th Edition; DP Publications
- Weaver, P. L. *Practical SSADM V4*; Pitman Publishing
- Laudon, K. J. C. and Laudon, J. P. *Management Information Systems*, 5th Edition; Prentice Hall (see chapter 11 in particular)

Study Unit 1

Information and Systems

Contents *Page*

(Continued over)

A. INFORMATION AND DATA

It is important that you understand the difference between information and data.

Data is raw facts; for example, a group of figures, a list of names and such like. By itself, data tells us nothing. Consider the following numbers:

212 263 189 220

In that form, the numbers could have a thousand different meanings. But if we add £ to the figures:

£212 £263 £189 £220

We can now see that they are monetary values.

If we add some dates:

January	=	£212
February	=	£263
March	=	£189
April	=	£220

Now we see that they demonstrate some kind of trend.

Alternatively:

212 screws, 263 brackets, 189 plugs, 220 bolts.

In this version we see that they are items on an order form.

We could rearrange the figures, we could add them up, find their differences, join them together and so on. Only when the figures have been further processed do they tell us something. They then convey **information**.

Information is raw data processed so as to convey a new meaning.

Theoretically, once we learn that new meaning, the information reverts to data, but in practice we continue to regard it as information.

B. INFORMATION NEEDS

In organisations, information is required as the basis of taking action – essentially by managers.

Role of Managers

Managers normally have two roles

- decision maker; and
- controller.

Stated simply, the processes involved in these roles is for managers to make a decision, check the outcome and either confirm or modify the decision in the light of events. This is a standard control process, involving five stages:

(a) Establish a plan.

(b) Record the plan.

(c) Implement the plan.

(d) Compare actual performance with the plan.

(e) Evaluate and decide further action.

You will readily see that information is needed at (a) in order to establish the plan, and at (d) to see how things are working out. There is a constant flow of information.

How Much Detail is Needed?

Now that we understand what information is, it is appropriate to gain further understanding of this information.

First of all, what level of detail is needed? Clearly this depends on the recipient or receiver of the information. For example, the shop manager may need the daily sales total, whereas area office may only need weekly totals. On the other hand, head office may only be interested in monthly totals. We should only provide as much information as is actually required.

"The more detail, the more information" is not necessarily true. Were head office provided with daily totals by each store, the "information" would simply be a mass of data requiring further processing before becoming information.

Therefore, although a mass of data is needed from which to draw the information, the detail being passed to managers should be no more than they require.

Information must be **relevant** to the purpose for which it is being used. Users should be provided with the minimum of information which will satisfy their needs. This is quite easy to achieve with computers, and information reports can be formatted according to individual user requirements.

Timeliness

"Information should be available instantly." This is possibly true at operating level, where the time-steps can be very short – for example, spoilage rate information on a production line needs to be readily available in order to check immediately on any deficiency as it arises. At managerial level, however, there is usually a longer thinking and decision time available, and so the need for immediate information is not necessarily valid.

The aim must be to provide **timely** information – that is, information at the time it is required. We can therefore say that information must be provided within a time-scale which enables it to be used effectively by management.

Accuracy

A high level of accuracy is usually achieved only at a high cost. Managers must be made aware of the cost impact if they specify very accurate information needs. Often slightly less accurate but sufficient information can be provided at a greatly reduced cost and it may also be more up-to-date.

Note the difference between **accuracy** and **precision**. Sales figures could be computed to the nearest penny, pound or thousand pounds – i.e. calculated to different levels of precision. However, if the data has been incorrectly entered, or the calculations carried out incorrectly, the resulting total will be inaccurate (wrong). Another type of inaccuracy is when approximate sales figures are collected because it would be much later in the month before exact figures were available. It is here that a compromise is made between accuracy and timeliness.

This compromise is important and we can think of a further example involving quotations for a proposed new product. Imagine the situation being one where the quotations are urgently required and yet 100% of supporting data is not known. In such cases, a 5% error margin may well be acceptable as a compromise in facilitating the progress of the quotations to the customer.

Rarity

Exception reporting draws the attention of users to changes in the situation. It avoids the need for managers to wade through long lists, looking for the one vital piece of information.

But who decides what is exceptional and how do they decide? When a system is being designed, careful thought must be given to this aspect and the ability to alter the parameters which denote exceptions should be built into the system.

Management by exception is an important approach which, if used correctly, can greatly increase the effectiveness of management. What criteria constitute "exceptions" must be regularly reviewed, so that they do not become outdated and thereby provide inaccurate reporting for consideration and actioning by management.

Quality of Information

Quality information needs to be relevant, reliable and robust.

- **Relevant** means it is pertinent to the recipient, who will then operate more effectively with the information than without it.
- **Reliable** means that the information is timely, accurate and verifiable.
- **Robust** means that the information will stand the test of time and failures of handling whether human, system or organisational.

C. TYPES OF INFORMATION

All businesses can be categorised into three main areas of operation:

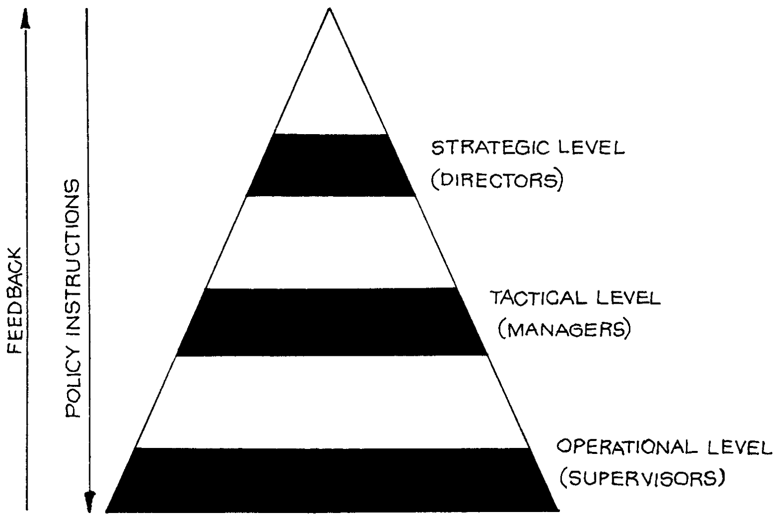


Figure 1.1

As we saw in Section B previously, all the information, at whatever level, should be:

- relevant to requirements
- at an acceptable level of accuracy
- up-to-date and timely.

Looking at information from a managerial viewpoint, we have the following.

(a) At the operational level, the shop-floor supervisory staff receive daily and weekly operational requirements such as order sheets, product changes and so on. They provide returns of orders filled, stock levels, spare capacity, manpower used and required and so on.

(b) Such information as above is used in summary form by managers at the tactical level in the actioning of reports on performance, budget versus actual and so on.

(c) At the strategic (director) level, information is much more global, having been highly summarised. It is at this level that policy making decisions, short- and long-term, are taken and passed down to the tactical level for actioning by the management.

In Figure 1.1, note how policies are passed down but their interpretation in terms of actioning is left to the management to administer. Feedback is then passed back up the strata originating in the operational level and finally, after much processing and refining, reaching the strategic level for overall review.

It is important that you are constantly aware that the above three information levels are not discrete. They all overlap, especially in smaller organisations where directors and managers may be the same people and where managers take a more “hands on” approach to operational matters.

It will also be useful for us to look more closely at each of these categorisations from an information viewpoint.

Operating Information

Operating information is used to instruct the employees of the business to ensure that they all know what is required of them. Despatch instructions to the warehouse, for example, state what is to be despatched and where it is to be sent.

Operating information – which can also be called routine information – contains the facts of what has been done, to provide a history of the actions carried out. A stores issue note records the fact that a quantity of raw materials has been given out to be used in production.

This type of information can flow to and from the outside world – orders received from customers, invoices sent to customers, or bank statements.

Without this information, the business could not operate. It is often termed “paperwork” since most operating information is presented in the form of paper documents.

When companies start to use computerised data processing, it is normally used first for the production of operating information. However, microcomputers are beginning to change this, particularly in very small companies where the amount of paperwork to be handled is relatively small and often very varied. In such circumstances, management techniques such as the use of spreadsheets often play a prominent role in the computer utilisation. Word processing also tends to dominate the small business area because of its universal application.

Management Information

All three categories of information can be collectively seen as management information.

(a) Firstly, we have those decisions associated with the day-to-day running of the business:

- "What action must be taken to bring the sales level up to the budget estimate?"
- "Will overtime be needed to complete a job?"

Most of these decisions are taken by junior levels of management – for example, by supervisors – and usually require that the information contains very detailed facts about the activities for which the manager is responsible. This information will come in the form of reports giving detailed summaries and analyses of the current situation.

This type of information is often categorised as operational information.

(b) Secondly, middle management is responsible for the overall running of the business on a week-to-week and month-to-month basis.

They must compare actual expenditure and sales with the targets provided by senior management, and require information to back up explanations of any divergences from the budgeted figures. They must also prepare the terms of reference to which junior management is to operate.

We can thus say that the information requirements of middle management are not so detailed, but they are wider-ranging – they will need to know about the outside world, as well as the progress of the business itself.

This type of information is generally categorised as tactical information.

(c) Finally, the senior management needs to plan into the future, often as far as five years ahead, and possibly further. They provide a framework for middle management by defining the policies to be followed and setting yearly budgets and targets. Their information requirements are therefore more general and each decision they take will have far-reaching effects.

This type of information is generally categorised as strategic information.

An organisation needs to collect data about items not immediately related to the running of the organisation – i.e. future prices of raw materials, competitive products, personnel skills, national statistics, new processes, new equipment, etc. A considerable volume of this data is to be found within any organisation and exists in many different forms. This data is used in considering strategy in medium- to long-term plans.

All the categories of information also include two sub-types: trigger information and background information.

Trigger Information

As its name implies, this triggers some action. In the case of operational information it is usually an instruction which is to be carried out, for example a job card which instructs a machine operator to make a particular item; or a request which requires a decision before an action is carried out, for example an order from a customer whose credit limit may need checking before the order is met.

A trigger for management could be a report indicating that production was lower than expected, and the manager would have to decide how the situation was to be rectified.

A trigger for the directors may be national statistics such as the exchange rates or a competitor's actions in introducing a new product. A corporate response will be required.

Background Information

This gives additional knowledge when trigger information is received. In the case of the customer order above, the clerk who checked the order would refer to the customer's account to find the credit limit and the current balance. These last two items are background information for the clerk.

In the case of the lower production report, relevant background information could be knowledge of seasonal demand cycles, or of a technical interruption to production which has been corrected.

In the case of the introduction of a new competing product, background information could be knowledge of the competing company's strengths or weaknesses.

D. MANAGEMENT INFORMATION

We have looked at information needs in the context of management decision-making. The computerised system that provides the relevant information is known as a Management Information System or simply as MIS.

We study these in much more detail later in the unit and elsewhere in the course

Whilst some of the information may be obtained directly from a screen display, as a result of an enquiry, most of it will be presented in the form of reports. Managers are now, however, beginning to access information in a third way – by extracting it from the main computer files and then manipulating it themselves, using a computer package such as a spreadsheet running on a personal computer or workstation. Various types of report are generated by such a system to help managers make decisions.

Regular Reports

These are the daily, weekly, monthly, annual reports produced automatically by the computer system. The problem here is to make sure that the information presented is relevant to the recipient and that regular reports do not have too wide a circulation list.

On-Demand Reports

The term "on demand" can be used in two ways:

(a) Particular managers may request a copy of a regular report only when they themselves want one.

(b) A standard report can be requested at a time when it is not normally produced; for example, if aged debtor reports are normally produced at the end of each month, a credit controller could request one in the middle of the month.

It is more efficient to provide information for just those people who require it but, especially in large organisations, on-demand reporting requires an organised response system. Otherwise, managers will become frustrated if they cannot get the requested information quickly when they do need it.

Ad-Hoc Reports

Managers sometimes require items of information in a form which has not been specified within the computer system. If the data is not available within the computer system then the request cannot be met. For example, if a Sales Director asks for monthly sales of a product for the past year but the computer holds only sales this month and sales for the year, the only method of obtaining the information is to go back to monthly sales analyses kept as hard copy – providing these exist in the form required.

Even if the data is within the computer system, extracting it may be a complicated process which cannot be carried out in the short time managers usually allow between issuing the request and requiring the output. To some extent, the links between main systems and spreadsheet packages, now quite common, make it easier to satisfy these requests. However, one way of coping with ad hoc reports is to try to foresee what might be required and to incorporate them as on-demand reports when the system is built.

Modern computer systems are now increasingly overcoming this problem by using report generators. Such programs allow managers to specify the format in which they want data to be printed out and they also allow for sub-totals, totals and other calculations to be incorporated therein. Provided that data is held on the computer system, the specified report extracts it quickly and accurately in the format specified. Such reports can either be used as they are or incorporated into word-processed reports of a longer nature. Whichever reason triggers the request, properly used they can provide a cost-effective means of enabling ad hoc reports, user specified, to be produced.

In many circumstances the ad hoc report tends to become a regular supplement to the standard report pending the next system revision. This would then establish its justification as being worthwhile of inclusion as a standard report and the system would be amended accordingly.

Exception Reports

Exception reporting was mentioned earlier. An exception report could also be a regular report, for example a weekly report showing all those stock items with orders overdue by more than a week.

Analyses

An analysis summarises information in a particular way. For example, sales analyses can be summaries by:

(a) Product

Which products were bought by which customers? The analysis could be summarised at the product group level or give information down to individual products.

(b) Customers

Which customers (type of customer) bought which products?

(c) Region

Which geographic (or sales) region bought which products?

Data held in a computer can be analysed in so many different ways that it is important that designers consider what will really be useful to their particular company. The ability to vary the grouping and selection criteria is also useful.

When packages are produced, sales analysis is one of the areas where it is often possible to customise the reports which are produced. This is because each company has its own individual requirements.

When deciding upon the depth of sales or other analyses, it is important to remember that storage has a cost and that this cost must be compared with the benefits of such storage. Despite storage having increasingly large capacity and its costs reducing greatly, this compromise remains valid today.

Forecasting

Many packages now include a forecasting module, but it is important that managers know how these forecasts are obtained, since there are a variety of techniques which can be used. Strictly speaking, information based on historical data projected into the future is a prediction, whilst forecasts are based on subjective judgements as to the effect of various factors. Many sales forecasting systems start with historical predictions.

E. SYSTEMS THEORY

A dictionary definition of "system" is:

"Anything formed of parts placed together or adjusted into a regular and connected whole."

It would be worthwhile if you were to pause at this point and think of some of the systems you are familiar with.

You may have thought of:

- central heating system
- road/railway system
- administrative system
- government system
- management system

and so on.

System Properties

Systems receive input and produce output.

Systems consist of sub-systems which are themselves systems performing some function on behalf of the larger system. A fully integrated system consists of sub-systems, each of which has, as input, the output of another sub-system so that all the sub-systems together fulfil the overall objective of the whole system.

A system must have a **boundary**; outside the boundary is the **environment**. The environment of a computer system includes any people or business activities which interact with it, the sources of the data which forms its input and the recipients of the information it provides.

The art of systems analysis is being able to define **system boundaries** – to decide which parts should be included within a particular study, so that a logical and convenient model can be prepared.

Another definition of system is "the method by which an individual, organisation or mechanism accomplishes the tasks necessary to achieve its objectives". The method used will be made up of a number of related **procedures**, and – in a large system – there may be groups of procedures called **sub-systems**. Figure 1.2 shows a business system consisting of four main sub-systems, each of which can be divided – as shown for accounting – into a number of smaller sub-systems.

A system can thus be thought of as **hierarchical**, and this hierarchical nature extends both ways, in that the system being described can also be looked at as being a sub-system of a larger or wider system. Staying with our accounting example, a sales ledger system is composed of a number of subsections, whilst itself being a sub-system of the **total accounting system** of the organisation.

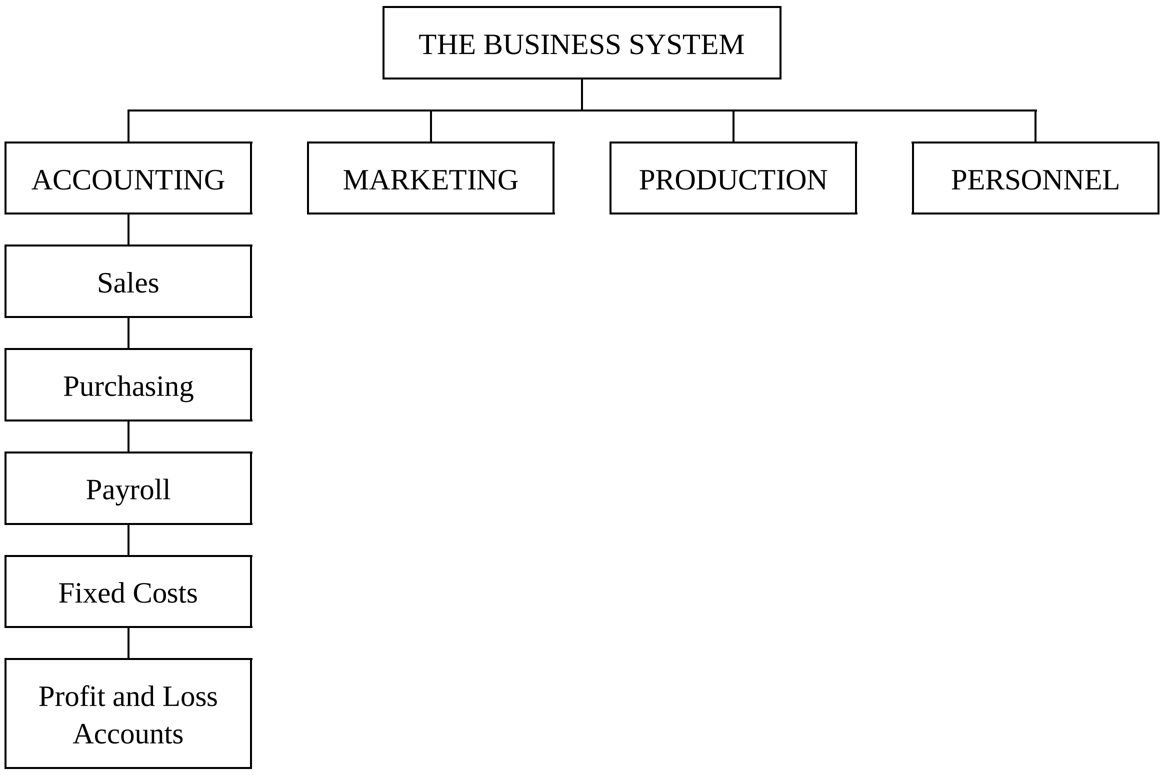


Figure 1.2

Probabilistic and Deterministic

Consider a slot machine. Here you place your coin in the machine and, provided the machine is stocked, the required item will be delivered automatically. The outcome is completely predictable; a slot machine is an example of a **deterministic** system. Whenever we take a particular action, the result or resulting action will invariably be the same, providing that the system is working correctly. Every step in the system has this feature and so the total system is deterministic in nature.

A computer can be considered to be a deterministic system since it automatically follows the series of instructions it has been given. At least this is true for traditional computer systems, which includes the vast majority of those currently in use and certainly all those used for data processing.

Alternatively there are **probabilistic** systems whose predictability is less than that of deterministic systems. If, instead of always getting an item out of a slot machine, you sometimes got nothing – for example, from a fruit machine or some other gambling device – then this system would be probabilistic. Stated simply, we can never be certain of how such a system will work, but we can assume that a specific action will take place, based on our previous experience or knowledge.

An example of a probabilistic system is a game of cards or the pricing system of an organisation.

Open and Closed

When a system is isolated from its surrounding environment it is a **closed** system. When a system responds to input from its environment and provides output to the environment, it is an **open** system. The same system can be seen as closed or open when viewed from different perspectives. For example, a payroll system is closed when viewed from an organisational position, but is open when seen from inside the payroll section.

Quantitative and Qualitative

A quantitative system will process and output actual values. All financial systems are quantitative.

A qualitative system processes and outputs less measurable quantities such as "a better service to clients".

F. OBJECTIVES OF A SYSTEM

Objectives are the **goals** towards which a system is working. In an organisation the overall objectives are seen differently in the separate departments, each of which tends to have its own objectives. For example, with inventory, Marketing wants to be able to supply the complete range of products whenever customer demand occurs; Production wants to manufacture products in long runs of each product to reduce set-up costs, etc.; and Accounting wants to keep stocks to a minimum.

Decisions selecting between conflicting objectives have to be made and once made they become **policies** and lower level decision-making will have to be made within the context of company policies. If, for example, it is company policy to buy computers from one particular supplier, then a proposal for a computer system will need to be designed round the models offered in a given range, unless a very good case can be made for using a machine from a competitor.

Necessity for an Overriding Objective

By definition, every system must have an objective. It is essential that an objective is decided upon even before the system is designed. At this stage it is likely that a number of conflicting objectives will be considered. For example, when discussing the setting up of a production line, the following objectives may be amongst those considered:

- the complete safety of the line.
- the best environment for the production line.
- the best equipped line to produce a better quality product.
- the most efficient line in terms of cost per unit of output.

No doubt there will be other objectives, but you can see that many of them will be in conflict. This conflict of objectives is a characteristic of all systems. It is essential for all objectives to be considered and an objective established that is a compromise between all those in conflict.

The overall objective that has been established for a system must be one that can be achieved. Although the objective may at times seem difficult or nearly impossible, it is of prime importance that in the long run it is achievable.

Objectives of a Wider Context

We have said that individual systems form part of a larger system. It is essential when setting out systems objectives that account is taken of the objectives of the next higher system.

It is better if the objectives of the higher systems are known before those of the sub-systems are established. The communication of the objectives from the higher to the lower systems is of great importance.

How Objectives are Created

The overall objective of a system is created firstly by listing all possible objectives. Many of these will be in open conflict and it is therefore necessary to weight, in order of importance, the objectives so far specified.

This list will have been formulated by consulting the interested parties. When an overall objective has been decided upon, it is necessary to "sell" it to those parties whose ideas were in conflict at the earlier stage. Perhaps at this stage minor alterations will be made after further consultations.

G. A BASIC INFORMATION SYSTEM

Information systems provide information to managers and whoever else requires it. It is difficult to classify information systems as deterministic or probabilistic. They are programmable (i.e. can be computerised) and thus appear deterministic. From the user viewpoint, however, there may sometimes be no satisfactory output, thus making them appear probabilistic.

In the same way, they can be seen as both open and closed. However, as we shall always want information systems to evolve, they are primarily open.

They do need to be quantitative though. The user needs concrete information on which to base decisions.

An efficient and effective information system will always give:

- THE RIGHT INFORMATION
- TO THE RIGHT PERSON
- AT THE RIGHT TIME
- AT THE RIGHT COST

(a) The Right Information

The information should be in detail appropriate to the recipient – detailed plant-by-plant figures for plant and works managers and only the summarised variances for the managing directors (although they can always get supporting information if they want it). Beware, however, of being too summary orientated: it is essential to measure what is happening from time to time, otherwise it is easy to become a summary expert without knowing any real facts. The information should be relevant.

However, a flow of information across specialised barriers is essential, otherwise managers get too shut up in their own worlds and may forget for whom they are working and what effect they have on the company's operations. It can be dangerous to feed summaries to persons other than the specialist who is in control, for it can lead to a lot of unfounded pessimism at bad figures or joy at good ones, when only the expert knows how bad is bad and how good is good!

(b) To the Right Person

This depends greatly on who set the budget and how. In the ideal situation, information is fed to the lowest possible point in the company hierarchy at which the recipient is happy to accept responsibility for the figures and do something if they are wrong.

(c) At the Right Time

In principle, the right time is "not too early, not too late", so that the best form of corrective action can be taken. There is an obvious difference between statistics on the temperature of a chicken run, where only several minutes outside certain temperature limits can cause death to the birds, and statistics on aviation insurance (figures more than once every six months are useless).

(d) At the Right Cost

There is no point in spending much time and money getting figures accurate to four percentage points if equally appropriate controlling action could have been taken on ± 5%. This quest for accuracy usually results in the information arriving too late to be of any use. It is a natural feature of industry to specify the tolerances to which any product should be made. This parallel should be extended to information: accountants, for instance, have to be told what information is required and how accurate it has to be.

H. DECISION SUPPORT SYSTEMS

We all make decisions each day. Most are of no significance. Some, however, are extremely important.

In taking decisions how do we know what is "right" or what is "wrong"? Perhaps rather than a black or white right or wrong consequence, there is a sliding scale of degrees of successful or otherwise decision-making.

Normally, in making decisions, we rely on past experience. However, with businesses operating in an increasingly complex and changing environment we may not be able to rely on past experience to lead us to the right decision, and other means must be found.

Decision Theory

Decisions have to be made whenever alternative courses of action are available. Each course of action will lead to one or more resultant states. If there is only one possible outcome as a result of a particular decision, the resultant state is certain.

Decision-making can be classified as being under:

(a) **Certainty:** once a decision is made, there is only one result. Deciding which decision to take is a matter of determining this resultant state for each possible decision and selecting the most desirable one.

(b) **Risk:** there are a number of possible outcomes for each course of action and the probability of occurrence of each resultant state can be determined. Various techniques can be used here, but the results obtained are completely dependent upon the accuracy with which the probability of each outcome can be estimated.

(c) **Uncertainty:** there are a number of possible outcomes for each course of action, but the probability of any particular one occurring cannot be determined.

(d) **Conflict:** the possible courses of action are affected by the behaviour of a competitor. **Game theory** is a method of developing a strategy for use in such circumstances but the situation is usually very dynamic, with competitors modifying their behaviour as soon as a decision made by their opponent is known.

Role of Computer Systems

Making decisions is the responsibility of managers but they can be helped to reach the "right" decision if they are provided with up-to-date and accurate information. Here is where computer systems play their part. Some of the facilities provided by a computer which can help a manager in reaching a decision are:

- Rapid processing of historical data.
- Ability to carry out simulations.
- Ability to carry out calculations rapidly so that it becomes feasible to use operational research techniques of various kinds; these techniques can calculate optimum solutions and show the effect of variation in some of the parameters.
- Highlighting of the unusual – the value of a piece of information often depends on its unusualness and, by means of exception reporting, the computer can draw the attention of the manager to outstanding items.
- Personal computer systems which allow managers to manipulate data and information themselves so that they become more aware of its implications.

Thus a computer provides managers with powerful and intelligent tools to assist them in their job; these tools are often called **decision support systems**.

MIS, EIS and DSS

Earlier in the unit we said a little about the Management Information System or MIS. This is the computerised information system that carries all the corporate information used in all the company departments.

Clearly, this information is provided to give a basis on which decisions can be made.

However, as part of the MIS, an EIS or Executive Information System may be provided. This system will draw information from the MIS files and provide particular executives with exactly the information they require to support their decision-making. It will assist direct access to the necessary data and provide a means of manipulating the data to compile reports as required. We have already seen the various reports that can be provided by such systems. The EIS will generate tailored reports for particular executives.

When an EIS is designed to provide information specifically for specified decisions, then we sometimes refer to the system as a DSS or Decision Support System.

Study Unit 2

Computing Within Organisations

Contents *Page*

A. THE DEVELOPMENT LIFE CYCLE

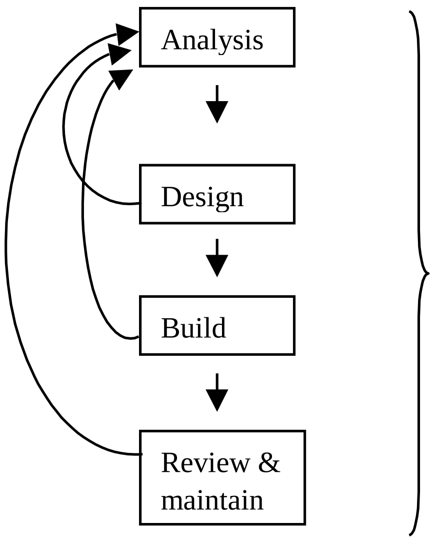
Introduction

In this study unit, we will be looking at the technique of systems analysis in detail. Systems analysis involves dividing a situation or process into constituent parts and then studying these parts to understand how they interact with each other.

We will begin our study of the subject right at the beginning with a brief look at the work of Frederick Winslow Taylor who is said to be the father of scientific management, method study and work measurement. He was also the proponent of systems linking work output directly to the level of reward. In the introduction to his book, the *Principles of Scientific Management*, F.W. Taylor states "In the past the man has been first; in the future the system must be first."

We will now look at how systems analysis is practised today.

Systems analysis is the first stage in the development of a structured system such as a computerised system. In general, we can break the development down into four distinct sections: -



The **development life cycle**, so called because development work continues throughout a system's life. Note how every subsequent stage refers back to the analysis. Even on-going maintenance during an operational system will refer to the analysis.

Figure 2.1 Development Life Cycle

As figure 2.1 shows, these stages are not strictly sequential in that the developers working through any specific stage will be continually referring back to any of the previous stages. At all times, a stage development must be consistent with all that has gone before, and in particular with the systems specification that was the outcome of the analysis stage.

Frederick Winslow Taylor

Taylor's work emerged in the late 19th, early 20th century with the rapid growth in the scale and complexity of industry. Agents or managers were replacing the enterprising owner who had become an absentee from the business. The old relationship between owner and worker changed as generally incompetent managers took over. Taylor saw these managers as ignorant, selfish and irrational.

His work drew on his experience of the old approach dominated by managerial ideas and actions. He argued that he was applying "scientific" method to the new practical management problems. His principles, which draw on a systematic study of work behaviour, and on his view of the direct relationship between incentive and efforts are still prevalent in modern organisations. Piecework, incentives and bonuses are still believed in by managers.

- **Work Measurement**

  Taylor's management devices of work measurement and job specialisation have decisively influenced the work culture. His models are based on assumptions about an "economically motivated” worker and are linked to notions of worker-management relations. Taylor’s diagnosis of industrial ills can be reduced to one principle – that wastage of resources and time in the workplace is causing inefficiency and disinclination to work! This is particularly evident where the working group controls the output. The situation is similar among managers who lack information about worker abilities and how long it takes to do tasks. When management discovers that the job is too easy they autocratically alter timings. In response, workers get together in order to find ways of overcoming this tendency and to gain proper reward for their efforts.

  Taylor introduced ways of measuring efficiency, such as work-study, method study and work measurement, so as to capitalise on worker rationality and improve management information. In the early part of the 20th century, Frank Gilbreth reduced the basic number of operations in bricklaying from 18 to 5. This was the beginning of the age of the management consultant and in many industrial corporations such as Ford, work-study was applied with considerable success.

  Taylor laid down the following management rules: -

  1. Find and then establish the best method and a reasonable time for completing a task.
  2. Select staff on technical criteria alone, such as their ability to do the job, and use rigorous training to develop their skills, using incentives where appropriate.
  3. The management structure and the technical planning, inspection and the clerical support work should be separated from workplace operations. Taylor proposed "a functional foremanship" described by the ‘order of work clerk’, the ‘time and cost clerk’ and the ‘inspector’, all in direct contact with the individual worker. These are equivalent to the modern roles of production planning, production control, inspections, maintenance, and personnel management.

- **Job Specialisation**

  Taylor encouraged the creation of specialist jobs and 'thinking departments'. For example, planning departments took planning responsibility away from line management. Taylor soon discovered, however, that conflicts emerge between these specialists and the older style generalists. In the early days, the specialists were greatly resented and Taylor had to modify his views to allow for this.

  Over time, the situation changed and specialist departments came into their own. Management recognised that even its tasks were governed by measurable rules. No longer could they rule the work place in an arbitrary manner. Managers had to accept advice from the specialists and had to hand over many management functions to specialists. The concept of **team co-operation** developed, and departments such as administration, accountancy, sales and personnel appeared in large organisations.

  An administrative bureaucracy soon followed and it was clear that there was a need to study the flow of paper through this bureaucracy. Taylor’s ideas were adapted and the science of **Organisation and Methods (O&M)** was born. Various tools were developed to help and **flow charts** were typical. Flow charts are a picture of an activity set out in sequential steps. The following simple flow chart, using traditional notation, shows the activity of making a pot of tea: -

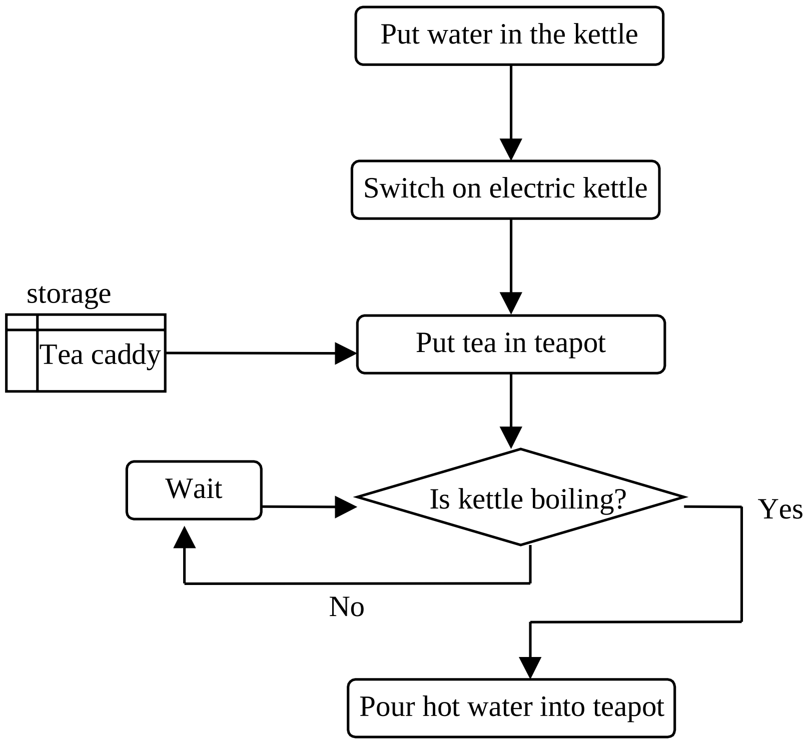


Figure 2.2: Flow Charts

During the middle years of the last century, O&M specialists found computers ideal for their analytical techniques. The techniques, as originally devised by Taylor, and subsequently refined by others, were 'scientific' and precise and so were easily computerised. The O&M specialist now had a powerful tool, and their particular place in every organisation was secure.

The term O&M was gradually replaced by another term, '**systems analyst'**. Today, no business project would be implemented without the attention of the systems analyst. Taylor's techniques, which had originally met with so much scepticism, are now accepted as the basis of all development projects.

SYSTEMS ANALYSIS – FEASIBILITY STUDY

If you were asked to organise the assembly of a large jigsaw with a number of helpers, some more skilled than others, you would probably break down the activity 'do jigsaw' into a series of sequential and parallel tasks and allocate resources or helpers to each task. You might adopt one of the following methods:

Method 1

- Sort out all the edge pieces (all helpers).
- Complete the edge pieces – adults and sort out the 'sky' pieces from the picture detail (children)
- Complete the sky – adults and extract and complete easily identifiable objects (children).
- Place objects into approximate positions (adults).
- Complete jigsaw (all).

Method 2

An alternative approach would be to discuss each piece in turn as a committee and either place it or discard it.

Method 3

A third approach would be to abandon the committee and allow each helper to select pieces at random, either placing them or discarding them.

Whilst we have all adopted the third method at some time or another, I think you will agree that the first is the more likely to succeed. Let us examine why.

(a) The problem has been broken down into manageable and understandable activities.

(b) The activities have been structured so that helpers can get on with their work with a minimum of dependence on each other.

(c) Each helper can advance the activity without the detailed involvement of you, the organiser. (However, co-ordination of the effort is still required – you need to see the overall picture).

All system developments follow a common pattern, known as the **development life cycle** (see figure 2.1). All subsequent stages of the development continually refer back to the analysis. In other words, it is very important that the whole development is built on a firm analytical foundation.

The **initial study** is, as you would expect, the beginning of analysis proper. But a word of caution first. There is often confusion in the naming of the various studies made at the beginning of a development project. Sometimes an initial study is made to quickly assess the proposals before any further work is done. If everything seems acceptable at this stage, then a **feasibility requirement** study is undertaken. This is used to investigate the situation in detail so that the new requirements can be formally reported as a **requirement report**.

However, not every organisation does separate reports and often, the initial study is very informal and is not really considered a report. The feasibility study becomes the first stage. The requirements are also often built into the feasibility study so that there is no separate requirement report. The approach used is not important so long as we understand that there are essentially three stages:

1. A brief assessment of the situation
2. A detailed assessment of the situation
3. A set of requirements

For our purposes, we shall consider the analysis study as encompassing three stages:

- the initial study
- the feasibility study
- the requirements analysis

Employee Consultation

At the very beginning of any systems project the golden rule is to get the user involved. If the user has *"had a good idea"*, or is in genuine need of help in the provision of information, or has developed a managerial style which always involves the specialist, then the analyst's approach will be consultative or catalytic; there will be participation on both sides and the study will make good progress. On the other hand, if the managerial attitude is, *"I know all there is to know about my work"*, or if a change is being imposed due to market conditions or from a higher level, then the analyst may adopt an innocent *"I wonder if you can help me"* or autocratic *"We have to get this* *done"* approach. Here, there may be antagonism or suspicion and the analyst will need to work at developing relationships as well as establishing facts. Of course, analysts, by their own attitude, will influence the relationships between systems staff and user staff. Good preparation is essential in all face-to-face discussions, but the analyst must resist the temptation to show off or impress with his grasp of the procedures. If a manager is determined to show that he understands every procedure in his department, this is a useful exercise for the analyst, who must become the interested listener.

The development manager will ask a whole series of questions. The answers to these are worked out between the development manager, the company management, users and others involved in the development. This initial study is designed to test the feasibility of the chosen system. Typical questions could be:

- What is the scope of the proposed system?
- What are its boundaries?
- What is the system for?
- How many people are involved and who are they?
- How many people will be affected?
- What development method is to be used?
- How will the system's success be measured?

There are many similar questions that could also be asked, but these will be specific to the project. It is also necessary for a decision to be made over which development method to use. This will depend upon the way the company sees itself and on the skills of the analyst. In most cases, the method called **SSADM** (Structured Systems Analysis and Design Methodology) will be the choice.

An important point to keep in mind is that few systems exist in isolation. This means that any new system will be required to fit into a pre-existing computerised function and methods used previously will have a bearing on what is used on the new job.

Initial Study

Should a system be accepted for development, then initial requirements or **terms of reference** are set. The following is an outline of a Terms of Reference memo:

From: Systems steering group

To: Systems development group

Terms of Reference

Recommend methods for the effective management of:

Speed of activity 1

Speed of activity 2

Number of errors allowed in documentation

Investigate and list the advantages and disadvantages of:

On-line data entry

Batch data entry, etc

Report on the feasibility of the project

Much preparatory work needs to be done before the actual development project begins. There are three distinct parts to the initial study:

1. **Personnel.** Is the expertise of the existing staff sufficient to tackle a development project? The same question must also be asked of any outside agency that may be proposed for the job. If the answer is ‘no’ then the expertise must be hired in or existing staff trained. There is then the question of whether the extra cost is justified by the proposed system?
2. **Technical.** Is the current hardware and software capable of supporting the proposed new system? If the answer is ‘no’, then there will be considerable extra cost which may affect the proposals.
3. **Financial.** Even if the proposed system does provide an acceptable return on investment, is the capital available for the development? Again, obtaining extra capital can be expensive.

As we have seen, the analysis stage seeks to provide an interface into the subsequent stages and it is this stage that links any existing system with the new system. It is therefore most important to fully understand the existing situation. All organisations must approach the establishment of computerised systems with care as there is a tendency for very senior managers to decide in some ad hoc way that they need such and such a system.

Feasibility Study

Once a project has been provisionally accepted for development, it is the task of the IT department to investigate its feasibility. The development request comes, initially, comes from a user department of the organisation. The user sets out the requirements formally and presents them to the appropriate committee. The IT section will then have to investigate the request.

The feasibility study is needed to obtain sufficient information for a decision on the next step in the project. The analyst looks for details of the organisation, staffing levels and cost, and sufficient information about procedures to relate to the problem in hand, whether it be an imposed problem through legislation and the market, or a required one originating from a good idea or a need.

Most of this level of information can be obtained from the manager. Quantitative information, such as the number of invoices produced, the number of products in stock, number of customers in debt, gives the analyst a basis for comparisons. Cut-off times, frequencies, peak periods and forecasts are all useful items and must be consciously sought. Again, the development of a checklist of questions for the discussions is essential and might be defined as a standard. From these initial facts, the analyst builds up a picture of the current activities and attempts to comprehend the scale of the project ahead. Whilst no firm decisions can yet be made, possible approaches to the solution can be examined. It may be quite apparent that a non-computer solution would be appropriate, or, in a labour-intensive procedure, that some form of automation would

The feasibility study involves a series of five steps:

1. Understanding the problem

The first task is to go back to users of the proposed system and check their understanding of the problem. Suppose, for example, the despatch department is recognised as being very slow and the project is intended to solve this by having a system to produce a list, in shelf sequence, of goods to be despatched for each order. Further checking may reveal that the management of the warehouse is inadequate and that goods are always placed on the nearest available shelf. There is then no fixed position for each item. The developers could have gone ahead and produced an excellent system, but the problem would not have been solved. This could very easily bring the developers into disrepute!

So first of all, the developers must obtain a clear definition of the user requirements. However, even before obtaining this, the developers need to establish who is the client: the person or group with overall responsibility for ensuring that the project's business objectives are met. Different people will almost certainly have differing views of what is required so we must be clear at the outset who will have the final say. In discussing objectives with end users, a number of techniques can be used, including the identification of critical success factors. These seek to define various business related attributes which, if achieved, will demonstrate the success of the particular project. Examples of critical success factors could be:

- To achieve a 15% market share
- To ensure all customers are invoiced in line with company policy
- To answer 80% of queries from clients within 5 minutes and all queries within two days
- To maximise profits from the sale of widgets

The high-level business functions necessary to achieve the critical success factors are then identified in conjunction with the user to form the basic objectives. These processes may relate directly or indirectly to IT or may be totally separate. The reason why queries are not answered, for example, may be that there are insufficient trained staff to deal with users or the telecommunications system may have become outdated and in need of change.

Constraints are also identified at this stage since they are likely to have significant implications for the project. Examples of constraints are:

- Cost not to exceed £250,000
- No additional staff to be employed
- No significant relocation of staff
- Project to be completed by end of financial year
- System to operate on existing hardware

More detailed objective requirements may also be agreed at this stage, such as:

- Response time/speed of transactions
- Return on investment
- Ease of use
- Job satisfaction

These objectives are intended to ensure that we solve the right problem for the right person. No matter how well a project progresses, if it is seeking to solve the wrong problem there is no way it can be successful.

Next, the purpose of the feasibility study will be agreed. Sometimes referred to as a **Scoping Study**, this phase of the project seeks to determine the overall viability of devising a product or service to meet the user's objectives within the constraints laid down. It will ensure that the high level functions identified above do, indeed, cover the full scope of the product required to meet the aims and objectives of the client. The skill of the IT professional is needed to determine whether or not it is feasible to develop such a product within the constraints. Reference can be made to similar systems in other organisations, to commercially available packages and/or to previous experience within the company of devising similar products. In summary, the main purpose of the feasibility study is to establish whether or not the proposed system is both feasible and desirable.

2. Preparing gross estimates of costs and benefits

The principal costing technique used is **cost/benefit analysis**. The costs of the existing system are evaluated, and then the costs of the proposed system are estimated, including the development costs. The net cost is then set against the likely benefits. These are in the categories:

- Direct savings, such as a cost reduction.
- Measurable benefits, such as actual financial return from the new system.
- Intangible benefits, such as an improved ordering system. Whilst these are difficult to quantify they generally give better customer service.

3. Full investigation and consideration of alternatives

When all the relevant facts about the existing situation have been collected, assembled and analysed, different approaches to solving the problem can be considered. It is not enough to add the costs, take the benefits in each alternative, and then recommend the least expensive. For example, one system may have a ten-year life and $100,000 net return. Another may have only a five-year life and a $60,000 return. On the face or it, the first is the better choice. Yet it has only a $10,000 per year return, whereas the second has a $12,000 per year return. So the latter example is the better one. On the other hand, should the costs of the second be all in the first year, and the costs of the first be evenly spread, then inflation will make the first the best choice.

It is also necessary to consider the risk factor. A system with a ten-year life is less precisely estimated than one with a five-year life.

4. Outline plan of proposed system

To lay the foundations for subsequent planning and control of the development project, it is necessary to prepare an outline plan. This will simply be a diagrammatic version of the proposed system requirements.

5. Presentation of findings

The conclusions of the study investigation and proposals need to be presented in a way to enable the steering committee to make decisions.

SYSTEMS ANALYSIS - DATA COLLECTION

Methodology

The first job is gather data about the existing situation. There are a number of techniques that can be used:

- Interviews with user staff
- Questionnaires issued to affected staff
- Observation of the existing situation in operation
- Examination of archival documentation on the existing situation

Interviews and questionnaires enable us to ask affected staff directly what their perceptions of their requirements are. In a large organisation, interviews can only be arranged with selected staff as each interview takes a considerable time. Questionnaires are then useful when many staff are to be canvassed. In each case, much preparation is required. Interviews must be constructive and sensitively conducted and the setting of questionnaires is a skill in itself.

Observation of a working situation can be very informative. It is a good way of identifying bottlenecks and such like. However, care is needed as most people work more diligently when being watched.

Examining documentation can also be informative, as it will show what was originally intended by the system. It also gives a good comparison for model overlay once a new model is built. You must however be aware that the documentation will show a historical perspective, and not necessarily what the system has evolved to.

Data Analysis

The next question is: 'What facts are required?' Typically these could be:

- Identification of the components of the system, (i.e. Products, personnel, departments, and so on)
- Details of all relevant items such as code numbers, descriptions, values, quantities, etc.,
- Volumes of data, (i.e. The flow of documents, data flows and fluctuations)
- The degree of accuracy in measurements
- Sequential steps of the existing situation
- Existing standards
- Available hardware
- The environment details and the framework of which the system is a component
- Details of all storage media
- Any existing problems
- All exceptions to the normal situation

You will no doubt think of other features that could be included in the list, as all systems are dynamic by nature. However, there is enough in the list to give you a sense of the range of the analysts' investigation.

Next, the fact must be verified. The usual method of verification is to collect similar facts from more than one source. It is surprising how often individuals give different versions of the same problem. There are also statistical techniques available to check the data, but we do not need to concern ourselves with these.

Data presentation

Now, assemble the collected facts into some meaningful form.

There are many ways of doing this but a diagram is normally the best form of communication, and figure 2.3 is one suggested starting point for a warehousing operation. This is a **block diagram**. This is a very general diagram with no specific notation. We use it simply to clarify the situation.

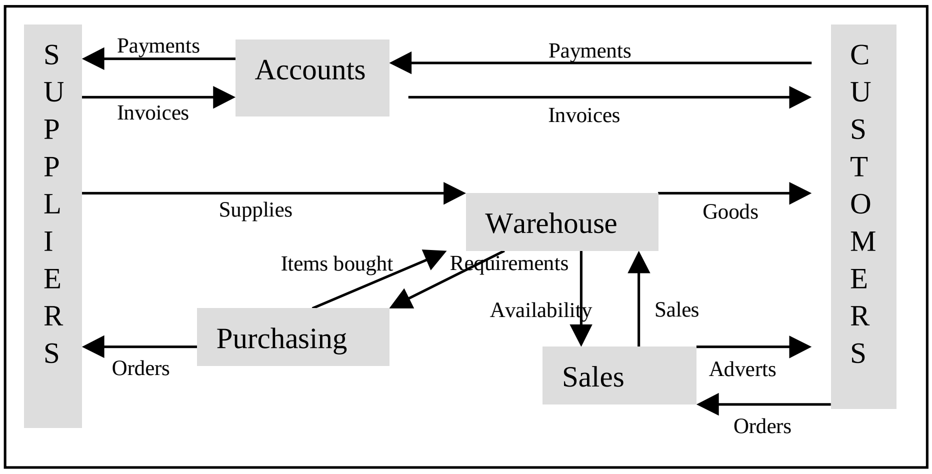


Figure 2.3: Block Diagram

Another useful diagram style is the **spider diagram**. This is a good way of identifying the important components on one side of a page and so can be helpful when describing the system.

Figure 2.4, which is generalised here, shows at the top left, the development of the conceptual or activity model and at the bottom left, the formal design structures required for the data and the files. Top right are all the inter-departmental details, as few systems will be contained within one department and bottom right, everything on files. It also shows how past errors and lessons are fed into the structures and model, and that file requirements are used when setting the file structures and that the conceptual model ideas are fed back into defining individual file requirements.

In this way we have a complete cyclical analysis.

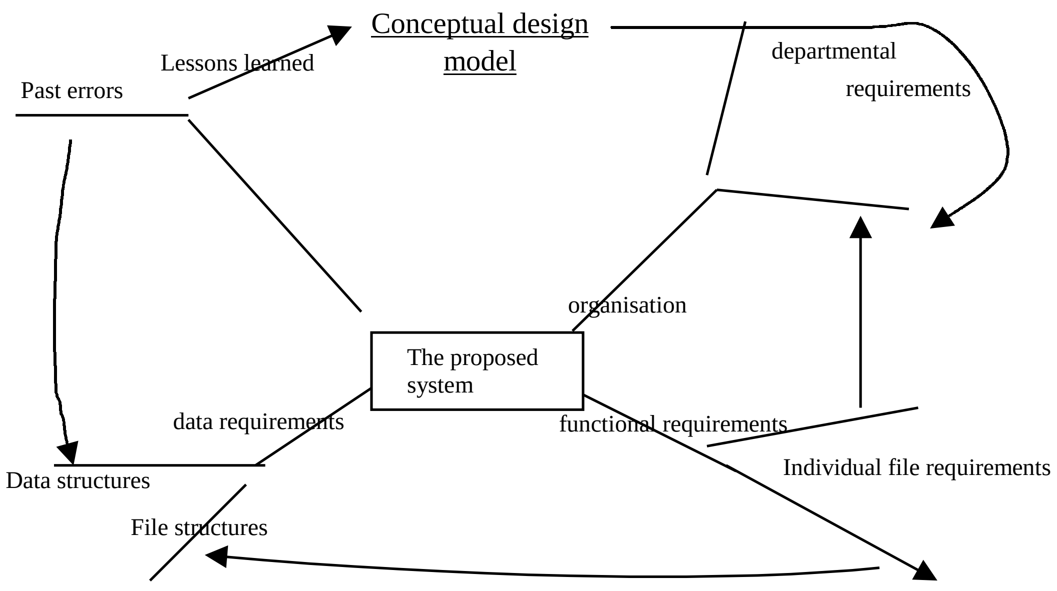


Figure 2.4: Spider Diagram

Study Unit 3

Systems Development and Project Management

(Continued over)

INTRODUCTION

We now start to concentrate on the development of computerised systems.

Computerised systems exist in one of two phases (development and operation) which together make up the system life-cycle. It is important that we regard system development as lasting throughout the life of the system. This is because a computerised system should always evolve and requires constant maintenance. The two phases of the life cycle are:

(a) Development

- Define client requirements.
- Undertake feasibility study.
- Investigation and analysis.
- Requirements definition.
- Design and build.
- Implementation.
- Client acceptance.

(b) Operational

- Post implementation review.
- Maintenance.

In this study unit, we will take a broad overall view of this life cycle, concentrating on the initial stages, and then consider the crucial management of the system project.

A. APPROACHES TO SYSTEM DEVELOPMENT

There are several approaches or methodologies that are used (although the trend nowadays is to avoid the traditional approach):

- the traditional approach
- the structured approach
- the database approach
- the rapid application development (RAD) approach.

We will look at them in more detail.

The Traditional Approach

This approach should not be used in modern development. It is essentially computerising an improvement on the existing system. Many of the defects are carried over to the new system and there is no allowance for user involvement.

Structured Approach

This is an improvement on the traditional system in that the aim is to build a new and better system and not just improve the old. It is, however, most applicable to large projects as it is essentially a team approach.

The technique is to use data flow diagrams and structure charts, with accompanying documentation, to define a logical model of **what** is required, leaving the **how** of implementation until later.

The basic approach is:

(a) Investigate the current system to see what it achieves.

(b) Together with any new requirements, produce a logical model of the full specifications.

A logical model is a fully worked out and formally defined model of the proposed system. It specifies what is required.

This specification is a very important detailed document as it forms the contract between the user and the computer department.

Traditional specifications were too narrative and were often couched in computer terminology. This made it difficult for users fully to understand the specification.

With the structured approach, using a largely diagrammatic form, the specifications are much more easily understood. Such specifications aim to be graphic, concise, easy to amend and capable of top-down development. Top-down development means that we start with the whole problem and identify within it the component parts and parts of the parts. This builds a hierarchy of detail down to the most basic level.

In order to identify the component parts, we use data flow diagrams, as in Figure 3.1, and from these we derive structure charts as in Figure 3.2. Then individual modules or components of the structure chart can be written up in design language called **pseudo code** or **structured English**, as in Figure 3.3.

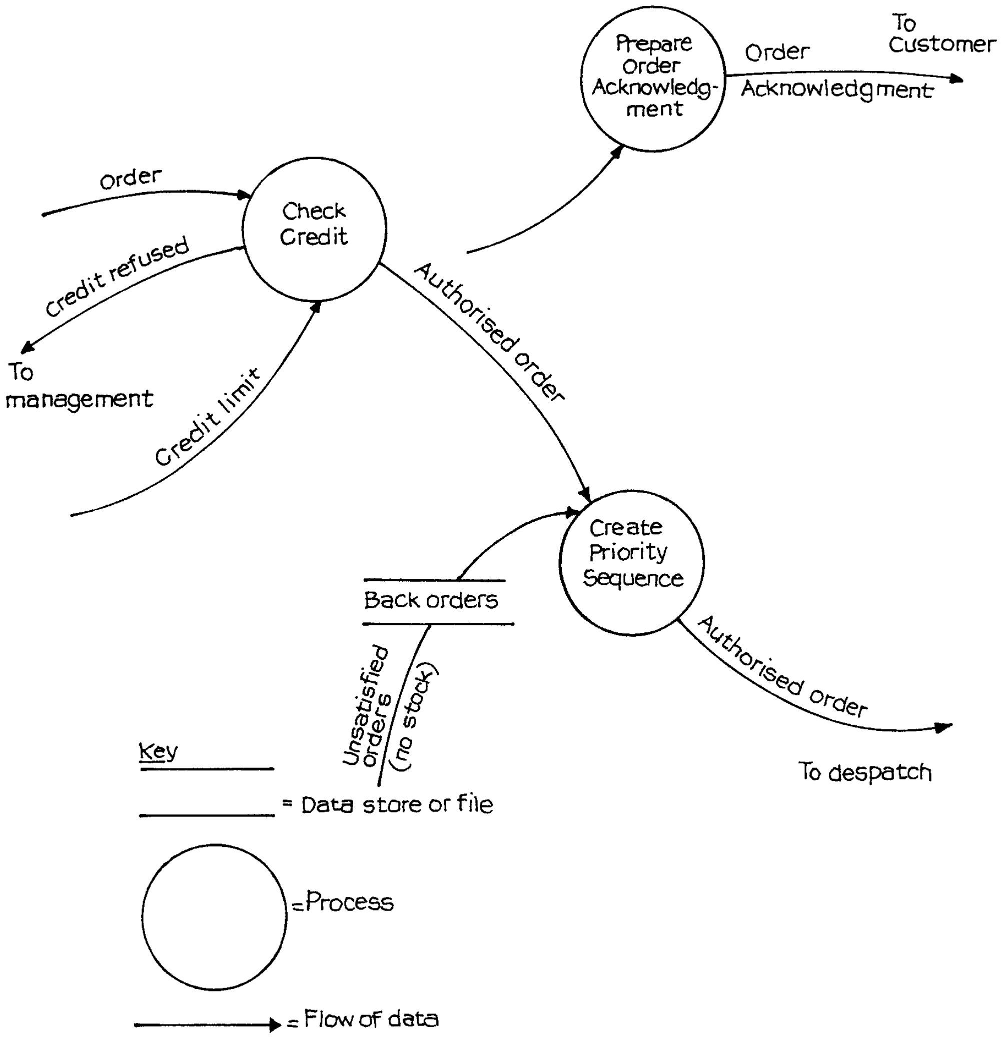


Figure 3.1: A Data Flow Diagram for an Order Entry Function

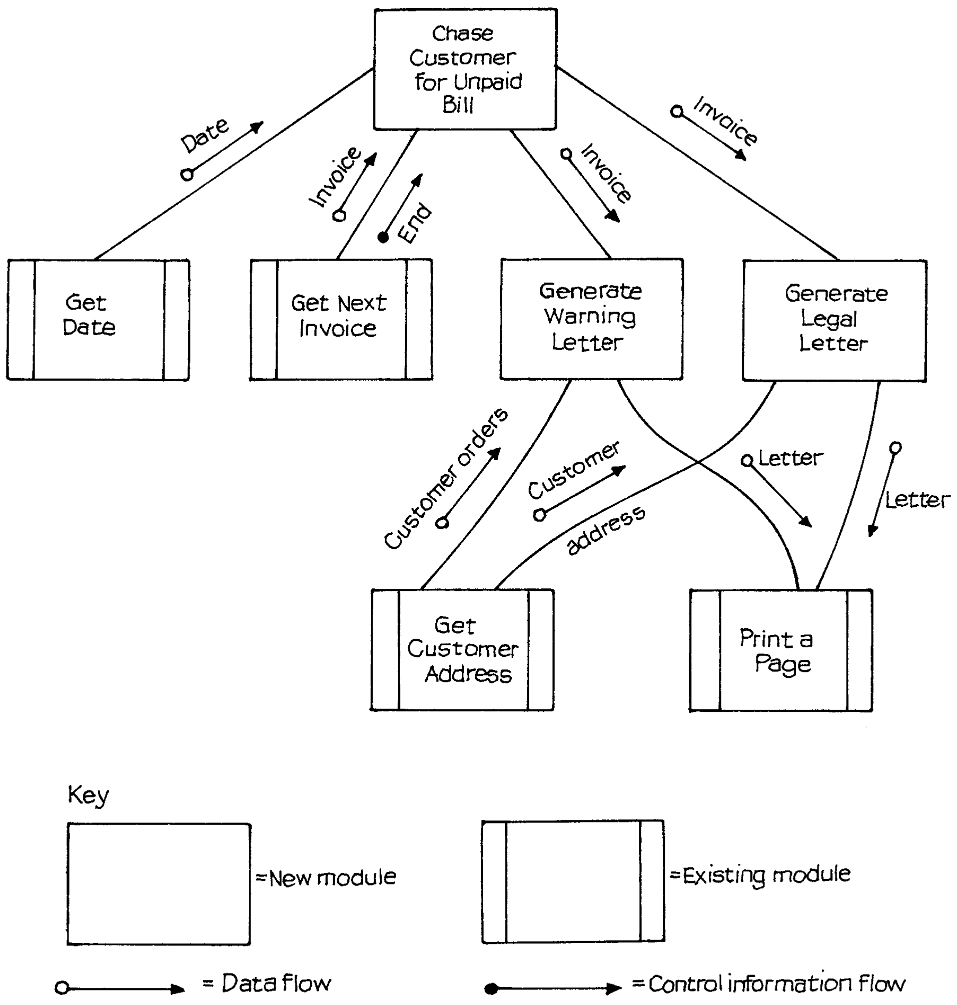


Figure 3.2: A Structure Chart for a System to Generate a Letter

(Note that this chart was not derived from Figure 3.1.)

The data flow diagram examines the sequences of processing involved. A broad-based diagram is produced first and then individual parts of this first diagram are examined to give a high level of detail.

The structure chart helps identify the individual modules that will require coding (programming). This chart deals with a further level of detail and there will be many such charts.

Yet another way to represent the logic is by pseudo code or structured English. Structured English avoids words such as "would" and "should" and uses a very limited subset.

```
IF    credit limit not exceeded
      THEN   OK order
      ELSE   IF pay experience good
                   THEN   OK order
                   ELSE   IF special clearance obtained
                                THEN   OK order
                                ELSE   Reject order
```

Figure 3.3: Pseudo Code

You will recognise a loose program format here. At this stage no specific language is considered and further design work will be needed depending upon the eventual language used.

Once the logical model is produced to give the full specifications, and the pseudo-code level of detail may or may not be included, the next stage is to derive a full design of the new system. The design is not produced until the specifications are accepted by the user. The design need not be totally readable by the user as it is a purely technical document; however, users must always be consulted and involved. The design will mostly consist of structure charts and pseudo code.

We will return to the study of structured methods later in the course.

Database Approach

This approach is based on the idea that the basic building block of any system is data. This contrasts with the structured approach which uses procedures as the basic building block.

The techniques of data models and normalisation lead to the implementation of database systems. The theory is that a flexible database can be constructed to service all the needs of the applications programs which require access. In this way, the database can be developed quite independently of programs.

We will return to a study of database, and normalisation, in the next module.

The steps are:

- Investigation of the objects (data) and relationships that exist in an organisation.
- Analysis of the data and the building of data models.
- Conversion of the model to fit a database management system and database implementation.

Data analysis is part of systems analysis; it is not free-standing.

Rapid Application Development (RAD) Approach

Various techniques are involved in this approach, including:

- Joint development or active participation of the user client.
- Integrated software tools that support the development.

Participation

All modern development approaches should require the full and active participation of the user client.

All systems developers should take into account the client's requirements. Techniques such as decision by consensus are advocated, which is very different from the approach where analysts and designers dictate.

This approach is a design aid which assists implementation. It is not a problem-solving methodology.

At this point it will prove constructive to return briefly to general systems theory.

B. "HARD" AND "SOFT" SYSTEMS

Traditional sciences (physics, chemistry, biology) led to the discovery of common laws (e.g. exponential decay) but are inadequate in explaining the behaviour of wholes. For example, can biology explain the whole human being as a collection of parts (brain, heart, etc.), or is there something more, called "vitalism"?

Definitions and Examples

The systems theorists have attempted to define systems. We previously looked at deterministic/probabilistic, quantitative/qualitative etc. examples; one important definition divides them into "hard" and "soft" systems.

- **"Hard" systems** are clearly defined systems with a single (or a low number of) aims/purposes, e.g. an engineer in the real world.
- **"Soft" systems** exhibit changing or conflicting aims, very often including humans, e.g. the political system.

As you will remember, a system:

(a) Is a complex grouping of people and machines.

(b) May be broken down into sub-systems which interact with each other.

(c) Is part of a hierarchy of systems.

(d) Must have an objective.

(e) Is capable of achieving its objective.

Some general examples of systems are the shown in the following table.

Example	**Characteristic**	**Class**
Bridge/crystal	Static	Hard
Clock	Predetermined motion	Hard
Thermostat	Closed loop feedback	Hard
A cell	Self-maintaining, adaptive	Hard
Plant	An organised whole	Hard
Any animal	Brain + learning ability	Hard
Human behaviour	Knowledge/language/responsive	Soft
Financial department	Predetermined	Hard
Sales department	Dynamic	Soft

Application to Systems Development

In information technology, we generally come across "hard" systems. However, we should be able to identify "soft" systems and not blindly apply our "hard" systems methodologies and expect them to produce results.

Let us differentiate between "hard" (well-structured) and "soft" (ill-structured) problem situations:

	SOFT SYSTEMS	**HARD SYSTEMS**
	Ill-structured	**Well-structured**
Objectives:	Vague, inconsistent, unrealistic	Agreed and realistic
Organisation:	Inhibiting, incompatible, inadequate	Consistent and effective
Authority:	Non-established, non-effective	Established and effective
Environment:	Unsustainable	Defined and sustainable
Control:	Unknown correlation between effort and results	Measures of performance can be set and applied
Communication:	Uncertain, unreliable	Reliable and effective
Information processing:	Insufficient, irrelevant	Necessary and sufficient
Attitudes:	Obstructive	Co-operative and flexible
Problem areas:	Vague and poorly appreciated	Agreed and appreciated
Solutions:	Intuitive, poorly supported	Known, consistent, relevant, useful

In our approach to systems development it is tempting to regard the system as being "hard", well structured and precisely defined. Indeed, most of our available techniques are built upon this assumption. In practice, the system will involve people and so, by definition, is inevitably "soft", less well structured, generally vague and difficult to define precisely. A cautionary point is now needed when considering systems development. All systems will have both "hard" and "soft" characteristics and the latter must be incorporated. The participation approach to development is now very applicable as it brings the user into the development process. Modern approaches consider it essential that users are involved at all stages of development and tools have been introduced to facilitate this. We shall return to this point in Study Unit 4 of the next module. Meanwhile, we can consider the principles of this "soft" systems approach.

"Soft" Systems Approach

(a) Define the rich picture.

 This is building up an in-depth portrait of the situation being considered. The source for this is the people actually involved or who are to be affected. The rich picture will be presented diagrammatically.

(b) Derive the root definition.

 Essentially derive a definition of what the system is to do, why this is so and what is actually wanted. Consider the whole environment and all the people involved.

(c) Build conceptual models.

 Compile on paper or a screen a model of the root definition.

(d) Compare the conceptual model with the original rich picture so as to highlight any problems between what is happening (the rich picture) and what is wanted (the root definition).

(e) Prepare a schedule for implementing (c) and dealing with problems identified at (d).

(f) Actually implement the changes. This will involve the people affected and obtaining their co-operation.

In summary, the "soft" systems approach involves consulting at every stage with those affected and obtaining their co-operation.

C. CHARACTERISTICS OF A "GOOD" SYSTEMS ANALYSIS METHODOLOGY

Client's List

Let us consider a list of qualities that clients have specified as desirable in systems development methodologies, and then attempt to evaluate this in "systems approach" terms.

(a) Decomposition of the System

 The systems we are concerned with are **complex**. Therefore **some form of decomposition** appears to be fundamental to facilitate analysis.

 With this approach, the developer is concerned with just **one level of detail at a time**; all consideration of lower levels of detail is postponed. Breadth of consideration is more important than depth at any level.

(b) Limited Number of Elements to be Considered

It is suggested by psychologists that the human brain cannot efficiently handle more than seven variables (ideas, thoughts) at one time; therefore coupled with the decomposition approach is the idea of limiting the number of elements being considered at one time to about seven.

This restriction focuses the developer's efforts on aggregating similar things and/or separate things with differences, which increases understanding. By looking at seven elements, he or she is forced to consider the relationships between these elements. This, as compared with looking at only one element at a time, also increases understanding.

(c) Precise Definition of Boundaries

This concept requires that the analyst identifies all processes, data types, and relationships between them, within the specific group of elements being analysed.

All that belong must be included; all that do not belong must be excluded.

(d) Analyse Activities and Data

Both the data and the activities should be analysed. The data is not an afterthought – it is the primary consideration.

(e) Graphical Notation

Graphical language can be both simple and powerful in conveying information. Because of its pictorial nature, it can show boundaries, activities, data types, relationships and the number of elements within the boundaries.

(f) Simplicity

An important factor of the methodology should be that it includes an effective communication bridge between the developer and clients. A graphical notation, with simple, uncluttered diagrams can provide this.

Criticism of User's List by the Systems Analyst

Such general lists are of value when assessing various development methodologies, but they must be treated with great caution. The systems analyst would level the following criticisms at the client's list:

- Decomposition and the study of a small number of elements in detail is a scientific reductionist approach, rather than the holistic approach recommended by systems thinkers.
- Definition of boundaries seems a good idea in principle but boundary definition is rarely a simple activity and we must avoid excluding potentially important items from a system in our desire to draw a boundary.
- Many current methodologies make great claims to replace all text in the system specification by a "simple graphical notation", but there are a number of important elements that cannot be shown by graphical notation, e.g. the human and behaviourist problems in the system.
- Discussion of notation, ease of modification and understanding are really about communication, not analysis, and obscure the search for analysis methodologies.

Qualities of a Structured Specification

These can be summarised as follows:

(a) Graphic and Concise

The structured specification contains pictures of the system, rather than large amounts of textual description. Users are more likely to look at pictures than read a lot of text. Also, designers are more likely to read it and understand it rather than ignore it and start again.

(b) Top-Down Partitioned

The system is broken into pieces that are as independent as possible, the **user can then review the system in small, digestible segments**.

The partitioning is done in a top-down fashion to show broad, general views of the system, followed by narrow, detailed views of a "local" area.

(c) Maintainable

The specification document is **highly maintainable**. Because of the graphic, partitioned, structured nature of the specification, modifications can be made quickly and easily (structured programming).

(d) Non-Redundant

In the structure specification, a **piece of information** is recorded **once and once only**. This non-redundant aspect ensures consistency and easy updating.

The user does not get bogged down in repetitious detail.

The designer can readily identify system components and then translate these into areas to be designed.

(e) Easy to Change

As diagrams go through many revisions during the analysis phase, it is important that they are not difficult to redraw. If they are simple combinations of lines and boxes with very few words, they will meet this objective.

(f) Numbering System

Closely related to this characteristic of ease of change is the system of numbering the different diagrams – as well as the different generations of each diagram.

(g) Defined Procedure of Use

The methodology should have a debugged, "prescribed" method of use – including suggestions on how to decompose systems.

The various types of user (analysts, designers, clients, programmers, managers, etc.) should all have the same understanding of the use, to the level of detail that their job requires.

(h) Leads into Design Method

There should be a natural way to move from the structure for displaying requirements to the design structure for the new system.

D. SELECTING AN APPROPRIATE METHODOLOGY

A valuable way of choosing a development methodology might be:

(a) Examine what the methodology claims to do, and where it suggests that it is useful.

(b) Assess whether the situation is appropriate for use of this methodology.

Possible situations are:

- No current system exists.
- Current system exists but design only needs tuning.
- System exists but is very "soft" and ill-defined.

The question of which approach to use is very difficult. However, we can classify the system as "hard" or "soft" and this then leads us into appropriate approaches, as illustrated by Figure 3.4.

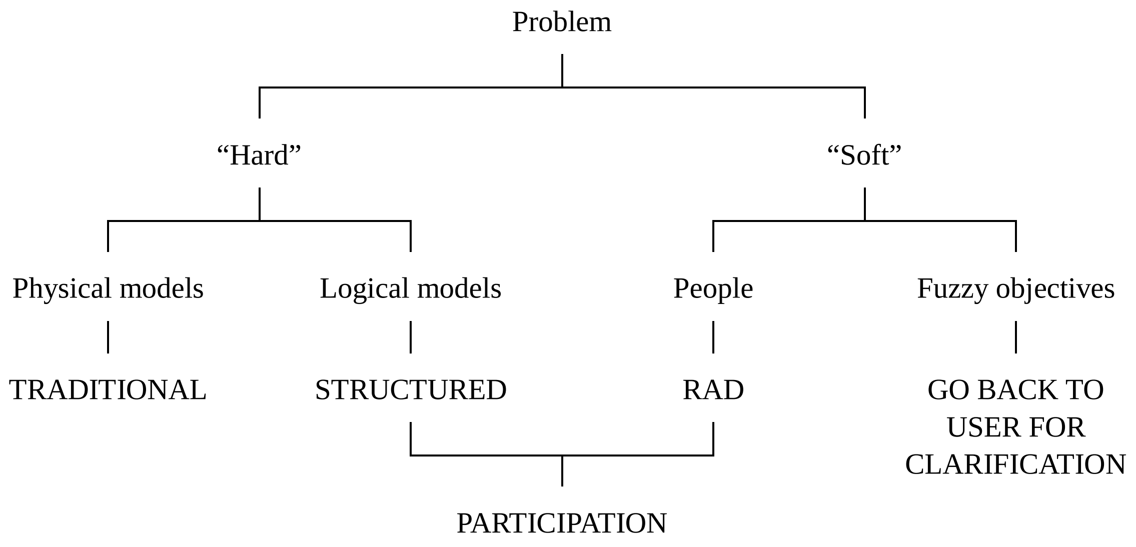


Figure 3.4: Decision tree for assessing appropriate solutions

E. MANAGEMENT OF CHANGE

A good deal of research and survey results confirm that, in practice, human problems are a major cause of failure in the effective development and management of computer systems.

- A technically feasible application may still collapse in operation. The reason could be ignorance, antagonism or sheer apathy on the part of user staff.
- Line managers may look upon change as a threat or merely as unhelpful, and withdraw their co-operation with the system. Managers have been known to "beat the system". For example, with a computer-based stores and spares system, they requisition for unusual specifications instead of making existing ones suffice, and build up the level of their stocks artificially through increasing the number of "lost" and "damaged" parts reported.
- Other managers may simply ignore the new system and continue using the former system – they may regard the new system as a waste of paper.

- We may also discover human problems which relate to the computer specialists. This often occurs, as the specialists tend to follow their own technical objectives and do not consider the overall corporate interest.

Sometimes these problems are, wrongly, treated as relatively temporary and unimportant. But most writers on this subject consider that it is very important to realise the complexity of such problems and to develop a planned strategy for the management of change, supported by continuous training.

Changes Resulting from Computer Systems

Some typical changes arising from the introduction of a computer system will be as follows.

(a) Changes in Relationships

Traditional assumptions are invariably challenged when a new system is envisaged and, when weaknesses are revealed, management will often feel resentful and less secure. Also managers involved may find that they have a heavier workload as a result of the change to the computer application, perhaps forcing them to neglect other matters.

On the other hand, the computer may sometimes be looked upon as breaking down traditional departmental and divisional barriers, the formation of project teams obliging personnel to work together.

(b) Changes in Controls

The system may be regarded as too rigid to be able to deal in practice with the many variable elements involved. But, it may also be that the organisation finds, for example, that the prompt and thorough market analysis made available by the computerised system leads to an increase in its market share.

On the other hand, management could be overwhelmed by the new data outputs, so that managers still secretly rely on the former, trusted system.

(c) Changes Relating to Organisation

Greater centralisation of decision-making is possible through the existence of the computer system. Maintenance of, say, customer accounts may be removed from the branches of an enterprise and undertaken in regional centres.

(d) Changes in Planning Activity

Planning may become a day-to-day activity whereas formerly it was an annual budget preparation exercise, so that the budget can now be recast to give the effect of pricing policies, actual performance and so on. Managers previously using the pragmatic planning approach may tend to feel that the system is in charge of them, as planning activity programs, scheduling and procedures become more exact.

(e) Inflexibility

One main alteration brought about is that the computer system is very often guilty of giving rise to rigid inflexibility. People cannot now behave on a “give and take” basis. Rules are now operated, quite irrespective of the circumstances. Much higher standards of accuracy and discipline generally bring about resentment – and, it must be added, much pleasure from staff in uncovering computer errors.

However, a good deal does depend also upon past relationships. Where these have been first-class between management services staff and the other personnel, then there is more than a good chance that this will give a flying start to management services work on the computer system.

Planning for Change

There are various ways of expressing this, but the following is a typical series of stages towards planning the change:

- commitment and approval of senior management;
- comprehension of the fundamental behavioural issues;
- objective clarification, manpower resources;
- creation of effective communications;
- growth and renewal planning;
- constructive use of a "change agent".

Now let us consider these separately.

(a) Senior Management

The commitment – as well as the approval – of senior management is important. Quite apart from the formal approval which has to be achieved in any case for the change involving the computer, management attitude and philosophy can make or mar the results. It must be realised that simply instructing people as to the pattern of behaviour expected from them will not produce the desired results.

During the feasibility study, management will have manifested its interest and will now already be aware of the considerable benefits planned for the organisation as a whole. Comparable computer installations will have been visited and discussions held with senior persons on other management implications of the computer.

Senior management will require the same stringency of planning, control and budgeting for the computer installation as for any other commercial activity. What is also important is that senior management should provide both the organisational and manpower resources for the project to be a success.

(b) Behavioural Issues

A deliberate learning strategy should be adopted to make sure that computer structure, relationships and values, plus technical and economic feasibility, are relevant. Remember the nature of motivation and the need for:

- self-actualisation
- esteem
- love and "belonging"
- safety
- psychological satisfaction.

Computer department labour turnover is often discovered to be high. Research points to the fact that this is because the needs for esteem and for self-actualisation are often not satisfied.

Another consideration is user satisfaction. Users often think the computer will eliminate their recognition and status if not their actual work. Whilst this is sometimes counteracted by the sense of achievement in using the computer, this in turn can pass after the initial impact of the computer.

(c) Human Resources

The sort of questions to be asked here are:

- How many computing specialists, and with what experience and skills, will be required?
- What internal promotions may be made?
- Is there sufficient awareness of achievements, expectations, knowledge and skills of our own personnel to plan succession and careers development?
- What sort of skills and knowledge will non-specialist management need in relation to the computer if they are to co-operate and contribute within the planned applications?
- What is to be done concerning training?

(d) Effective Communications

This, in itself, is a big question. It, too, requires a complete analysis and a clearly expressed plan. The question is how to ensure that everyone knows what he or she must do concerning the installation, how it is going to affect them, how it is now progressing, and so forth. Not only are we concerned about vertical communications (that higher levels actually listen to or consider the contributions of lower levels) but also about horizontal communications (especially between computer people and line management).

There are many possible approaches, ranging from visits of personnel to the computer installation to regular progress bulletins and items on management agendas.

(e) Growth and Renewal Planning

This refers to the fact that using the computer intelligently will mean a continuing process of increasing awareness and a renewal of purpose. The original objectives, assumptions and communication systems – sound enough for the early applications – may well have to be changed because they have become outdated and inappropriate.

(f) Use of a Change Agent

This may be an external element – a consultant or other person who acts as a catalyst and who provokes thought, reassessment of ideas, and criticism.

F. PROJECT MANAGEMENT

An alternative title to "project management" is "project control". By this change we can see that the whole purpose of the management of a project is to keep control of the project. As such, all the normal principles of any control system will apply:

- Pre-set a standard. In our case of a project, this will not only be a quality standard, but it will involve a schedule and a cost. It must also have the requirements specification as part of the standard.
- Arrange a flow of feedback. In an earlier study unit we discussed milestones or checkpoints. At such points, development staff will submit reports of results obtained, time taken, costs involved and objectives achieved. Reports will also be submitted at specified time intervals.
- The project manager makes a study of the feedback and makes a comparison with the pre-set standards and with the project plan. Any variations at all are taken into account.

- A response or corrective action will be made depending upon the comparison.

Establishment of Standard Procedures

By controlling the preparation and use of documentation, an organisation is able to achieve the high level of quality and reliability needed by users of its systems. It is thus important to establish and enforce a set of standard procedures for the development and operation of systems.

Standards are uniform practices which govern the methods by which systems are developed and operated and, further, they provide a basis of assessment for both system and personnel performance in terms of quality and quantity. They are decided upon jointly by the manager and user department staff, and cover areas like:

- computer operations such as log procedures, staff responsibilities, operating procedures;
- ancillary operations such as data preparation, control and job specification;
- system development topics such as guidelines, file usage, test data, audit trails, documentation and standards.

Project control is a very complex task. There are no routine activities involved. All problems tend to be unique. It is a "one-off" situation as no two projects are the same. To add to the complexity, in many instances, the project staff are working together for the first time outside the established organisational structure. Line management relationships are upset and yet the project manager is unlikely to have the authority to hire and fire staff. Strong leadership is called for.

Defining the Project

A major problem is the control of application development time and cost. For successful control of projects there has to be a project plan and structure defined clearly from the very start. If the project is a large one, then it ought to be divided and subdivided into modules. These could then be arranged, by analysis, to establish points for checks ("milestones"). Each milestone should have a visible output (e.g. a document) which may be used to determine the quality of output produced at that given point within the development cycle.

In project control the manager is attempting to find answers to questions such as:

(a) Cost

- What will be the total estimated cost of the project?
- To what extent is actual project cost to date higher or lower than estimates?

(b) Timing

- Is the project correlating with schedule?
- At what point is each activity scheduled to begin?
- How long (estimated) is the project to take?
- To what extent is the project now on time?

(c) Human Resources

- What number of staff is needed?
- In what way should assignments be allocated?
- What skills are required?
- What other resources (e.g. machine time) are required?

Project Manager's Responsibilities

- Co-ordination, administration and supervision of the team.
- Co-ordination of all elements in the development of the project activity schedule, critical path network, etc.
- Securing of approvals necessary.
- Distribution of the approved schedule and reports to all persons affected.
- Keeping of necessary records and a history of the project.
- Attainment of system specifications.

G. PROBLEMS DURING DEVELOPMENT

The development of a computer system is very often a long and complicated exercise, involving many staff and other resources in a highly organised manner. The costs of such an exercise are high.

With such a complex project, problems are inevitable. The skill of the project team is required to identify project problems and find solutions before the problems become major and jeopardise the total project.

Causes of Project Problems

When any project "goes wrong", it is all too easy to blame the staff who are working on it. It is exactly the same with computer projects.

When a project "goes wrong", the symptoms are:

(a) late running;

(b) cost escalation;

(c) general loss of control

Very often the causes are:

- poor project identification
- insufficient investigation
- incorrect design
- inadequate planning
- poor project management
- poor programming.

It is unfortunate that, as a result of poor project management, most problems are identified towards the end of a project, in system testing. The problems then result in program amendments and a great deal of programmer time is expended. As a result, substantial effort is made to improve programmer productivity, getting nowhere near the root of the problem. It is likely that the problem began much earlier in the project cycle.

Let us take a closer look at the causes and examine ways in which we might overcome them.

Poor Project Identification

For example, suppose a despatch department is taking too long to pack goods for delivery. The development team could prepare a system of great efficiency. However, if they have overlooked the real, original problem of products being placed on the wrong shelf, they will only bring themselves into disrepute however well the project progresses from that point. If a system perpetuates a manual problem then computerisation will only speed up the mess.

It is necessary, therefore, to spend time identifying the real problem. This work needs to be done by a senior and experienced analyst.

Insufficient Investigation

Having identified the real problem, we must ensure that we understand it. A thorough understanding of the problem can only be obtained by detailed investigation.

This investigation is often limited to a series of interviews with staff, proposed by the potential user of the system. The analyst conducting the investigation, therefore, obtains an understanding of the system as seen by the person interviewed. This could be a very biased view; what person is going to admit his or her laziness at completing the documentation correctly? The analyst must, therefore, use the complete range of investigation techniques. The essential point to remember is, that if a full range of techniques is not considered, then insufficient investigation will lead to a poor understanding of the problem.

Incorrect Design

This is where the technical expert takes over. The business analyst has completed his or her task.

The objective of the design work is simplicity and flexibility. Simplicity in that "a sledge-hammer must not be used to crack a nut" and flexibility to ensure that any future extensions to the proposed system do not create problem projects.

In practice, at least two, and ideally three, designs should be produced. The designs should then be validated against the objectives set in the initiation stage and, with the user's involvement, a decision reached as to the design with which to proceed. There is no correct design; any solution which meets the objectives and can be justified is acceptable.

Inadequate Planning

A computer project requires a large variety of resources: business analysts, system designers, programmers, trainers, computer power, transport, data preparation, users, etc. The project will require many more resources in the middle and slightly more at the end than at the beginning. While we could strive for continuity, it is likely that team members will change, especially in a long-running project.

This complex requirement for resource types demands an accurate plan of what resource is needed, when it is needed and for how long. Without such a plan, which must be updated, reviewed and amended as the project progresses, there is little chance of having the correct resource at the right time for the required length of time. The project time-scales will, therefore, increase, leading to a general loss of control. Careful planning is perhaps the most important activity of any project, and should be undertaken by the most experienced staff.

Poor Project Management

It is likely that the project manager will be the project planner. A poor project manager may, therefore, (but not necessarily) be a poor planner. An experienced computer professional could be an excellent, diligent planner but a very poor manager.

If we are to have high quality work, we cannot have the members of the project team inventing rules independently. Once a plan has been established we must adhere to it. We must also adhere to the standards laid down for a particular task, whether that be design, programming, documentation, or any other.

If we are going to have ground rules – which we must have to ensure that information is disseminated in a coherent way – then they must be enforced.

A strong leader or project manager is, therefore, necessary.

Poor Programming

This problem area is probably the least troublesome. If a program is not completed properly in terms of testing or documentation, then this should be quickly identified, and provided that good, experienced programmers are available, can be quickly corrected. The most likely cause of problems at this stage is either poor initial planning or extended project times before the commencement of programming, resulting in hurried, insufficient testing.

In fact, many good programmers, by asking questions and probing deeper into systems, identify problems from earlier stages within the project.

H. RULES OF PROJECT MANAGEMENT

The foregoing, then, are some of the reasons why projects "go wrong". Let us complete the picture by providing a summary of ground rules to guard against such situations arising.

- A computer project needs to be set up correctly at the start – ensure proper planning takes place.
- Get a slow start to ensure you solve the correct problem.
- Structure the project team so that the lines of communication are clear and the project manager can obtain a second opinion.
- Establish clear ground rules and enforce them.
- Pick the right size of project – the smaller the project, the better the chance of success.
- Choose the right project manager.

The above list is, of course, not exclusive. It is more than likely that your project will run into trouble at some stage and it is the skill of the project manager and his or her team which prevents problems becoming major hurdles by anticipating them and dealing with them promptly.

The list below sets out in a jovial manner the "Laws of Project Management":

(a) No major project is ever installed on time, within budgets, with the same staff that started it; yours will not be the first.

(b) Projects progress quickly until they become 90 per cent complete. They then remain at 90 per cent complete forever.

(c) One advantage of fuzzy project objectives is that they let you avoid the embarrassment of estimating the corresponding costs.

(d) When things are going well, something will go wrong. When things just cannot get any worse, they will. When things appear to be going better, you have overlooked something.

(e) If project content is allowed to change freely, the rate of change will exceed the rate of progress.

(f) No system is ever completely debugged: attempts to debug a system inevitably introduce new bugs that are even harder to find.

(g) A carelessly planned project will take three times longer to complete than expected; a carefully planned project will take only twice as long.

(h) Project teams detest progress reporting because it vividly manifests their lack of progress.

I. SPECIFIC DEVELOPMENT CONTROLS

These are four very specific areas which are crucial to project control. Adherence to these points will always make project management more effective.

Documentation Control

Documentation covers all aspects of systems development, from the initial problem outlined in layman's terms to the detailed program listings and finally to the comprehensive user manuals.

In a later study unit we will look at documentation in detail. From the project manager's point of view, detailed paperwork will not only aid project monitoring, it will help development staff. For example, full design instructions and explanations will ensure program coding consistency and faster, more efficient coding at that. Project management should therefore rigorously insist on development staff keeping up-to-date documentation and filing of the documentation, at all times.

Later in the project life-cycle, efficient maintenance is totally dependent upon the earlier documentation. This is because different personnel are involved and they will have no knowledge of the system design.

File Restriction

Development staff should never be allowed access to live data files. Inevitably such files would become corrupted. In addition, restrictions on the use of and access to sensitive files should equally apply to the development staff.

Work should only be allowed to be performed on data files created from live files, the creation of such files being carried out by non-development staff. Alternatively, if this is not possible, dummy files should be used.

Audit Trails

During development, paths should be created which will trace the operations being performed on source data. Source documents with consecutive or unique identifiers should be listed on entry and again in final tabulations. Programs should be capable of being "frozen" in mid-operation to allow enquiries of the current state of data. File dumping to printer and display units should be catered for.

Such features should not, of course, be confused with the audit trails built in by the system auditors. We will discuss these in a later study unit.

Amendment Control

Any amendments made during development and maintenance should be logged in the system documentation. For example, if a particular file is deleted from use, then all references throughout the document should also be deleted.

As amendments can significantly alter the way a system works and the controls upon the system, all amendments should require authorisation, however small.

The authorisation form can then be part of the documentation.

J. CONTROL TECHNIQUES

First of all, it is important to be aware that, although planning ahead is the most important project control technique, it is not the only one. Figure 3.5 shows the relationship between planning and the other processes.

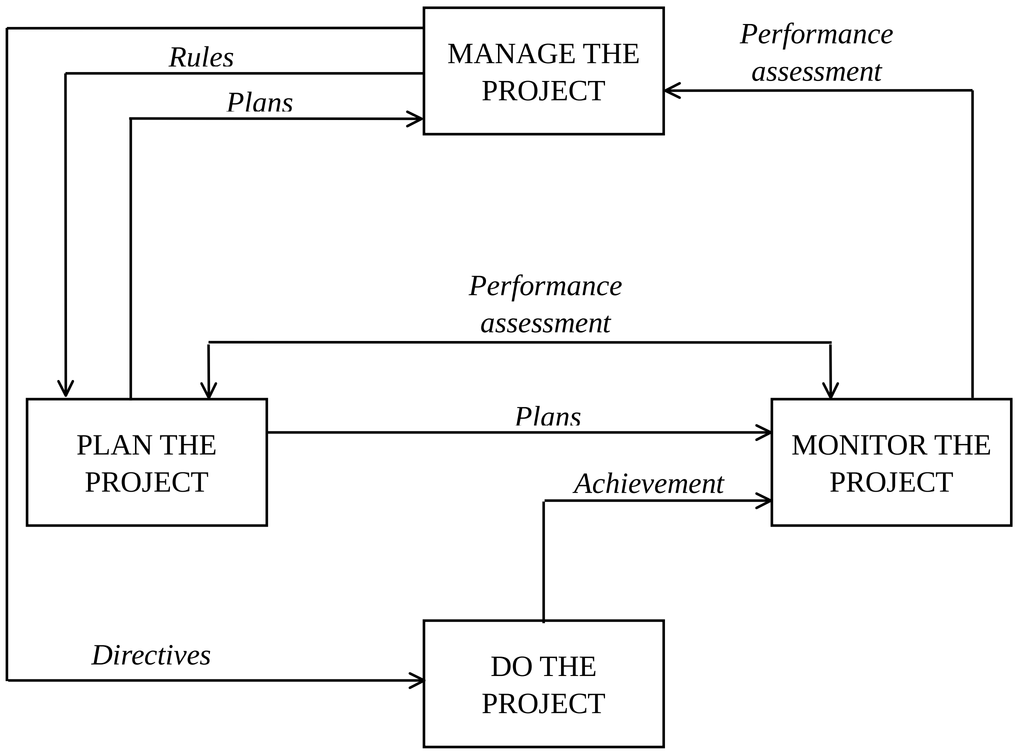


Figure 3.5: Processes in a Project Life

Activity Decomposition

When monitoring the project, we will need a clear idea of all the tasks within the project. It is an important project control technique to decompose the broad activities of investigation, analysis, design, etc., into task lists.

Task lists should be as detailed as possible. Tasks to be performed in the near future should be decomposed into smaller units than those which are planned for later completion. For example, programming tasks should be shown by individual program, but it will not be possible to do this until system design has been completed.

Examples of task lists:

(a) Investigation Activity

- Plan interviewing.
- Develop interview guides.
- Make appointments.
- Conduct interviews, gather volumes and samples.
- Summarise interview notes.
- Document investigation findings.

(b) Systems Analysis Activity

- Review all material gathered.
- Draw data-flow diagrams.
- Analyse the data and construct a data dictionary.
- Identify the scope of the new system.
- Define the essential procedures and data.
- Present results to user:
  - (i) Arrange time, place, attendance list.
  - (ii) Plan presentation.
  - (iii) Draw diagrams/slides.
  - (iv) Rehearse.
  - (v) Conduct presentation.
- Finalise documentation.

Standard task lists should be constructed for all the activities in the development cycle. Each task list can then be used in conjunction with the project detail to arrive at a specific task list for the project.

Milestones

A milestone occurs when a task or a major series of tasks has been completed. The major project milestones where major decisions are required, e.g. feasibility analysis, analysis/design, design/testing, testing/changeover, etc., have already been identified.

Milestones or checkpoints are used by the project manager, the computing management and the user management to enable them to conduct formal reviews of progress.

A suggestion for the major checkpoints in a development project is shown in the following table.

Table 3.1: Major Milestones

Completion of Task	**Project Manager**	**Computer Management**	**User Management**
Objectives/Terms of reference		x	x
Feasibility report	x	x	x
Project planning	x	x	x
Investigation			
Plan interviewing	x	x	
Develop interview guides	x		
Summarise interview notes	x		
Document investigation	x		
System analysis			
Draw data-flow diagrams	x		
Data dictionary	x		
Plan presentation	x		x
Rehearse	x	x	
Finalise presentation	x	x	
Conduct presentation	x	x	x
System design			
Outline design	x		
System specification	x	x	x
Program specification	x		
System test plan	x	x	
Conversion plan	x	x	x
Implementation			
Individual programs	x		
Completion of programming	x	x	
System test	x	x	
File conversion	x	x	x
Parallel operation	x	x	x
Live start-up	x	x	x

Software Engineering

The disciplines and procedures used to develop software for an information system are called software engineering methods and a set of procedures used from the feasibility stage to the operation of the system is called a software development methodology. A number of techniques and computer tools have been developed.

Network Analysis

Network analysis is a particular technique for use in project planning and control. It first attracted attention through its use in the development of Polaris rockets in the USA. Some network analysis methods select the more critical or strategic elements of a plan and concentrate mainly upon them – this is critical path analysis. Another term is PERT – Project Evaluation and Review Technique.

The first operation of the technique is to prepare a list of the activities with the time each will take and, for each activity, to define what other activities must be completed before that particular activity can be started. The resources needed for each activity will also be noted – if, for example, the development team has only one expert database designer, this could be a critical resource.

Just as there are a number of names for the technique, so also there are a number of different ways of drawing the network which shows the relationship between the different activities. The important items to be identified from the network are:

- The critical path which indicates those activities which, if not completed within their defined duration, will cause the overall completion to be delayed. They are, therefore, the activities which need closest control.
- The "spare" or float – for activities not on the critical path there is a certain amount of latitude permissible in which they actually start and how long they actually take, provided they do finish by the latest time.

The existence of float in the network gives the project managers some freedom of action in the planning of the network, permitting, for example, some reduction in resource demands.

The converse of this argument, of course, is that activities on the critical path merit adequate or even additional resources.

A further consideration is the analysis of the resources required for each activity. Each activity will need so much manpower and possibly the use of equipment. The allocation of appropriate resources to each activity permits a "profile" of resource demand to be built up. This will reveal peak periods. The network can then be reassessed in an attempt to reduce or eliminate these peaks.

Use of Computers for Network Analysis

Many small networks can be handled manually but, for large networks, the computer produces many benefits. In practice, there may be thousands of activities represented in a network and many of these will depend on more than one activity being completed. There are many computer packages available and these permit fuller use of the network in appraising different plans, minimising resources, progressing work, etc. – they are examples of a software engineering tool. Modern PC-machines are increasingly being used in this and related areas as a cost-effective tool for use by management.

Documentation

(a) Project Plan Form

Once authorisation to implement has been received, the manager assigns a schedule, allocation of staff, miscellaneous costs and operations cost to every activity, through use of a project plan form. Such a form is shown in Figure 3.6.

PROJECT PLAN

Project title ....

Project manager .... Date ....

Type of system ....

ACTIVITY		TIMING		STAFF ALLOCATION		COSTING		
Identification	Description	Start	Finish	Grade	Hours	Code	Description	Cost
001-003	Listing	1.9	10.9	06	35	65	Transport	£12.50
....	....	...	....	..	..	..		

Figure 3.6: Project Planning Form

(b) Staff Production Report

This gives the projects worked on, the kind of activity and staff hours (see Figure 3.7).

PRODUCTION BY MEMBER OF STAFF				
Staff No.	**Employee**	**Name of project worked upon**	**Code**	**Hours**
008	John Idler	Sales invoicing	11	12
		Stock control	15	9
		Stock control	11	13
		Total accountable hours:		34
		Total non-accountable hours:		3
025	Henry Workman	Payroll system	09	45
		Parts system	03	26
		Total accountable hours:		71
		Total non-accountable hours:		3

Figure 3.7: Staff production report

(c) Project Activity Report

This shows actual cost expenditures and timings by project activity (see Figure 3.8).

PROJECT ACTIVITY REPORT

Project name: Project manager:

Date:

Activity	Scheduled Completion	MAN-HOURS			MISCELLANEOUS COSTS		
		Est. Hours	Actual Hours	Variance	Est. Cost £	Act. Cost to date £	Variance to date £

Figure 3.8: Project activity report

Project Manager's Role

The project manager must accept complete responsibility for the achievement of the objectives of the project and he or she does this through leadership and through adopting the initiative in project management:

(a) Planning of Main Activities

This means identifying project activities and functional responsibilities.

(b) Scheduling of Activity

Here, there is identification of detailed activities and their interrelationships in timings, resources and sequence.

(c) Continuous Control of Project

Under this heading there is allocation and co-ordination of:

- assignments
- resources
- current project status maintenance
- documentation
- project problems in timing.

(d) Progress Reporting

There has to be preparation, and distribution, of milestone progress reports, including those concerning progress to completion date.

(e) Summary Recommendations and Reports

The preparation and appropriate distribution of regular summary reports has to be undertaken concerning:

- performance of the project
- intermediate objectives achievement
- interim recommendations.

Study Unit 4

System Specification

Contents *Page*

(Continued over)

INTRODUCTION

You should now have a general picture of the type of methodologies used for system development and of the overall management of the process. There is a wide choice, though, in practice, each organisation will have a favoured or preferred approach which has evolved through the experience of the systems manager and his or her predecessors, the nature of the organisation and the environment in which it operates. It is also important to be aware of the whole system into the sub-system under development will fit, even if the sub-system is extremely large itself. Remember, few systems exist in isolation.

Whatever methodology or approach is used, the system must undergo some definition process. We touched on this earlier when we were discussing computing systems, the role of steering committees and so on. This is an important topic, so we will return to the study of the first project stages and examine the specification of a system through the initial investigation and feasibility study.

First of all, though, we briefly review the whole development process.

A. THE DEVELOPMENT LIFE CYCLE

We have already looked at the life-cycle stages earlier. Figure 4.1 repeats these and shows how the stages are interrelated and how the whole cycle constantly involves referring back and going back to what has already been done.

(a) Work should proceed in orderly steps, with the output from each step being checked against the definition of that step before work proceeds.

(b) Notwithstanding the point made in (a), there must be some recycling, reworking of a previous stage in the light of subsequent work. There must be constant referral to the requirements specification. In other words, it is an iterative process.

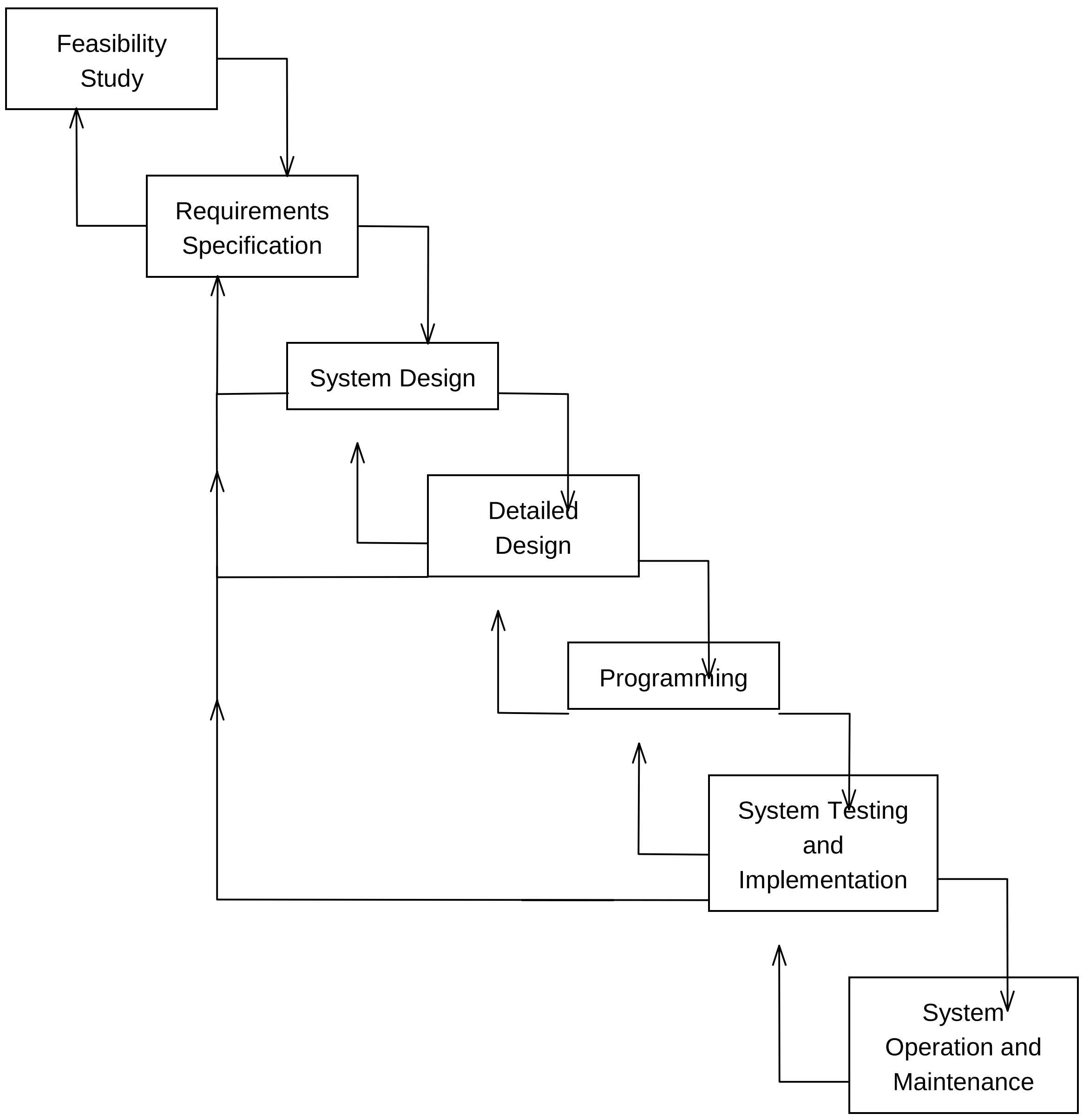

Feasibility Study
Requirements Specification
System Design
Detailed Design
Programming
System Testing and Implementation
System Operation and Maintenance

Figure 4.1: System development life cycle

(c) If the system is to be provided by an outside supplier, then Figure 4.1 can be reduced, as shown in Figure 4.2. **But**, remember, if the supplier is providing a tailor-made system, the purchaser should monitor progress, since the three stages of system design, detailed design and programming will have to be carried out by the supplier and, during these stages, there should be continuous feedback on how the work is progressing, whether any snags have developed, how costs and time-scales are matching up to budget, etc.

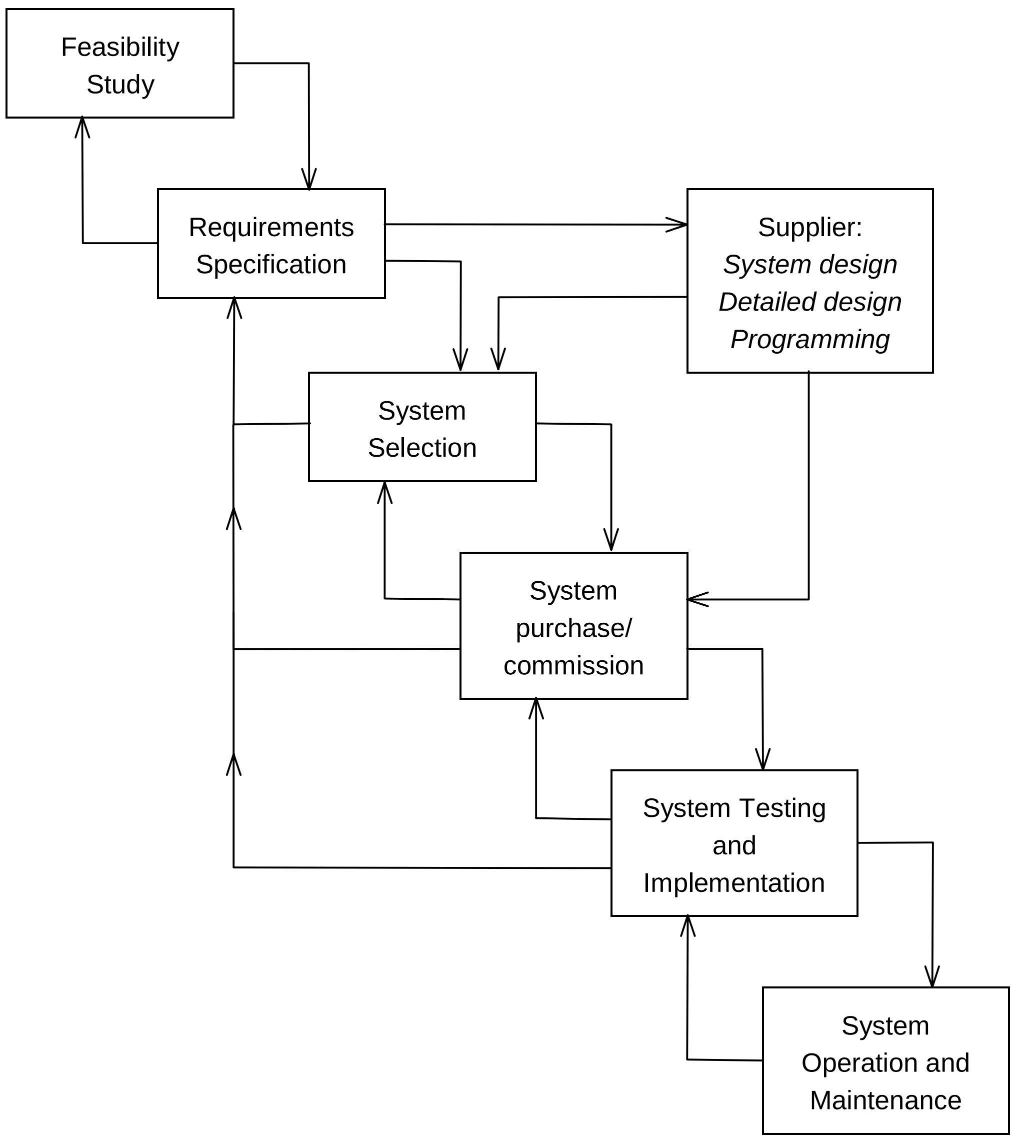

Feasibility Study
Requirements Specification
Supplier:
System design
Detailed design
Programming
System Selection
System purchase/ commission
System Testing and Implementation
System Operation and Maintenance

Figure 4.2: System development life cycle (with external support)

Key Points and their Output

There are certain key points within the life cycle, for example:

- Agreement of the specification of requirements, which should be clear, consistent and unambiguous to users as well as computer specialists.
- Systems testing and hand-over to the users.

These key points must be clearly defined and the output of each stage needs to be:

(a) specified before work starts; and

(b) agreed as fulfilling this specification before the stage is completed.

This production of specified items at each stage shows everyone how progress is being made, but it also highlights any areas where problems might develop.

Particularly with the development of software, it is most important that errors are discovered as early as possible. An error found at the operational stage can cost fifty to a hundred times as much to correct as if it had been found at the design stage.

Key points are sometimes called **milestones** and it is important that their position, and what is to be delivered at that stage, are agreed with both purchaser and supplier. Purchaser and supplier may both be within the same company; for example, user department and computer department. If the supplier is external, it is important that the actual people who will eventually be using the system are involved in its specification and acceptance.

The Importance of Documentation

Documentation should be going on in parallel with the other activities and, at least at the early stages, documents will be the items produced as output. These will need to be referred to at later stages. For example, the requirements specification will need to be checked against parts of the feasibility study and any major changes in plan discussed and agreed. This requirements specification will be the basis against which the design is checked. This does not mean that, once agreed, there can be no changes to the specification. Work carried out at the design stage may show how improvements can be made. The specification, after full steering committee and user consultation, will be altered to accommodate the improvements.

B. SPECIFICATION OF REQUIREMENTS

The first stage of all development cycles is a statement of requirements from the eventual user department.

This should state **what** is required both functionally and financially. It does not suggest **how** these requirements will be achieved.

It is also known as the **user requirement document**.

The first section will contain a general description of the system which has to be designed. It will act as a general guide to the rest of the document, giving an overall picture of what is needed.

(a) What is the system expected to do?

A precise and full statement of the objectives of the system is contained in this section. Mere generalisations are insufficient. Wherever possible, the objectives should be given some measure so that the success of the system, when operational, can be assessed.

(b) Who will use the system?

It is always important to identify clearly the eventual owners of the system. This is likely to be a department. The department will be responsible for accepting and approving each stage of development and they also act as the primary point for involvement and consultation of the users with the development team.

(c) What is the minimum the system must do?

It is important for the user department to specify the minimum requirements. This is also useful for the future, as it gives a better outline of the requirements and keeps the designer from developing an over-complex system that does as much as possible rather than just what is required.

System Selection

There are series of factors which need to be taken into account in identifying the system for development:

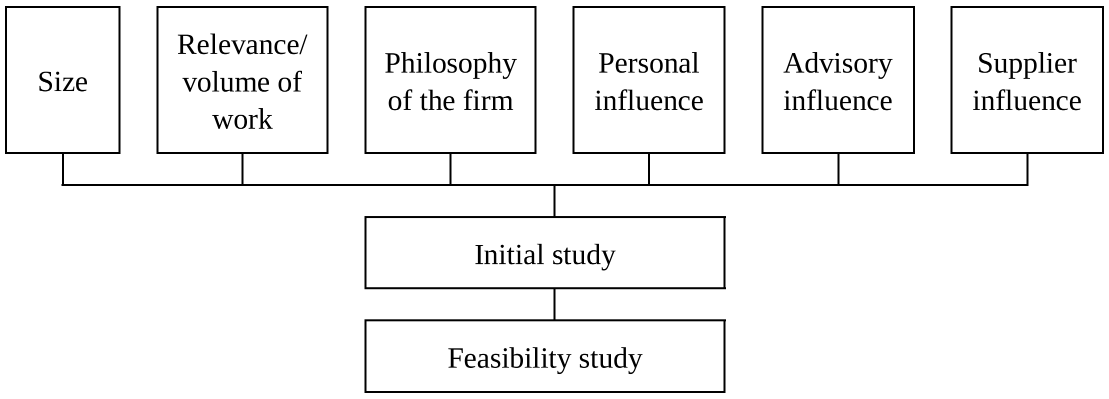


Figure 4.2: Influencing factors in system selection

We already know that computers can do repetitive jobs very accurately and quickly, provided they are given a suitable list of instructions – the program. In addition, large volumes of work provide even greater savings when carried out on computers. Therefore, jobs that come within these parameters are top of the list for consideration. In effect these criteria can be regarded as the fundamental "building blocks" for the study.

After these initial considerations we need to look at the future plans of the firm. How much expansion is anticipated in terms of output volumes, staff numbers and so on? After all a computer can sometimes be justified by allowing expansion without increasing staff to handle it.

There is yet another reason to consider! If all your records are kept manually, it is likely that they will be filed in a specific sequence. What happens if you require information to be extracted from them?

At best it involves your staff in selectively collecting the facts by laboriously searching through every record. At worst the information may take so long to collate that its use becomes nullified – remember what we said earlier about the timeliness of information? In such a case you don't bother to ask. The facts are there, somewhere; the information is theoretically available; unfortunately cost and time make its retrieval impractical.

So here is yet another area where a computer could possibly help your business – producing accurate, timely information that would either be difficult or impossible to retrieve without the speed of modern computer systems.

Let us summarise what we are looking at at this early stage:

- Ways in which the computer can help in your processing. This does not necessarily mean that the computer will take over completely – it may merely complement your staff.
- Operations which involve the frequent copying of data from one source to another, particularly if complex searching and sorting is involved.
- Routine work, particularly large volumes, which is carried out frequently in an identical way.
- Individual staff workloads – are some staff experiencing "peaks" which stretch them to their limit?
- Security – are checks on accuracy being omitted or non-existent because of the demands/limitations of the existing system?

- Any areas where information is not available or available too late for it to be used effectively.
- Speed – can the speed of response be useful in providing a better service whereby queries are answered "immediately"?

The System Environment

The need for a new development will be formulated within an existing environment, and it is important to have an understanding of that environment at the outset.

We have already considered many of these factors, but it will be useful to gather them together under three socio-technical, environmental headings.

- **Economic**

  Economic factors centre around the expected increase in profitability and/or improvement in both internal and external services provided.

  Reports should use actual factual values rather than "vague generalisations" – there are of course categories of improvement which, by their nature, are difficult to place a tangible value upon.

- **Technical**

  Technical considerations can be subdivided into those which affect the firm and those that relate to the computer – its operating constraints and environment. It is important that the organisational considerations include all aspects of the new system – those which are to be carried forward from existing systems and those that will be introduced as a result of the new system.

  Aims must be stated explicitly so that no ambiguity can possibly interfere with anyone's understanding of them. Computer constraints can be said to relate to the technical aspects of the system such as system design, programming language used and so on. Reports will not deal with these considerations alone – they would represent a consensus between the firm's management and the computer specialists.

- **Social**

  The third consideration, social, is often very difficult to quantify because it involves such issues as job satisfaction and industrial relations. Nevertheless, it is important to indicate in some way the expected effect in these areas.

The Initial Study

It is the **initial study** which is used to formulate the user requirements of the new system.

The initial study will be conducted by a small team consisting of the designated project manager, a senior user department manager and, possibly, an accountant. The actual composition of the team is not important, although it will be quite senior.

The project manager will need to gain the confidence of the staff – in other words get the staff on his or her side. This will have several benefits – staff will feel less threatened by the impending changes and will give more co-operation. The project manager will be wary of making educated guesses at what the outcome will be – people, being human, will often feel that an incorrect guess was deliberate. On the other hand, if staff on the current system are really heavily loaded, it may be possible to say that there will be no redundancies – a fact that could make a great deal of difference to people's attitudes.

User Involvement

This is another aspect that we have already discussed, and we will return to it again. However, it is probably one of the most important features of modern system development, and it is, of course, of particular importance at this early problem definition stage.

It is essential for the potential user of the future system to work with the system analyst from the very beginning. This provides clear advantages, including:

- acceptance of every stage of the development;
- ensuring the design meets the user needs;
- making the provision of training easier as there is a user representative who understands the system.

Other advantages are training within user areas of development staff and establishing good relations between all computing staff and the users. Thus, not only do the development staff understand the user's requirements, but the user understands the project as well.

We shall look in more detail at the whole area of development staff and relations in the next section, before moving on to examine the way in which systems are investigated.

C. THE PERSONNEL INVOLVED

The Computer Professionals

There are three key groups of specialist computer professionals involved in systems development – systems analysts, systems designers and programmers. However, the exact demarcation lines between analysis, design and programming are not very precise and vary from organisation to organisation. In some companies the programmers will design detailed file and report layouts, working from an overall specification provided by the systems designers. As the use of programming aids becomes more widespread, and the work involved in detailed coding is therefore reduced, this division between different kinds of computer professionals will become even more blurred. It is, though, a useful distinction to draw in order to consider the roles involved at different stages in the development process.

(a) The Systems Analyst

We can say that the general role of systems analysts is continually to update the information system of an organisation. They will maintain a continual survey of information requirements and propose changes in (or design new) systems, control the implementation of the designs and monitor their operation.

Systems analysts are responsible for the analysis, specification and implementation of computing or computing-related projects assigned to them. It is thus essential that the analyst has both computer and business knowledge and experience. The job has a very wide scope, and perhaps there is a need to divide it into two – an investigator with business training and a designer with a background in computers.

(b) The Systems Designer

System designing can be defined as the act of analysing and documenting existing systems, and – more particularly – the act of creating new systems meeting defined objectives.

Systems designers can work in one of two ways:

(a) converting an existing system (usually clerical) into another (usually computerised) system; or

(b) creating an entirely new system to meet an entirely new need.

It is obvious, therefore, that the specification of requirements stage is again very important. If management is looking for a sophisticated, far-seeing system but does not specify this, then what it may find itself provided with is merely the existing system mechanised or computerised.

Essentially, the activities of system designers centre around converting what is to be done (given by the requirement specification) into how it is to be done. They will thus have to undertake the following tasks:

- Study the requirement specification and determine the overall system in terms of input, output, processing and storage.
- Design, in detail, the layouts of all output documents, reports and displays to be produced by the system and define their expected frequencies and volumes, where these are not clearly expressed in the requirement specification.
- Determine the processing logic of the system; this will define the programs which are required, the functions they will carry out and the order in which they will run.
- Very carefully determine all the control, validation and audit procedures needed in the system, so that these can be incorporated into the design at the appropriate places.
- Design the input data layouts and define expected volumes, frequencies, etc.
- Specify and design the secondary memory files; this will include detailed record design, file organisation decisions, estimated volumes, update frequencies, retention periods, security and control procedures.
- Finalise the data specifications for input, output and storage to ensure nothing has been overlooked.
- Design the manual procedures associated with the new system.
- Define the system test requirements which are to be used, to ensure that the system is operating correctly.

(c) The Programmer

Programmers take the very precise design specifications and turn them into computer code which can be understood by the computer.

It is very important that they work closely with the design and code it as written and specified.

The programmer will run the tests of the code as set down by the designer. Any problems will be reported back to the designer for changes to be made to the design.

The programmer will also be involved in the actual implementation of the system in order to provide any necessary last minute coding changes.

A major area of work for programmers is in systems maintenance. During the operating life of all systems, various bugs will appear in the code. A programmer is responsible for correcting these in consultation with the designer.

From time to time, enhancements will be proposed for the system and again the programmer will be closely involved in coding the design changes.

Users as Clients

Gone are the days when computing staff were referred to as “data processors” and the people who relied on their systems were known as “users”, although you will have noticed that we continue to refer to “users” in this course. Information Technology staff now use systems engineering techniques to develop products for their clients. Such products may operate on the client’s own hardware or may form part of a total facilities management service provided by the IT department or external contractor.

A feature of business today is “total quality” and this introduces concepts such as:

- Get it right first time/zero defects.
- Internal clients (other departments in the company).
- Continual improvement.

It is now, therefore, more important than ever that the client participates fully in the design, development, implementation and operation of the product.

We have already reviewed the structure project teams for this process. For the purposes of this study unit, however, we shall consider the following categories of client:

- senior management
- junior management
- client staff.

Depending upon the size of system and organisation, there will be more or less overlap between these categories. In a small business one person may perform all roles, whereas in a large organisation many people may be involved.

(a) Senior Management

The client should be represented at a senior level by one person who is the final decision maker on all matters relating to the project. He or she will make the “go/no go” decision at the end of each stage of the development process but is unlikely to be involved in the day-to-day activities of the project team.

Senior management’s responsibilities can be summarised as:

- agreeing terms of reference for the project
- defining critical success factors
- reviewing progress on a regular basis
- “signing off” each major deliverable
- agreeing budgets and schedules.

(b) Junior Management

These are the people who will have regular contact with, and may even be part of, the project team. Working closely with the IT development staff they will need an appropriate degree of “IT literacy” and have a good understanding of the methodologies being used. That is not to say they should be IT experts since a good methodology will be readily understandable by most people.

They are likely to be managers/supervisors from the departments most affected by the introduction of the new system and thus in a good position to define the detailed requirements. During the operational phase of the system they will be key players in ensuring its success.

The responsibilities of junior management can be summarised as:

- Defining detailed objectives.
- Confirming "deliverables" meet client requirements.
- Making recommendations to senior management.
- Assisting in quality assurance.
- Participating in client acceptance.
- Helping to design training procedures.
- Participating in training activities.
- Assisting the implementation process.
- Using the implemented system.

(c) Client Staff

These are the people who will be using the system on a day-to-day basis. Some may be involved in the development process as part of the project team but their main responsibilities will be to:

- Assist the development process.
- Undertake training.
- Assist in client acceptance testing.
- Use the implemented system.
- Provide input to the post-investment appraisal.

Ensuring that the system is being designed to meet the "right" requirements is a key element of any successful project. The client has a major responsibility in this process not only in confirming the critical success factors but also at a more detailed level.

The development methodologies described earlier place great emphasis on diagrams on the basis that "a picture is worth a thousand words". The methodology can thus help the client as well as the IT staff to confirm that the design is aimed at meeting requirements. It is very much an individual decision on the level of detail one can expect a client to confirm, with the norm being that such involvement is restricted to the top two or perhaps three levels of the structure.

In the case of especially important aspects of the design it may be necessary to involve the client in much more detail and, if the client is willing to devote the appropriate level of resources, the more one is able to do this the greater should be the eventual benefit. Such involvement does not, of course, absolve the IT staff from responsibility. The client's main role is to confirm **what** is required; the IT developers have responsibility for confirming this is the case and also **how** best to achieve those defined objectives.

The client will confirm acceptance of the following prior to commencement of the next stage of the project cycle. Such "sign-off" may well entail the physical change of letters between the project manager and the senior client representative:

- statement of requirements/terms of reference

- feasibility report
- requirements definition.

Whilst IT staff involved with the project have responsibility for ensuring that the necessary project control systems are in place, clients can assist in the review process by co-operating fully in change control. Once an amendment has been identified and the implications for cost and schedule are known, an appropriate client representative should confirm agreement or otherwise to the change taking place. Junior management may "sign-off" individual changes to certain limits, with significant variations requiring senior management approval. All changes should eventually be submitted for formal review and approval by the senior client representative.

It would be unusual for even the simplest of projects to reach completion without some problems arising and when these do, all participants, both developers and clients, should seek appropriate solutions. Whether the client has forgotten to define a particular requirement, or the IT staff have underestimated the time for a particular task, the situation is perhaps best resolved by discussions at an appropriate level in the project team structure.

Of course, clients cannot be expected to accept such responsibilities without some sort of help or training. We will discuss user/client training in a later study unit, but meanwhile we can note some specific help available, such as:

- on-line help screens
- operations manual
- help desk.

Whilst most systems currently being designed will incorporate some on-line help facility, a more detailed procedure manual may also be needed to cover complex cases. Where difficulties arise, the client will need access to someone to assist in resolving the problem and one means of providing this is via a help desk which the client can telephone to discuss the problem. Whilst the help desk may not have all the answers, it will be responsible for "managing the problem" and ensuring that the client's difficulty is overcome.

Computer and User Staff Relations

Computer specialists do tend to be young, well educated but not very experienced in down-to-earth business activities. User managers tend to be older and much more experienced in the organisation as a whole, and in the work of their own departments. Full co-operation, as we have stressed repeatedly, is essential. But there may be particular problems arising between user staff and computer staff.

(a) Need for Skill in Human Relations

The general work of gaining good user and computing staff relationships is not always easy. It does need skill in human relations. This skill is vital in such tasks as:

- Obtaining all relevant data for studying and defining the initial problem – often from staff who are reluctant to define weaker points of their work.
- Training staff in work quite new to them.
- Working with future users of the system, actively involving them in the solution of problems in design, encouraging creation of a design which incorporates the consensus of everyone concerned.

- "Selling" (because that is what the activity is) the chosen solution, which may not always turn out to be the most popular one with individuals (or with the user department as a whole).
- Gaining complete co-operation in testing.
- Accepting criticisms – whether reasonably founded or not – and overcoming problems jointly with a wide range of personnel at different levels and with different responsibilities.

Note also that skills in presenting information are very important as part of the "armoury" of the analyst:

- written reports with the right approach
- oral presentations, including the help of visual aids
- demonstrations of equipment.

(b) Possible Problems

Here is a short list of some of the difficulties in computing/user staff relationships which may arise:

- There may be no line/staff relationship to cover certain difficulties.
- The line manager may not appreciate the advantages of computers.
- Every line (and senior) manager is apprehensive of the unknown.
- Managers may believe that their jobs will be superseded by the system.
- The line manager may be used to adopting a "rule of thumb" approach, contrary to the discipline needed by the computerised system.
- Some older managers may be happy to allow their careers to draw quietly to a close without upheaval, whereas the computer/information technology specialist may be go-ahead and ambitious.

It is quite wrong for all operations and decisions to be left to systems staff during the implementation of the system, after the agreement as to the specifications has been reached. The IT department constitutes a staff group which exists to gather and analyse information, and to make recommendations concerning the line unit's information system. It is the line department – the user – which has to make the decisions. Often, a user department will virtually abdicate its responsibility for design and implementation of a system when its current manual system is badly controlled and weak. This means that what appears to be a systems problem is only a symptom of the real problem – bad management in the user department. It is not unknown for users in these circumstances to try to transfer the blame to the IT department.

A user department, too, may not be properly staffed and may be unable to meet its obligations to the system. To help in starting the system, the computing department may then offer more help than would normally be wise. However, in this case, it should be made known clearly that the assistance in no way relieves the user department of responsibility.

(a) Role of Communication

Effective communication is vital to the operation of a systems department. Communication is vital in establishing formal aims and formal policies, and also in informal resolution of problems.

Personalities, it is true, tend to assume greater importance in informal problem-solving; the computing manager can frequently gain a great deal from the presentation of an unbiased and carefully prepared analysis of the situation, offering the advantages and disadvantages of the alternatives, and by assisting user staff to arrive at a reasonable solution.

We have said that communication is concerned with aspects of selling. Computing managers often have to sell their point of view to their line managers, who may well have a narrow perspective, as it is their own department with which they are primarily concerned. The computing department cannot have such a narrow viewpoint since its work reaches across all departmental boundaries. The computing manager is obliged to look at what is best for the organisation.

Co-operation and communication are also allied. When communication is good, then computing and user staff work harmoniously together. The representatives of the user department should be well known, and be able to devote enough time to successful systems application and development.

D. SYSTEMS INVESTIGATION

The actual project development begins with a very full investigation by the systems analyst.

To establish requirements and constraints, the systems analyst must elicit all the necessary facts from potential users and all other interested parties.

The feasibility study, the production of a specification of requirements and the design of the system, all require a considerable amount of investigation. The results of the investigation must be fully recorded. We will now look at some of the methods of fact finding.

The Scope of Fact Finding

“Investigation” is concerned with the review of what is known now, the discovery of the previously unknown, and the structuring of experiments to find out facts about known areas and unknown ones.

Merely finding out about what is known is insufficient. Systems investigation should include such activities as model building, “brainstorming” and simulation, as well as experimentation. Techniques, therefore, are often regarded as going beyond the fairly straightforward concepts of the interview, the questionnaire, observation and document analysis.

(a) Problems

The most essential feature of any fact-finding technique is that the facts so found must be correct. A number of factors can distort the truth of facts discovered:

- The facts may be distorted, intentionally but more likely unintentionally, by the person presenting the information.
- The communication containing the facts must be interpreted by the person receiving it and the interpretation may distort the truth.
- The written word creates potential difficulties when the writer and readers do not know each other.
- Observation of the actions of people does not always reveal the true facts, as again it is subject to the interpretation of the observer; also, the person observed can disguise what is actually being done – how else does the conjurer operate?

These problems can be overcome provided systems analysts take steps to ensure that all facts are thoroughly checked. They must also be aware of the distortion they themselves may impose.

(b) Sources of Information

Each project may be concerned with different sources of information – some may rely on internal, others on external data.

As far as the current system is concerned, users are the most important source. It is they who are aware of the significant and trivial facts. However, a worker within the system may easily have been misinformed or have misunderstood the duties of a colleague; conflicting information about a system is not at all uncommon. The supervisor may not know all the procedures undertaken by staff; duplication of effort is common.

Most investigations start by looking at the current system; documentation for this will vary greatly in quality. Occasionally, it is found to be correct to the smallest detail, or, alternatively, it may be almost non-existent.

(c) What Facts are Required?

If it were possible to list the facts that should be obtained during a systems investigation, then a systems analyst's job would be easy. This cannot be done because no two systems are alike, but it is possible to give guidelines that pertain to most commercial applications. Systems analysts must use their common sense when following these guidelines.

- ***The organisation's range***

  This includes not only the products or services supplied, but also the number of people employed, the number of suppliers' and customers' accounts serviced, etc. – in other words, any items relevant to the area under investigation.

- ***Details of items in the range***

  The quantities, values, descriptions and fluctuations are required. In addition, the type of code numbers and their range are very important.

- ***Volumes***

  Once the type of data has been established, the volumes, receipt and despatch of documents, peaks and troughs in data flow, etc. must be recorded.

- ***Calculations***

  It is necessary to record precisely the calculations performed, particularly with regard to fractions, rounding-off, and the degree of accuracy required.

- ***Movement***

  Documents will move between departments in the organisation and also in from, and out to, other organisations. These movements will probably be shown diagrammatically.

- ***Exceptions***

  It is relatively easy to record what the usual procedures are. Because the exceptions occur infrequently and unpredictably, it may be difficult to identify them. Most of the problems that occur after the computer system has been implemented result from the exceptions that were not recorded during the systems investigation.

- ***Staff***

  An organisation chart will be helpful, and details of the number of different grades of staff and a brief description of what they do should be obtained.

  In this context it is important to ensure that the data gathered reflects the actual situation in force. In other words, care needs to be exercised to ensure that the organisation chart and supporting documentation have been accurately updated upon each change in organisational structure.

(d) How are These Facts Gathered?

- ***Asking***

  This is the holding of discussions, whether formal or informal, between the investigator and the staff who operate the system. **Interviewing** may be one method here, although this suggests a more formal and stereotyped approach. Often, many useful ideas and suggestions can be obtained by an informal chat.

  Before systems analysts commence this work, they should arrange, via the department head, a convenient time to approach staff and have at least an outline (or, better still, a list) of the facts required.

  It may be impracticable to visit personally all the staff involved, because they may be at branches scattered throughout the country or there may be too many of them. If the facts required are of a relatively simple nature, a **questionnaire** could be drawn up and despatched.

- ***Observation***

  Systems analysts will spend a great deal of time in the department being investigated; they will be asking questions, examining records, etc. During this time, they will be in a position to observe what is happening. For example, they may notice:

  (i) excessive staff movement

  (ii) some very busy clerks and others with much spare time

  (iii) frequency of file usage

  (iv) communication problems both within the department and outside.

- ***Reading***

  There may have been previous investigations of the department at present under scrutiny.

  Systems analysts should read these records to help them understand the current system. However, too great a reliance should not be placed on these documents, particularly if they were prepared some time ago.

  Other documents, such as policy statements, standing instructions, memoranda and letters pertaining to the investigation should also be read.

- ***Measuring and counting***

  The number of items, documents, transactions, people and time taken for individual processes should be measured, either to confirm facts already recorded or to establish the current position. Sampling (see later) may be used, but care should be taken to avoid bias. The frequency of different activities and events should also be recorded.

- ***The organisational structure***

  The organisational chart of the user department will often highlight features useful to the investigation and individual job descriptions will identify an individual's area of responsibility.

(e) Relevance and Verification

You may think that it is obvious that only relevant facts should be gathered. However, it is not always easy to see what is relevant and what must be discarded. It is easy to gather large volumes of facts, but not so easy to glean the important ones. Systems analysts should constantly refer to their terms of reference or other documents setting down the areas to be investigated.

Ideally, every single fact obtained should be checked. Practically, this is not possible, and a degree of tact and diplomacy is often required where facts are being questioned.

Analysts can find themselves in a consultative role if user staff are able to carry out the fact finding themselves. Provided that the facts are being obtained correctly, this method is to be encouraged. It involves the user staff right from the start, and enables them to look at their work objectively. They will need encouragement to suggest changes, but they will most likely accept changes that they have initiated.

Interviewing

Personal interviewing is the most satisfactory method of obtaining information. It will yield the best results provided it is carried out properly. The interviewers' approach, whilst being disciplined, can be as flexible as the occasion demands, and they can pursue any lead provided by the respondent to obtain the maximum amount of information. A good interviewer will achieve a balance between discipline and flexibility in order to draw maximum benefit from the interview.

This method is often the only means of obtaining opinions from senior people, particularly where the subject under discussion is highly technical. Such information is usually qualitative. Personal interviews may also be the best means of obtaining a large amount of quantitative information from respondents. Only a personal visit allows the researcher to see written material which the respondent may otherwise forget or ignore.

Personal interviews are, however, time-consuming and expensive. Their use may, therefore, be limited to a few key interviews, if the remaining information can be obtained in other ways.

(a) Advantages

The main advantages of interviewing are:

- The written word and its associated problems are eliminated.
- An interview is a dialogue, which enables both parties to ensure that the other is not misinterpreting what is being said; immediate correction is possible.
- Important side issues raised during the discussion can be followed through immediately and placed in context.
- Since analysts are face to face with the other party, they can use appropriate language for the discussion (this, of course, may demand considerable ability in the use of language by the analyst).

(b) Disadvantages

The main disadvantages of interviewing are:

- It may be necessary to interview a number of people in order to eliminate any distortions caused by each single interviewee; this can be very time-consuming for the analyst.
- During the interview, interviewees will be unable to carry out their normal tasks.
- The technique relies heavily upon the skill of the analyst in creating a suitable environment to put the interviewee at ease and to elicit the necessary facts. For example, if the interviewees feel that their livelihood is at stake, they may be unwilling to co-operate or may deliberately give misleading information in order to protect themselves.
- Difficult situations can arise when a particular operation is not carried out at the "office floor" level in the way managers think it is being carried out; the analyst's report must contain the true facts of the situation, but the managers may, as a result, feel it is attacking their position and competence.

(c) Conducting an Interview

The analyst must be aware of the potential dangers of the technique, and the following points should always be borne in mind:

- The analyst must carefully consider what facts are to be obtained from a particular interview before the interview takes place. An appointment should be made, to ensure that the interviewee is free to discuss the matters in hand, and interviewees should be given notice of the matters to be discussed in order that they may collect any necessary material together.
- During the interview itself, it is important that the analyst uses the interviewee's own language. This will help to reduce misunderstandings.
- The purpose of the interview is to elicit all the facts. To do this effectively, discussion must take place. Analysts must avoid asking questions which merely require a "Yes" or "No" reply. However, they must also be careful not to ask leading questions.
- The analysts' task at the interview is to obtain information. They must not express opinions.
- Care must be taken to distinguish fact from opinion. Although opinions are important, they must act only as a guide which the analyst can follow to find further facts.
- After the conclusion of the interview, analysts should prepare a written report and cross-check the facts given.

(d) Summary of Procedure

We can now draw up four rules of correct procedure for any interview:

- Identify the correct respondent – if the respondent cannot provide the information, the interviewer should not continue the interview, but try to contact someone else who can.
- Arrange the interview – appointments for personal interviews should be made in advance.
- Plan the interview – the success of the interview will be proportional to the time spent on preparation. A questionnaire or list of questions should be drawn up. Each interviewer must be conversant with the subject under investigation, and be technically briefed from

information gathered by desk research. As far as possible, the interviewer should anticipate problems and reactions, and be prepared to handle them.

- Record the interview – when the results have been written up, the interviewer should review them, note any problems or mistakes, and record any important points that should be raised in subsequent interviews.

Observation

A second method of fact finding available to the analyst is simple observation. Often, many more facts can be obtained by watching a particular operation than by mere interviewing alone. But observation must be accompanied by other methods, and analysts must know what they are looking for, prior to the period of observation.

The technique can obviously be applied only when there is something to observe. Thus, it can be used only to investigate existing systems, and then only to observe the actual working of the system. It cannot be used to find out abstract information, such as the objectives or constraints. When investigating existing systems, observations can act as a very useful cross-check on facts gained by other techniques.

A major problem with observation is that an analyst can never be sure that the operation he or she is watching is not affected by his or her own presence. The person being observed will be conscious of the analyst's presence and may work more carefully or quickly or slowly, depending upon his or her attitude to the purpose of the observations.

Questionnaires

(a) Written Questionnaires

Sending written questions to people may appear as an attractive technique, in that it saves the time needed for interviewing. However, it is fraught with major problems.

- It is extremely difficult to phrase questions which are unambiguous and make it clear exactly what information is sought.

  For example, a questionnaire sent to departments in a college concerning their potential use of a computer asked: "Please list the courses which will use a computer in the coming year". One department interpreted this as any course where a student used a computer, even if that use was merely a brief demonstration by a lecturer of a computer terminal. Another department interpreted it as only those courses in which students were taught to write programs. The original intention of the question was to find out how many courses used the computer as a teaching aid to students, whether the course was computer studies, accountancy, English literature, or whatever.

- The replies to questions are subject to misinterpretation by the analyst, for similar reasons.

- The analyst can never be sure that the questionnaire was completed seriously by the recipient. For example, has the recipient really checked the facts written, or merely guessed?

- Many recipients of questionnaires simply throw them in their waste-paper bins. The results may thus not be a valid sample of certain opinions or facts.

Expert advice on framing questionnaires should be sought by the analyst, and possibly a junior member of the analyst's team should help the recipients complete them. These two actions can alleviate some of the problems.

Questionnaires are a last resort method, to be used only when other methods are too time-consuming, uneconomic or not practicable for other reasons. They are useful when a large number of responses is sought.

(b) Use of Questionnaires in an Interview

A questionnaire may be required when collecting information from respondents by personal interview, for the following reasons:

- ***To ensure consistency of approach***

  Use of a standard questionnaire will ensure that the same questions are asked in the same way throughout an assignment. This is particularly important when more than one interviewer is involved. The questionnaire need not be rigidly structured, but it should always provide a guide for the interviewer. This will enable a standardised approach to each interview to be adopted.

- ***To achieve a logical flow of questions***

  It is often important to ask questions in a particular sequence in order to obtain accurate results. Some questions should be asked only in the middle or at the end of an interview to avoid influencing a respondent's replies to earlier questions. The interviewer often does not want to give too much information at the beginning of an interview, as this would probably bias the respondent's opinion.

- ***To avoid omissions***

  It is essential to ask all the questions for which answers are required at every interview. An interviewer can easily omit important questions during an interview if he or she is not working in a systematic way. It may be difficult and time-consuming to go back to the respondent once the interview has been terminated. A questionnaire serves as a check-list of key questions, and thus avoids omissions.

- ***To ensure data is collected in a form suitable for tabulation***

  Collecting information is not an end in itself. It must be processed, analysed and presented in a useful form. The discipline involved in designing a questionnaire with a view to tabulating the answers, ensures a logical flow of the questions that it is essential to ask, and eliminates the unnecessary ones.

(c) Rules for Design

Whether the questionnaire is to be used at an interview or sent out to respondents to be completed and returned by them, there are **five** basic rules to be observed when designing it:

- ***Avoid bias***

  Questions must be phrased to avoid influencing the respondent's reply.

- ***Test out questions***

  The ideal way to test the efficiency of the questionnaire is to get a few pilot ones completed to check if respondents can provide the answers required. These will highlight difficulties and produce at least some of the possible "odd" answers, so that the questionnaire can be modified beforehand. (You must remember that, however well a questionnaire is designed, someone will give unexpected answers.)

- ***Design for subsequent tabulation***

  This is fairly straightforward when compiling structured questionnaires, but not so easy for those requiring extensive use of open-ended questions.

  When possible, items should be pre-coded to reduce the time spent on tabulation. It will be necessary to meet special layout requirements if the analysis and tabulation is to be done by computer.

- ***Be simple, concise and logical***

  Questionnaires and questions should be as short as possible. Jargon terms (which may not be understood by all respondents) should not be used as they may be misinterpreted. The questionnaire should follow a logical pattern and should be divided into sections, each dealing with one aspect of the investigation. If statistical information is especially important, it may be necessary to ask for it during the middle stage of an interview when the respondent is most co-operative.

- ***Use the correct terms***

  It is essential to use the correct technical terms which can be readily understood by all respondents. Data should be collected in units recognised by the industry concerned.

Existing Written Material

The analyst must not ignore any existing written material associated with the area under investigation. Such documentation may include:

- Organisation charts.
- Company manuals describing the terms of reference for departments.
- Operational procedure manuals for existing systems.
- Reports and specifications of previous investigations of the area being looked at.
- Reports and specifications of systems which are related to (or associated with) the system to be developed.
- Any statistical information from which the analyst can determine (or predict) the volume of data which may be processed by the new system.
- Any policy or planning documents which may indicate future changes that may affect the operation and information output of the system.

The documents listed above will suffer from all the problems associated with the written word. As with observation, written material is useful as a cross-check for interviewing. Discrepancies must be investigated carefully. Remember, even if the written material was accurate at the time of writing, it may have become outdated. All written material must be cross-checked before using its facts in any report or specification.

Sampling

Analysts will need to obtain some quantitative information about the systems they are studying. They need to know the amount of time spent on particular aspects of the work:

- How many invoices are processed each day?
- What is the average number of items per invoice?
- And the maximum number?

It is impossible for analysts to sit for hours on end, watching and recording every movement of the staff. Neither do they have the time to examine many thousands of invoices in an attempt to get an accurate count of the items of interest.

In each case, analysts will want to take a few figures, a sample, and use these figures as a basis for calculation.

The fundamental problem is, how many figures is “a few”, and how should the analyst obtain them? How reliable are figures based on samples?

Using the theory of probability and the mathematics associated with it, it is possible to take a sample and, from the characteristics, draw conclusions about the characteristics of the system from which we have taken it. Since the theoretical knowledge required is fairly extensive, we cannot describe it here, and if sampling is to be used to any extent, the analyst should take advice from a statistician. However, we will briefly describe the ideas involved in sampling so that you are aware of the possibilities of using these techniques in an investigation.

Sampling is a method for reducing the amount of observation which analysts have to do, as well as giving them a “feel” for the numerical information which can be obtained from records. The main lesson to learn is that averages are not a sound basis upon which to work.

When selecting a sample from a larger population, there are **two** main points to note:

- The number of items in the sample must be large enough to avoid any abnormal items having an undue effect on the average of the sample.
- Every item in the population must have the same chance as any other of being in the selected sample.

However much care is taken in selecting the sampling method, there will still be some sampling error – that is, the difference between the estimate of a value obtained from the sample and the actual value.

E. THE FEASIBILITY OR INITIAL STUDY

As a result of gathering and analysing facts about the existing system, the analyst will draw up a feasibility study. This is an extremely important document as it is used to justify (or not) expenditure on the development of a new system.

The proposals for new systems generally come from user departments. They draw up a short document, the user requirements, setting out what is wanted. This is submitted to the steering group, who will pass it to a small team for initial investigation.

The initial selection will have been based on relatively scant information, but the people carrying out the feasibility study need terms of reference, which will be drawn up by management and based on the user requirement.

Aims

The aims of the study are to investigate the proposed system and produce a feasibility report which will contain the results of the study and indicate possible methods of achieving the requirements. The alternatives will be accompanied by cost/benefit comparisons, to assist the steering committee in deciding which solution to adopt.

A very detailed investigation is usually unnecessary at this stage, as only broad estimates are required. This also keeps the cost of the investigation to a minimum.

There are **three** aspects to the question of feasibility. We have already discussed these from the perspective of influence on the proposed system:

(a) Technical

Is the current state of hardware and software technology capable of supporting the idea of the system?

(b) Personnel

Is the expertise of the systems development section sufficiently advanced to tackle (and successfully achieve) the production of the system? Such a question must also be asked of an outside agency if it is to produce the system. If the answer is "No", can the expertise be obtained by hiring new staff or training existing staff? Is the cost of this justifiable?

Many modern systems are run by users themselves who may, or may not, have the support of an IT department. With a small business system, it may be necessary to go back to the supplier if support is required. In both cases, it is necessary to consider the personnel within the user departments when deciding on the feasibility of a particular approach.

(c) Financial

Even if the system does provide an acceptable return on investment, is the capital available for outlay on the project?

Determining the Main Requirements of the System

The assignment brief will have specified in general terms the requirements of the proposed system. However, even at this preliminary stage, detailed requirements must be determined, otherwise the analyst will be unable to envisage the total scope of the problem and thereby suggest possible solutions.

Initially, therefore, the team will conduct a survey of all interested parties to find out exactly what will be required of the system.

(a) Main Characteristics

These will be defined in terms of the output information required. At this preliminary stage, the detailed layouts and methods of presentation will not be needed, but the team will need to find out:

- Volumes of output required.
- Frequencies of the output.
- Geographical location of the receivers of the information.
- Response times to requests for information if on-demand output is required.

When the outputs have been determined, the team will be able to ascertain – again in general terms – the volume, frequency, etc. of input data and the volume of stored data necessary to produce the output.

(b) Main Constraints

Not only must the requirements of the system be known, but also any constraints on its operation. We have indicated one of these above – that of geographical location. Others might include:

- Limitations caused by associated existing systems.

- Financial and staffing considerations.
- Statutory constraints.
- Auditing requirements.
- IT policies governing the inclusion of ultra-new technology (many firms will only use tried and tested technologies – many “fingers have been burnt” by pioneering organisations).
- Current computer equipment.
- Environmental constraints.
- Time, especially when information must be available at a particular time (as in a payroll system).
- Performance in terms of throughput, response times with a given number of users, etc.
- Accuracy of output information.
- Control and security procedures.

(c) Input Data

This section will describe all the input data. It will indicate the type of data, its quantity and frequency. If relevant, it will define the structure and layout of the data when the design of input data documents is outside the control of the systems designer. The medium for input may be a specific requirement.

(d) Output Information

All the output to be produced by the system will be specified in this section. Unless there is a specific requirement, it may be left to the systems designer to prepare the actual layouts and presentation of the information. However, this section will specify the elements of information required for each output, the frequency and volume of output, together with the medium on which the output is to be presented.

(e) Validation Requirements

The data input to any automatic data processing system must be correct. It is important that the validation procedures to be adopted in the new system are specified fully so that the designer can incorporate them in the design.

This section will specify both the validation procedures to be adopted, and the ranges of values and other associated information for every data element used by the system. For example, a name must be given a maximum length, a code number, its check digit mechanism and the range of values it can adopt.

In addition, if the system is to be on-line with direct data entry from keyboards, then the methods of indicating errors and the action to be taken by the system should be specified, especially if the user is to be allowed to make immediate corrections to any data erroneously input.

(f) File Requirements

Unless the system uses files created by other systems, it is usual to leave the file requirements to the system designer. If the system does use existing files, then their details will be included in this section.

This is a fundamentally important point with existing files increasingly being used to promote integration throughout systems methodology.

(g) Enhancements

It is quite usual for a system to be specified with the intention of enhancing its facilities at a later date, or for there to be a future system envisaged which will use the output or files of the specified system.

The system designer, in studying this section of the requirement specification, will be able to allow for such enhancements and facilitate their inclusion in the future.

Considering Alternatives

When all the relevant facts have been collected, differing approaches to solving the problem can be considered. The team should investigate solutions which have been found for systems with similar requirements both within and outside the organisation.

Cost/Benefit Analysis

This is the most important aspect of the feasibility study as its results will be the main information from which future decisions concerning the viability of the project will be determined.

Cost/benefit analysis is frequently used in management accountancy in order to compare solutions to a given problem and, by using value judgements, to assess savings and losses made by various parts of the system. Essentially, such an analysis attempts to place a monetary value on all aspects of a system. Some aspects will create a cost (e.g. the purchase of computer equipment) and others will create income or reduce cost (e.g. an increase in the number of transactions handled or the scrapping of existing computing machinery).

The method is not as straightforward as it at first appears since some items – usually benefits – are greatly influenced by subjective evaluation. For example, what is the benefit of providing a better service to customers? How much will sales increase if the sales director has more information on which to base marketing decisions?

When each aspect of each alternative has been given a monetary value, then their costs and benefits can be compared. We can therefore identify three main parts to such an analysis:

(a) Determine the costs

These may be divided into four parts:

- **Computing costs** – which start with the salaries and associated overheads for the systems analysts as they perform the feasibility study and investigate the requirements of the new system and then go on to develop and implement the system. This work may be carried out by people from an outside company, in which case the costs will be easier to itemise. Also under this heading comes the cost of any new computer hardware and any "bought-in" software. These can be regarded as **capital costs** of the new system.
- **User department costs** incurred during the investigation and implementation stages, which include wasted productive time as a result of the investigation disturbing the normal flow of business, education and training costs and "phasing in" costs when, for example, parallel running (see later) may make some overtime necessary. These again can be regarded as capital costs.
- **Operational costs**, which include salaries and overheads of the people running the system (who may be IT or user staff or both), consumable materials such as special

stationery and disks, communication costs (for example, telephone, if the system has some remote users) and hardware and software maintenance.

- **Other costs** – all systems must be monitored throughout their life to ensure that they continue to fulfil the original requirements, or to check that the requirements still apply. The system may thus require modification if it is found not to be achieving the best results. This evaluation and modification is often termed "system maintenance" and can involve major expense, especially if the design is poor and ease of maintenance was not allowed for in the system.

(b) Determine the benefits

These may be assessed under three headings:

- **Direct savings**, which can be defined as a definite reduction in cost as a result of changing from an old to a new system. These are fairly easy to identify and attach a monetary value to, and because of this they are sometimes used as the sole measure of benefit. However, this is incorrect, as the other two groups of benefits often give rise to much greater returns on the investment in the new system.
- **Measurable benefits**, which can be defined as "a monetary or financial return which accrues to the organisation as a result of the operation of the new system".

  These are by far the most important benefits to be balanced against the costs of the new system. They are often difficult to assess, yet effort should be expended to assess their monetary value. One example is a purchase ordering system which may produce great benefits by ensuring that the best possible terms for bulk purchase discounts and early payment discounts are obtained.
- **Intangible benefits**, which are those to which no monetary value can be attached. A new system will usually provide improved information. For example, an on-line order processing system will expedite the despatch of goods to customers. The organisation will thus provide a better service than its competitors, and this should mean higher sales, but the level will be difficult to assess.

(c) Perform a comparison to determine the "best" approach

The most important comparison is between the measured costs and benefits of the alternative systems. One might assume that this is a simple task of adding all the costs, subtracting this from the total of the benefits, and the system showing the largest net benefit is the one to choose.

However, factors such as return on investment, inflation and risk have to be taken into account.

The Report

We can now list the contents of a typical feasibility report produced by the team of systems analysts on completion of their investigation. These are as follows:

- Restatement of the **terms of reference**.
- Summary of the **investigation** (often the detailed findings are attached as an appendix to the report).
- Identification of the **requirements** and **constraints**.
- Descriptions of the **solutions**.
- Detailed **costings** and **benefits** of each solution.

- Summary **comparisons**.
- **Recommendations**.

This report will be sent to that part of management which requested it; it must enable them to determine the next course of action. Assuming that one of the proposed approaches is selected, then the next stage of the project will be authorised.

F. USING CONSULTANTS

In certain cases, an organisation may decide to hire a consultant or engage a consultancy firm which has expertise in the required area. This may be because the organisation lacks skill and knowledge of the techniques it wishes to employ, or because there are insufficient resources within the organisation to carry out the required work.

Consultancy firms with the requisite knowledge can also advise on the setting of standards, the adequacy of control procedures, assessing the performance of existing systems, etc.

Consultants should be able to formulate and evaluate alternative strategies and, if necessary, help in the implementation of the selected solution. In order to carry out the work successfully, consultants need the wholehearted support of everyone in the organisation concerned with the project.

Preparation is important. It is not enough simply to hire a consultancy and ask them to get on with the job. As with all such situations, the user organisation will need to think through in some detail what it is they want. They will need to formulate limitations of what should be done, both in the extent of the proposed system and its cost. It will also be necessary to specify any existing constraints as for any project. Typically, these may be existing hardware and software systems, time-scales, existing auditing procedures and environmental considerations.

In addition, the progress of the work should be carefully monitored in case unexpected snags occur. A senior person within the organisation should be appointed as the consultant’s main contact so that there is a clear way of reporting progress, obtaining information and sorting out any difficulties which may arise.

If consultants are used by a company with an existing computer manager, it is important that this manager should participate in the policy decisions relating to this arrangement. If this does not happen, the manager may find that he or she is left with unrealistic objectives established by the consultant or, at least, objectives with which he or she has not been associated.

Problems often arise when using consultancies. These are, of course, generally due to lack of prior preparation by the contracting organisation, although there are other possible reasons:

- **Lack of experience**, possibly on both sides as we can never assume the contractor is fully experienced in our requirement area.
- **Over-optimism** as everyone concerned wants to complete the job as soon as possible and, as with most projects, more time than that allocated is required. This can cause friction with the consultancy.
- **Communication problems** occur even when a senior member of staff is assigned as main contact with the consultancy. That member of staff may frequently be unavailable. This will cause delay as a minimum effect and maybe even misunderstandings over the requirements.

It will always be necessary to arrange for frequent expert reviews of progress and proposals to date. The computer manager should always be involved here, as well as the senior manager from the user department requiring the system. Such reviews should compare progress with the project plan, ensure adequate resources continue to be available and monitor the costs to date.

Study Unit 5

System Design

Contents | *Page*

A. LOGICAL AND PHYSICAL DESIGN

We can differentiate between logical system design and physical system design. The design of the logical system will take as its starting-point an outline model of user requirements. Assuming that this is in the form of a data flow diagram, the process of logical design involves successive refinement of the model until it meets all of the user requirements. The refinement normally starts with the outputs of the system and works back via files (or stored data) to inputs and procedures, taking into account the objectives and constraints that the system has to accommodate.

When the logical system has been defined to the satisfaction of the user, work can begin on detailed physical system design, i.e. detailed specification of the way the system will operate in a specific environment, using specific equipment and specific people.

There are lots of tasks involved in physical system design, covering all aspects of computer-based systems. For the sake of simplicity we will break them down into a series of separate tasks, though, in practice, they would not be carried out separately or in isolation.

The tasks we can identify are output design, input design, file design, program design, user procedure design, forms and dialogue design, and security and controls. Any approach to computer system design will involve all of these aspects in an integrated way. The design of files, for example, will be tied in with the design of input; input must closely relate to forms and dialogues; all data (input, file and output) has to be considered in relation to programs, and programs in relation to user procedures. In other words, they all relate to one another and cannot be carried out without these relationships being optimised.

The relationship between the logical and physical system design is shown in Figure 5.1.

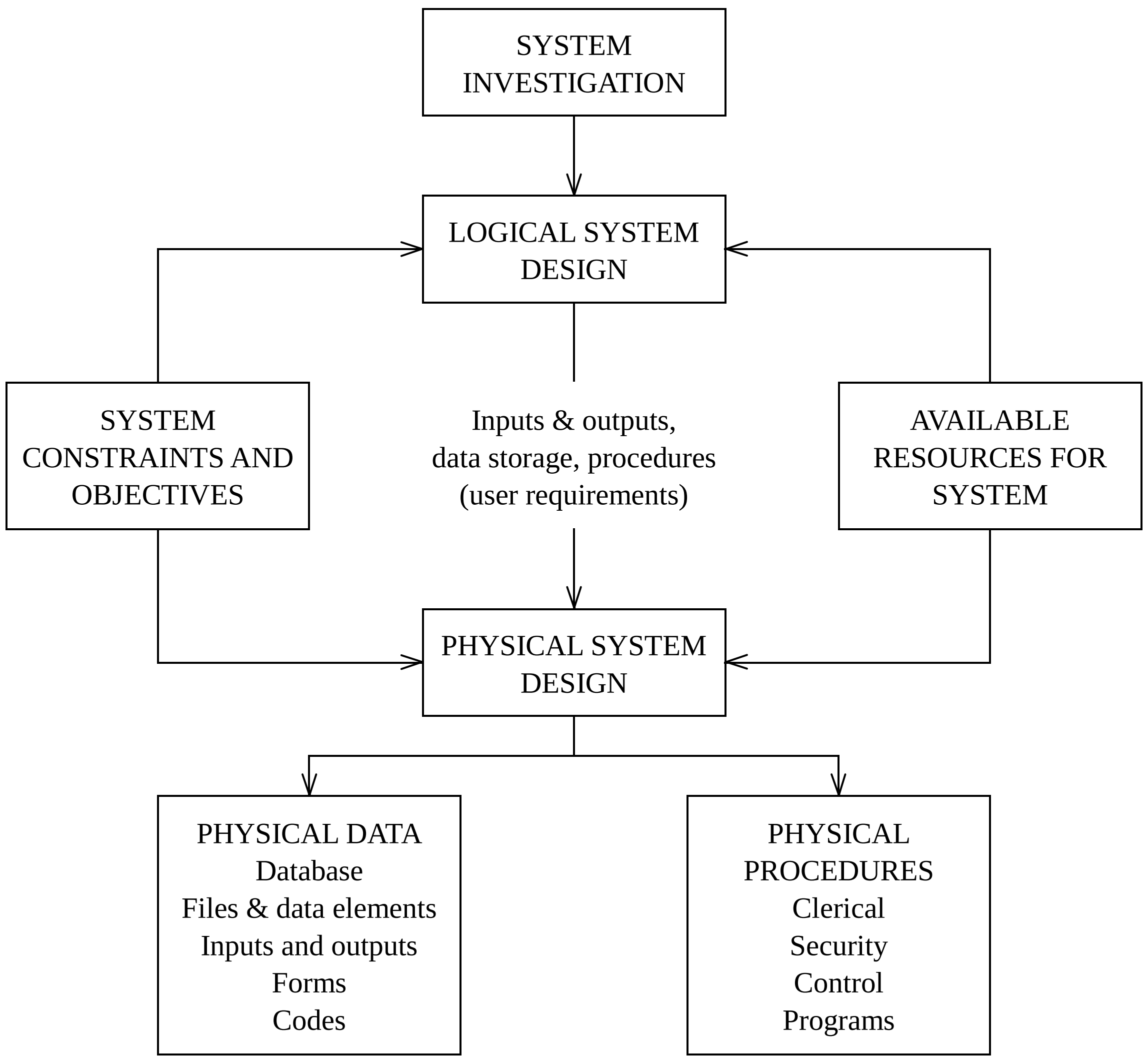


Figure 5.1: System investigation, logical design and physical design

Points to Note

In logical system definition the analyst must take the following into account:

(a) The design objectives must be completely specified.

(b) Data requiring storage must be structured.

(c) Conversion of the analysis results into inputs to and outputs from the system is required.

(d) The kind of processing needed to meet user requirements is considered at this stage.

B. THE DESIGN STAGE

We can now return to the design stage itself which will personify and embrace the points made above.

Importance of Requirement Specification

When the requirement specification is complete, it must be approved by management. The document is very important, as the new system will be judged on the basis of it.

Before the actual design and implementation of a system, the requirement specification must be agreed with the potential user as a full statement of facilities needed. When a system meets the requirements stated, then the designer can consider that the development has been concluded successfully.

Perhaps the most significant reason for the dissatisfaction many managers have with computer systems is the lack of importance placed on a good requirement specification. It clarifies the purpose of the new system – when one is not produced, or little attention is paid to it, nobody – user, designer, manager or clerk – will really know the purpose for which the system was intended. It also has a major advantage for the designers in that they can design a system without interference. (Without an agreed requirement specification, the user will often not accept that what is produced is what was asked for, and will keep thinking of "improvements" to the system.)

The design stage of development should thus not be started until a formal agreement is reached on the requirement specification. Ideally, the user department management and the commissioning management should sign a written statement of agreement.

When a system has been specified, the possibility of using ready-made packages should always be investigated, since this will save a large amount of development work. If packages which may be suitable are identified, the facilities they offer will need to be compared with those required by the users. Sometimes it may be more cost-effective for users to get 90% of what they said they required if this can be obtained from a proven package, rather than waiting for a "100%" system to be developed.

Influences on Design

Whether packages are used or a new system developed from scratch, inevitably the desired aims could probably be achieved in several different ways, and so the final design is almost bound to be a compromise based on a whole set of influences: cost, accuracy, control, security, availability, reliability, and so on. The design must be acceptable to programmers and users.

(a) Cost

This will involve:

- Development cost, including the current design stage. (The feasibility and analysis stages are separately funded.)
- Operations cost, including data preparation, output handling supplies and maintenance.

(b) Accuracy

This means appropriate accuracy, resulting from a compromise between error avoidance and the cost of this.

(c) Control

The design must permit management control over activities, one facility of this type being the provision of control data in the form of exception reporting.

(d) Security

This is a fairly complicated aspect of design, largely concerning data security and confidentiality. We shall be referring to this later.

(e) Availability

This refers to availability of the resources which will comprise the operational system (staff, stationery, software, space, etc.). This is the responsibility of the analyst in the sense that he or she must be sure that, at the time of implementation, these are available.

(f) Reliability

The design has to be reliable, that is sufficiently strong in itself to withstand operational problems. In addition, there must be alternative computing facilities as "back-up" in the case of breakdown, and sufficient staff to deal with peaking of workload.

The Design Process

System design is essentially a creative process, involving much conceptual thinking. Almost anyone with a basic knowledge of computing should be able to design a computer system which would produce the right results. However, there is a much more difficult part – namely to design a system which can be implemented and controlled in practice, and which will not only produce the right results but will additionally produce them to time and (especially) do so economically. It is in these latter areas that the expert knowledge of the designer manifests itself, and it is these aspects which must be readily discernible in the detail of the system specification.

We now take a quick look at the way in which we might approach the design process.

(a) Start with the Results

One approach to design is to consider the output requirement first; this is the main vehicle for achieving the objectives. We look at the output from the point of view of content (the information to be produced) and of format (the way in which this information is to be presented).

(a) Organise the Data

We need to input the variable and fixed information for each procedure we want to carry out. Derived information will be produced by the computer and constant information such as report headings, etc. can be built into the computer program.

(c) Design the logical system

This will be carried out by taking into account a series of considerations as we consider below.

(d) Design the physical system

The logical design will be shown to the user for approval or suggestions. When approval is obtained, the physical design is prepared by the designer. It will consist of:

- Screen presentations;
- Input understandings;
- Output understanding;
- Processing descriptions.

The physical design implements the logical design in the technical specifications of the proposed system so as to optimise the use of hardware and software.

All the service requirements must be achieved, both technical and administrative.

C. SYSTEM DESIGN SPECIFICATION

The designer receives the requirements specification and produces the systems design specification.

The requirement specification describes **what** is required from the system. The next stage is to decide exactly **how** the system will be developed and implemented to meet these requirements.

A typical system design specification will contain an introduction, briefly highlighting the main system requirements, and the following sections.

(a) General Overview

This will define, in not too much detail, the facilities which the system provides. It will contain a system diagram, and will give the environment and basic operations philosophy of the system and describe its interface with existing systems. Facilities will be presented in terms of the input, output and storage provisions of the system.

(b) System Operation

Systems should serve distinct business processes. Thus there should be individual systems for each business process and sub-systems for each activity within the business process.

The detailed specification of the logic of the system will be given. Essentially, this will show how the input data is converted into output information and when the files are updated. Each program will be shown and placed in relation to the other programs.

Exact details of volumes of data and frequencies of processing will be described, together with any special requirements for recovery and restart in the event of system failure.

(c) Hardware and Software Requirements

This section will define in detail the hardware and systems software which must be available to enable the new applications system to operate successfully. In particular, it will specify any new hardware or systems software which must be obtained. If the new system is to operate within an existing computer system, there may be little in this section, unless it requires additional hardware or software.

(d) Output Layouts

This section will contain detailed plans of the layouts of every output produced by the system.

(e) Input Layouts

This section will contain detailed plans of the layouts of all input to the system. This will also include layouts for all documents to be used to collect data.

(f) Data Storage Specifications

Exact plans for every record to be used by the system will be specified, and the structure of the files or database will be given.

(g) Detailed Specifications

This section of the design specification will be divided into a number of subsections. For each program or module the following will be given:

- A general functional description specifying the facilities to be provided by the program;
- Details of input;

- Details of output;
- A detailed specification of any particular processing required, e.g. actions in the event of errors validation procedures, any formulae to be applied, any standard subroutines to be used.

(h) Manual Procedures

This section of the system design specification will define the clerical procedures for the new system.

(i) Test Requirements

Before a system can become operational it must be thoroughly tested to ensure that (as far as possible) it contains no errors in the software, and that it meets the requirements laid down in the requirement specification. This section will define in detail the tests to which the system will be subjected, to prove that it has been correctly manufactured.

(j) System Set-Up Procedures

Unless a system is completely new, it will be replacing an existing computer or manual system. All the stored data of the existing system will thus have to be transferred to the new system. This will be a once-and-for-all task, but an essential one. The mechanisms for the transfer must be carefully designed. Their design will be written in this section of the specification.

The mechanisms will be both manual and computerised, requiring the definition of special forms, programs, clerical procedures, etc. – so much so that this section becomes a miniature design specification in its own right.

(k) Glossary of Terms

A design specification will use jargon terms in order to be precise. The jargon used must be defined in this section.

(l) Index

In a complex system, the specification will be very large, and an index to it will be a great aid.

D. DESIGN CONSIDERATIONS

Input Design

As you are already aware, commercial data processing comprises four main functions: input, output, processing and storage. The systems designer will consider each and decide the most appropriate method to be adopted.

The designer must specify a complete mechanism from the origination of the data through to its input to the computer. SIX important factors must be considered:

(a) The data capture system must be **reliable** and reduce the potential for error to a minimum. The more separate stages there are, the greater the chance of error.

(b) The system must be **cost-effective**; the cost of reliability must not exceed the cost of errors, should they occur. For example, there is little point in spending £10,000 per year on a very reliable system which is error-free when a £5,000 system would eliminate most errors and the errors would cost only £1,000 per year to correct.

(c) The geographical **location** of the places at which the data is originated will have a great effect on the method chosen.

(d) The **response time** for output must be considered. The faster the response time, the faster the data capture system must be.

So, what kind of objectives should the systems analyst have as far as data capture and output are concerned? We may categorise them like this:

- to minimise the total volume of input, as far as is practicable;
- to minimise the extent of manually prepared input;
- to design input to the system so that the work in preparing it is as simple as possible;
- to minimise the number of steps between origin of the data and its input to the computer.

We have said that two of the objectives are to minimise the total volume of input and to minimise the extent of manually prepared input. However, we must also consider who is entering the data. The less the input, the less time and effort needs to be used in preparing it. Also, the lower the input volume, the less is the extent of errors introduced into the system. There are FOUR aspects to the minimisation of input volume:

(a) Data which is repeated needs to be entered into the system only once.

(b) Editing and spacing can be omitted from input which is keyed. This makes for more rapid input. Editing (for example, inserting decimal points) can be implied from the general format when the input is read into the computer.

Output Design

Output can be presented in many ways. Consider first whether the information is necessarily to be a permanent record. For example, an on-line enquiry system needs to supply the user with an immediate report which will be acted upon; after this the information is no longer required. A screen display would thus be the most appropriate output. On the other hand, employee wage packets must be printed as a permanent record for the employee.

Four further crucial aspects are:

(a) WHERE is it to be Forwarded?

Which location or departments require the information? In some cases it may be necessary to send output to people at a different geographical location, and the method of transmitting it will affect the design of the system.

(b) WHEN is it to be Forwarded?

The timing and related accuracy with which information is made available are crucial. Not only must the time interval be adhered to (e.g. daily, weekly, etc.) but on several occasions specific times of days will be stipulated.

(c) WHAT is to be Forwarded?

Quite simply, only that which is required. Any additional information makes the output unfamiliar to the recipient, and the unexpected is often ignored, hence additional information – even if thought to be useful – will probably be rejected.

(d) HOW is it to be Forwarded?

This aspect covers such items as the method of delivery and the format in which it is provided.

Processing Considerations

(a) Batch Processing

The logical processes that have to be carried out assist us in breaking up the overall process into far smaller ones. As soon as we carry out this split in the overall process, we are faced with the need to convey information from one process to the next. We create an intermediate or work file in one process and use it as part of the input to the next process. Usually, the complete set of input transactions will be handled by the initial process, and an intermediate file will be created ready for the next stage. This approach assumes that all transactions will be available for processing at the same time, and is known as the batch processing approach, as we discussed in a previous module of the course.

(b) Transaction Processing

Many systems require that a transaction be processed as soon as it is received; it cannot wait to be batched. For such a system we cannot use an intermediate file. In such transaction processing systems (we previously called these on-line processing systems) we may need to cope with a number of transactions being processed simultaneously and independently through the system.

We can now see that the time taken to react to a transaction – the response time – required of the system will influence the way we tackle its design. The response time will also influence the way in which the data is stored.

Storage Considerations

The selection of an appropriate method for organising the stored data is another important feature of the designer's task. The main points to be considered are:

(a) Volume of storage required;

(b) Method of access (sequential or random) of all processes;

(c) Volatility of the data;

(d) Activity of the collection of data in relation to all processes using it;

(e) Access (response) times required;

(f) Predictable additional volumes and uses.

Maintenance and Expansion

The design of a system must take into account the more or less inevitable maintenance that will be needed to keep the system up-to-date. If this inevitability is accepted, then design may be developed in such a way as to make amendments easier to carry out than they would otherwise be.

No single person or group can foresee all possible future needs. A system which is flexible, therefore, and which can be reshaped when considered necessary in the future, is the ideal. Remember that the system, including a computer system, has an input, a process and an output which may well be extremely complex. Aiming for flexibility requires, therefore, that none of these three items is fixed in nature.

With advances in technology occurring all the time, it may be advantageous to introduce changes to the hardware configuration without drastically altering the application programs. There may be a need for more on-line data entry; a remote location may wish to connect in its computer to the central one; or some word processing applications may want to access the main files or database.

How easy such changes will be, will depend on the hardware and software selected: although systems may appear to be infinitely expandable, there are many conflicting "standards" for connecting equipment together and exchanging data between systems.

Design Constraints

Except with the most trivial of systems, there will be some design constraints imposed by, for example:

- software standards, languages and operating system;
- hardware configuration;
- interfaces to other systems and procedures;
- safety margins, fault recovery procedures, etc.

Whatever particular constraints apply in a given case, these must be considered from the beginning of the design process.

E. HUMAN-COMPUTER INTERFACE

As we have just discussed, everyone has an increasing contact with computers. The non-computer-literate person approaches them with wariness, whilst the fully literate person can be highly critical of a machine that is obtrusive in their contact with the system being used.

Between the user and the machine is what we call the **human-computer interface** or **HCI**. The interface should be as clear and accessible as possible; consistent in how the approach to the machine is made and how the machine responds; comfortable to use; and common sense should guide use of the machine.

Various styles of interface are available.

Command-driven Interfaces

With this type of interface, the user types in a command and the system responds with the appropriate action. These interfaces are simple to design and use the keyboard for input. Command processing languages have been developed which are based on simple system commands. For instance, the shell language in UNIX is a very powerful command processing language and complex commands can be built from a sequence of simple demands.

Usually there are abbreviations which mean the user does not have to spend extra effort in typing. Alternatively, commands may be stored in files and different activities invoked by the user by executing the files.

There are **disadvantages** in that users have to learn the command language and they are likely to make errors when keying commands. These also require additional software to handle errors and diagnostics, otherwise they are not popular.

These interfaces are not suitable for inexperienced users. It takes time to learn them.

Careful considerations have to be made in the choice of system commands. Meaningful abbreviations need to be used while keeping typing to a minimum, although some systems allow users to redefine the command names. Users who utilise more than one machine find this useful.

Menu-driven Interfaces

In a menu-based system, users have to select one of a number of possible choices. The user may type the name, or an abbreviation for the choice, through the keyboard or use a mouse to indicate the choice. With some touch-sensitive screens, choice can be made by placing a finger at the selection.

These systems are popular for a number of reasons:

- Users do not have to remember individual commands
- Typing is minimal even for those who cannot type
- It is impossible to make an incorrect choice and crash the system
- Help can be provided at each stage and in context

Experienced users may find menu systems are slower than command-driven systems.

There are two types of menu in use, the **pull-down** and the **pop-up**.

- The pull-down menu is on continuous display and picking up a title with a mouse causes the menu to appear. Inappropriate options are displayed in grey and may not be selected.
- The pop-up menu appears at the cursor position when the mouse button is pressed.

Graphical User Interfaces (GUI)

GUIs are now the most common type of interface. Most operate on a WIMPs basis where the user selects the required options from the screen using the mouse. It is now common for new application system developments to include a GUI interface.

One of the major benefits of the GUI approach is that the look and feel can be made consistent. This cuts the times to learn a new package and it is also claimed that GUIs are instinctive, and experienced users can easily switch between systems.

GUI and WIMP interfaces are popular because they are easy to learn. The user has multiple screen interaction, and full-screen interaction is also allowed rather than the line-orientated interaction allowed by command-driven systems.

Voice

This method of input is still not particularly common. The ability to recognise commands tends to rely on the user speaking very slowly and clearly. The number of commands which can be recognised is still quite limited.

Screen Design in Practice

Most modern computer systems include at least some input and output via a display screen and keyboard, with programs running interactively and the user entering data at the keyboard in response to messages appearing on the screen. The layout of the information on the screen, and the order in which it is presented, is a most important part of the design of a system.

We will now look at some of the additional points which have to be considered when such systems are designed.

(a) Presentation of the Information

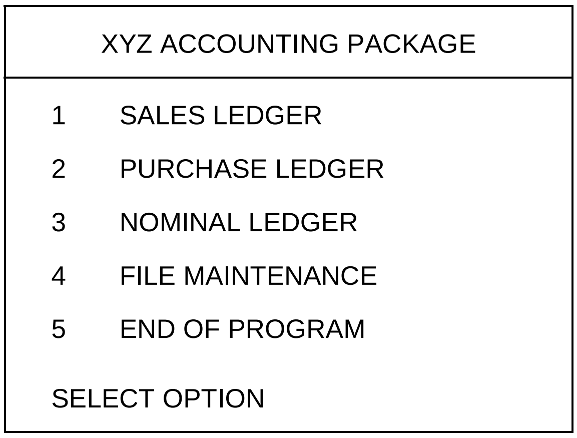
Seven points are important here:

- How the procedures can be divided into suitable “interactive screens” so that data is input in a logical order with only relevant responses being provided.

- Layout, so that important information is not lost amongst the data appearing on the screen.
- The order in which data is entered – this should not only be logical, but should also correspond (wherever possible) with its layout on the source document.
- What type of validation can usefully be carried out at this stage and how the operator should respond to errors – if data from a batch of forms is being entered, it may be more reasonable to note any errors which cannot be corrected directly and find answers to all the problems at the end of the run than to keep interrupting the data entry.
- Error messages must be designed so that they can be easily understood and the correct action taken.
- When a "flashing cursor" should be used.
- When it might be useful for the system to "bleep" at the user.

(b) Menus

A very common way of presenting a transaction processing system to the user is via a series of "menus". For example, an accounting package could have a main menu such as shown in Figure 5.2.



```
        XYZ ACCOUNTING PACKAGE

   1    SALES LEDGER
   2    PURCHASE LEDGER
   3    NOMINAL LEDGER
   4    FILE MAINTENANCE
   5    END OF PROGRAM

   SELECT OPTION
```

Figure 5.2: Sample main menu

This menu would appear on the screen once the user had entered the commands to start the accounting package. The initial commands might include a password, and sometimes people with different passwords will get different menus. For example, this menu may be available to senior accounting staff who are allowed to work on the nominal ledger and also carry out file maintenance procedures which include adding new customers, changing credit limits or codes. A junior clerk may get a reduced menu such as:

]

```
XYZ ACCOUNTING PACKAGE

1      SALES LEDGER
2      PURCHASE LEDGER
3      END OF PROGRAM

SELECT OPTION
```

Figure 5.3: Sample reduced menu

If Option 1 is selected, the sales ledger menu will appear.

```
XYZ SALES LEDGER

1      SALES LEDGER INPUT
2      TRANSACTIONS
3      STATEMENT PRINT
4      CUSTOMER AGED DEBTOR REPORT
5      RETURN TO MAIN MENU

SELECT OPTION
```

Figure 5.4: Sample second level menu

There are thus various levels of menu which can extend to a third or even fourth level. For example, entering Option 2 with the last screen will lead to a menu of the different types of transaction.

Again, care is required when designing menus, since, whilst being comprehensive and easy to follow, their use must not become tedious to the experienced user. One way of avoiding this is to allow progress through the menus to be accelerated by the experienced user. If an operator, given the first menu, knows that journal entries is Option 3 (say) on the transactions menu, it may be possible with some packages to enter 123 and get directly to the required program without waiting for the intermediate menus.

(c) Icons

An alternative to menus is the modern trend toward using icon-based screens. Here, small pictures representing the options are used in conjunction with a mouse to achieve selection. This approach has two main advantages in that the use of keyboard entry on selection is avoided and, for many users, a graphical option is easier to understand than lists of written dialogue. With increasing computer awareness these two alternatives become more “common” in their usage.

(d) MIS Design

The design of a management information system is little different in principle although a database is involved, which we have not yet discussed.

- The product of an MIS is information, mainly statistics, summaries and analysis of operational information.
- The input is operational information such as production and sales figures, stock levels, accounts ledgers, personnel levels, etc.,
- The processes are the manipulation of the operational information for MIS presentational purposes. The constraints imposed on the design will be the complexity of the company's activities, the authorised levels of access to specific data, the level of detail expected by the user and so on.

Batch processing will suffice as it is not the responsibility of the MIS to keep the data up-to-date. On the other hand, the MIS itself will certainly work on-line.

By using standardised procedures and documentation, maintenance will be relatively straightforward.

Flexibility for future development is unlikely with an MIS package, although an in-house built system may include facilities for further development, especially an on-line, user manipulation of the data provided.

Study Unit 6

Approaches to Systems Analysis and Design

Contents *Page*

A. STRUCTURED SYSTEMS ANALYSIS AND DESIGN METHODOLOGY

High maintenance systems usually result from an incomplete understanding of the requirements at the development stage. Structured Systems Analysis and Design Methodology (SSADM), developed originally in 1980 for central government use, provides a sound procedural framework for systems development from the beginning.

Since the early days of commercial computing, thousands of people have been involved in the production of computer systems. Their results range from acceptable to disastrous. In general, there has been a great deal of dissatisfaction amongst users of computer systems at the products they have been asked, or forced, to accept.

Most of these were produced using “traditional methods” of development, usually meaning hardly any method at all. Statements of requirements were produced in a rather sketchy manner, and as quickly as possible in order to get on with what was perceived to be the real work – writing programs and performing other activities associated with producing computer systems. In other words, producing a physical solution before the problem had been fully understood. The results of this approach can be seen in the form of systems which do not satisfy users’ real requirements and which are difficult and sometimes impossible to change.

Having been implemented, these systems usually consume significant amounts of computer department resources in attempts to make amendments. This is known as “maintenance”. Perhaps a better name for much of it would be “retrospective analysis and design” since it is largely concerned with tacking on processes and data which should have been incorporated, had they been discovered, during development rather than after implementation.

It is because there has been an eventual realisation of the need to improve the product, that more structured and rigorous methods of analysis and design have been developed. One of these is the Structured Systems Analysis and Design Method, more popularly known as SSADM.

Its purpose is to provide the analyst and designer with the tools and procedures to enable the production of computer systems which provide what people want, and which are robust and capable of amendments to cope with future changes.

SSADM consists of six stages:

1. Investigation of current system;
2. Specification of required system;
3. Selection of technical system;
4. Data design;
5. Process design;
6. Detailed physical design.

Each stage is broken down into steps – which are named and have unique numeric identifiers – and steps are further subdivided into tasks, which are identified by numbers within the appropriate step.

What SSADM Will Do

Many computer departments and development teams engaged in the production of a new computer system without the use of a standard development method, find themselves struggling with a communication problem. Communication between themselves, that is, let alone user departments! How do we communicate if not by the use of a common language? If the language used is one which allows description of a physical computer system using such documentation as program specification, system flowcharts, runcharts, program logic charts, etc., then little communication is possible before the proposed system is designed, often in great detail and perhaps even with some programs already written. At this time the die is very nearly cast when problems are encountered in the design, changes are often difficult and complicated.

SSADM provides the means by which to communicate knowledge and perceptions about the requirements right from the initial stages of system investigations when changes are relatively easy to cope with. To achieve this, the method utilises techniques, such as data flow diagrams (DFDs), logical data structuring (LDS) and entity life histories (ELHs), to provide pictorial views of systems which can be used as both development and discussion aids. (We examine these three techniques in detail in the following units.)

Since these techniques provide views of systems from different perspectives – DFDs providing the processing view, LDS the underlying logic of the data structure and ELHs the effect of time on the system – an amount of cross checking is available between the different views, invariably resulting in an increased understanding of the requirements. This understanding is further enhanced by involving users throughout development, the principle being that the common language resulting from the application of SSADM can be understood by non-computer people.

The statement of requirement which emerges from the use of SSADM, describes the required system in terms independent of any physical hardware and software constraints and characteristics, allowing the decisions about physical implementation to be made quite separately.

Stages 4 and 5 of SSADM are concerned with the production of a logical design which provides a basis for physical design and implementation, with the aim of fulfilling the known requirements and also providing a design which is amenable to change, enabling future needs to be incorporated with the minimum of effort and expense.

Conversion of this logical design into a physical equivalent consisting of program specifications and a set of files or a database design, takes place in Stage 6 of SSADM.

What SSADM Will Not Do

SSADM undoubtedly provides a sound procedural framework for systems development. Within this framework lies the application of some very useful techniques. What the method does not do, is replace the skills and expertise of the systems analyst and designer. Given these skills, the use of SSADM will almost certainly produce better systems – those which meet the current need and are able to cope with future needs. It is, therefore, a mistake to view the method, as some organisations have done, as a substitute for trained and experienced staff. To do so is to invite disaster in the form of projects which grossly overrun their planned time-scale and, in some cases, are never ending, i.e. abandoned.

Since one of the overriding principles of the method is that users should be involved throughout the development life-cycle, it follows that there must exist an atmosphere of co-operation and enthusiasm for the approach. SSADM cannot engender this. It must be created within the organisation, firstly by management acceptance and secondly by adequate training and education. Only in this way can the people involved gain an understanding of what can be achieved, their role in achieving it and the commitment to make it happen.

SSADM is a method of analysis and design. The final output, essentially, is a physical data design, program specifications and a plan for the programming and implementation phases. It does not produce coding nor does it cover the use of program design methods. It is not intended to! It also does not cover a whole range of activities associated with computer systems design and implementation, such as equipment procurement, site preparation, etc., which must take place in parallel to the SSADM tasks.

Impact on Project Time-Scales

It is often assumed that the use of the method will almost certainly extend the time-scale of the project being undertaken. There is no real evidence that this assumption is true. Since it is unlikely that information will be available as to how long a given project would have taken without the use of SSADM, as opposed to with it, then usually no comparison can be made.

What is evident, however, is that a more complete system encompassing more of the users' requirements, should emerge from the use of the method. What is also evident is that the taking of short cuts to implementation becomes more difficult since, using SSADM, the development of a physical system in terms of programs and files is likely to begin much later in the project life-cycle than would be the case without its use. This severely restricts the developers' opportunity for implementing something which satisfies only part of the requirement in the hope that the rest can be added later in that part of the life-cycle referred to previously as "maintenance".

Although SSADM is not a project control method, it is well suited to environments in which formal methods of controlling projects are used. The decomposition of the work involved in developing systems into specific stages, steps and tasks has two distinct advantages to the project controller – it provides a basis for estimating and, through the existence of formal products signifying completion to a certain point, a means of monitoring progress.

The method recommends the setting up of a **steering committee** on which should be represented the IT department, the management of every user department involved in the development, and corporate management. The latter is needed because of the likelihood that decisions will have to be made at this level and because of the financial implications arising from the cost of development. The exact composition of this committee will obviously vary from organisation to organisation.

Room for Improvement?

SSADM has been further defined over the years since its first implementation. It is probably the best defined method for developing computer systems and is generally recognised by the Central Computer & Telecommunications Agency (CCTA), and other users. Scope still exists for further improvement and, indeed, it is likely that the process of evolution will continue.

Criticism has been expressed at the starting point of SSADM, beginning with the first stage devoted to a detailed investigation of the current system and its problems. These criticisms are largely associated with the amount of time which this can consume. A consensus view is that this stage is vital to the success of the project and that there is no better place to start, although some guidance is needed on how to approach situations where multiple versions of current systems exist in perhaps separate geographical areas or outposts of the same organisation, when the objective is to produce one common solution. In a similar vein, the method does not make any reference to starting up projects where no current systems exist at all.

Whilst both these scenarios can be coped with given knowledge of the method, specific guidelines would be useful. Concern has also been expressed about the difficulties encountered in using some of the techniques, specifically Entity Life Histories and Logical Dialogue Design – the former because it is a complex and difficult technique anyway, and the latter because it is not particularly well defined.

Further difficulties are caused by the poor definitions of some key terminology in the method, e.g. “events”, “functions”, “operations” and “processes”. These examples, whilst not in themselves serious defects in the method, and which can be handled by the experienced practitioner, still prove rather daunting to the analyst embarking upon his or her first SSADM project.

Finally, although making a brave attempt in Stage 6, the method does not solve the problems associated with making the transition from a logical design to a physical computer system. It must be said, however, that attempts to generalise about this minefield of a topic almost always run into trouble because of the widely varying hardware and software features which abound from installation to installation.

There is no substitute for the designer’s knowledge, skill and awareness in this area.

Taking on SSADM

If you, as computing manager, believe that SSADM has a useful part to play in systems development within your corporate environment, then an action plan will be required. Here is a plan for the introduction of the method:

(a) Obtain Corporate Approval

To be successful requires commitment at all levels. Systems development using SSADM is no longer solely a computer department job; it demands user and management involvement.

(b) Pick a Pilot Project

Such a project might be one which can be started within, say, the forthcoming six months and can be finished within a year and has clear objectives. It is vital to achieve some quick success!

(c) Train Development Staff

Analysts and designers should receive full training in the method from a CCTA accredited training provider. You may wish to take advantage of the relatively new examination courses which lead to the nationally recognised Proficiency in SSADM Certificate as administered by the British Computer Society’s Systems Analysis Examination Board.

(d) Train Users

Users will require some training. As a minimum they should be briefed on the SSADM products with which they will come into frequent contact.

(e) Plan for Post-Training Support

Most development teams embarking upon their first project feel the need for some expert guidance from time to time. The concept of using some outside consultancy help should be given serious consideration. The resultant saving in time and improvement in quality will invariably justify the cost.

If the use of a structured method is accepted, then SSADM is a good way forward, as long as the commitment is not underestimated by all concerned.

B. WORKING WITH USERS

We have already seen how the SSADM technique requires user involvement and, in an earlier unit, we emphasised the value of user involvement at all stages of a project. As we have said, then, it is desirable for the potential users of the system being designed to be consulted from the very beginning. But what is meant, exactly, by “user involvement”? How do we involve the users?

Development Stage

The answer to these questions will depend upon the user's level of responsibility. We can identify users as senior management, line management and non-managerial staff. These are concerned to a different extent, and in a different manner, with line managers probably being the most involved during the design and implementation stages.

(a) Senior Management

Total organisational involvement is essential to the success of any system effort, and the participation of top managers is necessary in the decision-making process. The use of committees and special presentations may be effective at different times, but two-way communication is necessary. The DP department must know the overall requirements and objectives of the organisation it serves and to feel a part of this organisation.

The other aspect is that senior management may not be aware of the potential advantages of new or developed applications, and these should be brought to their attention, by analysts, to make management more sensitive to new approaches and concepts in DP.

It is part of the systems analyst's job to persuade people of the importance of (and benefits accruing from) a project, but this may not always be successful as the personal motivation of the staff will affect their attitude. The knowledge, however, that senior management is associated with the project raises the chances of its being accepted.

(b) Non-Managerial Staff

Important ways of involving non-management user staff include:

- providing them with an understanding of the DP department objectives;
- keeping them informed as to progress;
- consulting them constantly during fact-gathering and design stages;
- keeping in touch with user staff during the working-in of the new system, and working with them wherever possible;
- training them in the use of the new system;
- post-implementation consultation.

(c) Line Management

Because line managers are so affected by new systems, it is important that they give their support to the project. SIX main areas of involvement may help to achieve this.

- ***Sharing DP Objectives***

  Periodic discussions and occasional formal presentations will inform managers of (and involve them in) systems development plans.

- ***Appointment of Representatives***

  Line managers could be asked to appoint project team or liaison staff. For every systems project, irrespective of its size, it is sound policy to ask the department(s) affected to appoint some representative(s) to be the chief point of contact. The representatives' function would be to make sure that the needs of their department are made known and are being met, and generally to make themselves felt. They would also assist in gathering information and in implementing the system. Where the project is medium-size or large-size, the representative would be a full-time member of the project team.

- ***Report on Progress***

  Frank and honest reports should be made concerning the progress of the project, whether it is fully operational or still being developed. Line managers will thus not anticipate too much too soon.

- ***Assistance with Project***

  Managers concerned with a particular department will surely know more about its work than anyone else. The team itself can comprehend the workings of any department only with management assistance. Line managers can therefore help by:

  (i) Providing the team with any documentation, such as organisation charts, work flow diagrams, data flowcharts, procedure manuals, job descriptions, sample documents and forms, etc.

  (ii) Discussing with the team, problems which may have arisen in the existing system.

  (iii) Clarifying and explaining their department's work.

  (iv) Making arrangements for the team to interview more junior members of staff at appropriate and convenient times, to appreciate the detailed and finer points of the present system. (Managers should, of course, know about the shortcomings of their own departments and have ideas as to how these problems may be tackled – the feasibility study team will welcome discussions on these matters and possible solutions, and ideas will be exchanged before the proposed courses of action are set down in the feasibility report.)

  (v) Managers should also give considerable thought to the nature of the computer outputs provided by the new system – computers are good at making comparisons and at data analysis and they can provide exception reports instead of large volumes of routine data; management (not the computer specialists) must decide what an "exception" is, and what analyses are going to be of use to them.

- ***System Testing***

  We will look at some aspects of this later in this section.

- ***Operation of the System***

  The user involvement here, especially of line management, is fairly obvious. With modern distributed systems, it is the user department personnel who actually use the system, via keyboard terminals.

Operational Users

Whilst it will be management who are primarily consulted during a development project, once a system is up and running, the users of a modern system become the ordinary staff of the organisation, and others. We are then able to identify four different types of user who are able to access, and be involved in, a system. This, in turn, justifies their claim to be consulted during the development stage so that particular requirements can be satisfied.

(a) The Parametric User

The most common type of computer system user uses some kind of terminal. Such use is often routine and clearly defined, with predefined responses.

Typically, a telephone order or enquiry clerk has a screen and keyboard. He or she asks for specific details (e.g. the caller's customer number), keys this in via the keyboard and then continues to respond to requirements of the system. The screen will initially generate menus and eventually a single entry.

The clerk will probably be able to request further access via specified paths through the system.

Such a user is termed "parametric" as strict parameters of use must be followed.

Another example of a parametric user is the shop checkout operator who uses a computer terminal under very narrow pre-set parameters. The shop supervisor, who can override these parameters, is still a parametric user as the supervisor still uses a (different) set of parameters.

(b) The Casual User

At some time or another we are all casual users of a computer system. Such a user makes infrequent use and knows nothing of how the system works.

The only viable interface such a user has to the computer is screen menus, or more likely nowadays, a graphics/windows screen. These are just sophisticated menus with the user selecting a graphic screen symbol using a mouse device or a touch-screen device. In turn, further screen options are generated either for the whole screen or for parts of the screen, i.e. windows.

(c) The General User

The general user needs a fairly comprehensive knowledge of how to use the system. He or she does not need to know how the system is constructed or even which files are used, but does need to know how to access the system and manipulate the results.

A typical example of a general user is a manager accessing a management information system. He or she will use the system in an unstructured way, making unpredictable and *ad-hoc* access. The interface needed for this will be a query language and we shall discuss these in a later study unit.

Clearly the general user will need to be trained in the commands of the query language. In their use, such a user will himself or herself be driving the system, i.e. initiating action.

(d) The Programming User

This user has a fairly predictable role in using the system. The programming user needs access to the actual code and to test data, but not necessarily to live data.

It is important to be aware that none of the above users need to be given access to the actual computer room. This is a basic security precaution. However, all of them should be consulted in some way before the system is finally designed.

Walk-Throughs

Another development stage at which users become involved is testing the proposed design. Before the design is sent forward for programming, it is important to assess its ability to meet the original requirements.

At this stage, the design will be almost complete and will consist of the system design as a whole and the individual program designs. The programs will, in turn, consist of a series of self-contained modules, for which there will also be designs.

(a) Purpose

One of the main problems in systems design is that of **logical errors**. These arise from a fault in the flow of a program or in a control structure within a program. Typically, wrong selections are set or a loop is executed the wrong number of times. A good way to eliminate at least some of these errors is to perform a “dry run” or “walk-through” of the program design. This will involve assuming some initial data and then following its processing through the design steps as a paper exercise.

It is generally advised that this exercise be performed by someone other than the designer, or at least in the presence of someone else, since the designer will be influenced by his or her expectation of the program design’s behaviour and may not see obvious faults. Ideally, a knowledgeable user could make the walk-through. Such a walk-through can be applied to the original algorithm in the design language or even to the original structure charts.

It is much easier to analyse the design in this way if it uses structured techniques. These techniques essentially involve subdividing a design into separate components, which may then be independently called by the main program design. This means there are many self-contained pieces of design that can be easily analysed, as each will do one job only.

In fact, walk-throughs can be conducted at any stage of the design, although the principal structured walk-through will be made on the completion. As well as the knowledgeable user, or peer designer with the user, there should be representatives of the project manager, quality control section, internal auditors, and others as appropriate.

The structured walk-through review process is not a problem solving session. It is a detached review of the design work to date. Any problems found should be referred back to the designer.

(b) Value

The main benefits of a walk-through are:

- It brings a fresh check on the effectiveness of the design;
- It develops trust in the design;
- It reassures all the parties affected by the design;
- It ensures the participation and approval of all the parties affected by the design;
- It encourages open communication between all those involved.

(c) Hand-checking

It is impossible to overemphasise the fact that successful programming is based on willingness and thoroughness in hand-checking. Each stage from design to program should be hand-checked. All are before valuable computer time is spent in looking for program faults. The cost of computer processing time is such that a moment of program failure is far more expensive than, perhaps, an hour of a designer’s time. Silly design errors are therefore very costly in computer development time and should be totally eradicated by good hand-check walk-throughs.

Hand-checking involves the following steps:

- Examining the program specification in detail and ensuring that the objectives are being achieved.
- Ensuring that the program instructions agree with the program flowchart.
- Checking that individual program instructions are correct. Remember that a programming language requires 100% precision in the use of characters, words and other symbols.

- Preparing test data and working through the program, instruction by instruction, to ensure that the test data is processed correctly. It may take a programmer several hours to hand-check what a computer can do in minutes. However, a computer may stop after one program error and need several runs before all errors are corrected. A programmer should find all these errors in one detailed examination.

The use of modular construction allows each independent part of the system to be tested before it is integrated into the rest of the system. This principle can and should be extended to smaller components such as individual procedures, where these are relatively self-contained. If, however, they depend on the presence of a complex data structure, it is more practical to test the parts of the system that build the data structure before testing the parts that access it. The system can, in fact, be built up in stages, each stage adding some more facilities to the ones that are already (partially) tested. The degree to which completely independent testing of modules is possible depends on the particular design, and it is impossible to generalise. As in so much design work, common sense and experience are both valuable in choosing the correct course.

C. RAPID APPLICATION DEVELOPMENT (RAD)

Until fairly recently, all project development work was a lengthy and very slow process. The development life cycle dictated a highly structured, standardised approach in order to cover each element precisely. Over the last twenty years, though, the cycle has been broken down by the application of new methods and techniques which may be grouped together under the general title of Rapid Application Development (RAD).

There are four principal approaches within RAD:

- Joint Application Development;
- Skilled small team development, or SWAT (Skilled With Advanced Tools) teams;
- Prototyping;
- Computer Aided Software Engineering (CASE).

We shall study these approaches in the next sections of this unit. However, it is important to remain aware that these are just aids to development. In themselves, these tools do not provide designs, code, etc., other than what the developer decides.

Underlying all RAD approaches is the active participation of the user in development. The user should be involved in all structured development techniques. However, SSADM, for example, only requires the user's approval of the separate stages; and sometimes to act as a reference point to clear up requirement ambiguities. On the other hand, RAD requires the user to be part of the actual development team and not just an approval/reference point. It is this involvement which enhances the speed of development.

D. JOINT APPLICATION DEVELOPMENT (JAD)

Common system development is the development of a system that is going to run on more than one computer (even a different manufacturer's hardware) and possibly for more than one organisation. During such development, several user departments, or companies, will be involved. Because of the complexity of such development, it is even more important than normal for all the users to be consulted and to be aware of every stage of the project.

Methods of Development

JAD is a technique particularly applicable to the analysis stage of the life cycle.

It usually involves a series of intensive workshops between the users and the developers and it replaces the conventional interview/questionnaire fact-finding activities. During the workshops the exact requirements are identified by those who have the authority to agree them. The development staff, who will also be expert in business applications, will help with technical design matters which arise and then record the agreed requirements in a CASE software tool.

Following the JAD sessions the development staff will prepare detailed models of the proposed system using the agreed requirements.

Several approaches are possible.

(a) Customer Specification

A customer specification can first be derived and then used as the minimum requirement for each organisation to develop its own system. This approach implies that each organisation or part of the organisation that requires the system has its own complete DP set-up for the development and operational running of the system. At first sight, this might seem to be very wasteful of development resources. However, if the system involves interfacing into a number of other systems which are local to that organisation, this could be a more attractive approach.

(b) Program Specifications

Some systems design effort can be saved by designing the system in one location and then using the program specifications as the basis for achieving commonality between systems. This approach, like the previous one, relies on having development staff at all locations. It certainly overcomes the problems of language as it would be possible for the different locations to use their own high-level language.

(c) High-Level Language Code

Another approach is to have a common high-level language code. This is probably best achieved by developing just the central core of the code using an agreed subset of the language that is acceptable to all machines, and letting each site develop the input/output part of the code. This achieves a good commonality without having to cater for many different interface problems. Also, as with the previous approaches, it relies on development staff being available at the separate sites. This certainly makes maintenance and operational troubleshooting easier.

(d) Low-Level Compiled Code, or Absolute Code

As a final illustration of an approach, the low-level compiled code or the absolute code can be produced centrally and distributed to the individual sites. This certainly gives the tightest control to the central organisation, and ensures that each site is using exactly the same system. It also means that the central site has suitable facilities to compile into the individual machine codes, and will subsequently have to cater for all the modification requirements.

Flexibility

Systems development for several common situations must have some in-built flexibility. The designer will take the differences between the systems into account.

Typical differences are:

- Common terms are notorious in that they are not "common"; hence "despatch date", "days of credit", etc. can vary just sufficiently from one organisation to another to require parameter definition.
- Policy data will require changing, for organisations will differ in what defines a debt, for example, and this is associated with the policy of running the business.
- Different users will require slightly different reports from a system, in terms of frequency of production, depth of detail and order of data in the report.
- Different media are used for data capture, printing, processing speeds, etc.

E. SKILLED SMALL TEAM DEVELOPMENT

Skilled small team development or, as it is often known, Skilled With Advanced Tools (SWAT) teams describes the development team in a RAD environment.

SWAT teams should ideally comprise about four to six highly motivated people with complementary roles in the development process. They need to be skilled in the various techniques associated with rapid application development, and are most effective in the following situations:

- On small projects of limited scope and impact, although larger projects can be divided into smaller sections to make the JAD approach more effective;
- On projects not requiring new hardware or software;
- When CASE (or 4GL) tools are available.

F. PROTOTYPING

The models which are prepared following either detailed structured methods or the JAD sessions are essentially pictorial representations of the system. They may be used to build a prototype, or initial working version of the system.

Prototyping is a useful development tool. It is a screen-based development tool, the two main features of which are:

(a) Users should be involved in development.

(b) Users do not know what they want until they see it.

The concept is that we view data as an important entity in its own right. So the first consideration is the data. Alongside this we consider the type of structure that will hold the data. By manipulation of a screen interface, the data and the data structures are fashioned to particular requirements.

Techniques and Usage

Prototyping uses a special generator package. The construction is rapid, enabling the generator to go through many interactions until the user sees a screen layout and dialogue which suits him or her. This will, in fact, represent a data structure and data definition.

The technique involves sitting the user in front of a screen and, working closely with the systems designer, evolving a screen interface satisfactory to the user.

The systems designer then goes away and develops the real system based on the prototype, using other development tools, and in particular the graphics package. Occasionally the prototype is itself adopted as the real system, but this is normally avoided because the prototype is generally inefficient in its use of computer resources and, usually it will not contain all the features required by the user.

The prototyping technique has come a long way from its early days. Originally it was used for fairly simple systems and to develop straightforward screen formats. Later it was realised that by getting the screen correct, the designer had, in fact, developed an interface with the system itself. This made prototyping a powerful tool as, with such a large user involvement, and bringing the system to a point where it responds directly to the user, the designer was able to meet directly the requirements specification as perceived by the user.

The designer is often keen to exploit this advantage and to use prototyping in even more complex project developments. Not only that, but by using prototyping, and its necessary user involvement, at every stage of development, the designer is able to enhance the project management and aim for a "getting it right first time" approach, or as it is nowadays labelled, the "**total quality approach**".

Types of Prototyping

There are two types of prototype, depending upon the stage at which the technique is first used.

- If the prototype is used to define further the user requirements, the resulting whole system will be coded, maybe even in a different language to that used for the prototype. This then replaces the prototype, which is known as a "throw away prototype".
- The second type of prototype is that favoured in a RAD environment. The prototype is repeatedly refined until the final system evolves. This is "**evolutionary or incremental prototyping**".

  This means, in effect, laying a prototype interface for testing on to a prototype interface design which, in turn, was laid on a prototype of the data structures. In other words, the prototype for each stage is further developed to the next stage, giving evolutionary prototyping. Whilst this seems very much common sense, it does depend on the availability of the supporting development software. The designer is thus using certain software tools, for example a data dictionary, to enable him or her to use another development tool, prototyping. This is another way of viewing evolutionary prototyping, as it means using tools in certain particular ways to bring development closer to the user.

Advantages and Disadvantages

Prototyping has many benefits:

- Most of all, it satisfies users, especially if it grows out of JAD sessions, as users are fully involved;
- It is quicker. It has only very necessary documentation and training occurs during the prototyping sessions;
- It produces a working solution;
- It eases communication problems between users and development staff.

It does, though, have disadvantages:

- They are difficult to handle at first because of development staff inexperienced in the technique, their lack of control during prototyping sessions and the probable lack of reasonable prior planning;

- It is difficult to prototype large systems because of their complexity;
- Users usually want to use the prototype immediately without the necessary background support work.

G. COMPUTER AIDED SOFTWARE ENGINEERING (CASE)

The most useful software tools during evolutionary prototyping are Computer Aided Software Engineering tools, as the evolutionary process will be based on the CASE models, not the actual code.

Computer staff have been developing systems to help production controllers, accountants, personnel officers, etc., for the last 30 years. Yet it has only been in recent years that attention has focused on using computer systems to help **all** systems development staff. Programmers have been well supported for many years, with editors, compilers, linkage editors, loaders and run-time systems, but what about project managers, systems analysts, systems designers, program designers? Various individual pieces of software have been produced over the years, e.g. to assist in project planning or in database sizing, which help out on a piecemeal basis.

CASE Tools

CASE provides a range of tools which can be combined in various ways to meet the particular requirements of the development project. The minimum necessary components will be:

- A graphical tool to enable the analysis and design models to be displayed and manipulated on screen;
- A data dictionary to hold common data definitions and hence avoid repeating this task;
- An application generator to produce program code;
- Management tools to aid the monitoring of the project and to produce the necessary documentation.

Other facilities which can be provided by CASE tools are:

- Maintaining and verifying consistency during development;
- Prototyping directly with the user to produce acceptable screen interfaces.

Use of CASE

The benefits achieved in using CASE certainly outweigh the upheaval and cost of its introduction. System development is much more streamlined and standardised due to the integration of all the development activities. With the increasing complexity of systems and of new applications, organisations are beginning to realise that integration of all the methods used is essential.

CASE tools are of most use when linked to a development method. In doing this, the existing method must be reassessed to accommodate CASE, with consequent retraining of technical staff.

CASE tools are particularly useful in keeping the compatibility between the models and the code. This accounts for their usefulness in prototyping.

The one drawback, at present, is that the standardisation required throughout the industry is not yet in place. As CASE operates through a variety of environments, from PCs to mainframes, it cannot be fully adopted until full standardisation is agreed.

Finally, a word of caution. There is nothing very fanciful about CASE. Normally, it is just a set of documentation tools which allow the designer to manipulate development diagrams on a screen and to store design definitions of the components of the system being built.

Thus, you will readily see that CASE, sets of documentation tools, whatever we call them, are simply software packages. They run on stand-alone personal computers with graphics facilities. Sometimes they run on work-stations linked in to a project database held on a supermicro.

Fourth Generation Languages (4GLs)

Unfortunately, there is no strict definition of the term "fourth generation" – it is primarily used in sales literature and has been applied to a wide range of software products offering appreciably different facilities. In general, however, products advertised as being fourth generation enable business computer systems to be constructed much more quickly than earlier generation products, such as assembly language and COBOL.

You may have been involved in setting up a number of business computer systems using third generation language. If so, you will have noticed that the same basic types of processing occur time and time again, e.g. validation of new data entered into the system, insertion/updating/deletion of records, production of reports with headings and totals. The designers of fourth generation products have identified what they consider to be the basic types of processing, and have built these into their "language" from day one in order to speed up the construction of application systems. The architecture and use of a typical fourth generation product is shown in Figure 6.1.

4GLs should provide a complete life-cycle development tool, from the beginning to the end of the project.

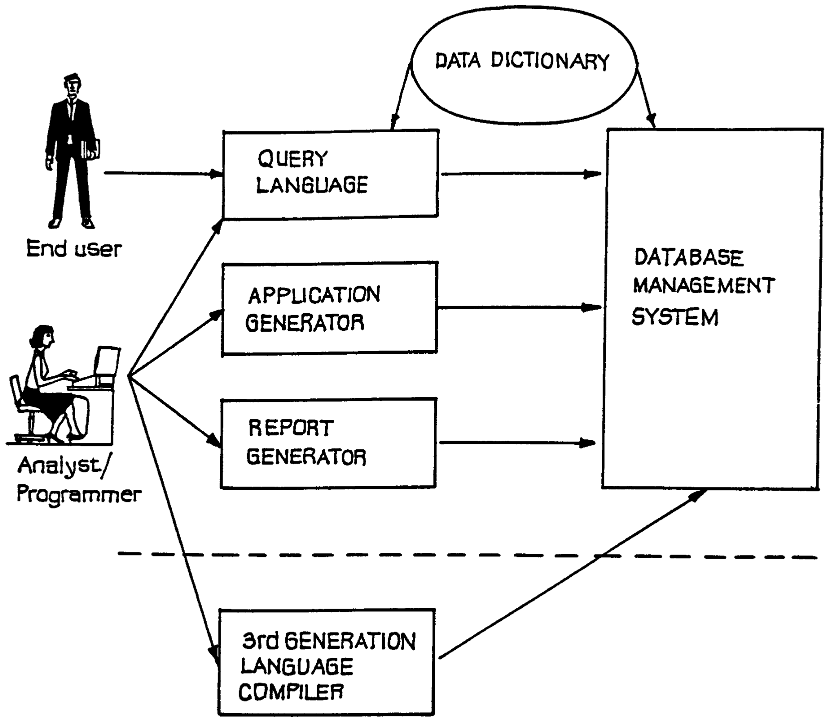


Figure 6.1

The facilities above the dotted line in Figure 6.1 may be used where the processing involved in an actual application system fits into one of the defined types supported by the product being used.

Many application systems, however, will have a proportion of processing that does **not** fit, e.g. the production of tape files in a special format for another computer system, and this proportion will need to be handled using third generation tools. Note also that some fourth generation facilities are designed for end users, whereas other facilities can only be realistically used by professional system development staff.

Typical features offered by 4GLs are:

- File definition and maintenance procedures.
- Screen-based input and validation, allowing users to specify the screen format and validation approach.
- A query language which can access the data via simple, albeit structured statements.
- A report generator which will produce reports when given certain parameters.

At the centre of a 4GL is the data dictionary. This is just a database full of relevant design facts. Any amendments required can be made to the data dictionary and will then be reflected throughout the 4GL system.

As we have said, different 4GLs have different features and will help the system designer in different ways. For instance, some are heavily database-orientated and have a query language, report writer, screen handler and graphics. Others are form-orientated and produce screen forms that are to be filled in. When backed up by a procedural language, this type is useful for inputting system requirements.

Yet others are basically procedural languages in a new high-level style. This just means normal procedural languages such as COBOL, FORTRAN, etc., but supported with the main requirements of validation, data structure building and so on, all built in. They are therefore much more powerful design tools.

Study Unit 7

Data Flow Diagrams

Contents *Page*

INTRODUCTION

Communication and information lines are vitally important to the smooth running of any organisation. Where documents are passed through several departments or sections, the flow of information cannot be easily described by narrative alone and are best shown through the use of different types of diagram. The most commonly used diagram is the **data flow diagram**.

A. SYSTEM DEFINITION

Before we look at data flow diagrams, it is useful to understand what a system is, and define such a notion.

A **system** has: **-**

- An **input**
- An **output**
- A **process** that transforms the input into the output
- The process is itself a system and consists of one or more **sub-systems**, each of which are themselves systems. The sub-systems act in a co-ordinated way, so that the output of one sub-system is the input of another and they all act towards the common function of the overall system process.
- A **boundary** which is the limit of the influence of the process
- An **environment** which is everything beyond the boundary. The input source and output destination lie in the environment

As an example we can think of a college system.

- The **input** is the learning conveyed by tutors and the output is the learning received by students.
- The **process** is the total functioning within the college, including all the lectures, the administration and the heating and other building and catering functions.
- The **process** system consists of innumerable subsystems such as the programme of lectures, the employment of lecturers, the ancillary staff, the kitchens, the janitorial functions, and so on.
- The **boundary** is the limit of everything done by the college. This may be the courses themselves, but if the college does outside catering or puts on a theatre programme, then the boundary is defined wider.
- The **environment** is the city or town in which the college exists, and the educational system of which the college is merely a subsystem.

B. STRUCTURED SYSTEMS ANALYSIS AND DESIGN METHODOLOGY (SSADM)

We always need to decide which development method is to be used for a particular system development. This will usually depend on the way the company sees itself and on the skills of the analyst. In most cases, the method called **SSADM (Structured Systems Analysis and Design Methodology)** will be the choice. SSADM is a widely used analysis and design methodology. It is also the methodology that uses the techniques of analysis that we are going to study.

There are six stages that generally follow a fixed series of steps that broadly follow the development cycle of a project. However, there is a strong emphasis in the involvement of the commissioning user in the methodology.

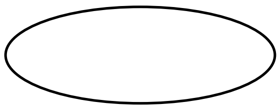
The six steps are: -

Stage	Action	Comment
Stage 1	Analyse the existing situation to provide current data flows and entity-relationship models.	Stages 1 and 2 are the analysis stages.
Stage 2	Produce a requirement specification with proposed data flow diagrams and entity-relationship models. It is at this stage that WHAT the system is to do is separated from HOW it is to do it.	
Stage 3	Involve the user in choosing the alternative which best suits their needs. This stage is most important, as, at the end of the day, it is the user who best knows what is wanted.	Stage 3 brings the commissioning user into the development so that the user can approve what is proposed.
Stage 4	Produce a complete design of the proposed system. Whereas Stage 3 says WHAT the system is to do, Stage 4 says HOW this is to be achieved.	Stages 4, 5 and 6 take the developer through the design and implementation stages of the system project.
Stage 5	Break the design down into detailed parts in order to evolve the structure required.	
Stage 6	Provide the program specifications and file/database definitions.	

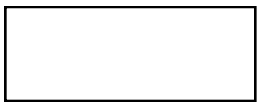
The main principle behind analysis, design and implementation is that the whole system is divided into separate modules, each module generally covering only one activity within the system. Many modules are linked together to make up the whole system. This ensures the integrity of the development as the modules can be tested individually. It also helps any subsequent maintenance or update of the system, following implementation, as individual modules can be identified, accessed and changed without undue effect on the system as a whole. In addition, the SSADM methodology separates WHAT the system is expected to do from HOW it is to do it. This is a fundamental point which ensures a logical view of the system is taken at first, and on when that is satisfactory are the physical aspects considered.

C. DATA FLOW DIAGRAMS (DFDS)

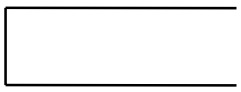
Data flow diagrams are used to represent the system being analysed. The diagrams use four basic elements, the styles of symbol used are not intended to imply any physical representation.



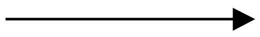
External **entities** are things which exist outside the system and which send or receive messages from the system. These may be users or other systems. By definition these are the source and destination of the information flow that is being modelled.



Processes, any activity that alters data in any way.



Data stores correspond to master files of data whether they are manual, computerised or a database, or indeed any unit of stored data.



Data flows are the pipelines along which data moves between the other components of the system.

Although the style of these symbols does vary in different publications, the meaning of the diagram should be clear to you.

As with other graphic techniques, there is a limit to the amount of detail that can be shown while retaining the essential clarity of the diagram. It is, therefore, necessary that every symbol be given a name that can then be looked up in the documentation to obtain more details. The simplest situation is where data flow diagrams and data dictionaries are used together with corresponding entries in the dictionary for each process, store or flow. However, data dictionaries are not essential and data flow diagrams can cross-refer to any type of documentation. (N.B. A data dictionary is a computerised reference point containing all the details of a system.)

There are several methods by which data flow diagrams can be constructed, each corresponding to a system design method. One large multinational uses a bottom-up approach, starting with a flow to or from a user (external entity) and progressing through the whole system at this detailed level. Flow diagrams are a popular technique; they emphasise the use of a top-down approach, starting with the most global/high-level functions and the major data flows in and out of these. A diagram with a small number of boxes is constructed to represent the top level. This is then expanded to produce a more detailed diagram, and so on. A halt is called when each process box corresponds to a single task or module, which can be clearly and simply defined. They are a very useful and flexible charting technique.

Example 1

The following example illustrates the use of data flow diagrams to show the progressive development of detail within a system.

In figure 7.1, there are only three boxes; these show that the basic function of the SALES LEDGER SYSTEM is to process the INVOICES and PAYMENTS. This is a **level 0 diagram**, also called a **context diagram**. It has only one process.

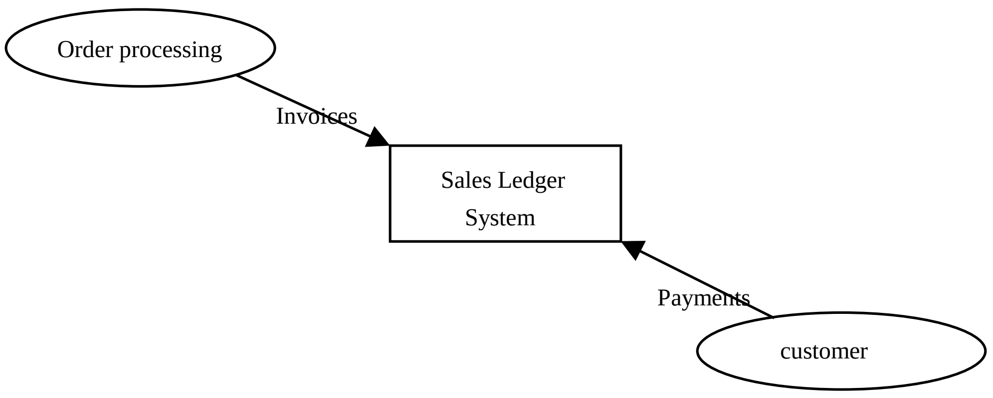


Figure 7.1: Context Diagram

The second version (figure 7.2) describes the Sales Ledger System in more detail, showing that apart from accounting for Invoices and Payments, a very important function is sending Statements to Customers. The system itself is seen as consisting primarily of three processes and one data store. This is a **level 1 diagram** and it identifies the individual processes.

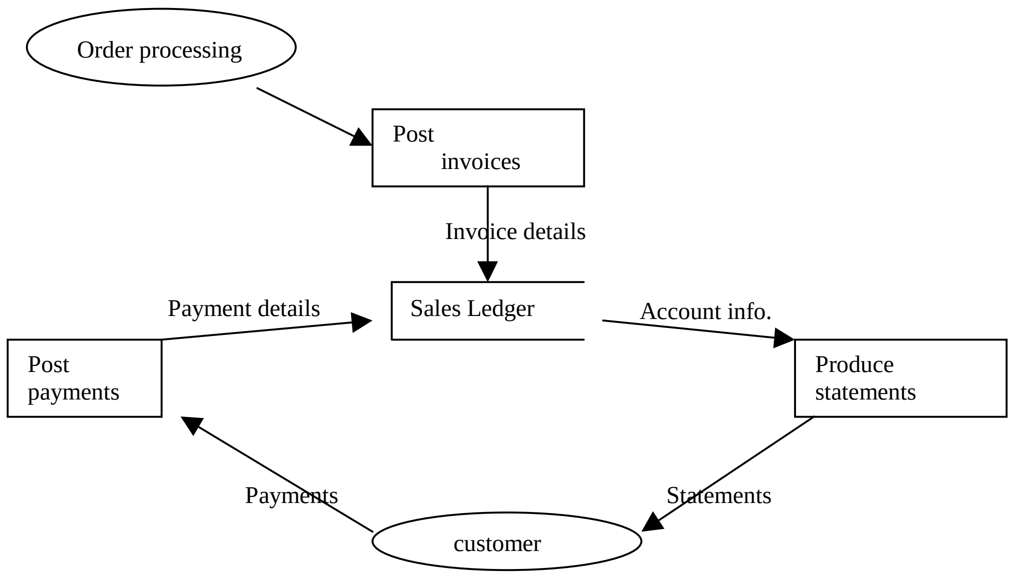


Figure 7.2: Process Diagram

A **level 2 diagram** will show the sublevels of each of the processes. We will not take the above problem any further as it is the context diagram and the level 1 diagram that are of most interest to you on this course. For more understanding, we will consider another example.

Example 2

In this example we are considering an ordering system, and in particular entering an order into a system.

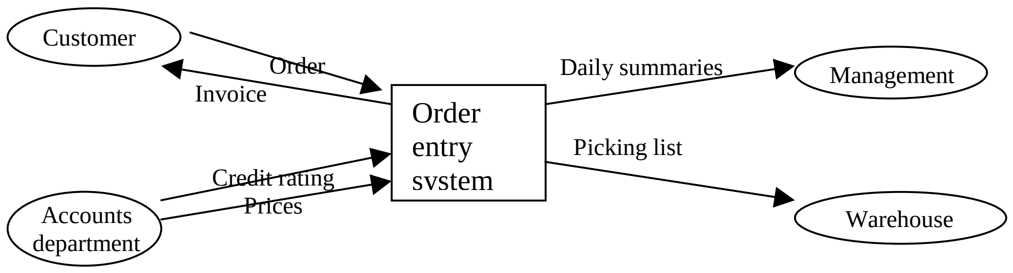


Figure 7.3: A context or level 0 diagram.

At level 0 there is only one process, which is the actual system being modelled (figure 7.3). This process is expanded at level 1 (figure 7.4) to show the component processes of the system:

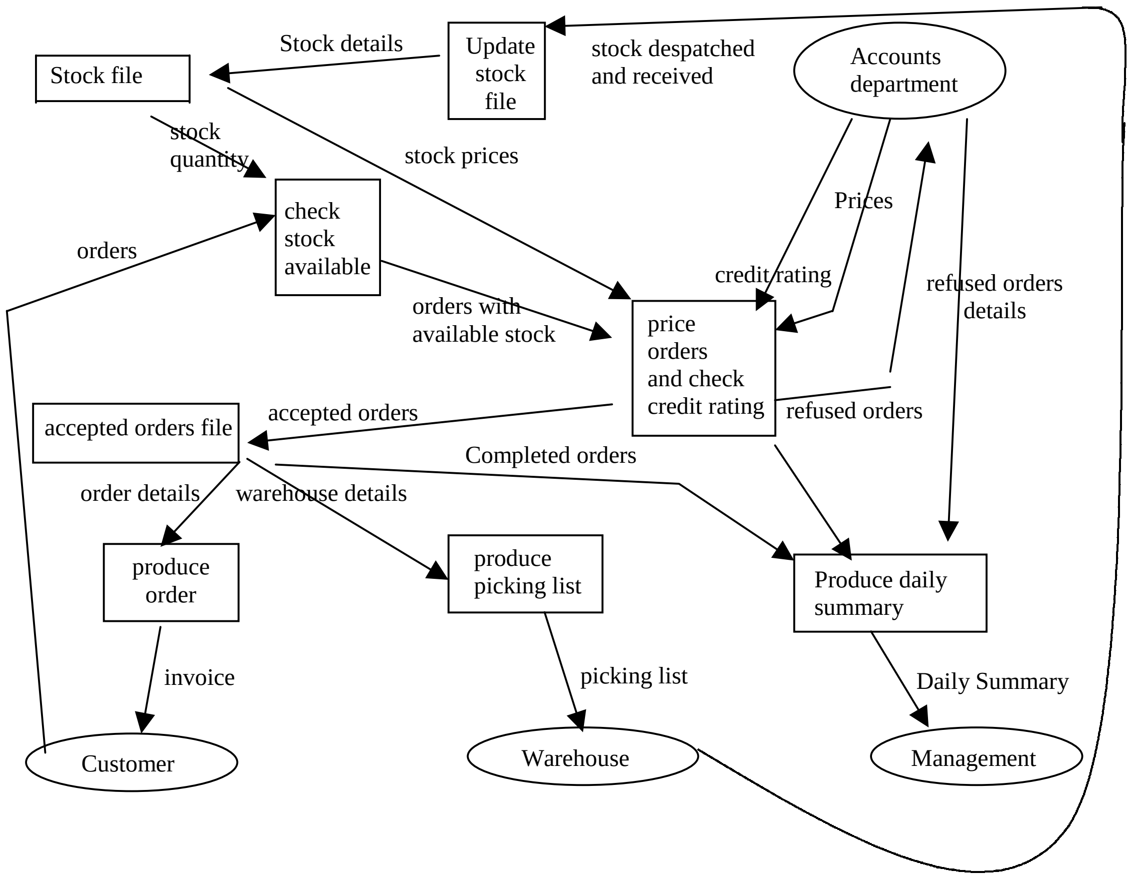


Figure 7.4: Level 1 Diagram

Next, we will develop the level 2 DFD for the ‘Produce Order’ process.

The following DFD is the level 2 expansion of the ‘Produce order’ process from the previous level 1 DFD showing the individual processes involved within. As you will see, raising an order is not straightforward, as the stock availability must be continually checked. Also, any outstanding amounts owed must be taken into account and rebilled for. The outer box identifies those processes that are sub-processes of the ‘Produce order’ level 1 process. Everything outside of the outer box clarifies the interaction of this process and the complete DFD model.

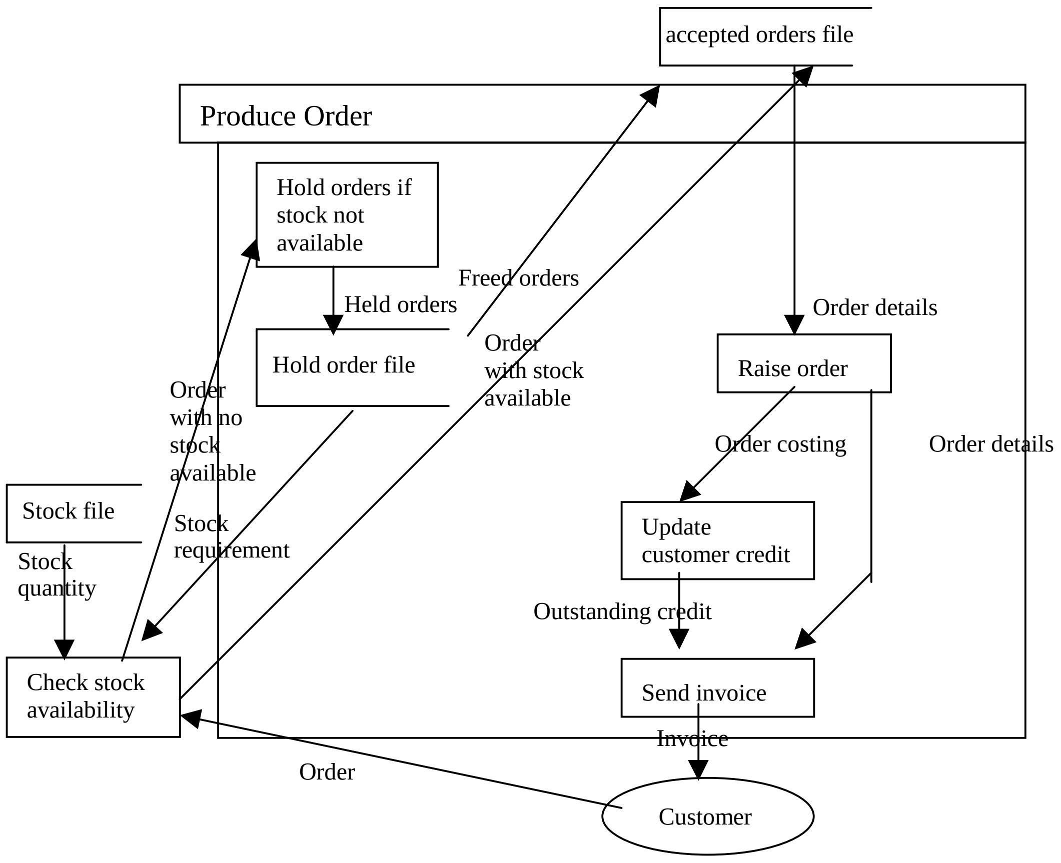


Figure 7.5: Level 2 Diagram

D. DRAWING DATA FLOW DIAGRAMS

The important point to bear in mind is that we are still at the analysis stage of the system development and so are solely concerned with WHAT we require of the system. HOW this will be achieved is left until the next stage of the development, the design stage.

1. Start by identifying all the source and destination **entities** of the system to be modelled. This is unlikely to be the total system under development as it is normally necessary to partition the whole development into manageable parts which can then be separately developed. (See Step 7 below). It is one of these parts that the DFD will represent. As we have seen in our definition of a system, the sources and destinations are not part of the system as the system is what happens as the information flows from the sources to the destinations. The **nouns** and **noun phrases** in the scenario description give a useful guide and identify the entities and the data stores.

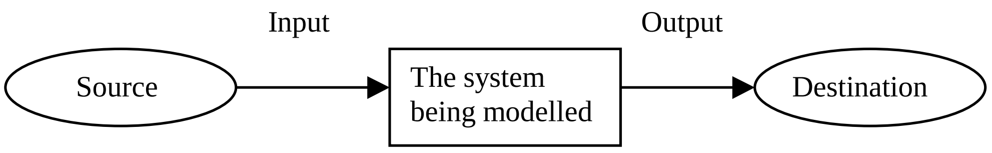


Figure 7.6

2. In the style of the above standardised DFD, draw the appropriate context, or level 1, diagram.
3. Refine the context diagram by identifying all the **processes** within the main system process whilst at the same time maintaining all the input and output flows given in the context diagram. Again the **verbs** in the scenario description tend to define the processes.
4. Once all the processes are identified, begin at a source and decide which process is to receive that input. Then decide where the output of that process is to go. By repeating this exercise for all the input and output processes, a full data flow chain is produced, and this is the level 1 DFD.
5. Once step 4 is completed, take each process in turn, and repeat step 4 for each of these, drawing a separate diagram for each. These are the level 2 DFDs.
6. In summary, there will be one context diagram and one level 1 diagram for each of the partitioned parts of the whole development project. There will be several level 2 diagrams for each of these partitioned parts.
7. A good rule is to keep the number of processes in any one diagram to no more than seven. This is the reason for the original partitioning and the identification of the level 1 processes.
8. Never connect a source directly with a destination

General Rules

- Within the scenario description, only the essential words and phrases need to be modelled. Synonyms should always be rejected.
- Words like ‘detect’, ‘use’, ‘accept’, ‘retrieve’ imply input.
- Words like ‘written to’, ‘display’, ‘print’, ‘operate on’, ‘update’ and such like imply output.
- The user and the computer media used are not modelled as these are beyond the sources and destinations.
- In the context diagram, work left to right with input to output.
- At levels 1 and 2, ensure there is a balance in the data flows. This generally means that there should be appropriate input to a process before there can be a required output. In our Example 2, level 1 DFD, the process ‘Update stock file’ has an input of stock receives and used which is then processed and sent onto the file as formatted details. On the other hand, this does not mean that every output must have a corresponding input. Sometimes more than one output is produced, sometimes more than one input is required to give the output. And the inputs and outputs from the context diagram are just used as presented.

E. ADVANTAGES AND DISADVANTAGES OF DFDS

Advantages

- DFDs are used to specify the functions of a system..
- They highlight the data flows through a system.
- They also identify the components of a system.
- It has very simple notation.
- It is easy to learn.
- The nouns and verbs of the system description translate directly into the diagram.

Disadvantages

- DFDs soon become cluttered with too much detail.
- They can be difficult to read if not clearly set out.
- They can be difficult to correct or maintain without redrawing the whole diagram. The normal way is to use complementary DFDs. These are 'second editions' of the original.
- There are no control elements in the diagrams such as 'does a process use all its inputs?'
- The notation is not standardised and so must be defined separately each time.

F. WORKED EXAMPLES

(Attempt the exercise then refer to the answer which follows)

The task is to draw a context diagram and level 1 DFD for tuition part of our college example. The scenario is that students enrol for a course and this information is held on file. The student receives tuition and is examined for each course. The tuition and exam are given and set by the college staff.

Suggested Answer

The following are just suggested answers for these two diagrams. Your answer may look different but be quite correct. The important points are the entities and the processes, not the actual names used.

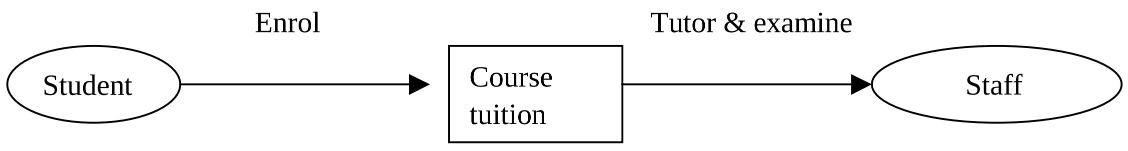


Figure 7.7: Context Diagram

The exercise scenario has the nouns 'student', 'course' and 'staff'. As the courses are totally within the system, they have not been identified as external entities. The verbs in the scenario suggest 'enrol', 'tutor' and 'examine' data flows.

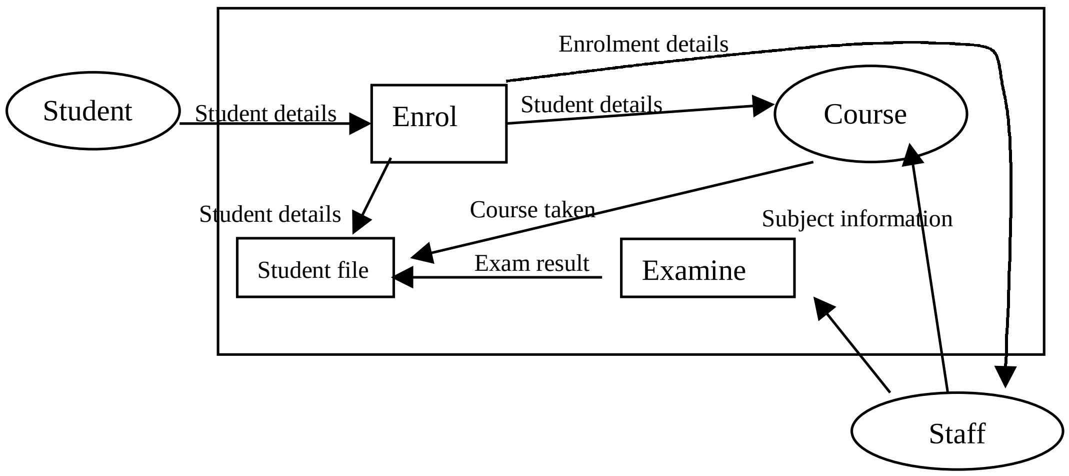


Figure 7.8: Level 1 DFD

From the context diagram (figure 7.7), *Student* and *Staff* are source entities and *Staff* is also a **destination entity**.

Enrol and *Examine* are the two processes identified. *Enrol* passes the student details to the internal Course entity, the enrolment details to Staff entity and the student details to the student file. The student file store holds student details, including the courses they have taken and the exam results obtained.

Study Unit 8

Data Modelling

Contents ***Page***

INTRODUCTION

In this Study Unit, you will be studying:-

1. **Entity modelling** which is a form of data modelling where we examine a scenario and decompose it into a formal structured diagram. This is then used as the base document for the design process that follows. At this point, with the DFDs and the entity-relationship models, we can understand the existing situation and what is required in the new system.
2. The use of a **data dictionary** as a computerised modelling tool. Modelling was originally a paper and pencil exercise, and still is in many cases. But nowadays there are a variety of computerised tools available, of which the data dictionary is one.

   As you will recollect from the previous Study Unit, SSADM methodology makes extensive use of DFDs and entity-relationship models.

A. ENTITY-RELATIONSHIP MODELS (OR LOGICAL DATA STRUCTURES)

Entity-relationship (**E-R**)modelling follows the production of DFDs and the models act as the bridge from the analysis stage to the design stage of the development. They define the data that is held by the system and they should reflect the real world situation of the business. The E-R model defines the way in which data entities are related and we should clarify what is meant by 'entities'.

In DFD models, entities are the sources and destinations of the data. In E-R models, entities are simply concrete aspects of the real world. So there is a little bit of difference in definition between the two modelling techniques. On the other hand, the similarity lies in the fact that entities are the nouns of the scenario.

As we stated in the previous study unit, SSADM separates the way in which items are physically stored on file from the way in which we view them logically in our models. The logical view of the situation is depicted in the entity-relationship model. This model evolves by taking a top-down view of the situation and identifying items and the relationships between them.

- The items are known as **entities** and are things of the system that we wish to collect and store data about.
- The entities are described by attributes or data items. These are similar to adjectives. For example, a car entity can be described, and has attributes such as, the colour red, an engine sizeof 1500, a saloon type, automatic for its gear system, and so on. We model entities as rectangles.
- Relationships between entities are expressed in the verbs of the scenario. For example, in the scenario '*the driver drives the car*', there are two entities, '*driver*' and '*car*'. A relationship between these entities '*drives*'. We model relationships as lines between entities.

Figure 8.1 shows a simple entity-relationship model of this same scenario: -

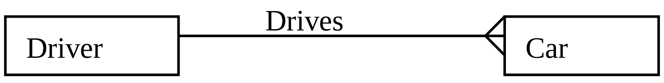


Figure 8.1

This model tells us more than the short scenario. The 'crows feet' at one end of the relationship means '**many'**. The lack of a 'crows feet' at the other end means '**one'**. Therefore, the model can be read as: -

A driver drives many cars.

A car is driven by one driver

This is called a 1 to many, or 1:M **degree** relationship, where M = many.

We read the relationship in both directions.

Other variations are possible, depending on the actual scenario (see figures 8.2, 8.3 and 8.4).

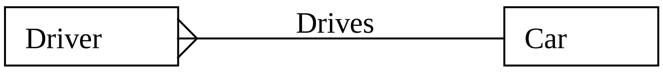


Figure 8.2

A driver drives one car.

A car is driven by many drivers

This is called a many to 1, but, as before, it is still has 1:M degree.

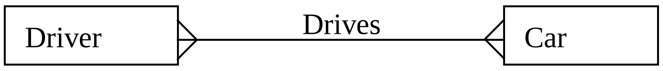


Figure 8.3

A driver drives many cars.

A car is driven by many drivers.

This is called a many to many, or M: M degree.

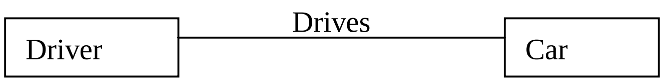


Figure 8.4

A driver drives one car.

A car is driven by one driver.

This is called a 1 to 1 or 1: 1 degree.

We also need to note that 'one' = 0 or 1, and 'many' = 0,1 or many. In other words, the one' and the 'many' are maximum values.

The E-R model diagrams **do not include the attributes**. These are held separately as a list. An attribute list for the above models could be: -

- Driver : **Name**, Address, Age, Experience
- Car : **Manufacturer**, Car Model, Colour, Engine size, Mileage covered

In each case, one of the attributes is in bold, showing that it has a unique value and can be used to help identify a particular record. These attributes are known as the **identifier** attributes.

An E-R model consists of both the E-R diagram and the attribute list.

Cautionary Points

1. We have already referred to entities and relationships. However, we must keep in mind that when we draw a 'Car' entity, this is representative of ALL cars. The proper name is therefore a 'Car' **entity type**. It does not stipulate any specific car. A specific car would be an **occurrence** of the 'Car' entity.

2. When drawing E-R diagrams, we use 1:1 degree relationships with caution as very often, one of the entities is simply an attribute of the other.

   For example, we can consider the following E-R diagram (figure 8.5).

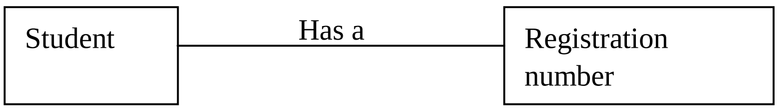


Figure 8.5

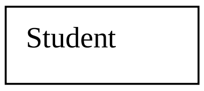
   With attribute lists:

   Student : **Name**, **Address**, Course-Taken, Age

   Registration-Number : **Value**

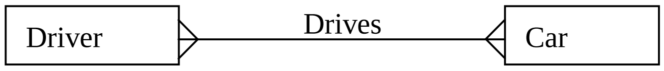
   (As an extra point, 'Student' entity has a composite of identifier attributes as neither Name nor address on their own will be unique and so cannot uniquely identify a Student entity, but taken together, they will be unique in all but exceptional cases.

   Now, returning to our cautionary point, we know very little about the registration-Number other than it has a value. There are no other attributes in the attribute list. We also note that the relationship called Has-A is 1: 1 degree. This just means there is a one-to-one correspondence between the entities. Putting these two points together we can draw the E-R diagram more succinctly as: -

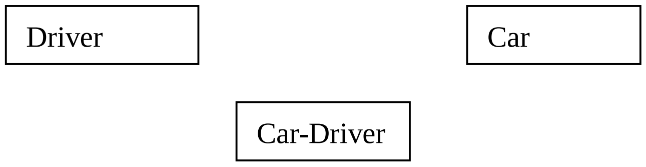


   Student: **Name**, **Address**, Course-Taken, Age, Registration-Number

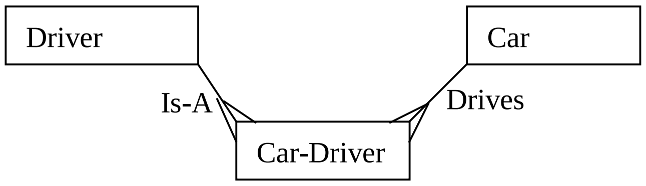
3. We do not like M:M degree relationships. These will arise during our modelling exercise, but they are not very helpful as all they tell us is that many of one entity is related to many of another. In other words, it is not possible to pin down which entity occurrence is being related to. We will always remove an M:M relationship using the following technique. Suppose we have the following:



Step 1: Create a third entity box and give it a suitable name.



Step 2: Draw two new relationships from the original entities to the new entity and make both with 1:M degree towards the new entity and give both suitable names.



So now we can say, each Car-Driver is one Driver and drives one Car.

Step 3: Set out the identifying attributes of the new entity as a composite of the original identifying attributes. Suppose:

- Driver entity has Identifier 'Name' and
- Car has Identifier 'registration-Number', then
- Car-Driver can have Identifier 'Name, Registration-Number'.

B. WORKED EXAMPLE

Scenario

A bus company has several buses and drivers. It has one garage and covers several routes. At any one time, some buses will be on routes whilst others are garaged. Individual buses are used on any route and drivers are allocated to any bus. Each route has only one bus travelling on it at any one time. Passengers may have to take more than one bus on a journey.

This exercise aims to identify the principal information needed by the bus company to maximise efficiency.

Suggested Answer

The first task is to identify the entities in the given description. These are:

bus, driver, garage, route, passenger.

For each, give and justify their attributes:

(a) **Bus** will have some kind of identifying number, either registration number or a company number. Its capacity is important in this context and whether it is in the garage or in use.

(b) **Driver** will have a unique company number as his or her name may not be unique. However, the name is also required and which route he or she is allocated to.

(c) **Garage** will, like all entities require an identifier of some kind and I suggest its name. Other possibilities might be its address or even the company name, etc. The address is needed in some form.

(d) **Route** will probably have a unique identifying number and, from the efficiency point of view, how long it takes to drive round it. The average number of passengers carried might also be relevant.

(e) **Passenger** can only be identified by name and address as unique numbers are not practical. The normal route used could also be relevant.

These entities can now be given formally:

Bus	**Number**, Capacity, In/Out-use
Driver	**Driver number**, Name, Allocated-route
Garage	**Name**, Location
Route	**Route-number**, Time-to-drive-round
Passenger	**Passenger-name, Address**, Usual-route

This question also calls for relationships:

- A bus can be driven by different drivers, and a driver is allocated different buses.
- A bus can be kept in the garage, which may have several buses parked there.
- A bus can travel on several routes, but each route can be covered by only one bus at a time.
- Drivers are allocated to any route, although the route can have only one driver allocated to it as there is only one bus.
- Passengers travel on several buses, each of which, of course, has several passengers.

An entity-relationship diagram will display them formally (see figure 8.6):

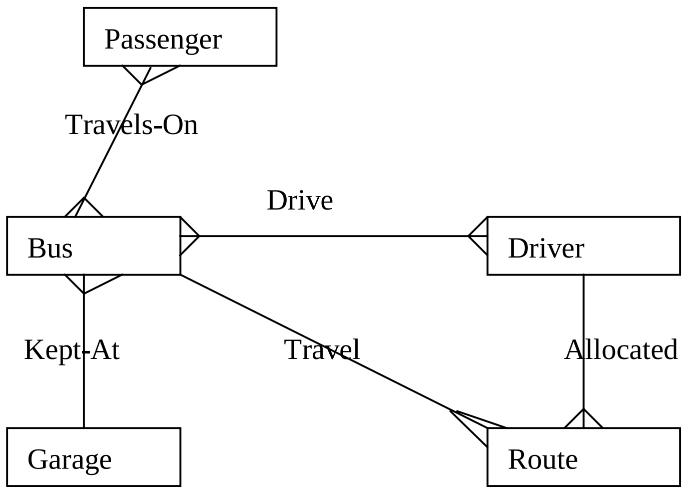


Figure 8.6

This first attempt at the E-R diagram (figure 8.6) is fine except that we have two M:M degree relationships. We do not like these and so will break them down as previously. However, before doing this it is worth re-examining the relationship called "drive". As this can be deduced from the fact that a bus can travel on many routes each of which is effectively allocated a driver, we can say it is **redundant**. Not only are many drivers involved, a driver may also be allocated to any of several routes, each of which is travelled by a bus. Hence many buses are involved. These two relationships tell us the same information as the "drive" relationship. So we can eliminate this relationship entirely. This means the 'Travels-On' relationship can be broken up as before. This then is the E-R diagram (figure 8.7).

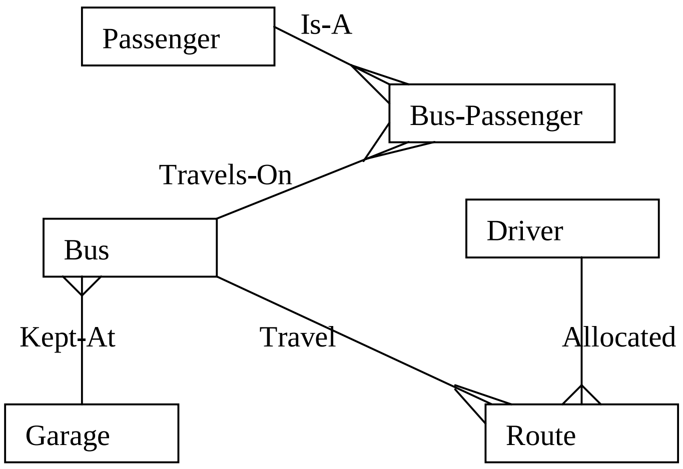


Figure 8.7

The other part of the E-R model is the attribute list: -

Bus	**Number,** Capacity, In/Out-use
Driver	**Driver-number**, Name, Allocated-route
Garage	**Name**, Location
Route	**Route-number**, Time-to-drive-round
Passenger	**Passenger-name, Address**, Usual-route

Together, the diagram and the attribute list are the E-R model.

C. DATA DICTIONARIES

Description

A data dictionary is a specialised form of database. Whereas a normal database is a repository of all the data that is related to an organisation and its activities, a data dictionary is specialised in that it contains everything about the data being held in the database and so provides an extremely useful reference store for developers. So we have a database containing all the data, and running alongside is the data dictionary with everything about that data (see figure 8.8).

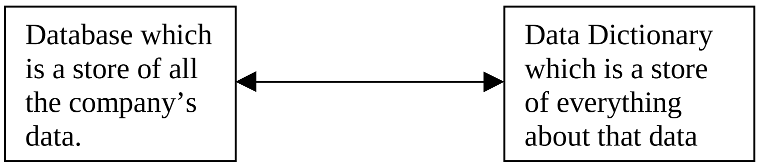

Database which is a store of all the company's data.
Data Dictionary which is a store of everything about that data

Figure 8.8

The data dictionary contains detailed descriptions of all the files, their indexes, the data structures used, the users, the analysis results including data flow diagrams information and entity-relationship definitions. This is not an exhaustive list but it will give some idea of the range of information held by the data dictionary.

Because everything about the information that flows through the company is held by the data dictionary, it makes it an ideal tool when analysing a new system. It will reveal considerable information about the existing system, and together with the data flow diagram technique, a full analysis of any of the company's systems is possible.

Each data item will include:

(a) A definition of the item.

(b) Its relationship to other items

(c) The programs which use it.

(d) Different names which may be used for the same item, e.g. "part number" and "product code" could refer to the same part or product.

(e) How it is validated, e.g. "numeric in a given range", "alphanumeric, 6 characters long". If all applications use the data dictionary, then all data items input will be validated in the same way.

The data dictionary will also hold information on the programs that use the data, including how often they are run, which peripherals they use, etc. The data dictionary also holds all the definitions of user views and physical definitions of how the data is stored and where, in fact everything about everything in the information environment.

The particular way in which the data dictionary helps during analysis is by providing a reference point for: -

- All the definitions created by the DBMS
- The physical characteristics of the data
- How the database is used and by whom
- The standards for using the system
- The data flow diagrams and other diagrams and reports prepared during development
- The authorisations for using the database system, and so on
- The data requirements of the system to be developed
- The details of the relationships between data items, records, files and so on
- Providing support for software modelling tools

The data dictionary is updated automatically. This is an important aspect as the developers will rely on the data dictionary's consistency and that it is up-to-date. It is liked directly with the main database, and every time the database is accessed, a record is kept of that access and any changes that are made to the database system are recorded immediately in the data dictionary.

The last point is worth dwelling upon as the data dictionary is at its most useful when supporting software tools.

D. CASE TOOLS

Professionals use the data dictionary during maintenance activities and during development. Should some work be needed on the database system, the maintenance staff will look up what the existing definitions are and what previous work has been done. At development times, CASE software tools are used and the tool will access the data dictionary for existing definitions and design specifications.

A CASE tool is special software designed for drawing diagrams, writing reports on the process specifications and is another source for maintaining the data dictionary. It maintains the data flow diagrams and their labelling such that a change made in one place is automatically revised elsewhere. Developers can use CASE tools in two broad ways:

- to help with analysis and design, and,
- to help with the construction of systems.

Both ways make use of the CASE repository. This is a data dictionary store of all the data definitions and other artefacts arising through this and other development processes. The repository is fundamental to the concept of CASE as it gives reuse of all aspects of the development process.

- During analysis and design, CASE gives a tool for the production of data flow models and, Entity-relationship models. This is a big help to developers as it save so much time.
- During the construction of the system, a CASE tool uses an **application generator** software tool to produce the actual code. This can be a very tedious task otherwise, and so again time is saved.
- Finally, the CASE tool will contain a Project Management tool that is useful for displaying the progress of the project, verifying its consistency and for documenting it.

When started, the CASE tool produces a graphical screen window interface surrounded by various GUIs and menus. An existing data flow diagram file can be opened within the screen window and then by clicking on the menus and GUIs as appropriate, changes to the data flow diagram are easily made. Once stored, these changes are replicated in all the other relevant places within the data dictionary repository. In this way, there is no need to repeat the exercise and so maintaining both consistency and saving time.

Yet another CASE tool facility is the provision of a **prototyping** facility. This is a fast, user-orientated development technique.

What is Prototyping?

Prototyping is a preliminary working model of an application developed by users using a CASE tool. Its strength is that it requires users to give an unambiguous definition as to what is required by them. A prototype pays particular attention to screen and report layouts, and the primary systems functions. Once modified by a professional developer, the prototype evolves into a fully implemented version of the required system.

The prototyping technique involves siting the user in front of a screen and then with assistance from the developer and with the use of a CASE tool, the user works out a screen interface that they are happy with.

Prototyping has many advantages:

- Most of all, it satisfies users, as they are fully involved.
- It is quicker and has only essential documentation. Training occurs during the prototyping sessions.
- It produces a working solution.
- It eases communication problems between users and development staff.

However there are also disadvantages:

- They are difficult to handle at first because of a lack of control during prototyping sessions and the probable lack of reasonable prior planning.
- It is difficult to prototype large systems because of their complexity.
- Users usually want to use the prototype immediately without the essential background support work.

E. NORMALISATION

Another use of CASE tools is in the technique of **normalisation**. This technique is also known as **relational data analysis** (**RDA**). It is a complementary technique in model building to the entity-relationship approach. Whereas entity-relationship is a top-down approach, normalisation is a bottom-up approach. Taking both approaches to model building will clearly identify all the relationships between entities.

The biggest problem when storing data is duplication. Duplicate copies of data are not only wasteful of storage and inefficient in searches, they also introduce inconsistency if only some of the copies are updated. For example, should your personal details be kept by a company on traditional flat files, it is likely that your address occurs in more than one place. This is the case in the following table of records.

Table 8.1

Item bought	**Name**	**Address**
computer	John	123 The Street, London
book	Peter	67 The Road, Bristol
printer	John	123 The Street, London
software	Peter	67 The Road, Bristol

The addresses of John and Peter are held in duplicate, and possibly elsewhere as well. Should Peter move address, then perhaps only the ‘book’ record will be updated with the new address so that there is then two different addresses for Peter held.

The answer to this problem is to structure the files to remove unnecessary duplications and to only allow duplication where it is known about and where it is necessary for a specific purpose. This is achieved using the normalisation technique.

We can best demonstrate the technique using an example.

Suppose the details of a student seminar are recorded on a form as in table 8.2.

Table 8.2

Seminar-Id	**Student-Id**	**Date**	**Type**	**Room**	**Book**	**Score**
DB1	X123	2-5-01	Database	2	Green	55
DB2	X123	8-6-01	Database	4	Green	65
DB2	Y345	8-6-01	Database	4	Green	50
C1	W567	2-5-01	C++	3	Black	80

Table 8.2 gives us the unnormalised list:

Seminar:

Seminar-Id, **Student-Id**, Date, Type, Room, Book, Score

We will designate the first two attributes as the unique, identifying key and identify it in bold print.

Normalisation follows a series of steps moving from First Normal Form to Second normal Form, and so on. We will take our example as far as Third Normal Form. As we move from step to step a different rule is applied in each case: -

- **First Normal Form -** the rule is "ensure all the attributes depend on the Identifier.".

By examining data values, we can say that all the attributes depend upon at least one of the identifying attributes. For example, *Date* depends on *Seminar-Id*, as does *Room*, *Type* and *Book*. On the other hand, *Score* depends upon the composite identifier. Therefore we can say the list is in First Normal Form.

- **Second Normal Form -** this time, the rule is "ensure every attribute depends on the whole identifier."

In our work on First Normal Form, we noted that *Score* depended on the whole composite identifier, but none of the other attributes did. They only depended on *Seminar-Id.*

We will separate these by writing two attribute lists each containing full dependency. We now have two relations: -

Seminar - **Seminar-Id**, Room, Book, Date, Type
Stud-Seminar - **Seminar-Id, Student-Id**, Score

Our example is now in Second normal Form.

- **Third Normal Form -** this time the rule is "separate off any non-identifying attributes dependent on other non-identifying values".

As *Stud-Seminar* has only one non-key value it is already in third normal form.

However, from the table we can see that *Book* depends on the *Type*. We will therefore remove *Book* from the *Seminar* list and make a new list called *Type* and showing this newly identified dependency. This gives: -

Type

Type, Book

Seminar

Seminar-Id, Room, Date, Type

Stud- Seminar

Seminar-Id, Student-Id, Score

Using the third normal form we can now construct a new set of tables equivalent to the original but which is unambiguous, flexible and easy to use on a shared basis.

Table 8.3: SEMINAR

Seminar-Id	**Room**	**Date**	**Type**
DB1	2	2-5-01	Database
DB2	4	8-6-01	Database
C1	3	2-5-01	C++

Table 8.4: STUD-SEMINAR

Seminar-Id	**Student**	**Score**
DB1	X123	55
DB2	X123	65
DB2	Y345	50
C1	W567	80

Table 8.5: TYPE

Type	**Book**
Database	Green
C++	Black

We can then build the entity-relationship diagram from the Third normal Form.

- We note that *Seminar-Id* is the identifier of table 8.3, but only part of the identifier of table 8.4. There is therefore a one-to-many relationship between these tables.
- Also, *Type* is the identifier of table 8.5, but is not the identifier in table 8.3. There is therefore a one-to-many relationship between these tables.

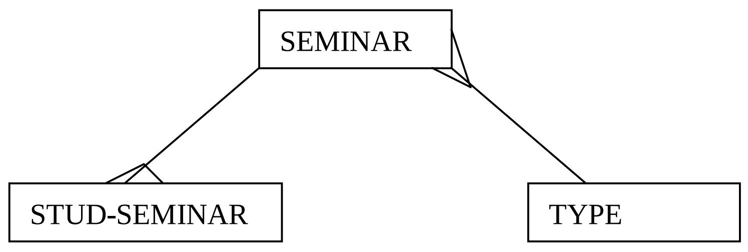


Figure 8.9

F. FITTING THE TECHNIQUES TOGETHER

We have studied a range of development techniques; -

- Data Flow diagrams

- Entity-relationship models
- Data Dictionaries and CASE tools
- Normalisation

So all that remains is to see how they all fit together into the development process.

1. The Data Flow Diagrams highlight the flow of information through the system being studied. They help the developer see exactly what is happening.
2. The Entity–Relationship model helps to identify the detailed structure of the data collected during analysis.
3. Normalisation complements the entity-relationship model by approaching the same task from a different perspective. Together they show they true structure of the data.
4. The Data Dictionary is a repository of everything about the data within the system and so, using a CASE tool, this repository can be used as a computerised tool to help the developer with each of the first three items above.

Study Unit 9

Entity Life Histories (ELH)

INTRODUCTION

In the previous study unit, we looked at entities and when they occurred. You will remember that whilst 'Car' is an entity type, any specific car is an occurrence of the entity. Suppose this is your own specific car. During the lifetime of the entity occurrence, you will buy several different cars, updating the occurrence each time. If at some stage you decide not to own a car, you will no longer be interested in the 'Car' entity. What we can note from this is that the entity occurrence has its own history. Your car occurrence history will be very similar to all the other car occurrence histories. There will be a common pattern.

An **Entity Life History** (**ELH**) diagram is a diagram that attempts to map the possible life of each occurrence from creation to deletion.

Compiling ELHs is quite a difficult exercise as there is so much to consider. It is however, a very powerful technique that takes the developer from the analysis stage towards the design stage.

A. ELH NOTATION

Events

During the lifetime of each system, happenings take place in the outside world that can affect the system's entities by changing them. The happenings taking place outside the system that in some way excite that system's data are known in SSADM as **events**. By definition, only events can cause a change of the system's data.

Effects

An event may affect attributes of more than one entity. The particular set of changes caused within an entity by an event is called an **effect**.

Notation

The drawing of ELHs takes place when the requirements specification is almost complete and when all the entities of the system have been identified and the DFD has provided most of the events which affect these entities.

ELH notation is based on the main 'structured' principle that states that any programming activity can be described using only the three basic constructs of **sequence, selection** and **iteration**.

An ELH consists of a set of connected boxes. A single box containing the name of the entity is placed at the top. Each box may have other boxes hanging beneath it. A box that has no other boxes beneath it constitutes a 'leaf or 'element'. Leaves contain the effects that particular events have on the entity.

Sequence

The simplest construction in an ELH is that of **sequence**, which shows things that are expected to happen one after the other. The thing that is expected to happen first is positioned to the left, and at the same height, as the thing that is expected to follow it.

For example, if the team of analysts has decided that three events, A, B, and C, are to affect an entity in that order, then these events are shown on the ELH (see Figure 9.1).

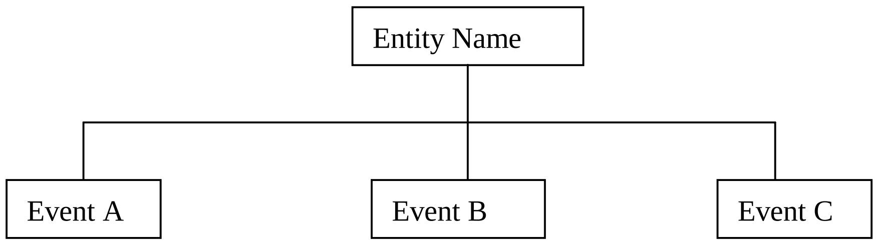


Figure 9.1

The ELH shows that for every occurrence of the named entity, event B can only affect it **after** event A, and **before** event **C**.

Selection

Often, a choice exists between alternatives that can affect an entity.

This choice is shown on separate 'branches' of the structure, each branch containing a box with a little 'o' (for option) at the top right.

Consider, for example, the Employee entity of any business. An employee can either be a fully trained or just a trainee. This means that, from the point of view of the business system, a particular person will be recorded for the first time as an employee either because they are trained or because they are trainees. Therefore, two different events give rise to an Employee occurrence. One is the 'New Staff Recruited' event and the other is 'New Trainee Accepted' event. This choice is shown in figure 9.2.

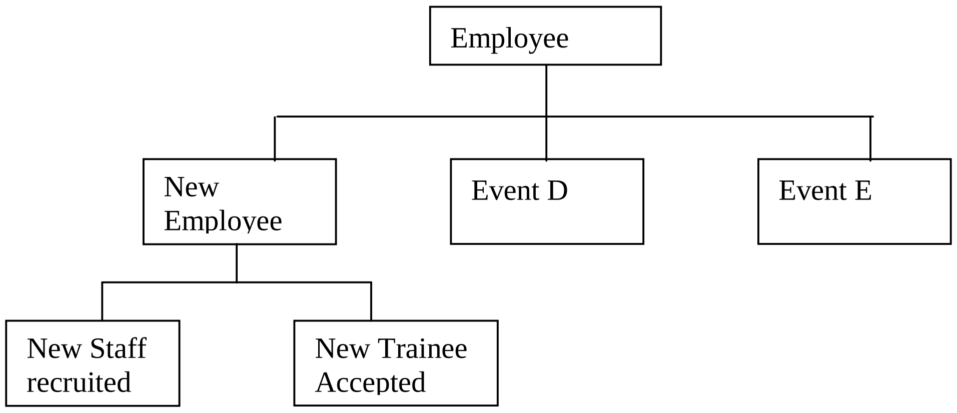
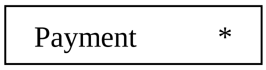


Figure 9.2

To read the diagram, we start from the left-most box hanging under the entity name (named 'new employee'). From that box we move down to exhaust all the possibilities of the events that hang under it, following the same rules at each level. When we finish with this branch we move to the next branch, as indicated by the uppermost sequence. So, in the example the first thing to happen to an Employee is the recording of either the recruitment of a new, fully paid, member of staff or the acceptance of a new trainee. The next thing that can happen to an Employee of this system is the effect of event D, whatever that is, followed later on by event E. Note how the structure only contains events at the bottom (i.e. the leaves) of each branch. Also note that, according to the sequencing rule, event D cannot occur for any employee for whom the 'New Staff Recruited' or the 'New Trainee Accepted' event has not preceded it.

Iteration

Events that occur many times for each particular occurrence of an entity are represented on the diagram with boxes containing an asterisk.



We can now construct a first ELH. Putting together the first two diagrams already drawn, and taking the history through to a conclusion, we have figure 9.3.

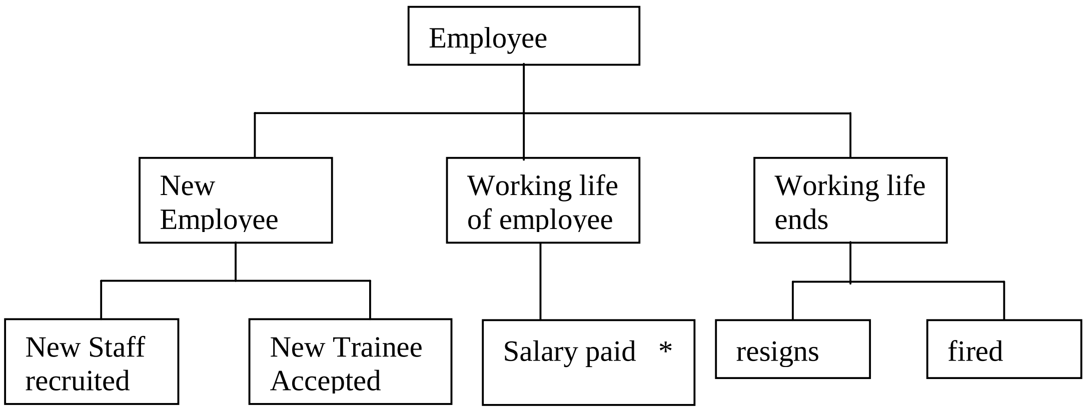


Figure 9.3

Some rules

1. All boxes hanging from one node must be of the same type.
2. All iterations must have a sequence or selection box immediately above them.
3. All selections must hang under a structure box.

Parallel Structures

Sometimes, an event occurs that does **not** change the status, as far as the system being considered is concerned, of an entity occurrence. For instance, employees can change their address at any moment during their employment. This event has no effect on the business apart from the fact that the new address of the employee is now on record. This type of 'uneventful' event typically occurs without any predictable sequence or even while other events are occurring. The ELH notation accommodates these situations by introducing a parallel structure, depicted on the diagram with a parallel bar, placing the exception to the right (see figure 9.4).

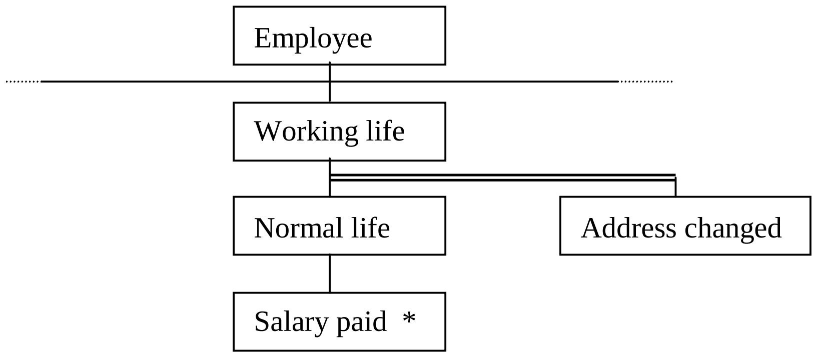


Figure 9.4

Quit and Resume

Sometimes extraordinary events occur and disrupt the expected life pattern of an entity. Attempts to anticipate those events on an ELH may lead to a complicated diagram that obscures the normal entity life. To avoid this tangle, **Quits** and **Resumes** are used on the diagram to leap from one event to another.

For example, suppose we return to our employee. Part of an employee's life involves receiving 'salary payments', attending to 'new appointments' and 'going for training'.

Now, suppose that while on training, an employee cannot be assigned to a new appointment nor can they receive their salary. This rule leads to the identification of the 'return from training' event which then has to be placed in a sequence after the 'go to training' event so as to enforce it.

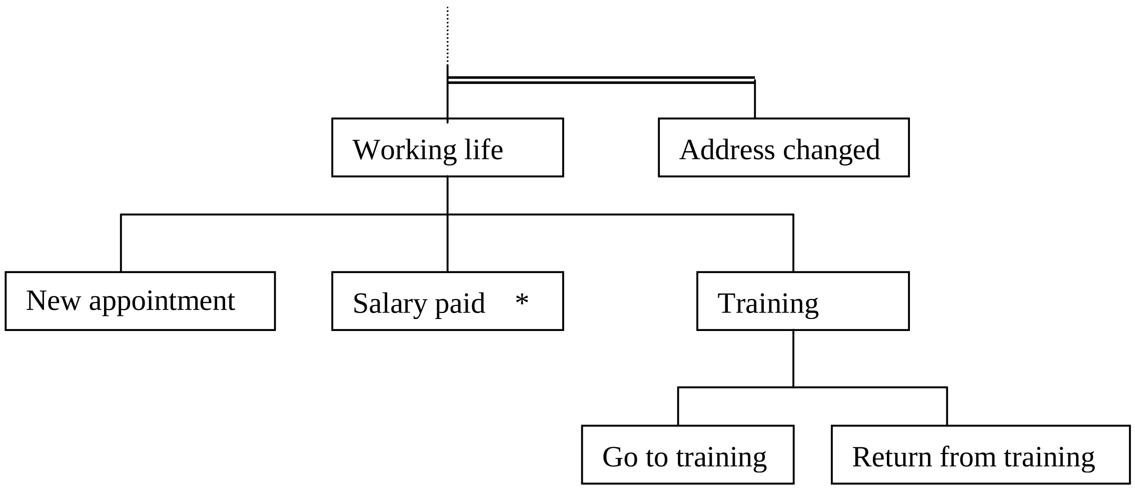


Figure 9.5

Suppose now that an employee who has gone for training decides not to come back. This means that the employee has either resigned or is fired. In other words, the system has to be able to jump to the events at the 'end of working life' node instead of having to go through 'return from training'.

The **quit** and **resume** mechanism allows the system to anticipate this situation by providing for 'two' lives, the 'normal' and the 'alternative'. These two lives are each placed on a branch of a new option, as shown in figure 9.6. The event which can be jumped over, in this case 'return from training', is marked with a 'Q', while the events on the point of resumption are marked with a 'R'. Since many *quits* and *resumes* may be present on the same ELH, the Q's and R's are followed by a number that indicates which R's correspond to which Q's.

Additionally, purely so that the alternative structure is more clearly visible to the human eye, it is suggested (in the official manual) that a box with the words 'start of alternative case' under which an iteration box with the word 'event' are placed on the structure. These conventions are shown in figure 9.6.

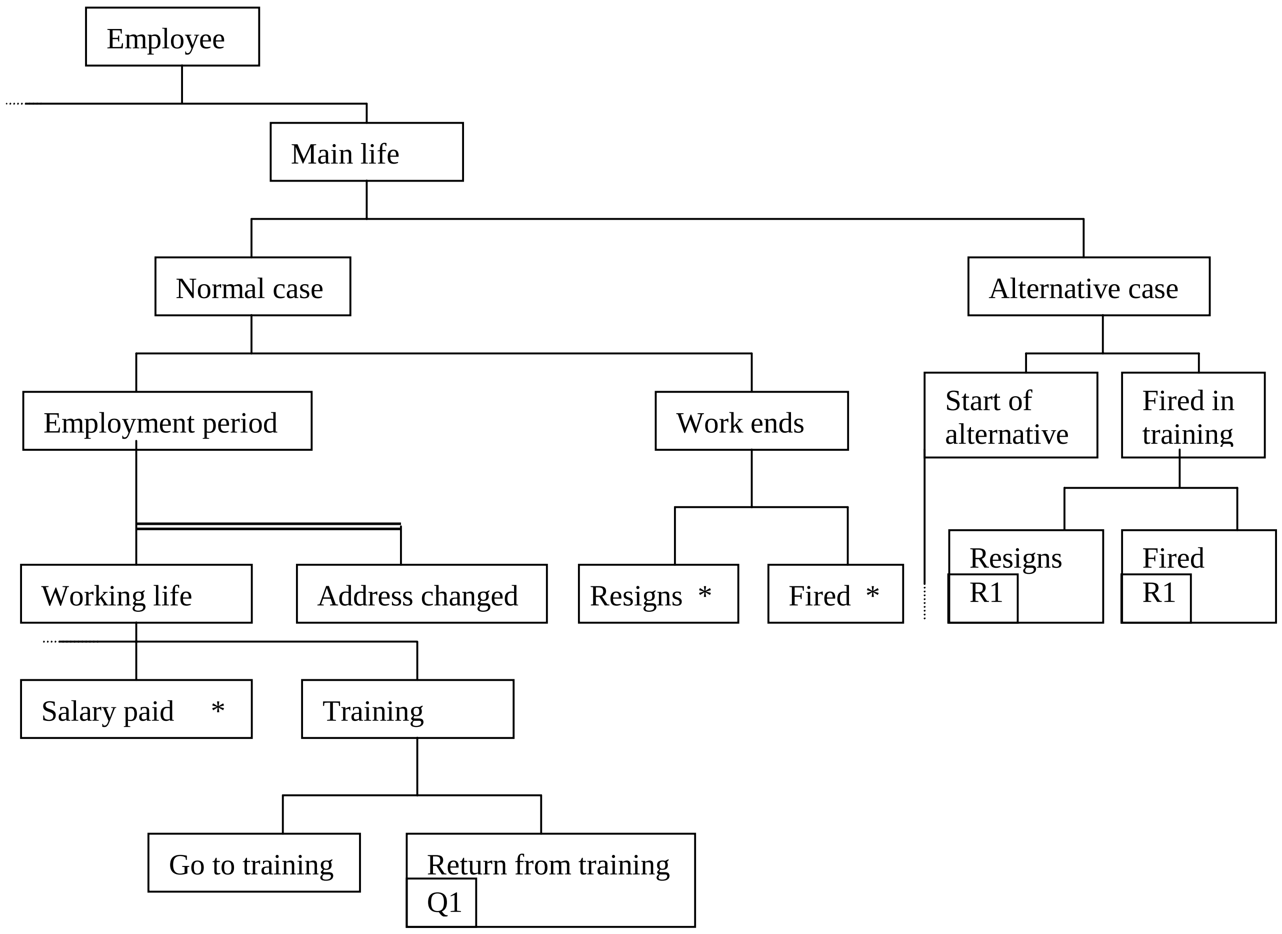


Figure 9.6

Occasionally, an extraordinary event may occur which suspends the normal life of an entity in such a way that it calls for a leap to a totally new event box placed at the foot of the diagram. Consider that employees may be drafted into the army where they have to serve for, say, six months. During that period nothing really happens as far as the business is concerned and everything concerning the employee is suspended awaiting their return. We could show the existence of such an 'employee drafted' event by placing a suspended event box somewhere in the vicinity of the ELH. Q's and R's could then be used to show that this event is another legitimate eventuality in the life of an employee.

Quits and resumes have to be used with great caution. Over-use leads to bad programs and the reappearance of 'spaghetti' codes. The main rule governing Quits and Resumes is that the event containing the Q may be replaced by the event containing the R.

Convention stipulates the use of a 'selection' whenever Quits and Resumes appear. So it soon becomes clear that Quits and Resumes can only be used to jump from one side of a selection to another or from an iteration into the main structure or towards an off-the-structure 'random' event.

State Indicators

Entity Life Histories follow the effect that time has on every occurrence of an entity. This means that it should be possible to know, for each entity occurrence, which event was the last one to affect it.

This information is important because it can be used to create a design which will not permit certain events to affect an entity occurrence if certain other events, as dictated by the business' rules, have not taken place. For example, the event of paying for an invoice should never be allowed to take place before the event that shows that the invoiced goods have been delivered by the supplier and accepted by the business.

State indicators (**SIs**) are the ‘markers’ placed on each entity occurrence to indicate the last event that affected it.

State indicators are numbers allocated to an entity whenever an event affects it. The value of the SI set by an event is shown on the right hand side of a slash placed on the ELH just under the event itself. The values of the SIs of all the events that are allowed to precede a specific event are placed on the left of each of these slashes (figure 9.7).

Each birth event has no previous valid SI, so a - is placed on the left of the slash. Similarly, an event that leads to the deletion of an occurrence of an entity does not set the SI to anything, so a - is placed on the right of the slash. A random event, as seen under a parallel life, does not set a SI value and so a * is placed on the right of the slash.

The placement of SIs on an ELH is fairly straightforward. Each event sets the SI to one value. It is natural to start with I for a birth event and then proceed until the last event.

- In a sequence, the SI set by the box on the left is the ‘valid previous’ value of the box to its right.
- Selection elements hanging under the same node should all have identical ‘valid previous’ SIs.
- Since an iteration means that the indicated event can occur after itself again, the ‘valid previous’ of an iterated event should include the ‘set to’ value of that same event.
- Since, in common with all computing conventions, an iteration is allowed to happen, the event following an iteration should contain in its ‘valid previous’ values the SIs set by the events immediately preceding the iteration.
- Quits and Resumes force the ‘valid previous’ SI for the resume event to include the ‘valid previous’ values of the event it is replacing (i.e. the event that contains the corresponding Q).

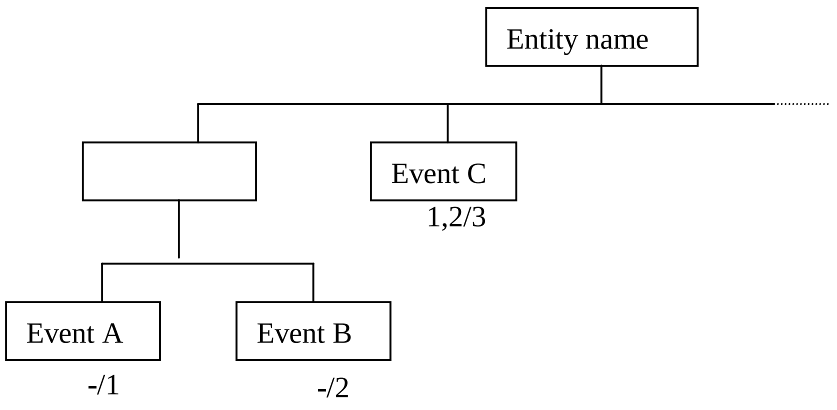


Figure 9.7

State indicators are a device to stop an event occurring if the business rules do not allow it. We therefore see that SIs can be used as blocking mechanisms.

B. INTERRELATIONSHIP BETWEEN THE DFD, ENTITY-RELATIONSHIP MODEL AND THE ELH.

The Data Flow Model, the Entity-relationship Model and the Entity Life Histories are three views of one and the same system. They are related since: -

- The data of the system which constitutes the entity-relationship model is the data contained in the data stores of the DFD.
- The data coming in from an external entity of the DFD represents an event, since this data is trying to enter, and thus affect, the system.
- The entities for which an ELH is drawn are the very entities of the entity-relationship model.

SSADM uses techniques in a well prescribed sequence. Among these techniques, Data Flow Diagrams, Entity-relationship Modelling and Entity Life Histories used to specify the requirements of the system under development. Each one of these techniques relies heavily on a diagram which becomes, in effect, a pictorial description of what the model is trying to represent. Each diagram consists of not more than four main symbols appropriately intertwined.

- The Data Flow Diagram represents the flow of information within a system. It consists of: external entities (source/recipient); data flows; processes; data stores. While the diagram is very powerful, there still is need for some documentation that will avoid any possible ambiguities in the interpretation of the diagram.
- The Entity relationship Model describes the information actually held by the system and its interrelationships. It consists of entities and relationships. Every entity consists of a specific list of attributes that define it.
- Entity Life Histories model the effect of time on each entity. They are based on structured notation that allows constructs for Sequence, Selection and Repetition.

C. WORKED EXAMPLE

Draw an ELH for the following description:

A delegate can be booked on a course. If the delegate has been to the training centre before, then their records will be held on file, otherwise their personal details must be taken. The booking can be either provisional or firm. Once the booking is firm a deposit is taken, the delegate then attends the course and an invoice is sent out to their company. If the company has not paid the bill after 30 days a second invoice is sent. If the company do not pay after 3 months the debt is referred to the training centres solicitor.

Points to remember:

- The Entity Life History that is being modelled is ‘Delegate’.
- At a high level there are a sequence of activities: *Take Details, Attend, Remove Details.*
- A delegate’s details are only taken once, similarly they are only removed once.
- A delegate can attend many courses.
- A booking can be *Provisional* or *Firm.*
- Once a booking is firm a sequence of activities takes place: *Take Deposit, Attend Course, Send Invoice.*

- After the invoice has been sent a series of subsequent activities may occur.

Suggested Answer

This answer uses a small circle in the top right hand corner to indicate alternative events.

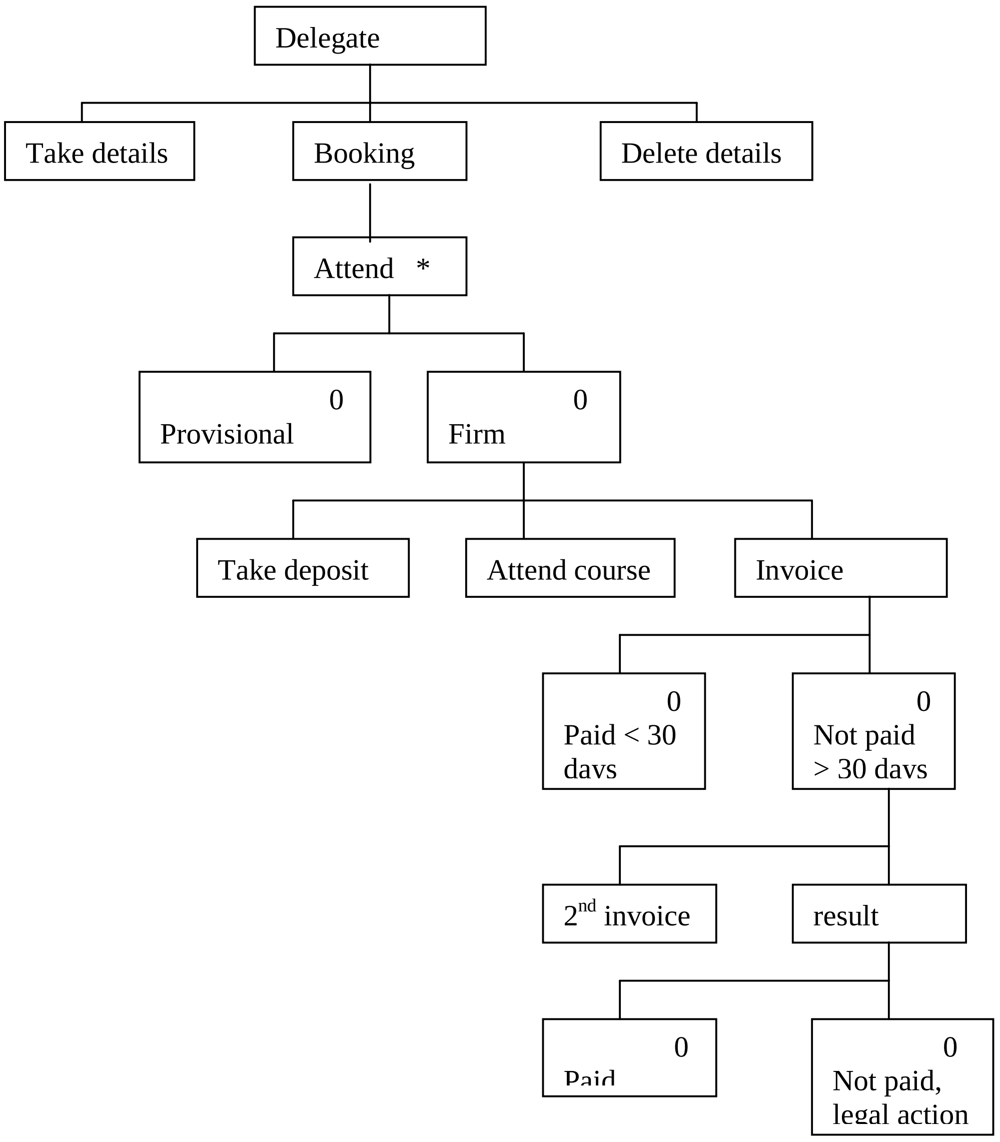


Figure 9.8

Study Unit 10

Standards and Documentation

Contents *Page*

INTRODUCTION

It is perfectly natural to group standards and documentation together. This is because standards are communicated and enforced through the documentation. To many data processing personnel, standards mean documentation standards because, in many organisations, these are the only ones in existence.

Standards are uniform practices which govern the methods by which systems are developed and operated, and provide a basis for assessing both system and personnel performance in terms of quality and quantity. They give a clear understanding of what is required to both management and data processing staff. Standard procedures have to be established for the development and operation of computer systems, but this is not enough. Once set up and agreed, they must also be enforced. As new methods of working are introduced, standards have to be updated to suit the new environment – enforcing out-of-date standards is counter-productive.

Of course, documentation is much more than the communication and enforcement of standards. Modern system development methods rely upon full and comprehensive documentation. Every step is fully documented and all through the designs, the coding and the testing, information on each of the steps is set out in manuals. Subsequently, the auditors, manager and quality assurance people can check the documentation to obtain information on the steps. After implementation, the documentation will greatly aid the maintenance staff.

A. THE ROLE AND SCOPE OF STANDARDS

Essentially, a data processing department should have standard procedures laid down for all aspects of its work, as this is how a quality project management system is maintained. The purpose, content and usage of each document should be specified in a consistent way.

Methods standards specify an orderly way of working for new systems development and for maintenance. They may be subdivided into systems, programming and documentation standards. They will include:

- Control procedures
- Documentation standards
- Development guidelines
- Programming and testing standards
- Maintenance procedures
- Performance standards
- Operating standards.

One of the benefits claimed for structured systems analysis is that it has a defined method of use which produces a systems methodology standard. Similarly, defined methodologies with regard to structured programming may form the programming methodology standard.

Documentation standards can run to many volumes. It is an extremely valuable asset to any organisation to have a set of standards which set out in detail what is required to be documented on completion of each activity.

All aspects of an IT Department's work should be standardised, in particular controls on development projects, expected level of performance and operating methods.

This covers almost everything, including:

- Adherence to the standards adopted by the department;
- The reporting hierarchy;
- The stationery used;
- The salary cycles and all similar aspects.

Development Controls

Project control standards specify the method of planning and controlling the progress and quality of systems development. They are concerned with the content of the work rather than its method of execution.

As we have noted structured development imposes its own standard methods, within which the management must adopt and enforce the use of standard styles of documentation and standard reporting points (milestones), as well as the requirements of the methodology and programming language used.

The control of a project has to be keyed to a defined set of activities, to the items of documentation to be produced during the project, and to other standard methods of working.

Performance Standards

Performance standards set base lines against which actual performance is measured. In this sense, performance standards are not estimates of how long tasks will take, but are used to specify how long tasks **should** take. Their primary use is for comparison with actual performance to pinpoint deviations that call for investigation and, possibly, action.

Operating Standards

The operating environment is, in many ways, a more difficult function than systems and programming for management control. Operating standards will need to cover:

- Physical security procedures
- File security procedures
- Work log procedures
- Error log procedures
- Definition of staff responsibilities (probably complete job specifications)
- Data control procedures
- Computer operating procedures.

Each installation must develop its own procedures, but the following is an example of what might be included in the standards for computer operation:

(a) use of documentation – each system should have an operations manual with a section detailing machine operation;

(b) control over stationery – both standard and pre-printed;

(c) control over magnetic media – use of write inhibit rings, etc.;

(d) exception procedures – what to do in the case of software or hardware failure;

(e) emergency procedures – details, for example, of automatic fire control equipment.

Benefits of Standards

(a) Saving Time

Elimination of indecision and the consequent economy in time are major benefits of standards. Good standards reduce the amount of repetitive work and leave more time for the more challenging aspects of development work – for example, problem analysis.

Suppose a project team is beginning the analysis activity, the major output of which will be the structured system specification. The standards can immediately answer a whole series of questions such as:

- What would be in the specification? and
- Who authorises the work?

(b) Management Control

The management of all aspects of the department is greatly assisted by the use of standards.

- All work is carried out to a prescribed quality that can be measured.
- Monitoring of progress of work is easy, since all tasks follow a defined sequence.
- Precise responsibilities can be allocated to staff.
- Estimates of time-scales can be more accurately predicted.
- Schedules of work can be prepared more easily.
- Staffing levels can be determined.

(c) Improved System Design and Development

- Standard procedures provide a checklist to ensure that important aspects of the development of a system are not ignored.
- Adequate documentation is produced at all stages.
- Standard design methods ensure that similar systems are designed in similar ways, thus aiding maintenance.
- Programming standards ensure that reliable and well-controlled software is produced, and that standard problems are solved in a standard way.
- Well-defined maintenance procedures ensure that this expensive aspect of software development is well controlled.

(d) Computer Operations

The standards in this area ensure that all computer and related operations are carried out correctly, reliably and fully.

(e) Other Advantages

- ***Training***

  When a set of standards is in force, the training of new staff is greatly eased since all aspects of the work have been defined – the actual content of a training programme is clear.

- ***Reduction of Dependence on Individuals***

  In a data processing department there is a fairly high rate of turnover of staff, especially in the systems and programming areas. It is essential that the absence of an individual does not have a catastrophic effect on the work of the department. The adoption of standards means that it will not be difficult for another member of staff to take over, since the correct documentation will be present.

- ***Improved Communication***

  (i) Standard presentation of information reduces misunderstandings.

  (ii) A well-defined set of procedures ensures that all parties who should receive information will, in practice, actually receive it.

B. THE STANDARDS MANUAL

The most common and perhaps best technique for the presentation of standards is via a standards manual or series of manuals. This is the “bible” of the department and will include all the aspects we have mentioned above.

The production of such manuals can be difficult and time-consuming, but it is well worth the effort. It is important to remember the following points:

- A standards manual is not a once-and-for-all production. It must be constantly reviewed to ensure that it is up-to-date and in line with experience.
- It is the “spirit” of standards which must be carried out, rather than the letter. It is far more important that a job is completed well and to a high standard than that the exact standard is followed. Insistence on “going by the book” defeats the whole object.

  It is thus essential that the staff using the standards want to use them and really understand their value and importance. Education and involvement in the production of standards is a very good method of ensuring staff co-operation.
- The management of the data processing department must believe in the standards and ensure that encouragement is given to the staff to follow them.
- The use of standards means that short cuts are never taken in the production of software and the operation of systems. This has to be appreciated by both user and data processing department. It will mean the time taken to develop systems will be extended, and this can lead to misunderstanding by users. Producing quality software takes longer but, once produced, it does not fail to the same extent. A user has to be convinced that the extra development time will in fact save time in the long run.
- The role of the standards manual will not be to supersede the computer manufacturer’s technical literature, such as programming, operating, training, site installation and software manuals, but to complement such literature with a more personalised document detailing the particular requirements, policies and practices of the organisation’s DP management. All manufacturers offer a manual of standards in one form or another, but none of these can be said to be ideal for everyone’s use.

C. OTHER DOCUMENTATION

As we have seen, a number of areas must be described to specify what a computerised system is to look like. This includes describing the input methods used, the files held, the output which is to be produced and the processes by which these relate. This is most frequently achieved by using a number of standard documents.
This approach has the advantage that all the relevant aspects of the process are recorded; for each file or screen or form, the person filling in the document is reminded of the questions which have to be answered.

System Design and Specification

We know that once all the facts are established and recorded to everyone's agreement and satisfaction, the next step is the design of a solution to the problem. This will take a number of forms and, although an optimum solution may appear to exist, it is important for the analyst to draft out a number of alternative plans.

The job as outlined to the client will be in the broad terms of the layman. The report will include the diagrammatic representation of the new work flows and procedures, supported by draft samples of all computer printed output such as reports, summaries, invoices, etc. It will also include an indication of the main forms of data preparation and input to the computer system.
It is of no interest to the non-data-processing people how disk files, for example, are to be organised, but such details are of prime importance to the computer programmer, as well as layouts for all input and output activities. The program specification should include full descriptions of the job being tackled, together with any necessary calculations or assumptions. Arithmetical requirements would include the instructions necessary for extreme conditions, the rounding up or down of answers, conversions of results (such as from metric to imperial units, for example).

Program or Module Documentation

We shall think in terms of the programmer writing and testing a program or module, completing the project only when it is fully working and when the following documentation has been satisfactorily prepared. The specification will refer to other areas of the manual of standards for such things as subroutines usage, file housekeeping requirements, error conditions and end activities.

Different computer installations have various levels of program documentation, but in essence they all include the following items.

(a) A large part of the program documentation will consist of a copy of the original program specification, updated as necessary.

(b) There should be actual samples of all printed output as produced by the working program.

(c) The documentation must also include the latest program listing or compilation which, of course, should be completely free of errors.

(d) Programs should include not only the individual ideas of their creators but also their positive identities together with the dates, names and details of any subsequent alterations.

(e) Finally, as a permanent reference to the degree of program testing undertaken, the documentation should include details of test data submitted, along with examples of results produced.

All these items should be prepared according to the installation's particular manual of standards. The acid test as to how well a program is documented is whether an average programmer can take hold of another’s program and perform amendments and alterations with ease and confidence. If the task is too great, then either the original program was not written and documented according to the installation’s standards, or the standards are too loosely written.

Operating Instructions

Operating instructions for an application system should:

(a) Describe the system briefly, most especially the file processing aspects, which are of particular concern to operators.

(b) List the error and informative messages produced by the program and indicate what action, if any, is to be taken.

(c) Specify the actions to be taken if any problems occur with the computer itself; these may range from a straightforward rerun to quite complex sets of instructions designed to back-out dubious on-line updates to a database.

User Documentation

User documentation is a reference manual for experienced users, who will need to refer to it when queries or exceptional circumstances arise; it also provides the basics for training of new staff.

The importance of this class of documentation should be clear but, historically, it has often been conspicuous by its absence, its incomprehensibility or by being out-of-date. It is not unusual to find that, for example, the accounts department in a company is using a system which nobody in the section really understands. The consequence is that nobody can check it, and many aspects of it are not used because nobody knows of their existence.

Some modern on-line systems include “help” facilities, which reduce user documentation, but do not make it completely unnecessary.

In general, users require documentation which tells them:

- What the system does and how it works.
- How to provide input data to the system and how to control it.
- How to identify and correct errors.
- What are their particular responsibilities.

As described above, each user manual must be tailored to suit the system and the particular users but, in general, the contents will include:

(a) An introduction giving a simple overview of the system.

(b) Running the system; when users run the system themselves, the user guide must include details of how to switch on the equipment, call up the programs they use, and also how to end sessions. In some cases, instructions for taking back-up copies of data, etc. will need to be included.

(c) Input requirements: with some users, data input forms will need to be completed and sent to the data input department. In other cases, users will themselves input the data, either from individual source documents or in batches which have been assembled in the user department.

(d) Output with examples of all the different types relevant to the particular users. Some of these will be screen displays whilst others will be printed reports available either as standard output or on request.

(e) Error messages: an explanation of all the error messages which might occur, together with the appropriate action to be taken. This section should include details of the person to contact if problems are experienced.

(f) Logging procedures: each installation should set up standards for manual logging of any exceptional occurrences, with details of the action taken.

(g) A glossary of terms used.

(h) Index: all except the shortest of manuals should have a comprehensive index, since the experienced user is likely to refer to it only occasionally but wants to be able to find the specific item quickly.

D. USING STANDARD FORMS

We have seen that forms are used to record, transport, present and store data. If they are well thought out, they can be used efficiently. If they are carelessly designed, they can hamper the efficient working of a system.

The systems analyst and designer should be aware of the basic ideas behind sound form design.

(a) Abolish Unnecessary Forms

We should begin by asking whether a given form is essential or not – can it be either eliminated or combined with another form?

(b) Establish Objectives

The objectives must be established before designing the form. At each point at which the form arrives, it should contribute to efficiency. Forms are "mobile". Their efficiency can be achieved only by consultation with the people concerned.

(c) Clarify Functions

The functions of the form must be clear. A descriptive title should be given to it, together with pre-printed advice and guidance and an indication of the route to be followed by the document.

(d) Ease of Use

Clerical work should be minimised. This may be achieved by pre-printing fixed data – especially codes to be used in data processing.

Ease of use should be a priority in the design. Various points could help here – elimination of carbon copying, pre-punching filing holes, avoiding misleading and unnecessary information, provision of a filing margin, grouping together in the order in which data items are to be entered into the computer, permitting enough space for manual entries, correct and appropriate spacing if typing will be undertaken, supplying as continuous stationery if it is in constant use.

(e) Appearance

Attitudes may well be influenced (at least in part) by the document's appearance, and so the analyst should design the form in such a way that important items are underlined, the number of typefaces is not excessive, various colours are used, as necessary.

(f) Special Considerations

In addition, special care has to be taken when designing certain types of document.

- ***Computer Print-Out Forms***

  These may be statements, invoices, advice notes and so on. The basic computer output must be structured on the manufacturer's printed layout forms. Built up around the data are the form outline and the appropriate headings. There must be enough horizontal and vertical latitude or it will reduce the speed of the set-up operation of the printer and could also spoil the final document.

- ***Computer Input Documents***

  These must have maximum clarity, as already described.

Study Unit 11

System Implementation

Contents *Page*

A. THE IMPLEMENTATION PROCESS

The purpose of implementation is to put the theoretical design into practice. It can involve the installation of a complete system or the introduction of a small subsystem. A particular project was selected. Its objectives, requirements and constraints were defined in the requirement specification. A designer, or a team of designers, has specified a suitable system, in the systems design specification. Now, the designed system must be developed and implemented. So, we can say that the aim of this phase is:

> to implement a fully-documented operational system which meets the original requirements according to the design given in the systems design specification.

Implementation involves the following activities:

(a) Writing, documentation and testing of all the programs required.

(b) Creation of all the master files required in the system.

(c) Preparation of user and data processing department operating instructions.

(d) Commissioning of the new system.

(e) Education and training of all staff who will use the system.

Implementation of a Package

We have said that the possibility of using a ready-made package should always be investigated before the decision is taken to write a tailor-made system. If it is possible to use a package, activity step (a) will not be required. There may, however, be some subsidiary programs to write to supplement the package and, often, these will be produced by the supplier. Some of the operating instructions will be supplied as part of the package but these will probably require "personalising" to suit the individual company. In particular, methods of collecting and entering data, and the control procedures connected with these activities, will be unique to the company and, therefore, they will still have to be produced.

All the other activities, (b) to (e), will have to be carried out, and it may look as if we are not saving much effort by choosing a package. This is not true, however, since the short description given under (a), often covers months (or even years) of work. With a package, all the other activities can build on the foundation of a tried and tested set of programs which can often be seen operating in another company before work starts.

System Testing

The transfer to operational status should begin with a positive written statement by the project leader and system designer to the effect that the system is fit for operation. This, of course, is the whole purpose of the testing procedures.

System testing will follow a similar pattern to the testing of programs. First, simple files will be created and the sequence of the data flow diagram followed. The output from each program will be scrutinised to see that the data has been processed correctly.

One of the main errors that may come to light is the misinterpretation of the design specification, so that, although a program works correctly in its individual tests, its input or output is not compatible with another program. For example, one program may expect a data item to be three digits without allowing a space at the beginning of a two-digit number (so that, for example, 12 has to be entered as 012), while another will allow a space.

When the programs can pass data reliably between themselves, then the data is made more complex, until there is sufficient confidence to use live data.

Systems testing builds up the confidence and experience of the users and data processing staff. Wherever possible, data should be prepared on the new system's input forms.

File Conversion/Creation

When a new system is to be implemented, it is likely that the master files either do not exist, or, if they do, that they are not organised as required by the new system. Before the system becomes operational, the master files must be created. This can be a major task, and it may involve the production of a file-conversion system, with its own programs. This is expensive, because the conversion will be a once-and-for-all operation, and the programs will be used only once.

With many systems, it is possible to enter the static data into the new files over an extended period of time prior to the changeover to the new system, leaving only the dynamic data for later entry. For example, product descriptions, code number, supplier details, etc. could all be entered into a product file quite a long time before the stock levels.

When the time comes to enter the active (or "volatile") data, the data files must be "frozen", so that the time chosen to do this is often over a weekend or other non-working period. Any transactions and amendments that occur while the file is being created will then have to be entered.

However, this can be a blessing in disguise, since the updates can be carried out using the new system, and then the state of the new file after updating can be compared with the old file which has been updated via the old system, and thus prove the validity of the new system.

As with the entry of static data, control totals should be used but manual checking of a listing of the files will usually be necessary. Even so, however good the original data collection and the subsequent data entry, some errors must definitely be expected. These will be straightforward mistakes, and also omissions and conflicting data items. Therefore, a procedure for handling these as they are discovered must be devised prior to the file-creation stage.

File creation imposes a heavy load on user departments, particularly if the information is coming from a manual system. Not only do the users have to extract the required data from their manual records but, after it has been put on to the computer, they have to check it for accuracy. If a system starts with inaccurate data, the confidence of the users will very quickly be undermined, so it is most important that adequate time is allowed for this part of the work.

If the volume of data to be entered is large, the company may be incapable of handling the additional load. Also, if conversion is from a manual system, there may be insufficient clerical staff available to fill in the conversion input forms. Sometimes, it is possible to employ temporary staff but, often, the only solution is to use the services of a computer bureau which specialises in file conversion and/or data entry. The service may include the clerical aspects, data preparation, and even the actual creation of the files. The specification of the new file is given, and access to the data of the old file, after which the bureau does everything necessary to prepare the new file.

Education and Training

An essential feature of the implementation of a new computer system is the education and training of all staff associated with it. All staff must appreciate the objectives of the new system, and how it will operate, as well as the facilities it will provide. Those staff who prepare data, operate the system and use the output will require detailed training and practice.

The detailed training must be supported by adequate documentation. This will be the user manual which is written by the system designer. There must be practice in the training programme. Special practice sessions must be arranged, especially where there is a direct changeover, when there is no "running-in" practice period of pilot or parallel running (see later).

Managers want to know what they can expect from the system, and where their responsibilities lie with regard to the provision of data. With terminals and personal computers in user departments and on managers' desks, the necessity for everyone to understand the basic operation of the system is even greater.

Training must also cover the procedures to be followed when something goes wrong. Many are the stories of back-up copies being taken scrupulously every day, only to find, when they are really needed, that the "recovery program" cannot read them. Training procedures should, thus, cover loss of data during a session, owing to a disk which cannot be read. Also, what happens if there is a failure in the electricity supply? Even if recovery is very simple, the first time a screen goes blank in the middle of an operation, it can easily lead to panic.

We shall return to this subject with a more detailed examination of the training process later in the unit.

Implementation Planning

The successful implementation of any system is based upon the following points:

- A project control monitoring time, cost and quality of output.
- Managerial commitment and involvement at all levels.
- Analysts who are good communicators and have a thorough knowledge of the organisation's operations and applications.
- The users' knowledge of and agreement with the system objectives.
- Recognition of user responsibilities in the system development.
- A computer manager capable of getting user support and of instilling confidence in users.

Most of all, sound planning beforehand is essential.

The planning must be very thorough and include all activities and related responsibilities to make the new system work and to withdraw the old one with its documentation.

In larger installations a co-ordination committee will be appointed. Their purpose is to ensure a smooth implementation. The analyst will be the committee secretary and will have the major responsibility. A timescale is established and regular progress meetings are held to ensure the timescale is being kept to.

Implementation Personnel

As well as the specialist computing staff responsible for the implementation of a system, other personnel have an essential role:

- The business manager and user group, including those involved in the prototyping, will be brought in to make the final test.
- The technical manager will assist the users with the mechanics of actually running the machine(s).
- Hardware representatives will be consulted over problems and for general advice.

- Consultants will be available for specialist advice on larger projects.
- The administrative section will be advised of new personnel, job and responsibility changes and all the necessary clerical backup; again this is applicable to larger projects.

B. USER INVOLVEMENT

The object of the transfer to operational status is to shift responsibility for the system from development staff to the users. So users need to be involved in both testing and then transfer.

Note that, if the system is large and complex, then there is no reason why parts of it should not be tested and transferred at different times.

User Involvement in System Testing

Line managers will submit their own data and check the results of testing. This is incidentally, the most difficult area of testing, owing to the need for the current procedure to continue until the new system is proved and staff feel confident to undertake new tasks. There is also the problem of how staff can carry out normal duties, while at the same time being concerned with their new roles.

User testing and carrying out totally new tasks may well involve evening and weekend working. Checking outputs is not a familiar task, and the sheer amount of work involved may be daunting. Managers may find it hard to check output which is in an unfamiliar form. Despite such problems, though, attempts must be made to carry out tests involving the user as much as possible.

User Involvement in Implementation

At the implementation stage, managers will formally accept the system. They will be closely involved in clerical procedures and in staff training.

In operation of the system, the line management must know the new duties that their staff will be required to perform. They have to make sure that:

(a) Input data is being prepared correctly, and on time.

(b) New reports are being properly used.

(c) Their staff are able to use and understand the system.

During the project development period, the user management will have a great deal more work to do in helping with the new system.

C. CHANGEOVER STRATEGIES

Importance of Successful Changeover

We now move on to consider the actual changing over of the systems. We are changing from a **development environment** (old system working, new system being developed) to a **maintenance environment** (old system abandoned, new system working).

The changeover implies changes in working practices: from clerical to computerised, from centralised computing to distributed computing; from one type of machine to another; and so on. Staff tend to resent change, and so to ease the way they must be kept fully informed, and in a direct manner. Any individuals adversely affected must be told personally.

A perfectly sound system can be completely destroyed by poor changeover. To be successful, remember, changeover has to have the **support and involvement of managers and the co-operation of systems staff and users**.

Thus, prior to changeover, management must verify that the system does actually satisfy defined information needs; that the equipment, software and staff necessary for successful changeover are available; that control and audit procedures are in existence to ensure system integrity; and that performance requirements have been established for the system's assessment in operation.

It is the **analyst's** responsibility to ensure that staff information is complete and accurate, the object being to obtain co-operation and a smooth, trouble-free changeover.

There are two basic methods of changeover – direct and parallel – and some variations of these.

Direct Changeover

Using direct changeover, at a specified time the old system is switched off and the new switched on, as shown in Figure 11.1:

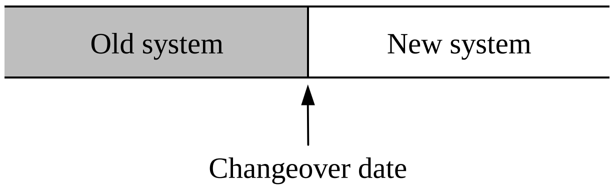


Figure 11.1: Direct Changeover

This is advantageous in that resources are spared – the method involves the immediate discontinuance of the old system. However, the new system must have been thoroughly tested so as to minimise risks in initial operation. Should the new system meet with unexpected problems – hardware, software, or design – then the old system may not be able to be retrieved.

As you will realise, this technique is potentially dangerous since it implies transfer of dependence from a current working system to a new system which, although tested, has not been used in a real situation. However, there are several situations where the technique is applicable or unavoidable:

- In very small systems it is often not worthwhile considering any other technique, owing to the inherent simplicity of the system.
- In very large systems it is sometimes not feasible to maintain two systems simultaneously (as in parallel and pilot running) owing to the work involved.
- Where there is little similarity between the old and new systems, the simultaneous running of both systems may be unhelpful.

When direct changeover has been decided upon, it is usual to carry it out at a time when work is slack, to assist staff to concentrate upon it. In direct changeover it is also important that everyone has confidence in the new system. However, direct changeover is probably the most fearsome for staff and it is not uncommon for absenteeism to rise sharply on the day of direct changeover or immediately thereafter.

Parallel Changeover

In parallel changeover the old and new systems are run with the same data until there is confidence in the new system, whereupon the old system is dropped. This is illustrated in Figure 11.2:

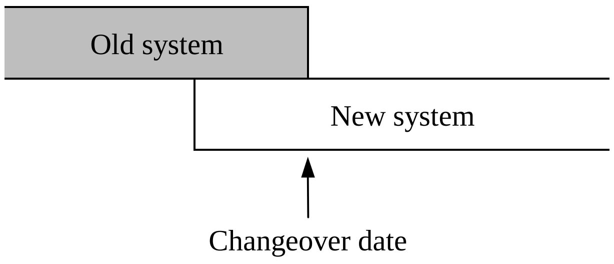


Figure 11.2: Parallel Changeover

Parallel changeover or parallel running of the old and the new systems simultaneously allows a comparison of output to be made between them. Any shortcomings of the new system can be rectified, and continuous cross-checks made. This is the most common method of changeover, but it is important to identify objectives, and a timescale must be established.

This method may be regarded as an extension of the testing of the new system, but this is really only a sound approach if the two systems are really comparable, a somewhat unusual condition in real life.

There may be problems in the cross-checks. What, for instance, is the significance of any differences between the old system and the new one which are discovered? Which is right and which is wrong? Unhappily there is a tendency to put the blame for differences onto the new system! It may be that differences arise merely from the greater sophistication of the new system.

Parallel changeover is often used so that the old system may still be operated when there is a breakdown in the new system. If this is the main reason then there must be a specific limit to the number of production cycles for which the parallel runs are to be carried out. What has to be remembered in this particular context is that running in parallel means double the cost.

Another problem concerns the staff and other resources used to run the two systems together. There may well need to be separate controls for the two systems, to be maintained and then reconciled. Where the reconciliation is difficult, the period of parallel running may have to be prolonged. A delay such as this could create tension and strain for the user department(s) because of the need to undertake two operations.

The objective should be to terminate the running of the old system as soon as is conveniently possible.

Generally you find that a database has to be created, or else existing files must be converted into a usable format, for the new system. The creation of files takes up a good deal of effort and resources. After the construction of the database for the new system is complete, there is the important question of maintenance. Any premature maintenance must be avoided, because maintenance of files for both old and new systems can stretch staff to the fullest extent.

Some specialists argue that it is beneficial not to carry out a parallel run, as this imposes more rigorous discipline on programmers and analysts to ensure that the system is viable. In any case, parallel running should not be used as a means of showing up inherent faults in the new system – programs should be properly pre-tested with live data beforehand.

Phased Changeover

Within the two basic methods discussed above we find a number of variations, of which the most common is **phased changeover**, where the new system is introduced in phases or stages as each stage is implemented and operating correctly. The phases continue until the whole system is in operation.

This method would be used for very large information systems which possess many complex components and which cross organisational frontiers.

The method consists of a series of direct changes. The implementation of each phase can be controlled, and risk to the user department is thus reduced considerably.

This method allows easier transfer of staff and is probably the most satisfactory method of working, where it is possible. It permits thorough testing under real conditions while limiting the risk of system failure. It requires, however, that part of the system functioning can be conveniently separated from the rest. It also requires some additional clerical effort in handling two different systems simultaneously. This method is sometimes called '**pilot running**' (see later), although note that pilot running can also be achieved under the parallel running method.

The great disadvantage of using any phased or pilot implementation is that users often have to wait many months, or even years, for a system to be available to them which completely fulfils their needs. Their needs may well change during the implementation period, and if their new needs are to be featured in the system, the final system may never be seen. This 'moving target' phenomenon experienced in many phased changeover implementations does much to frustrate and annoy both user staff and data processing staff, and suggests that the original system as a whole was ill-conceived.

Figures 11.3 and 11.4 show how phased changeover can be accomplished via either direct or parallel changeover. More complex combinations are also possible.

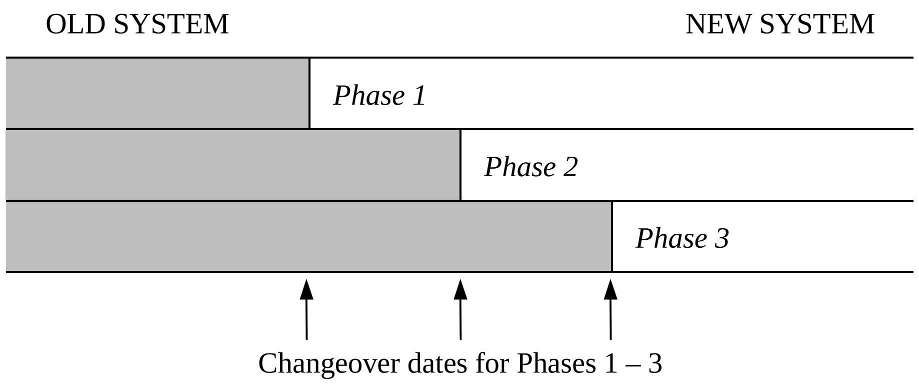


Figure 11.3: Phased Changeover Using Direct Changeover

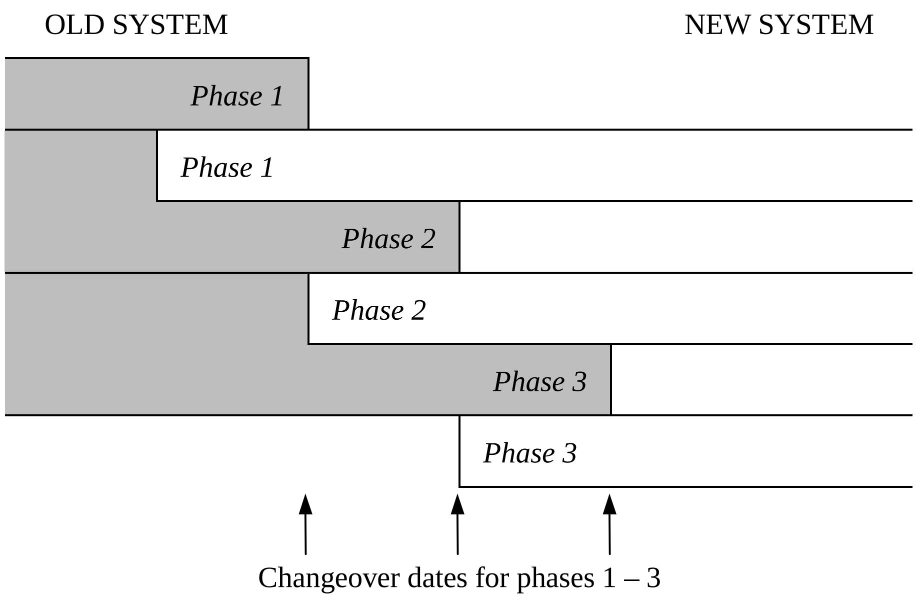


Figure 11.4: Phased Changeover Using Parallel Changeover

Pilot Running

Pilot running is usually used to mean where the new system is run under controlled conditions using **old data**, where a small representative part of the old system is used as a test area, as shown in Figure 11.5:

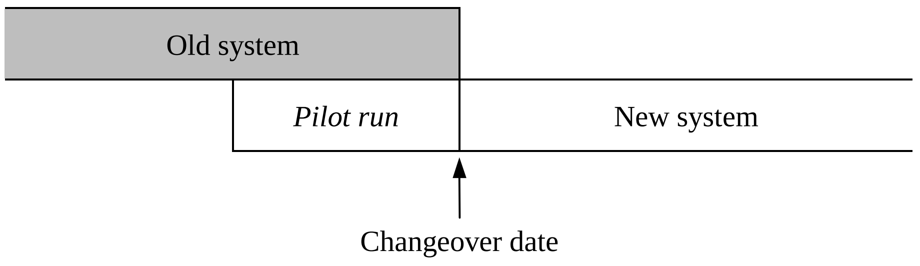


Figure 11.5: Pilot Running

This is a similar concept to parallel running but is less disruptive – data from one or more previous periods is run first on the old system and then on the new one. Timings are thus less critical, although realistic timing and data capture/conversion are not simulated.

Changeover Methods Compared

We will now give a brief list of the main advantages and disadvantages for each of the above changeover approaches.

(a) Direct Changeover

Advantages:

- This is the simplest method: stop one system, start another. It is usually only undertaken over a weekend or holiday period.
- No extra work in running two systems together for a time.

Disadvantage:

- Very high-risk – if the new system is disastrously wrong, it is difficult to recreate the old system.

(b) Parallel Changeover

Advantages:

- This is a safer method as the old system is kept operational while the new system is brought in.
- Much greater security.

Disadvantages:

- Greater effort is needed to run the two systems and to compare the outputs.
- It may not be very easy to revert to the old system should things go wrong. The new system may handle the information differently, making it awkward to compare outputs.
- The responsibilities of staff may well change between systems, leading to confusion.
- Knowing when to make the actual changeover. This is usually a compromise between too short a time, keeping costs to a minimum, and too long a time, allowing for extensive testing.

(c) Phased Changeover

Advantage:

- There is considerable control as only manageable chunks are being changed over at a time.

Disadvantages:

- The system may not easily be split into phases.
- The phases may well be different in the two systems.
- The interfaces between remaining old system phases and the new system phases already changed over, are extremely difficult to control.

(d) Pilot running

This is often the preferred method.

Advantage:

- Considerable control is retained and no risks are taken even if direct changeover is applied to each area.

Disadvantages:

- Time is needed to collect and collate the data to be used.
- The two systems may handle the data differently, making comparison of outputs difficult.

In practice a combination of methods is used. Rarely is a complex system subject to direct changeover, nor a simple system to any form of drawn-out parallel running. Whichever method is used, it is essential that data passed from the old system to the new is complete and accurate.

D. POST-IMPLEMENTATION REVIEWS

Project Review

Management will require a review to check that the system is up to requirements and to note lessons for future use.

In an ideal situation, two reviews would be made:

(a) At the implementation stage;

(b) Six months later.

The first is unlikely to be undertaken in full, as implementation of a system is an extremely busy time, and most DP staff feel that all the tests made will compensate for this. However, if an early review is made, then the essential review three to six months later has some guidelines to follow and data to compare with.

The review after a few months should be made by an independent consultant, who should assess whether the projected costs and benefits are being realised. He should check if the system requirements are being achieved and he should identify the strengths and weaknesses of the system.

The review consultant should be assisted in the review by an audit team who will undertake a parallel audit review of the system. In addition, a user representative and a representative of the development staff should be available for consultation as required.

(a) Topics for Review

The following points should be reported on by the review:

- The manpower estimates, compared with the actual manpower effort and skills achieved.
- The amount of machine time used during testing.
- The overall costs, analysed in detail.
- The delivery plan, with any differences explained.
- Lessons learned on the techniques and approach used for:
  - (i) project management;
  - (ii) technical aspects;
  - (iii) development techniques.
- Productivity achieved since implementation.
- Program sizes and any lessons learned here.
- Relationships between the DP department and:
  - (i) users;
  - (ii) other departments;
  - (iii) DP procedures;
  - (iv) other systems;
  - (v) the lessons learned here.

(b) Scope of Review

Having studied the reasons for a review after six months or so, and the aspects to be reported on, you should have a fair idea of the scope of the review. It is, however, worth setting it out formally.

The review consultant should examine:

- The documentation of the system, programs and operator and user procedures.
- The test packs developed for future maintenance.
- The test data, results and test logs already used and achieved.
- The errors found to date, the type of error, the department and program responsible, the cause and status of the errors.
- The user's opinion and view of the system – problems and benefits, deficiencies and training needs, and input and output.
- The design of the system and its operation; the interaction with other systems; the system's control functions and audit trail (see later in this study unit).
- Details of data capture procedures.
- A study of the operations department.
- full cost/benefit analysis, to decide if the system justifies its costs.
- Any recommendations to be made – enhancements required, operational changes, any changes to documentation, data integrity, testing facilities, controls, training, user procedures and personnel, and anything else that can be discovered.

The review consultant has a wide brief and is given plenty of time in which to carry out his review. He will inevitably find defects, but hopefully will also identify strengths in the system.

Quality Assurance Development Review

Quality assurance is the task of ensuring that the system is built to the specified quality and, as such, can be considered the responsibility of the technical staff. However, a separate quality assurance function is useful on all but the smallest projects.

We have already discussed some of the QA functions, which, broadly are:

(a) **Monitoring** – checking that standard procedures of working are adhered to and that standardised documentation is used. Monitoring involves producing independent reports on the current state of the project.

(b) **Advising** -providing independent expertise on any aspect of the project, particularly with regard to set procedures (such as requests for tenders, etc.) and standards.

(c) **Auditing** – carrying out formal checks to ensure that the system is working as specified.

At the start of a project, someone should be given responsibility for quality assurance with respect to the provision of user facilities and the satisfying of user requirements. This person should be involved from the feasibility study through to implementation, and beyond.

There also should be a quality assurance review made following the design stage. This is not the same as the full project review discussed above.

The QA review is to ensure that the design is valid. It is important to incorporate testing standards to ensure a high detection rate for potential failures as soon as possible in the system cycle, since the cost of correcting faults escalates rapidly as the cycle progresses. The following checklist may provide a guide to the quality assurance review:

- Have the objectives been met?
- Have the problems been solved?
- Have any new problems been introduced?
- Is the design vulnerable to change?
- What is the weakest link in the system, and which precautions have been taken to minimise the effect, should the system break down?
- What if the computer breaks down?
- Is the design practical for implementation and operation?
- Is the design dependent on key staff?
- Is the system economically viable?
- What are the security risks?
- What are the known disadvantages?

You should notice that this is really a risk assessment.
In the above list of QA functions, the first mentioned is "monitoring". We shall discuss this in the next unit and you will note that it is an ongoing activity throughout a system's life.
Also in the list of functions is "auditing". Whilst auditing may or may not formally be classed as a QA responsibility, it is closely related, especially in respect of internal audits.

Auditing and Audits

We have seen that quality assurance is an activity which must continue throughout the duration of a project – it must not be added on at the end. The management must ensure not only that standards are laid down but also that they are followed.

The term "audit" is often used, in a more general sense, to mean the independent examination of any completed work to ensure that it carries out the tasks required of it with the correct level of control to ensure that errors are minimised. For example, the coding of a program could be audited by a more senior programmer to ensure that standards had been adhered to and that the coding was correct.

It is a legal requirement that the financial transactions and the systems used to record and process these transactions are subject to audit, to ensure that they show "a true and fair view" of the activities of the organisation. It is also in the interests of the organisation to make certain that it is conducting its business properly. For this reason, many organisations have their own **internal audit departments**.

A computer system designer must accept that (even though the system itself does not require it) auditors will require outputs (usually printed), so that they may be satisfied that all is in order. With the advent of computers, the task of auditors is made much more difficult; no longer can they trace a complete transaction through a series of handwritten ledgers. Indeed, if they ask to see a sales ledger, they may well be handed a reel of tape. Auditors must, therefore, have control systems, themselves properly documented and containing such items as a pack of test data, as well as full details of the system and its controls (equally well documented).

(a) Work of auditors

The work of internal auditors can be divided into two categories:

- **Installation auditing** covering, in particular:
  - (i) security aspects, which will be discussed in the next study unit;
  - (ii) the evaluation of controls over the processing of data and, in particular personal data with regard to the **Data Protection Act**;
  - (iii) production of standards, best practice, etc.
- **Application auditing** where it is important that auditors retain their objectivity whilst providing as much assistance as possible to the development team.

Advances in information technology enable money, mail, police detective work, meetings, information enquiries and very many other activities to take place using computers and data transmission. Such transactions provide a whole host of confidentiality, privacy and security issues for management. Auditors are able to evaluate controls, pinpoint weaknesses or risks, and advise management of appropriate action to take.

The objectives of an external auditor are to assess whether the company's accounts give a "true and fair view" of the company's affairs and whether proper books of accounts have been kept. With computer systems, auditors will be concerned that control procedures for the operation of the programs and the storage of data have been incorporated in the system and are working properly. When a company is considering installing a computer system, the opinion of its auditors, with regard to the system being selected, can be very useful, since the auditors will know the type of controls which should be included.

When carrying out an audit, auditors will want to check the various control procedures which have been set up: these will include controls within the computer programs and also in the manual procedures which feed input into the system and ensure that output is handled correctly.

Auditors will be required to test the system (and its controls) not only during implementation and the actual changeover to the new system but also at intervals during the normal operation of the system. Auditors should have their own test data, containing both correct and irregular information, so that the system is adequately tested. They should be informed whenever programs or procedures are changed, so that they can ensure that all the control procedures are still effective.

(b) Audit Trail

Even if security procedures did not require it, auditors will ask that an "audit trail" be provided. This will be printed evidence, showing the passage of transactions through the system, giving not merely totals but also information as to the way in which those totals are obtained. The audit trail may make use of existing control procedures, and it will cover source documents; coded documents (from source); control totals; prints of master files; error messages generated during processing; and the processing log itself. In most cases, auditors will require their own copy of this printed information; it will not be sufficient for the designer to lend a report on a "see and return" basis.

It must be expected that during the operation of some procedures, something will go wrong and the run will be abandoned. It should be (but rarely is) common practice to have a processing log, in which the start and finish of every operation is recorded and remarks added where justified, such as runs which have been stopped before completion. These logs must be regularly examined by the auditor, to ensure that controls are adequate. It is highly probable that security precautions will have been introduced to ensure that sensitive or valuable output (for example, printed cheques) are not duplicated, or that duplicate copies are destroyed. It may not be so certain that a sales invoice which has been, say, cancelled because of a malfunction is correctly reinvoiced. Yet, there is a possible loss to the company in that event, comparable to the duplication of a payment.

(c) Operational Audit

After a system has been running for some time, an operational audit may be carried out to see if the real system comes up to what was hoped from it. For example, a stock control feasibility study may have shown that introduction of a particular system should reduce stock holdings by 50% while, at the same time, reducing the number of stockouts by 60%. These results will not be achieved in the first month's operation of the new system. After it has been running for a few months and stock levels have settled down, an audit might be carried out to see if the projected benefits have been obtained.

E. TRAINING

End-users will expect thorough training as part of their acceptance of the new system, and it is also in the interests of the organisation that their staff should be properly trained in its use.

There are three aspects to be considered in developing effective training:

- Who should be trained?
- How should they be trained?
- What levels of training are needed?

These questions can only be completely answered with a full knowledge of the system involved. But generally:

- Those who just need knowledge can attend a manufacturer's training school or attend targeted in-house courses.
- Those requiring a skill can go on specialist training courses, which can be conducted internally or by the product manufacturer.

The training should be provided in **relays**, i.e. only so many staff at a time, whilst the others continue to operate the old system and receive information and instructions relating to the new system. With careful planning, the training should not create many problems. At all times, care should be taken with staff problems and worries.

This suggested approach does have drawbacks though:

- The users who will actually use the system are precluded from early involvement.
- Considerable despondency results as the staff involved resent cramming facts about the new system at brief meetings.
- Before the system is even implemented, it can be doomed to failure if the training is not done properly and earlier groups of trainees report back the shortcomings to the scheduled later groups.

The way around the problem is simply systematic consideration and planning of training requirements, with **full communication** of all the training plans to staff at all levels.

Who Needs Training?

The short answer is, everyone who is involved with the system. All grades of staff from management downwards will, at the very least, need to be educated in the new system.

Training is best given just before any new duties are to be undertaken, so that the training is not forgotten.

Types of Training

The principles governing the appropriateness of different types of training include:

- Full-time training is best carried out away from the normal place of work, to prevent distractions.
- Generalised lectures giving background to the system should be given to all who are to be involved.
- Detailed training about particular aspects should be given to small groups and undertaken in situ, between normal work periods.
- Each type of training should cater precisely for its audience.
- On-the-job training can be beneficial under certain circumstances.

We can distinguish between **action learning** and **formal learning**.

- Action learning is learning through participation in change. By doing, people achieve a concrete feeling of what is possible or what the change will mean. This approach can be used to actually experience working with new technology.
- Formal learning is learning through a set of training and education programmes that are linked to the change sought.

The integration of formal learning with experience in action will frequently be effective in IT training.

Computer-Aided Learning (CAL)

CAL is widely used to deliver "individualised" instruction. It is most successful in teaching procedures and skills, such as how to use a software package. Also called CBT (computer-based training), it is a development of programmed learning books.

Advancing technology now permits CAL systems to be much more interactive – generating questions randomly rather than just in a fixed order, and allowing free-format answers rather than just multiple-choice selection as a result of rapid database searching. Intelligent CAL can modify the teaching strategy to suit the individual user, by modelling the pattern of the student's responses.

On-line training facilities can be called **open learning centres**, which may be defined as:

> Training facilities set up on site where training materials may be accessed at any time via CBT, multi-media, CD-ROMs or via a network for distance learning activities, ultimately leading to the **"virtual classroom"** using interactive training on the Internet.

The issues an organisation should consider when assessing the viability of using open learning centres and on-line systems for staff development and training include:

- It provides **'Just In Time'** training.
- May cut down the cost of sending personnel away for training.

- May save training costs/course fees, e.g. cost-benefit of using open learning or on-line which reaches a wider audience than traditional methods.
- CBT training fits around the working day.
- Setting up and managing a centre needs to be discussed:
  - (i) Full-time administrator
  - (ii) Automatic booking system
  - (iii) Evaluation of software/course used
  - (iv) Need to design and develop own support material for own specific requirements.
- A key advantage of this method of training is that it allows staff to train themselves on a **need-to-know** basis.

Training Plans

The employment interview is the first stage in the employment process, seeking to match the needs of an organisation with the experience of possible employees. The next stage is to develop employees' potential so that they can carry out a widening range of tasks. Some of these tasks will be ones currently being carried out by more senior employees but, with ever-changing technology, others will be new to the organisation as well as the employee. It is thus even more important to develop training plans, which should commence at recruitment and be revised at appraisal. These plans must be well thought-out so that the needs of the organisation are matched with each employee's aspirations.

Training should be related not only to the needs of the individual but to the organisation also. The appraisal systems will have identified the objectives for the short and medium term and the skills required to achieve those objectives. These skill requirements can be matched against those already possessed by the employee, thus identifying any training needs. It may be that some skills have not been used for a period of time and that a refresher course is all that is needed. Alternatively it may be a skill that the employee does not at present have, and thus a full training programme will be necessary.

Training plans, as for all plans, should be drawn up in a systematic, well conducted and cost-effective way. The development of a training plan is a six-part process:

(a) Specification of Training Objectives

This should cover educational and personal objectives, and the experience and ability required prior to training, as well as the training needed.

(b) Analysis of Training Needed

Individual training needs are assessed based on job descriptions and individual career paths to date.

(c) Design of Training Programme

There is a choice of standard college programmes, specialist organisation courses, specific short courses or specially created courses.

(d) Selection of Training Method

Whether training is to be on or off the job. Also a range of methods are available, from the traditional 'chalk and talk' to the use of interactive video and computer-aided learning. A one-off session for a small number of people would perhaps use a formal lecture followed by supervised hands-on training, whilst a need to train large numbers of staff on an on-going basis could warrant the investment in computer-aided training methods.

(e) Implementation of Programme

This covers booking courses and arranging job back-up and learning motivation. The timing of the training is important. The impact on current system operation should be minimised whilst at the same time ensuring staff are ready for the change. On the other hand, if training is completed too early then staff will soon lose some of their skills as there will be no opportunity to reinforce the training with practice, the new situation not being ready yet.

(f) Review of Training

This is a lengthy process of recording feedback from trainees, testing the knowledge required and eventually evaluating the trainees' job performance.

There are a range of training techniques available, and it is more effective to keep formal lectures to a minimum and to provide hands-on training as much as possible. Some lectures and talks are necessary, though.

End-user management should receive the same training as the hands-on end user.

Training and Competitive Advantage

Effective IT training will bring benefits not only to the staff but to the organisation as a whole. Sometimes the need for training is recognised but not followed through in practice. The organisation may thus miss out on the following kinds of benefits:

- Employees often judge an organisation on the opportunities/expectations for personal development, so they will be keen to work for one where these are good.
- Replacement skills for technological change: highly developed, flexible workforce.
- Organisations with a good training reputation are able to attract good recruits/improved work performance/increased morale and self-confidence/positive working environment/reduced error rate/less supervision.

Study Unit 12

System Maintenance and Security

Contents ***Page***

A. MONITORING

Monitoring is an essential activity in the routine running of a system. Monitoring of the system begins immediately the system is implemented and continues throughout the system's life. Monitoring is one of the principal activities of the Data Manager or Administrator. It involves keeping a close and formal watch on everything that happens within the database environment, such as the frequency of access of each user, the type and level of access, the most-accessed and least-accessed data items, failure rates, proper use and generation of documentation, and so on.
The results of monitoring lead to maintenance of the system.

Change Monitoring

There might be a number of reasons for changing systems. Assessing the results of such changes, or the prediction of the results for the system, is one of the purposes of monitoring.

Typical reasons for change – and, hence, monitoring objectives – are:

(a) Changed management with different requirements.

(b) New products incorporated in the system.

(c) Different manufacturing and administrative procedures being adopted by the organisation.

(d) All organisations evolve, bringing various degrees of change.

(e) External influences such as new legislation.

Threat Monitoring

As well as monitoring the effects of change, it is necessary to watch for signs of weakness in the system. This is called **threat monitoring**. Typical tell-tale signs of weakness are:

(a) Input documents not quite meeting requirements.

(b) Output reports not being used.

(c) Requests for speeded-up output.

(d) Complaints.

(e) Continual hardware breakdown.

(f) A high incidence of error.

(g) Too much being done manually.

System Appraisal

Yet another reason for monitoring is as a check that the system's actual performance matches the design predictions. This is known as **system appraisal**. This kind of monitoring is done at the early stages soon after implementation. Clearly, all the threat monitoring symptoms apply here. Also, cost must be a factor in these checks, as must the degree of satisfaction of the user.

B. SYSTEMS MAINTENANCE

Maintenance Problems

Maintenance of systems cannot be avoided but it can be made easier if ease of maintenance is designed in at the early stages of system development. In fact, maintenance is the principal reason for the employment of analysts. There is very little need for them to write completely new work.

Earlier in the course we discussed structured design of systems, particularly SSADM. All modern design methodologies use a structured approach, which requires comprehensive documentation, because not only does this give easier to develop and easier to test systems, but also it gives easier maintenance of systems. Structured methodologies lead to more or less self-contained program modules or subroutines. Each of these performs one processing task. It is therefore easy to identify the section of program requiring maintenance.

Whilst this will be true at the early stages of the operational system's life cycle, soon after implementation, it is unlikely to remain so and the maintenance team must be alert to the problems. Maintenance activities must be fully documented to assist future maintenance programmers.

When systems are changed, they become rather more complicated, cumulatively. Then, the user's requirements change, so that they no longer conform to those specified when the original systems were implemented. This, in turn, means that maintenance tends to be progressively harder, and uses more and more resources as the systems become older. As a rule, we can say that experience teaches that a system which is five to six years old is likely to need complete replacement, rather than modification.

This means that organisations which adopted computerised systems 20 years ago (or even a decade ago) have many "vintage" versions. While users simply see a service which is steadily worsening, the DP department is painfully aware that more and more of its efforts go towards maintenance – often 50%, or even more!

The introduction of new computer technology and new user aspirations adds to the difficulty of the situation.

Program Maintenance

Programmers tend to dislike maintenance as it can be very routine, repetitive and tedious. It is necessary though.

Whilst predictable changes should be built-in to the system, unpredictable changes such as latent bugs, law changes, changes in business practice will all have to be dealt with.

Maintenance Team

Large installations will probably have a distinct maintenance team. They have the responsibility to check the documentation of new systems at the handover stage and to act as a quality control unit on the work produced by the development team.

The maintenance team is an excellent training ground for new programmers and analysts. Sometimes the development staff rotate with the maintenance team for specific projects. This has the effect of spreading good ideas right through the development group.

In smaller installations, outside contractors are sometimes employed. In other instances a more ad-hoc approach is used with whoever happens to be available being allocated the maintenance task. This lacks continuity and is probably the worst option.

C. DATABASE MAINTENANCE

The reasons for database maintenance are similar to those for systems maintenance. However, as with everything else pertaining to the database, maintenance is controlled and authorised by the DBA.

Scope

Exactly what is meant by "maintenance" in a database context varies. In the early days of computing, it was used solely to refer to changes of content and structure of a database. Nowadays it also includes data definition and creation, file definition and generation, revision and conversion as well as update changes.

Means

Database maintenance is generally made using a software tool of an appropriate type. **Application generators** are useful, as are **impact analysis** tools which can predict the effect of changes. Most of all the **data dictionary** provides a comprehensive referencing tool. In theory, IPSE packages incorporate maintenance tools, but in practice they have avoided this activity. Because maintenance is the principal DP activity, this exclusion from IPSE has greatly hindered the adoption of IPSE.

Database maintenance is generally undertaken via the DBMS. As this is always a proprietary package, there should be a maintenance contract accompanying it. Thus any DBMS problems are covered for maintenance.

As with systems maintenance, the route to efficient database maintenance is via the documentation. It is the DBA's responsibility to see that all documentation is up-to-date and complete.

Database Revision

Occasionally more than simple maintenance of the database is necessary and a full revision is undertaken.

This involves:

(a) **Redefinition** of the logical data structures.

(b) **Restructuring** of the physical data storage to accommodate redefinition.

(c) **Reorganisation** of storage as a tidying-up exercise.

For example, suppose it is decided to divide a large department into two smaller departments. As a result the records defining the departmental structure will be redefined to show that it is now two departments and the appropriate personnel records will be linked into this new definition.

As a knock-on effect the physical storage of the records will be changed, as there is now a whole new department and new linkages have been defined.

Finally, the physical storage will be reorganised to aid the efficient use of the linkages.

In short, revision of a database has several far reaching knock-on effects, which have to be accommodated.

Database Back-up and Recovery

Databases are subject to much activity of a complex nature. From time to time, something will go wrong.

Typical examples are:

- A transaction error such as the wrong record being retrieved or a programming error being highlighted.
- The whole system going down due to power loss or software failure.
- Complete or partial database destruction due to some natural disaster or major operator error.

Whichever fault occurs, it is necessary that the database be **recovered**. To make this possible, a **back-up** strategy is necessary and a simple model is regularly to **dump** the database contents in a safe area (e.g. copy it to separate disks), and to keep a **log** of all subsequent transactions to the next dump, and so on. Clearly, the more frequent the dump, the faster and more complete the recovery. On the other hand, normal database processing activities must be suspended during the dump, thus inhibiting the processing.

It is the Data Manager's responsibility to decide on a balance between frequent dumping and uninterrupted processing. The balance will depend upon the nature of the data held and its security sensitivity; the number of data updates made as opposed to simple accessing of the data; and the urgency of uninterrupted processing.

There are various ways in which transactions between dumps can be logged. The transactions themselves can be logged or the state of the records being processed either before or after the transaction can be logged. The choice will depend upon the DBMS being used.

The following recovery techniques, are in general use:

- Following a transaction error, roll the database back (back out) to the state prior to the transaction using the original unaltered versions of the records and start again.
- Following a whole system failure roll back the database, as for a transaction error, through all the uncompleted transactions and start again.

Whilst it may be necessary to reapply failed transactions manually so as to avoid the original problem, the back-up and recovery mechanisms are contained within the DBMS and are activated automatically.

D. SYSTEM ENHANCEMENTS

Once a system is live, users frequently become more ambitious for their system. They will then ask the computer staff to add on extra functions. This would be an enhancement of the system.

Computer staff tend to be rather wary of such proposals, as they will be only too well aware of the uneasy balance that exists in any system, especially at an early stage of its life.

Enhancement or Development?

Every system will require enhancement. There is the choice of producing and developing an entirely new system. But even then, changes may be required to existing programs, as the new system will have to incorporate the previous system, and it will be sensible to use already prepared software.

Sooner or later, a choice must be made between enhancement (of the existing system) and development (of a new system).

(a) Reasons for Change

Several reasons for change lead to this point. Examples could be:

- General frustration with the limitations of the present system.
- The attraction of new technology.
- A relatively new system that has several severe design faults.
- Management decision on an expansion of the computing function.

It will depend very much on the particular circumstances.

(b) Points to Consider

There are several points to consider.

- Enhancement entails less risk, as it involves a proven system.
- Documentation is easier – most is already in use.
- Documentation can be quite informal, as staff are already familiar with the system.
- However, amending and testing existing code is far more time-consuming than writing brand new code.

After some time with a mature system, several enhancements may well have been made. Eventually, the user may be faced with inevitable development of a new system. Successive enhancements will tend to distort the original system quite considerably. An example of this would be a batch processing system that has been entirely converted to immediate access. The original system will have had other activities timetabled into it when the batch runs were not being done. There will now be considerable upset with the system going entirely open.

Software Packages

When the software is bought as a pre-prepared package (as they often are), it may not be quite right for the desired project plan, and so enhancement will be required to adapt the package to the exact requirements. Thus enhancement is not confined to mature systems.

E. NEED FOR SYSTEM SECURITY

Publicity is frequently given to damage to computer equipment through fire or flood, and to computer-related crimes. These breaches of security bring to our attention the need to protect computer systems. Breaches such as minor accidents, omissions and errors may seem insignificant in each individual case, but these can accumulate into serious loss. It is the responsibility of the design team to design systems which are secure against such events.

A computer-based system is a combination of many resources, all of which are required for the efficient operation of the system. These resources include hardware, systems software, applications software, files and data, documentation, people, data transmission facilities, etc. A computer system cannot be 100% secure, but measures must be taken to reduce the risk to an acceptable level.

Categories of Threat

There are two main categories of threat to security – accidental and deliberate threats. These can be split into the following sub-categories:

(a) Accidental Threats

These include:

- Malfunctioning of a hardware component
- Modification of software
- Naturally occurring threats such as fire/flood/earthquake, and interruptions or surges in power supply
- Death or injury of people, affecting their capacity to do normal work
- Interruption of data transmission lines

(b) Deliberate Threats

These are:

- Removal or corruption of programs/data
- Disclosure of information
- Withdrawal of labour

The following order of threats was produced as the result of a study by IBM. It is significant in that over 50% of all threats originate from errors and omissions:

- Errors and omissions (over 50%)
- Dishonest employees
- Fire
- Disgruntled employees
- Water
- Intruders and others (less than 5%)

As you can see, most of the security breaks are of the accidental variety, but these are harder to detect. Examples are when a programmer accidentally overwrites someone else's file or program, or when a machine or operating error causes the destruction of a direct access file. However, if we concentrate on the other type of security hazard, the wilful or mischievous, we will almost certainly be protected against the casual or involuntary threat.

There is an increasing problem in the proliferation of on-line terminals and readily learnt programming languages which make the computer more accessible to more people than was the case in the past. Thus it is most important to ensure that the operating system has means of identifying the users and logging their activity. No on-line terminal system should allow users to log on and off from the system without some reference to the activity being stored in a journal file to monitor, at the very least, connect time.

Sources of Breaches of Security

(a) Accidental Damage

We cannot ignore the possibility of fire, water or explosion in the building where the computer installation is sited. This may not be directed against, or caused in or near, the computer itself but the machine may be at risk in the overall damage. A fire may destroy not only the machine, which is replaceable, but also the magnetic media storing programs or data, and also the documentation of the programs. These losses may be irretrievable or, at the very least, expensive.

Accidental damage can also be done to equipment or storage media through physical threats such as dust, magnetic influence, high temperatures or dampness.

(b) Accidental Errors

These may be the mistakes of programmers, operators or users, which somehow cause an error; for example, loading the wrong tape, loss of output, or errors in despatch of output. They may be the result of a latent bug in the computer program which slipped through the testing phases and only reveals itself after many successful runs. (Users who inadvertently leave their terminal logged on when they are not present may unintentionally open the way for someone else to sabotage the system.)

(c) Dishonesty

This may take the form of the removal and holding for ransom of master tapes/disks containing the programs required by the company, or of the corruption of a payroll or payments system, or even of stock control, to enable an employee, or a confederate, to embezzle sums of money. Alternatively there is a possibility that exception reports may be suppressed, so that the existence of some condition, such as unpaid bills or excessive credit limits, is not reported.

(d) Sabotage and Espionage

This heading may include both the financially motivated operations of someone acting on behalf of a competitor to gain access to information or to destroy records belonging to the company, and the actions of a disgruntled employee seeking revenge over some grievance, such as sacking.

There was a classic example of this in an American company which gave one month's notice to their file librarian. At the end of the period of notice the employee left, and the company discovered that all the disks had been recycled in such a way that all their files were lost. So great was the chaos (the creditors would still send in their accounts, but debtors would not pay without notice) that there was a loss running into millions of dollars, and the company eventually had to consider liquidation.

The third area in this category is trade union activity. A small number of key employees in the computer room can, by striking, "hold the company to ransom". Failure to send out the quarterly accounts would cost more money, in loss of revenue and cash flow, than perhaps 10 years' salary for the staff concerned.

(e) Mischief

We have included a separate category here which is really part of sabotage but which is more difficult to detect and deter. There is a certain intellectual challenge in trying to "beat the system". This syndrome is seen in the current craze for **hacking** – the deduction of secret access codes to allow telephone calls to be made to an information base.

In one English school, the children were given free access to a timesharing system on the massive computer system operated by a local company. One enterprising group of boys devised a means of logging on to the system even in the periods when they were supposedly barred, and they wrote a special program which, in turn, loaded more and more programs simultaneously, monitoring the progress with relevant messages on the terminal, until they had monopolised the processor to such an extent that they effectively brought all other work to a standstill. There was no criminal intent and no thought of vandalism; it was part game and part "research".

(f) Viruses

In recent years we have heard much about computer viruses. They are particularly prevalent in personal computer systems.

A computer virus is a self-loading program which automatically spreads itself to every disk and personal computer machine that it makes contact with. It loads itself into the personal computer's system software. Once there it will be dormant until it is triggered by some event. This can be a date (for example, any Friday 13th) or particular classes of password, or just by logging on, and so on.

Once activated, the virus will cause problems. Some merely give a harmless message; others cause all the screen output to fall to the bottom of the screen; others wipe files clean; others corrupt the data files, and so on. Once a virus is lodged in a machine it is not an easy task to get rid of it. And if the machine is part of a network, the virus will almost certainly spread to every other machine in the network. Given that many networks are international, in a very short time a virus is spread throughout the world.

All kinds of precautions should be taken against viruses. For example, use a special program to check the files on a disk before they are activated. If a virus is found, destroy the disk. Always obtain software from reputable sources, but even then check all disks before using them. In an earlier study unit we listed some control measures to prevent viruses or other illegal codes entering a system.

On no account be complacent about viruses. Their effects can be disastrous and they are endemic in computer systems.

F. SECURITY MEASURES

Security measures for a computer system should be designed and specified as early as possible in the project cycle. It is often uneconomic or impractical to add security features to an existing system.

Prevention and Detection of System Errors

There are, basically three categories of system error which need to be guarded against:

- **Hardware errors** may be transient or permanent. Transient errors are normally detected, overcome and reported by the operating system. Recovery from permanent errors has to be treated in a similar way to program and system error recovery.
- **Software and program errors**, should not strictly exist. However, it would be nonsense to expect this to be the case. The very nature of software errors makes them unpredictable, and in every case the recovery procedure may differ.
- **Invalid data** is the final class of error. Under no circumstances should invalid data cause an unscheduled halt in a program. Errors in source data, input data and output data should be detected by validation procedures or programs, usually involving sampling, prenumbering limit checks, etc. Stages of data control in a typical system are shown in Figure 12.1.

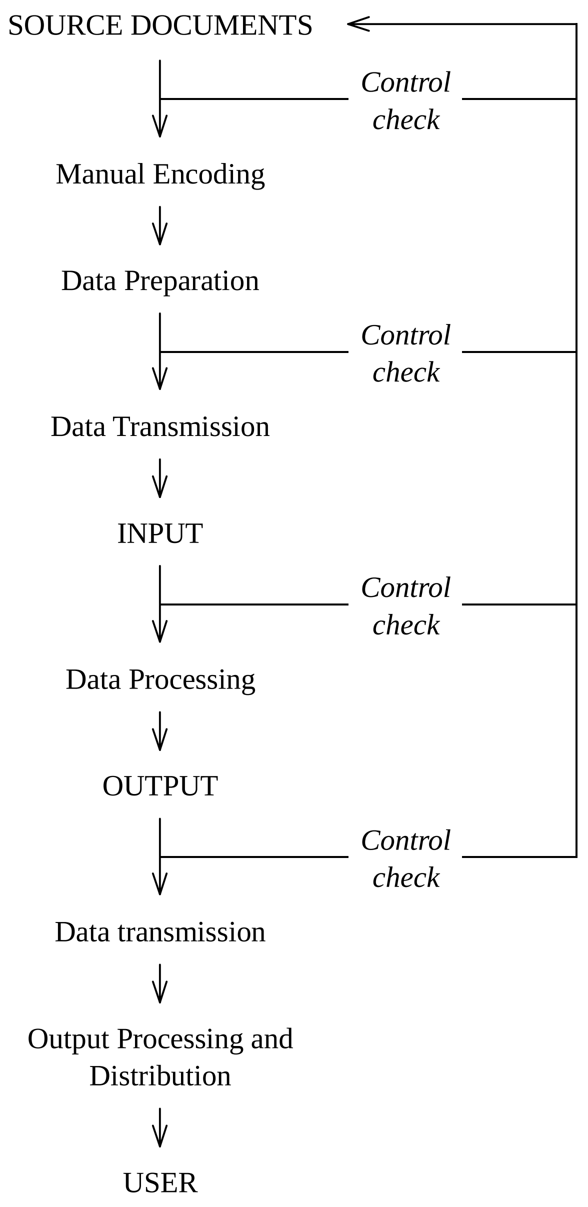


Figure 12.1: Data control processes

Administrators can assist in achieving successful operation of IT activities and enhancing security according to the following guidelines:

- Division of responsibility between users and those responsible for the system installation
- Monitor staff performance (can be achieved by supervision)
- Training is essential
- Users should only be able to access their own control records
- Co-ordination should develop from work schedule and monitoring of progress
- Documentation should be effectively maintained
- Protect confidential data
- Protect hardware and software

The adoption of industry standards provides a considerable security protection. For example, standard development and testing ensures the reliability of the system, standard documentation enables full checks to be made without misunderstanding or confusion, standard operating procedures are designed to minimise misuse and careless use of hardware and of disks, and professional standards among the staff ensure the highest quality of personnel.

Staff should always "double-up". No part of a system, whether in development or operational stage, should be reliant on only one member of staff. For obvious reasons, this is a recipe for disaster. At all times, at least two and preferably more members of staff should be familiar with development stages and be involved in operational procedures.

Not only does this give protection against fraud (though admittedly, not absolute protection), it also gives protection against staff absence.

Physical Security

Failure of physical security is common. Sound buildings and regular maintenance of both premises and equipment are essential.

(a) File Backups

To safeguard files and data continuity, there must be an effective system providing several generations of backups (backups to backups) for all files. In the event of a system failure, systems could then be restored.

Whenever a program is implemented or amended, a security copy on disk or tape should be stored in a secure, off-site place. There should also be a complete and up-to-date copy of all associated documentation. Information held on magnetic media does not last indefinitely. Thus, when copies are likely to be held for a number of years, procedures for re-recording at suitable intervals must be made.

Data security is provided by copies of master files and, for complete security, a copy of each file should be kept off-site. These copies should be taken at regular intervals, and the best way of taking back-up copies of data and how often they should be taken must be considered when the system is designed. Adequate time must be allowed for security procedures in the daily running of the computer system. These precautions apply just as much to a PC system running applications vital to the business as to minicomputer and mainframe installations.

It is important to practise data recovery procedures which, of course, should be fully documented.

(b) Power Sources

Computer equipment should, usually, be run on its own line, and the earthing arrangements must be examined carefully. Specialist knowledge is required if random, rather obscure errors are not to occur because of an inadequate electrical earth. If a "clean" power line is not possible, then using a constant voltage transformer (CVT) will enable the current to be stabilised by "smoothing out" spikes (see below) as they occur within the supply. Even a photocopier, for example, would cause sufficient power fluctuation to disrupt a computer system.

The two main problem areas are possible breaks in the power supply and "spikes" – transient disturbances which last only a fraction of a second but which can be very large in amplitude. The latter, which can be caused by switching on or off of large pieces of machinery (or even, sometimes, equipment such as a photocopier), can cause peculiar data errors or corruptions and unprogrammed jumps in the sequence of software instructions. The local electricity board should be asked to investigate if unexplainable faults occur on a random basis.

(c) Security of Access

In terms of physical security, we are concerned with access to the computer environment.

There are a number of standard procedures that may be adopted – for example:

- Standard burglar alarm equipment is available, and relatively easy to install.
- Strict control over visitor and staff access – staff entering must be identified.
- Minimal number of entrances, ideally only one.
- Use of guards or receptionists.
- Closed-circuit TV to monitor access and movement.

The more complicated question is access to the computer environment by employees. The use of access control mechanisms such as card keys and key pads on doors is one answer.

However, necessity for access should be questioned. It is not necessary for application programmers to enter computer rooms as they work on terminal PCs which can be sited anywhere. Similarly, analysts work either within the user's own environment or at a PC. Only operators and systems programmers (who program the computer's own internal software) require access directly to the computer room.

Even then, a system of authorisation can be installed, either via key cards/pads or simple "signing-in".

(d) Control of Keys

Losses of keys are often not reported simply because it costs little to get another key cut! Duplicate keys may be readily available in cabinets and desk drawers which are themselves not locked. Spare keys are often labelled invitingly to indicate their use. Locks and padlocks often bear serial numbers which, when quoted to manufacturers, permit extra keys to be issued. Another common feature is that a single master key is used to open all or most locks – so that this key, in the wrong hands, virtually puts the entire system in jeopardy.

Key control should be introduced as an added security measure. Keys should only be issued on signature to authorised staff and should be periodically checked to see if they are still held by those staff. Each separate set of keys should be clearly and individually marked – it has been known for a single set of keys to be passed from hand to hand by members of the staff so that an inspection of individual key holding was satisfied!

It is better not to issue a key when this is possible. Instead, the main key holder should go with an authorised member of staff to unlock the door himself or herself, making sure that the door is subsequently locked again. When a member of staff leaves the organisation, the return of any keys held must be part of the normal clearance procedure.

(e) Fire Hazard

Fire and smoke constitute a major hazard. Automatic fire detectors (e.g. smoke detectors) and controls can, and do, malfunction and are often slow to react properly. If a fire-quenching system (e.g. halogenated hydrocarbon) is installed, or if appropriate fire extinguishers are provided, human reaction can still cause problems. Actual combustion (as opposed to burning) can cause more damage and the introduction of large quantities of water will often damage media maintained in a humidity-controlled environment.

(d) Environmental Control

The power consumption of modern machines is very much smaller than for older ones, so that air-conditioning is becoming less necessary, and it is more likely to be installed for the comfort of the staff than as a machine requirement. However, the tolerances on air temperature and humidity should always be checked with the supplier, who will often send round an engineer to advise on the installation of the machine. Air-conditioning may be advised in order to reduce the amount of dust in the computer room.

Anti-static carpeting is often used on the floor, to prevent a build-up of static electricity, which can play havoc with magnetically-recorded data.

The insurance company must be consulted for its approval of the proposed computer room. The company will, probably, have strict standards for fire detection, extinguishing equipment, and the construction materials to be used. A security system to prevent unauthorised access is also necessary.

Also to be considered are vibration, lighting, decoration, sound-deadening and sound-proofing.

Data Security

Data must be protected against:

- Accidental modification or destruction
- Intentional modification or destruction
- Unauthorised disclosure.

Restrictions on access are the main precautions against these threats. However, whilst such measures are all important, it is important to realise that they may not be sufficient. Problems may also arise during “normal” running of the system, and data can be acquired for unauthorised purposes by:

- Producing extra copies of print-outs, either by re-running or by photocopying;
- Reading of sensitive output;
- Availability of waste output;
- Unauthorised copying of data files – for example, to diskettes;
- Using data from the live files for testing – this can also lead to accidental modification or destruction of the data, and it is a practice to be completely forbidden.

Considerable care should be taken over the reliability of staff members who have actual access to data and systems. Only trusted staff should access the most sensitive data. In addition, measures should be adopted to prevent staff who leave the organisation, disclosing sensitive data and procedures to others. Whilst this is clearly against the law, it requires positive precautions to be taken.

(a) Restricting Access to Data Transmissions

The increased use of data transmission, often over public lines, has increased the vulnerability of data. One means of ensuring that sensitive data does not fall into the wrong hands is to make use of **encryption techniques**.

Encryption involves the data, either in storage or (more commonly) in transit, being encrypted or translated into seemingly meaningless code, so that someone with unauthorised access to the data cannot make sense of it because he does not know the encryption algorithm. When the encrypted data arrives at the destination it must be decoded using the same algorithm.

(b) Passwords and Answerback

This constitutes the simplest method of security. However, it is not as secure a control as it is often considered to be. The principle is simple enough. An identity (a number, as a rule) is keyed in and then the system seeks the appropriate password from the user. When this is keyed in it must not be "echoed", or it should be overprinted so that it cannot be read over the user's shoulder (or retrieved from a waste-paper basket!).

Two questions arise, however:

- How is the remote user to be given the password? By letter, or telephone?
- What action can be taken to prevent the password from being known to unauthorised persons at this particular stage?

Also, the system must necessarily store a list of passwords, so that it is able to compare incoming words with this list. There have been instances where the complete list of passwords has been improperly extracted. The passwords themselves are often scrambled in some way to offset this possibility.

Perhaps the biggest security threat may come from the user himself or herself, through carelessness. If the remote user prints the password (as has been known!) by the side of the terminal, this largely negates the security attempts. A common example of this is found in the banks' cashpoint or ATM systems where a card is inserted into a machine and a PIN number (password) is keyed in by the card holder, to request cash from the machine. Some card holders actually write the password on a piece of paper in the wallet in which the cash card is kept – thus being extremely helpful to pickpockets!

The **answerback** system on the terminal may be regarded as perhaps the most cost-effective system in our present context. After setting up the link by using the normal routine of the password, the system calls the remote terminal and uses the answerback drum to generate a specific fixed series of characters. A similar method disconnects the original call and then re-establishes it by dialling the identified subscriber.

The setting-up procedures are often called **extended handshake** techniques. These do safeguard against unauthorised users who obtain a password and then try to use this from a terminal other than the appropriate one.

(c) Restricting Access to Specific Files

Control methods include the provision of an access authority level for each file. Users of the system are given an authority level associated with their password. The level is compared with the authority level required for each file access, and if the password level is greater than the file level, access is denied.

Aspects of Network Security

Our above discussion of computer system security includes networked systems, and we have mentioned the need for data encryption to protect transmitted data. We also need to consider some specific points relating to intranets and the Internet.

(a) Privacy of E-mail

A key point for employers is that they need to establish clear employee expectations with regard to e-mail privacy. A clear policy is required so that all employees are aware of the following:

- Who is legally able to read your e-mail?

- What are the different levels of access to the system?
- Why a monitoring system is being implemented, e.g. for users' satisfaction and systems usage.
- Setting of standards (tone, language usage, culture etc.) of e-mail messages within and outside the company.

This policy should be communicated to employees so that they fully understand the issues involved.

(b) Control of Web Pages

We have noted how easy it is for companies or individuals to set up Web Pages. As you might expect, this can present considerable security problems, particularly if individual employees of an organisation start setting up pages on their own. The dangers of this include:

- Possible inaccuracy of information provided by an unsupervised individual.
- Violation of corporate standards for Web Pages, and possible harm to the company's image.
- Providing an access route for hackers to gain entry to the company network and possibly damage the system or obtain confidential information.
- Employees may conduct private business on the Web during company time.

These dangers can be minimised by having:

- Corporate **firewalls** – these are complex pieces of software to protect private intranet sites. Possible breaches in firewalls can be avoided e.g. by having fax modems controlled through a server.
- Effectively controlled corporate standards for web site development.
- One prime home page only.

Contingency Planning

Practically all precautions against disasters, major and minor, will involve some cost. Therefore, in order that money is spent where it will be of most benefit, the management must determine where the system is most vulnerable, and where sensible precautions will reduce the risks in these areas.

Contingency plans should be set up to cover the various levels of disaster which might occur. Recovery after data has been corrupted, for whatever reason, will mean starting again, using back-up copies and, perhaps, entering some transactions again. Complete destruction of the installation by fire, flood, etc. will mean using security copies of the data and programs on another computer system. Where the processing is very critical to a company, the organisation of a stand-by system may be considered.

(a) Standby

As many of the accidental threats to physical security are very real and have a considerable impact should they happen, many organisations arrange a standby agreement with a similar site. These arrangements range from informal mutual assistance to formal contractual agreements. However, only the formal agreements can be relied upon.

The standby may be for only one processing function or application or it may be for the whole computing system. Whichever is the case, the standby must be fully tested and kept up-to-date so as to take over at short notice.

All updates to the main site hardware or software must be duplicated in the standby. Copies of all processing runs should be passed to the standby. In fact, if the standby is going to be really effective, then real activity must take place at it and very regular testing needs to be undertaken.

A common arrangement nowadays is to have more than one processor connected to a common store and common set of peripherals. They can do duplicate work, or one can be devoted to on-line work and the other to background work but ready to take over if the on-line machine malfunctions.

(b) Contingency Plans

More generally than just standby facilities, an organisation should have some form of disaster contingency plan. Several subplans that address specific contingencies may be better than a single large contingency plan.

Typical subplans include:

- ***The Emergency Plan***

  This will specify those measures that ensure the safety of employees when disaster strikes. The measures include alarm systems, evacuation procedure and fire suppression systems.

- ***The Backup Plan***

  The organisation should make arrangements for backup (standby) computing facilities in the event that the regular facilities are destroyed or damaged beyond use. These arrangements constitute the backup plan. Backup can be achieved by means of some combination of redundancy, diversity and mobility:

  (i) Redundancy – duplication of hardware, software and data facilitating continuous processing by the backup system

  (ii) Diversity – by having spare computer centres, risk of loss is reduced

  (iii) Mobility – may involve: reciprocal arrangements; hot site – complete facility; cool site – constructed at a separate location – is effectively an empty shell without computers.

- ***The Vital Records Plan***

  These are the documents, microforms, disks etc. which are necessary for carrying on the organisation's business. The vital records plan specifies how the vital records will be protected. This will also detail siting of backup copies. May involve physical or electronic transportation.