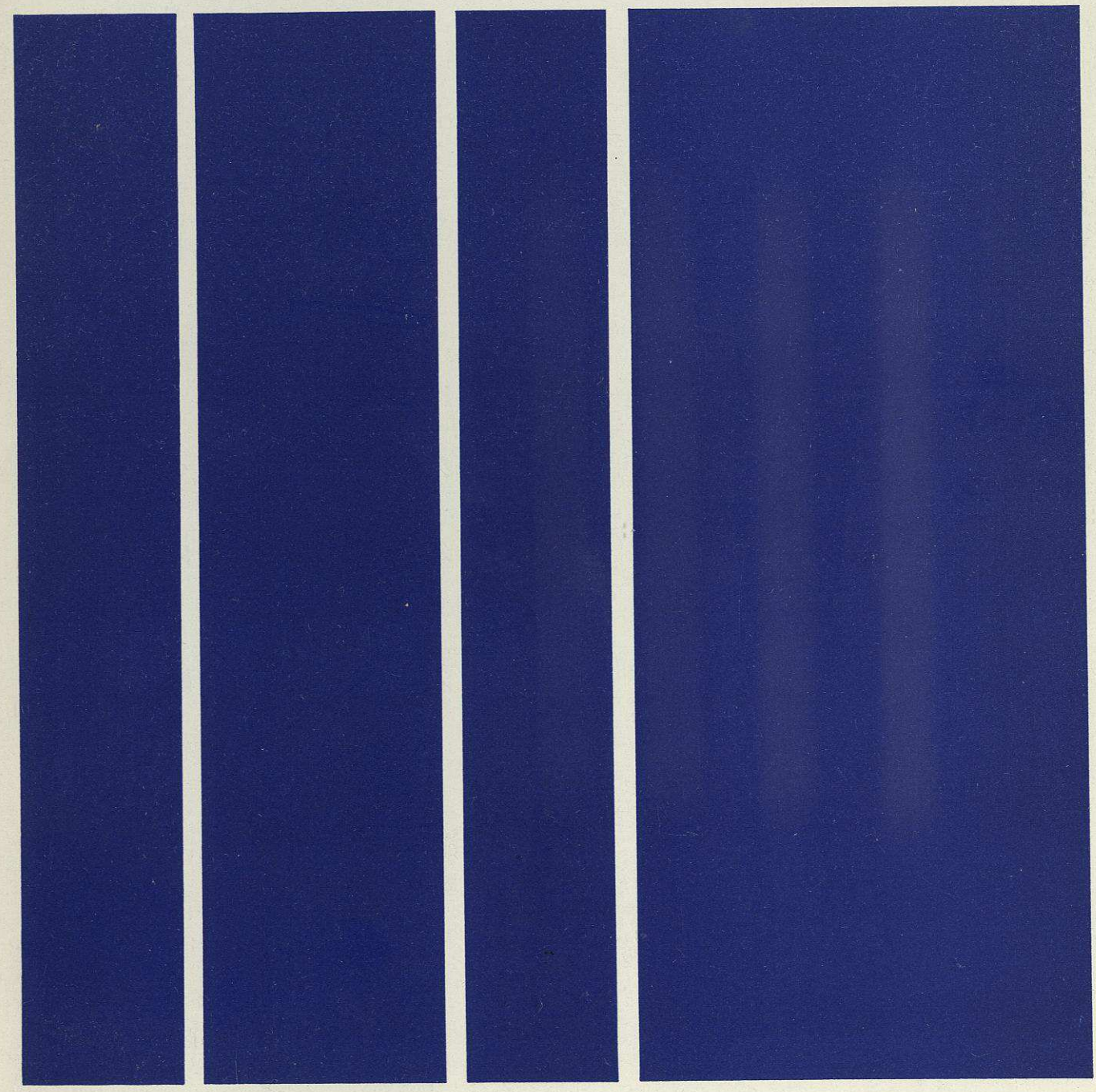
Report Series
No 8

Project Management

June 1978



The Butler Cox Foundation

Abstract

Report Series No. 8

Project Management

by Hamish Donaldson

June 1978

Many systems projects fall short of the expectations of both end users and management services managers.

While the manifestations of failure or partial success are usually technical ones in many cases the underlying causes are concerned with the management of the project.

This report sets out to explain a set of rules for managing projects which are known to work well in practice. They are applied to both large and small projects, using a variety of equipment.

These rules are largely concerned with the structuring of the project – both to minimise management 'interference' and to minimise the penalty to be paid for failure in any one area.

The content of the report is detailed, and embraces a number of technical issues which are encountered in system development projects, since the effective management of computer projects does require an understanding of these matters.

The Butler Cox Foundation

The Butler Cox Foundation is a research group which examines major developments in its field — computers, telecommunications, and office automation — on behalf of subscribing members. It provides a set of 'eyes and ears' on the world for the systems departments of some of Europe's largest concerns.

The Foundation collects its information in Europe and the US, where it has offices through its associated company. It transmits its findings to members in three main ways.

- as regular *written reports,* giving detailed findings and substantiating evidence.

- through *management conferences,* stressing the policy implications of the subjects studied for management services directors and their senior colleagues.

- through *professional and technical seminars,* where the members' own specialist managers and technicians can meet with the Foundation research teams to review their findings in depth.

The Foundation is controlled by a Management Board upon which the members are represented. Its responsibilities include the selection of topics for research, and approval of the Foundation's annual report and accounts, showing how the subscribed research funds have been employed.

Report Series No. 8

PROJECT MANAGEMENT

by Hamish Donaldson

June 1978

Butler Cox & Partners Limited Morley House 26 Holborn Viaduct London EC1A 2BP

TABLE OF CONTENTS

Acknowledgement

The material used in this report is to be published in a book entitled 'A guide to the successful management of computer projects' by Hamish Donaldson. The publisher will be Associated Business Press and it will be available in Autumn 1978.

I. INTRODUCTION

Many systems development projects fall short of the expectations of both end users and management services managers. While the manifestations of failure or partial success are usually technical ones in many cases the underlying causes are concerned with the management of the project.

An important aspect of project management is the structuring of tasks so that work can be delegated. We want to avoid continual management 'interference' and fussy supervision. We recognise that mistakes will be made but by choosing the tasks correctly we minimise any risk of overall failure.

This report shows how to apply this style of management to the development of computer projects. It recognises that the individual members of the project team are likely to be very different. They have different skills, different levels of competence and require different degrees of supervision. So the tasks need to be cut to suit the individual.

It is recognised that team members work best in an organised and stable atmosphere; an atmosphere where the ground rules are clearly established in advance and do not change to suit the whim of managers. It turns out that if we are to deliver a high quality product on time and within budget, *there are not many ways of doing it right.*

Assembled here are a set of rules which work well in practice. They have been applied to large projects and small, on mainframe computers and minis, for batch processing and on-line systems. Generally, computer projects have a high failure rate, but it is through no shortage of brains, effort and motivation. It is inexperience (and cutting corners) that leads to trouble.

If you follow the rules thoroughly, leaving out none of the steps, you will be a long way to achieving success.

II. AN OVERVIEW OF COMPUTER PROJECT MANAGEMENT

When a project 'goes wrong' it is all too easy to blame the staff working on it at the time – like the unfortunate Greek messenger. The Ancient Greeks had the understandable if irrational habit of executing the *bearer* of bad tidings.

So it is with computer projects (character assassination if not physical assassination). The symptoms of failure in a project are late running, cost escalation and a feeling of being 'out of control'. The causes are most commonly incorrect design (business problem wrong or technical solution wrong leading to rework), inadequate planning and possibly poor programming.

But notice when the problems become identifiable. It is towards the end of the project, in system testing, and the people most active at the time are of course the programmers. As a result, a great deal of attention has been directed at improving programmer productivity, at improving the construction of programs and at making the programs easier to maintain; all laudable aims but getting nowhere near the root cause of the problem.

The problems began much, much earlier – possibly as the project was being set up. So before looking at the techniques of project management let us establish a framework which is likely to lead to success.

A A Framework For Project Management

All of our education and training suggests that the best way of ensuring success is to get a good start; but a good start on a systems project is almost always a slow start. Systems are run by people and the relationship is complex. As a result things are not always what they seem.

It is necessary to spend time understanding the real problem, the work needs to be done by a senior and experienced analyst. *Use your best staff for the initial investigations* – never use trainees.

Structuring Project Responsibilities

A computer system is not an end in itself. It only exists to support a business need. It follows that the line manager who is going to run the system must be involved in the design – in fact more than involved; committed to its success.

The User Executive is a key appointment. He should be at director level and have line responsibility for the ongoing business system. His role is not full time and he will delegate day-to-day involvement to members of his staff. He has, however, overall responsibility for the project, for ensuring that the business needs are correctly stated, and that the user department wholeheartedly support the new system.

If a suitable User Executive cannot be found the project should be stopped. If nobody will sponsor it at this stage they are even less likely to later on when the going gets tough (as it does).

The Project Manager is the full time leader of the project. To be successful he needs to have a good technical knowledge, a sound business knowledge and be good at project management. Usually he is supplied by Management Services and it is up to the User Executive to ensure any gaps in his business understanding are remedied.

However good the Project Manager, he always benefits from a second opinion. The job of the *Supervising Manager* is to provide such a second opinion. He has a relationship with both the User Executive and the Project Manager that allows him to counsel in three specific areas:

- solving the right problem (has a greater understanding caused a shift in requirement?)
- assisting with project planning (planning to avoid problems)
- ensuring high quality work (to agreed standards of performance)

Let us be clear on the responsibilities:

The User Executive is responsible for:

- correct statement of business problem
- right level of user involvement
- satisfying, eventually, the user needs

The Project Manager is responsible for:

- day-to-day running of the project
- all staff working on the project
- meeting time, cost and quality objectives

The Supervising Manager is responsible for:

- quality control
- providing a second opinion
- obtaining resources

Organising the Project Department

A computer project requires a wide variety of skills:

- business analysis
- systems design
- systems analysis
- programming
- organisation and methods
- work measurement
- technical authorship
- operations
- training

The project will also require more staff in the middle and end than it did at the beginning. While we should strive for continuity we can see that team members will change during the life of the project. All of this instability creates something of a man management problem, as by their nature people prefer a stable reporting environment.

The solution is to distinguish line management of the staff (pay and rations, development and progression) from the project management (mixed skills to solve a particular problem).

When staff appraisals are carried out the project leader reports performance on the job, while the line manager reviews performance and counsels on personal development and career progression. In practice it is also possible to arrange that most of the staff reporting to a given manager are also on projects which he supervises.

Establishing Clear Ground Rules

If we are to get high quality work we cannot have all the members of the project team inventing their own rules as they go along.

We need to identify the various activities that need to be done and make sure that we set about every one in the best possible way. Whatever the standards are for (writing programs, system testing, writing procedure manuals, survey reports etc) there are common aims. The standard should be:

- simple to understand and work
- comprehensive, allowing no gaps in the analysis
- clear, uniform and unambiguous in application
- enforced

Setting standards of performance (which is what this is all about) is a key responsibility area for the Head of Department. He must think carefully before introducing them, because once introduced they must be enforced.

The corollary is obvious. If you are not prepared to enforce them there is no point in wasting time on a standards manual.

Selecting the Right Projects

What is the right size for a computer project? The question is not quite as silly as it may look at first sight – and there is an answer:

> Not much bigger than the last project we completed successfully.

Computer projects can be extremely complex and they always demand a high degree of technical and management skill. As they grow in size the complexity grows in a way that seems to be exponential.

There is another reason also for limiting their size. It is difficult to maintain enthusiasm and momentum for an interminably long project – and it could be that by the time it is introduced it is no longer solving the right problem.

Experience suggests that most projects should show results within one year and even the most complex within two years. That is not to say that very large projects are never undertaken; but a large project should be broken up into sub-projects, sub-projects which can stand alone and demonstrate progressive achievement.

This idea of breaking up a large problem into a more manageable size is a key to success in project management. We will be applying this principle at all levels of the project.

It is not quite as easy as it looks, however. If the division of work is carried out badly there may be too much interaction between the components – instead of within them. Getting it wrong adds complexity instead of removing it.

Understanding the Nature of Computer Projects

Computer projects have a creative phase and an execution phase, and the two are very different. The creative phase, at the start, is concerned with identifying the problem to be solved and deciding on a good technical solution which meets the real business need.

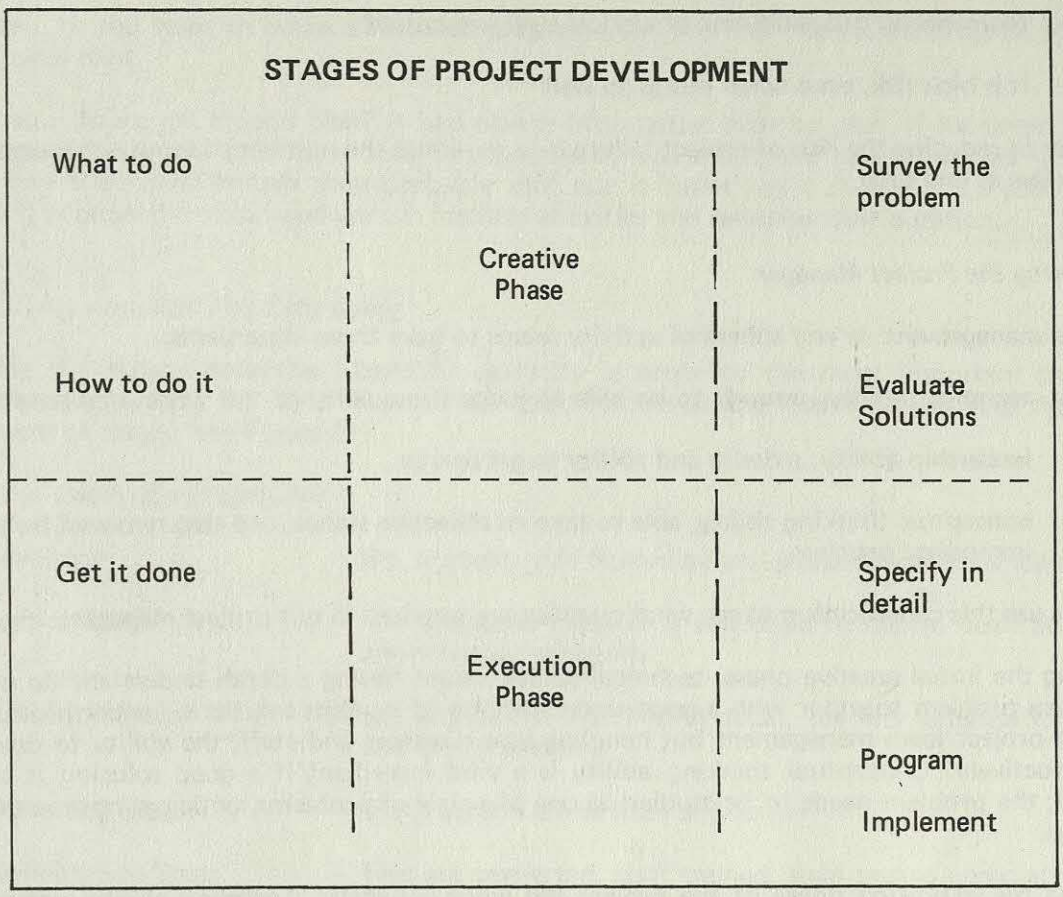


Figure 1

The execution phase, which follows, is all about getting the solution installed – detailed specifications, programming and implementation.

The creative start, surprisingly, may take almost as long (in elapsed time) as the execution, but it only involves one or two people (with rather special skills). The execution may involve a small army of people providing a very different management challenge.

There is a second way of looking at computer projects which is equally important. Is the system input driven or output driven?

An output-driven system (e.g. management reporting, stock control, sales statistics) is very much easier to manage:

- files already exist giving the data base
- there is little impact on the way people work
- it is low risk, even when things go wrong

Input-driven systems (e.g. order processing, payroll, foreign exchange dealing) are substantially more difficult:

- main files have to be set up and maintained
- transaction files have to be set up and maintained
- extensive control and error correction procedures are needed

- reorganising and retraining of clerical staff is required
- it is high risk, even when things go well

A way of reducing the risk of project failure is to minimise the number of input-driven systems going live at one time.

Choosing the Project Manager

'Good management' in any sphere of activity seems to have three ingredients:

- technical ability, enough to be able to judge the quality of the work of subordinates
- leadership ability, a desire and ability to get results
- conceptual thinking ability, able to take an objective stance, one step removed from the immediate problem.

Let us use this classification to see what qualities are required in our project manager.

During the initial creative phase, technical ability means having a depth understanding of the business problem together with a good understanding of systems solutions. Leadership ability is not project team management but handling user directors and staff; the ability to disagree constructively. Conceptual thinking ability is a vital ingredient if a good solution is to be found; the problem needs to be studied as one of a class of problems, anticipating growth and change.

During the execution phase of the project the emphasis shifts. Technical ability means not only a sound computer knowledge but also an awareness of people and systems, the human interface and what can be achieved with a computer system. Leadership ability means giving out the right size tasks and checking that they get done; a systematic recording of all agreements and meticulous attention to detail. Conceptual thinking ability is shown in the desire to look ahead, avoiding problems by forward planning.

It is not often that all these qualities are found in one man; but if we are aware that they are needed we can build a more balanced management team. A project manager whose strength is attention to detail might best be supported by an experienced supervising manager who is good at forward planning.

Of all the qualities, giving out the right size tasks and then checking that they get done is the one to prize most highly in a project manager.

The quality to avoid is lack of success. In this business people do not seem to learn by their mistakes (at least not in a global sense). The qualities that caused the problems on the first project will emerge again in the stress of the second – in spite of temporary reforms. So do not pander to your instinctive sense of fair play by giving him a second chance; give him a different job which he can do well and get another project manager. It is more realistic, and kinder.

B The Project Management Cycle

All managers want to feel that they are 'in control'. Being out of control means that you are rudderless – being driven by events instead of driving them.

But good control depends on a good initial plan. It is no good seeing your team members once a week and asking them how it is all going. When they reply 'Just fine', what have you really

learned? If you want to make a reasoned judgment you will want to review progress against an original plan.

Now how do we get a good plan? A bad plan is little better than no plan. If the target dates are unrealistic or the individual activities incorrectly identified, the team members will tell you they tried hard (which they probably did) but it 'never had a chance'. So a good plan is going to depend on how well we can identify activities and estimate their duration.

Identifying Activities and Estimating

Getting the project activities identified correctly is probably the most important part of project management. To make it easier we start by recognising that a project goes through a number of stages (see Figure 2).

The five stages of a project are:

Survey Stage	— the problem area is studied and possible solutions identified
Evaluation Stage	— the proposed solution is examined in depth, both business and technical solutions
Specification Stage	— the problem is specified in detail, the link between the user and the programmer
Programming Stage	— the programs are written and tested
Implementation Stage	— files are converted, staff trained, final testing completed and the new system introduced

The division into these stages is not arbitrary. It recognises the way any good solution will be developed for a computer project. It also allows that the best solution may not involve a computer — if that is the outcome of the Survey and Evaluation stages.

So we will tackle the project one stage at a time. Every stage has a visible end result which can and must be checked before the next stage is begun. These checkpoints force the user executive and the project manager to confirm the work which has been done before the next level of detail.

Authority to start a stage is provided by the Project Brief. It is prepared by the project manager in conjunction with his supervising manager and the user executive. Typically the user executive will require the approval of his board to incur the expenditure on the project stage. Similarly the Head of Management Services must approve the time and cost estimates.

The Brief sets out what we are trying to achieve, the time-scale, cost and lines of responsibility. This formal requirement to record what has been agreed (and to allow time for second thoughts) is a vital prerequisite to a successful study.

Each *Stage* is broken down into *Activities* which in turn are broken down into *Tasks.*

A *Task* is a specific job, with clear start and end points, performed by one person. The duration is four to eight days' work and it is the basic element for short term planning and control.

An *Activity* is a group of related tasks under the control of one person and ending in a *Milestone.* The duration is typically one to two elapsed months and it is the basic element for planning the stage.

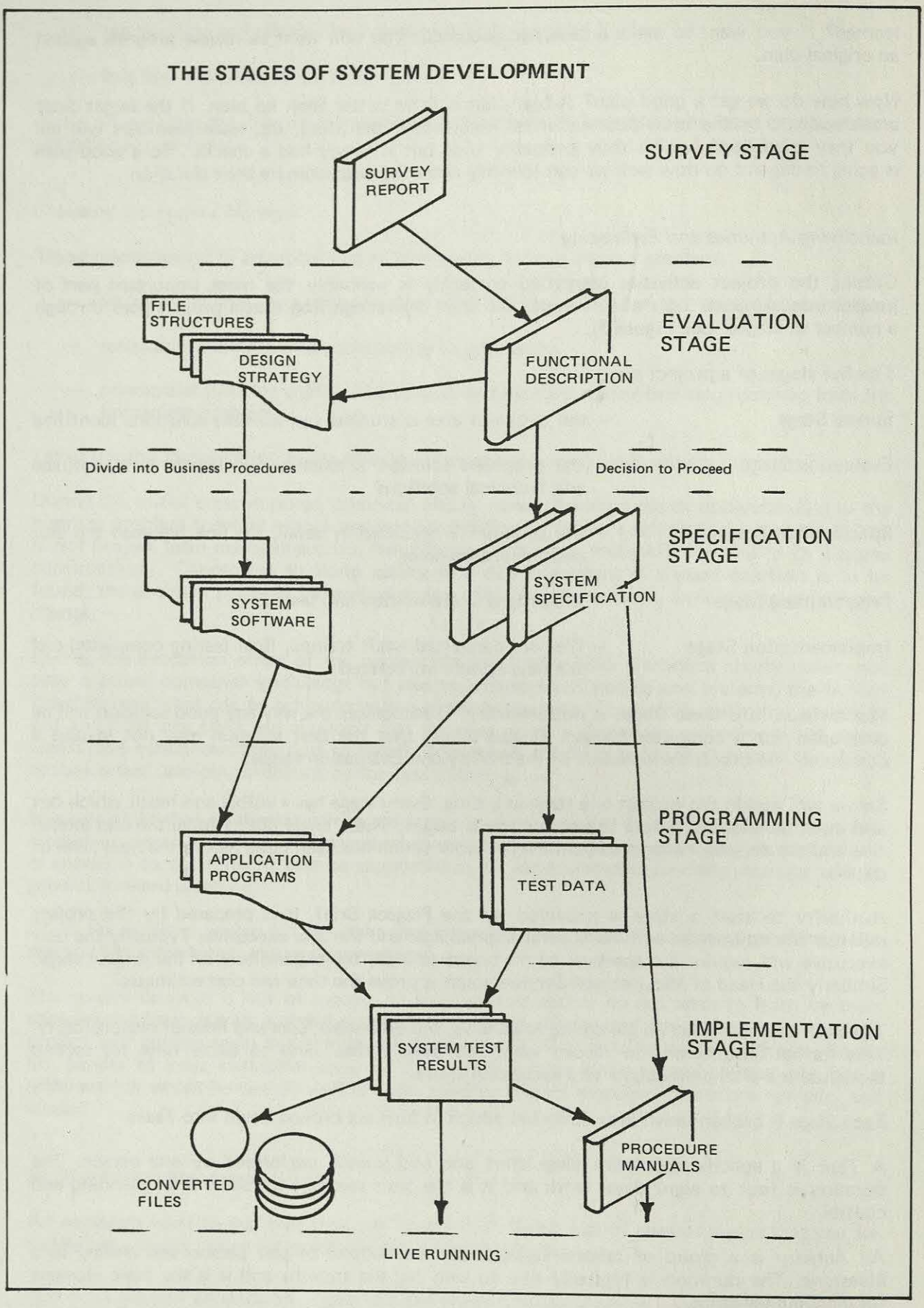

THE STAGES OF SYSTEM DEVELOPMENT
SURVEY STAGE
SURVEY REPORT
EVALUATION STAGE
FILE STRUCTURES
DESIGN STRATEGY
FUNCTIONAL DESCRIPTION
Divide into Business Procedures
Decision to Proceed
SPECIFICATION STAGE
SYSTEM SOFTWARE
SYSTEM SPECIFICATION
PROGRAMMING STAGE
APPLICATION PROGRAMS
TEST DATA
IMPLEMENTATION STAGE
SYSTEM TEST RESULTS
CONVERTED FILES
PROCEDURE MANUALS
LIVE RUNNING

Figure 2

PROJECT BRIEF

PROJECT NAME:

PROJECT PHASE:

PROJECT INITIATED BY:

PURPOSE OF PROJECT:

DURATION OF PROJECT:

TERMS OF REFERENCE:

PROJECT LEADER AND TEAM:

PROJECT SUPERVISOR:

USER EXECUTIVE:

BUDGET:

AUTHORISED: ..... DATE: .....

Figure 3

Identifying project *Milestones* is a technique for focusing attention on critical events during the progress of a project. Typically milestones occur at the end of an activity, but it may be convenient to have an additional one in a long activity. While it may be that occasional tasks run late, project milestones should never be missed (without early warning and a very good reason).

Ways of identifying activities will be covered as we examine each stage, but the following principles apply to all stages:

- Look for well balanced activities, about the same size. There is no point in having one activity of 200 man days if the others are about ten days each.
- Isolate innovation. Anything you have not done before is unlikely to be easier than you expect (you only know the good things and not the bad).
- Work to minimise the interaction between activities. Group the tasks to achieve independence and define unavoidable interface points as early as possible.

Once the activities have been identified there remains the task of accurate estimating.

Estimating has a reputation (undeserved) for being difficult. However, provided you are reasonably systematic it is not difficult to estimate the duration of an activity, the problem is to identify the right activities.

If you continue to find that estimating is a problem, a solution is to appoint an estimator whose job is to provide a second opinion to all project managers. The approach ensures that what information is available is assembled at one point, and the approach is visibly fair. Being

an estimator is not a full time job — it is a staff role which somebody adopts when the need is there.

Planning for Real

There is an old military maxim that 'no plan stands contact with the enemy'. One computer manager when first introduced to project planning prepared a very detailed plan for his project, six months ahead. It took him two weeks to do the work and by the end of the month the inevitable changes meant that the plan would have to be completely redrawn. He did not bother to replan, however, as it was clear to him that the exercise wasted two weeks without advancing the project much.

And yet we do need to plan ahead if we are to schedule the various resources and avoid wasted effort. Planning, then, is needed most where resources interact; and hopefully we have reduced the amount of planning required by organising our activities to minimise interaction between them.

What is needed is a two-level plan: an overall plan at Activity level which will be stable, in spite of short term variations, and a short term plan at the detailed task level.

The short term plan recognises that problems occur on a day-to-day basis which require re-scheduling and maybe rework. It should be only two to four weeks ahead and use tasks (four to eight days each) as the planning element.

Short term planning can be done with confidence, provided it is within the framework of the overall Activity plan. *The overall plan is designed to come true;* and that is what we mean by 'planning for real'.

If the plan is to come true, how long should you draw the lines on the bar chart?

Suppose, for example, an activity has been estimated at forty-five man days; will it take nine weeks for one man to complete? Surely it would be wise to allow longer. We do not know what will happen, but there may be holidays, sickness, training or similar unproductive time. Statistically you will find that staff work an average of four productive days per week, so eleven weeks to complete the job looks more realistic.

Further analysis of time spent suggests that we can only *guarantee* three and a half days per week on a given job. Emergency maintenance on another project, unforeseen activities on this project and similar unplanned but productive work can reduce the time to three and a half days per week — even when we aim to have the man full time on our project (as we must).

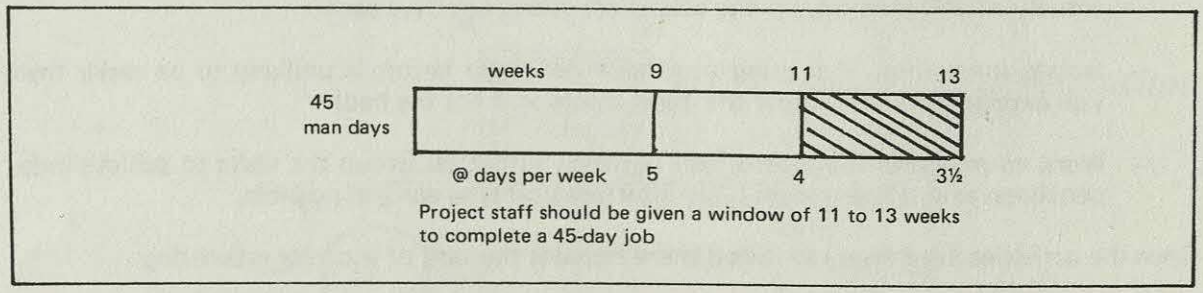


Figure 4

If the system is to work, the original estimate of forty-five days must be fair. One person is given responsibility for the activity and the way the plan has been drawn is explained to him. Eleven weeks is his personal target and if he achieves that result he will have done a first class job. Anywhere between eleven and thirteen weeks is a good job — but with all the contingencies built in there can be no excuse for exceeding thirteen weeks.

The approach is fair and easy to understand. Nobody wastes time developing alibis and excuses for failure because it is clear that there are likely to be none valid enough to be accepted. Claiming that the project was delayed by the users is no defence either – if users are given all the facts and allowed two to three weeks to make a decision, they are likely to come up with a good one on time. Give them the facts late or without warning and it should be no surprise if they dig their toes in.

The overall plan then is based on planning the activities on the basis of three and a half days' work accomplished per week. But there are two dangers to guard against. The first is to think that the built-in contingency allows for sloppy estimating; it does not. The contingency reflects practical experience of computer project pitfalls *when the estimating is pretty good.* So estimate all the activities for the stage thoroughly by breaking out the individual tasks and considering them carefully.

The second danger is to suppose that, because the overall plan is based on three and a half days' work accomplished per week, we can take it all fairly easily. On the contrary, to achieve an *overall* result of three and a half days a week, we need to get a day's work done every available working day. There are going to be any number of ways for the contingency to be eaten away as the project progresses.

Here are the main elements of the two-level plan summarised:

Activity Bar Chart	–	four to six months ahead
	–	two-month activities
	–	detailed estimating
	–	3½ days' work per week
	–	milestones identified
	–	two to three weeks for important decisions
Short Term Scheduling	–	two to four weeks ahead
	–	two-week tasks
	–	five days' work a week
	–	weekly review

Now let us look at ways of planning the individual activities and tasks. There are a number of planning tips to observe.

1. Aim for parallel, not serial activities
   Professor Brooks in his software essay 'The Mythical Man Month' argues that software projects always run out of time and that adding more resources makes them even later. If you organise your projects in a *serial* way you will find that he is all too correct.

   But you do not have to accept his hypothesis.

   Organising the project activities to achieve parallelism and independence guards against the problem. Adding one man to a one-man activity which is running behind can just about double the rate of progress on that activity.

2. Front End Load
   Critical path planning methods have one grave disadvantage. They tend to encourage activities to begin at their latest start date – and on a computer project that approach is

entirely wrong. We are not dealing with an exact science; there is almost always some degree of innovation in the project and it is consequently very difficult to know in practice where the critical path really lies. It is almost worse if we do get the plan right – for every job will have become critical at the end.

The right approach is to *bring every activity forward to its earliest possible start date.* In this way a buffer is created to absorb any shocks. The attitude of putting off problems to tomorrow is fatal. The big decisions are the ones to reflect on and to allow time before reaching a final conclusion; as the little problems arise they need to be explored and resolved with urgency. 'It'll be all right on the night' is not a very good theatrical maxim; it also leads to dreadful problems towards the end of a computer project.

3. Clear Responsibilities
The need for clear responsibilities has been implicit in many of the project management principles already mentioned. The idea that one person should be responsible for an Activity and that one person should carry out a Task recognises the importance of clear responsibilities.

4. When Things Go Wrong
When things go wrong it is time to take a long hard look at the project manager. Small things will always go wrong and the occasional program may need to be re-written but the good project manager takes it all in his stride.

Computer projects are inherently difficult and the manager who does not have problems is not just lucky. He has usually made his good luck by:

- bringing forward innovation
- giving his team clear jobs to do
- getting deeply involved in the critical parts of the project (e.g. system design, conversion strategy)
- not doing too much himself
- recording all agreements and following them up

Look at it this way. We want our plans to work, and if they do not it means that either we have a leader who is good at planning but bad at management, or we have one who is bad at planning and good at management, or maybe one who is bad at both. Whatever the reason, he looks to be the wrong man to lead a project.

When things go wrong the natural inclination is to change the plan and hope to do better. A more logical action would be to replace the project manager.

Positive Control

There seem to be three general approaches to project control:

- the subjective: 'How's it going?'
- the pseudo precise: 'I'm 90% through'
- the precise: 'Is the task completed?' Yes or No?

The first two methods are hopelessly vague. You will find that they obey the 90:90 rule, i.e. the first 90% of the project takes 90% of the time and the last 10% also takes 90% of the time.

The only realistic way to be in control is to see regular evidence of progress – evidence of jobs completed.

'Have you finished the survey?'

'Yes.'

'Can you give me the survey report?'

'Yes, here it is.'

'Do you have user agreement?'

'Yes, here is the user department acceptance.'

By organising the size of the tasks the project manager can limit his exposure and limit the amount he can get out of control before knowing about it. Very few tasks should be over two weeks long and, if they must, they should be carefully planned (e.g. a programme of visits by an analyst carrying out a survey).

1. Act on Deviations
   Before we go further in discussing methods of control we need to be clear on what we are going to do if things do go wrong. Suppose for example that you find that at the end of the first month of a six-month project you are three days behind schedule, what do you do?

   The easy and common reaction is to think that we have five months left to pick up the lost days. But in reality they will never be picked up – unless you act decisively. The project team will understand that losing three days per month is an acceptable slippage and your chance of hitting the target date has vanished.

   If you were thirty days behind after a month it would be more forgivable – it would probably mean you were solving a different problem. But three days behind means that the attitude is wrong. You will need to get the offender in over the next weekend or working late – whatever is needed to get the project back on course within a week.

   The team members now know that you regard hitting target dates as important and that you will take action when there is a slippage. That is not to say that we are trying to be unreasonable. We want to make a fair estimate for the job and to allow a very reasonable target date, and in return we expect a fair day's work.

2. Short Interval Scheduling
   The best way to exercise control of the day-to-day project running is to use the approach known as Short Interval Scheduling.

   The principles to apply are:

   – clear start to the task

   – clear end to the task

   – duration estimated before the start

   – expected results communicated

   – methods and standards laid down in advance

   – minimum of forms and controls

Two further points which apply to project management are:

- limit the size of the teams
- co-ordination and management are always necessary

Short interval scheduling on a computer project follows these general principles. Every week the project manager prepares a list of all the jobs to be done in the next fortnight. He issues the tasks one job at a time and must know what is happening on every day, and who is doing it. In this short term scheduling the project manager plans on five days' work per week (less known absences).

3. One Man – One Job
Rather than allowing team members time to finish a job, most project managers, anxious to avoid lost time, push the next task in too early.

It is common practice to allow programmers to 'study the next specification' while finishing off the program testing of the current job. Nothing could be more misguided. Programmers tend not to like testing much – it is difficult to admit that there are going to be any errors – and the more deeply they get involved in the new challenge, the more difficult it is to turn back to the old. So testing gets skimped and the problems show up later.

Look at it this way. If you have a good programmer he will not lose too much time waiting for test results. If you have a bad programmer you cannot afford to let him loose on a second job.

4. Short Term Estimating
Short term estimating is bound up with allocating jobs to individuals and controlling results. Up to now our estimates have been of a broadbrush nature and possibly made without knowing who is to be doing the individual jobs. Here are the main points to look for when making short term estimates:

- The estimates must be seen to be fair. Where possible publish the estimating basis.
- The time spent on the job will be heavily influenced by the quality of earlier work.
- The estimate must take account of the ability and experience of the man going to do the job.
- Time pressures should not permit any deviation from agreed quality standards.
- Some 'perfectionists' will work forever perfecting a piece of design.
- Estimating in areas of innovation is very difficult.

and perhaps most important:

- Do not change the job once it has been issued – give the man a fixed target rather than one which is constantly moving.

5. Keeping Control
The only effective way of keeping control is to document what everybody has agreed to do – then check up that it has been done.

Start by *issuing the work clearly.* If you are inexperienced, use a document along the lines of that illustrated below.

Make a note of all the work issued and the scheduled finish dates. Go through the list every

ACTIVITY SLIP

Name:

Date:

Activity Description:

Starting point (documents etc attached):

Expected result:

Likely problem areas:

Standards to be used:

Estimated duration and completion date:

Actual duration and completion date:

Remarks:

Figure 5

week with the team members individually. If you go through what they have agreed point by point you will find that they get into the habit of getting the jobs done that you want done.

Insist on Meeting Notes which are issued shortly after any meeting. Record all agreements (if there have been any misunderstandings they will emerge quickly) and note who is responsible for action. Never have two sets of initials in the action column – joint responsibility just does not work.

Serially number all paperwork in the top righthand corner. It is easier to refer to the document you mean and easy for the user executive, supervising manager and project manager to check that they have received all the facts.

Keep a project workbook and note all ideas in it. Most problems will have been anticipated earlier, but may be forgotten. So get in the habit of writing ideas down before they are lost – then systematically clear the items as each stage of the project progresses.

6. Senior Management Control
If the project manager concentrates on getting all the *tasks* done on time and on budget, then his *activities* will come out on time and on budget.

Senior management should leave task progression to the project manager. They should concentrate on activities. If every *activity* is completed on time and on budget then the *project stage* and indeed the whole project will be on time and on budget.

So top management control is down to two simple questions:

- For every activity completed in the month, was it completed on time and within budget?
- For every activity started, do we still think it will be completed on time and on budget?

When reduced to these basic questions, we can see that it is fairly easy to know what is going on.

C Project Documentation and Reporting

The documents to control project progress need to be a balance between a desire for control and a desire to keep forms to a minimum.

In practice we have found the following forms to be the working minimum:

1. *Project Assignment Brief* (one for each stage of the project), used to initiate a project stage and containing objectives, timescale and expected resources.
2. *Activity Bar Chart* (one for each stage of the project), used to identify the main activities, to plan completion dates and to allocate responsibilities.
3. *Activity Time Estimate* (one for every activity of the stage), used to identify *tasks*, to estimate duration and to record progress.
4. *Activity Progress Record* (one for every three activities of the stage), used to record weekly progress and to forecast completion date and man days against budget.
5. *Project Progress Report* (one for every month for every project), used to summarise progress, to warn of problem areas and to note all major agreements and commitments.

III. MANAGING THE WORK OF A PROJECT

A The Survey

The Survey Stage is the formal start of a project. It may have been preceded by a preliminary survey which has identified the area under review and set terms of reference.

The Survey Stage is concerned primarily with WHAT is the problem to be solved. Rarely is it possible, however, to separate the problem to be solved entirely from possible solutions. A batch computer solution will tend to solve a different problem from an on-line computer solution or a no-computer solution. So the Survey Stage also includes a preliminary evaluation of alternative solutions.

It may be that more detailed analysis, during the Evaluation Stage which follows, will reveal a problem in the chosen solution. We will just have to accept that occasionally we must return to look at other solutions again; it will not happen often and is better than wasting too much time evaluating all solutions in depth.

The Survey Stage then is an analysis of what needs to be done with an evaluation of the main alternative solutions. The work of the analyst falls into six phases:

Fact finding:

1. Orientation

2. Fact gathering

Developing solutions:

3. Understanding the business need

4. Verifying the findings

5. Developing solutions

Presenting the findings:

6. Writing the Survey Report .

It goes without saying that the Survey Stage is the most important part of the project. No amount of excellent programming will make up for getting the problem wrong in the first place.

The Survey then starts with fact finding.

There is a view that fact finding interviews are difficult – and they are, if the right sort of preparation is not carried out. If, however, you know what you are trying to find out and you also know the responsibilities of the person being interviewed, every analyst has a good chance of success.

What is needed is a systematic approach.

The approach to be described will work for a survey at more than one level. In the first phase, interviews start at the top of the organisational unit (for orientation), and work their way to the bottom. In the second phase interviews start at the bottom (for verification and consolidation) and work their way back to the top.

It is no good going into detail before seeing the context of the problem; similarly it is no good discussing new ideas with more senior managers without hard facts to support the argument. Therein lies the rub. *The one thing likely to emerge from the fact finding interviews is opinion — not fact.* The top-down then bottom-up interviewing style is designed to avoid the trap, by delaying discussion on change until the real facts can be established.

Phase 1 — Orientation

Starting at the top of the organisational unit under study, the aim of the analyst is to understand the main business components and their relationship.

Start by getting the person being interviewed to go through his organisation chart and explain the physical layout of the staff and equipment under his control. From this information prepare a physical layout chart. Make the level of detail appropriate to the level of responsibility of your informant. At section leader level record where individuals sit and the equipment they use. At divisional level show all the sections with their staff numbers and the major units of equipment — and so on. A physical layout chart reveals much more than an official organisation chart alone — but add any reporting relationships which are not explicit.

Having understood the makeup of the organisational unit, the next step is to find out what they all do. For every sub-unit make out an *Activity List* covering:

- functions performed
- approximate proportion of time on each
- functions which are difficult or are problem areas.

Use this list to home in on key parts of the work to be studied. Work which is not time consuming, difficult or a problem area is unlikely to affect the outcome of the study; concentrate then on the remainder.

The third part of the interview is used to get an understanding of the flow of paperwork between the sub-units, and the files which are maintained. Use a simple *Flow-diagram* to record this data flow (see Figure 7).

The end result of the interview will be three documents:

- physical layout chart
- activity list
- flow-diagram.

If you start off with the aim of getting these documents completed your interview will have direction and purpose. Any points which are missing can be left with the interviewee who can pass them on to you later.

Now for the danger areas. Note down any opinions expressed, you will need to check them later on, but refrain from passing any judgment until more facts have been observed. Be particularly careful about opinions expressed about another area — the opinion can only be part of the story.

Try, also, to limit the questions to the work of the interviewee and his immediate subordinates. Questions at too detailed a level will get answers about what he thinks ought to happen — and at this stage we are more interested in what does happen.

ACTIVITY LIST — Order Clerks

INTERVIEW WITH: H Adams, Sales Office Manager 11.1.78

FUNCTIONS	% TIME	DIFFICULT/PROBLEM AREAS
– Maintain Customer History and do Credit Check	15%	
– Batch orders for typing and check typing of desp. set	7%	
– File rep. copy and file office copy	7%	
– Marry up despatches with order copy	10%	

Figure 6

The final danger is that not everything you hear is completely correct. Make a habit of verifying what you are told at the level below and reconciling any differences.

Phase 2 – Fact Gathering

The orientation interviews continue (concentrating on the high volume, difficult and problem areas) until you reach the bottom level in that part of the organisation. It is at the bottom level that the work gets done and where true facts can be obtained.

True facts, however, are not found by interviewing. They are found by counting pieces of paper, counting entries and checking serial numbers. To find out how many invoices there are in a month, for example, check the invoice numbers on the first working day of all the last six months. To find out the average number of lines per invoice take a random sample of, say, fifty invoices and count the lines on every one. Where possible check statistics in two ways and explore apparent discrepancies until the cause is fully explained.

Here then are the things to do:

- Start by understanding the purpose of all the accounting entries generated in the section. Work from the back to the sharp end of the business (the customer contact).
- Collect specimens of all key documents (the ones on your flow diagram). Make sure their names are correct on the flow diagram.
- Identify controls on accuracy and quality – what errors and queries are there?
- Establish volumes (high, low, average and peak).

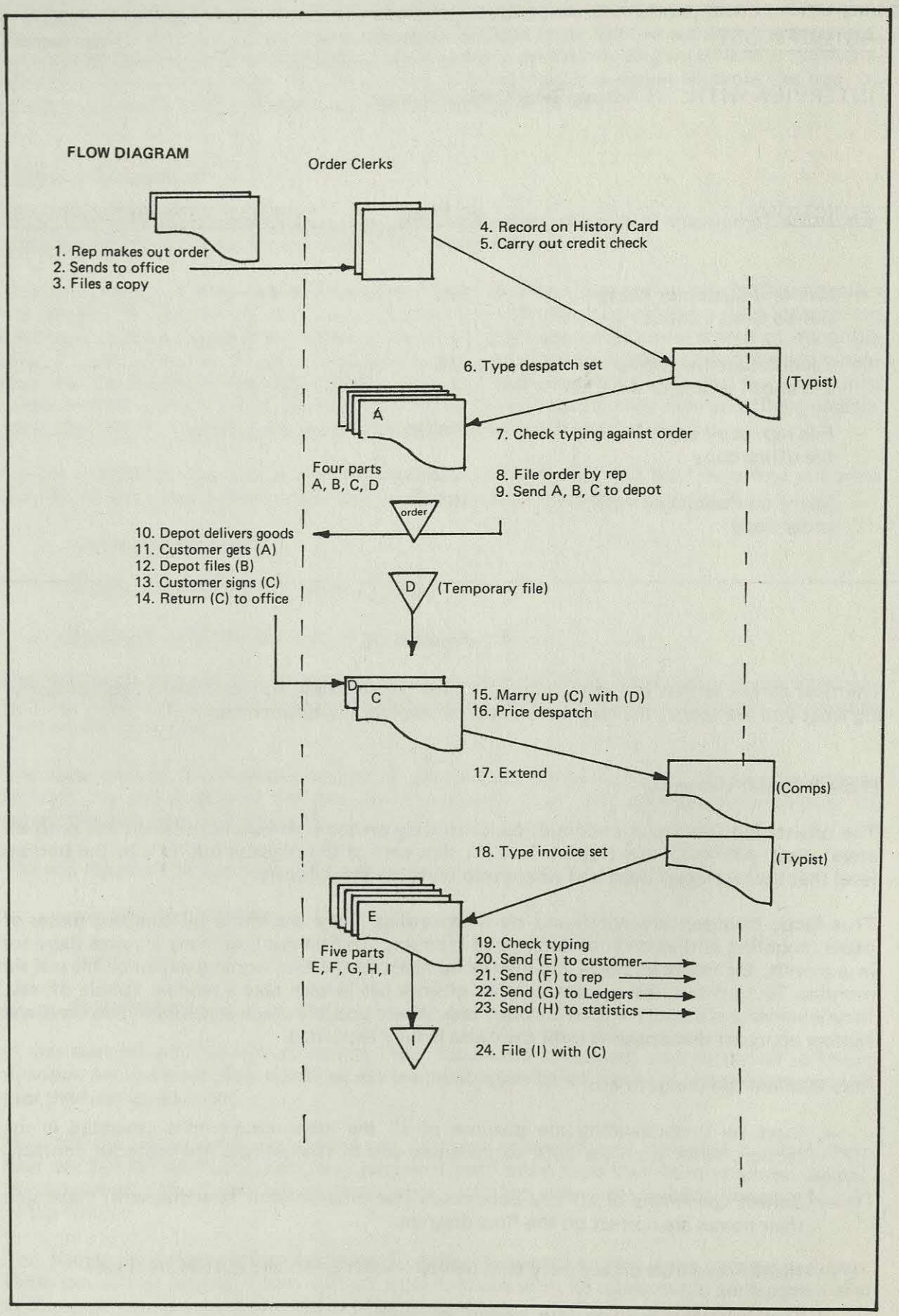

FLOW DIAGRAM
Order Clerks
1. Rep makes out order
2. Sends to office
3. Files a copy
4. Record on History Card
5. Carry out credit check
6. Type despatch set
(Typist)
A
7. Check typing against order
Four parts
A, B, C, D
8. File order by rep
9. Send A, B, C to depot
order
10. Depot delivers goods
11. Customer gets (A)
12. Depot files (B)
13. Customer signs (C)
14. Return (C) to office
D
(Temporary file)
D
C
15. Marry up (C) with (D)
16. Price despatch
17. Extend
(Comps)
18. Type invoice set
(Typist)
E
Five parts
E, F, G, H, I
19. Check typing
20. Send (E) to customer
21. Send (F) to rep
22. Send (G) to Ledgers
23. Send (H) to statistics
I
24. File (I) with (C)

Figure 7

When the facts have been established they should be agreed with the section leader. Prepare a meeting note which covers the facts which have been agreed and send a copy to the section leader as a formal record. A great deal is going to be decided on the basis of these facts and it is important that they are not in dispute.

During the fact finding you will probably form a view of the work output of the section (based on flow of work, timing, peaking, overtime, utilisation of staff across sections, quality of supervision). Most sections have room for a 25% improvement (by training the first level supervision, improving the flow of work and merging small sections), but precise work measurement is not likely to be worth while if the existing systems are to be changed substantially.

Phase 3 — Understanding the Business Need

The orientation and fact finding stages have helped the analyst to home in on the main issues — but true understanding requires a detached sifting of the facts. The problem is to separate true business needs from the methods which have grown up to satisfy them.

The starting point is the flow-diagram which describes the system under review. It may be necessary to redraw it, as the flow will cross departments, to keep it clear.

From this flowchart examine every operation critically:

- clearly state the business *Purpose* behind the operation; avoid suggesting *Method*
- if one operation achieves two Purposes, list them both separately
- an operation performed to enable some subsequent step to be taken is ignored; it is Method
- ignore such operations as typing, coding or transmitting information; they are also Method
- checking documents which enter the system from outside has a valid Purpose, but checking an operation within a system is again ignored, as being Method
- documents filed for reference have a Purpose, but intermediate files on a computer system are usually Method.

As this systematic examination continues it will be found that the list requires careful re-checking and usually revision. Understanding is being developed.

Make a point of verifying the true facts — not what people think. A daily list of balances may also be used to look up customer codes; a printout for answering queries may go to the sales office while the queries actually come to the depot.

The Purposes which are eliminated are the ones likely to provoke most reaction — they are also the ones likely to yield substantial benefits. Check them thoroughly and be sure of your ground.

Phase 4 — Verification

The fact finding is almost complete and the analyst has developed a good understanding for the needs of the business system. It is now time to verify the findings and to encourage ideas for improvement.

The next step is to *challenge the real need for every purpose* on the list.

Eliminate any unnecessary Purposes and star any in doubt – they will probably be necessary in the final system but should not determine its design. Figure 8 shows the way the key Purposes can be analysed and reduced to essentials.

IDENTIFYING KEY PURPOSES

Operation	Purpose?	Comments
1. Establish requirements	yes	
2. Send to office		Method
3. Customer history for Rep	yes	Rep needs for next call
4. Customer history in office	yes*	Referred to sometimes
5. Credit Control	yes*	No problem at present
6. Type despatch set		Method
7. Check despatch set		Method
8. Rep history in office		Never used in practice
9. Send despatch set to depot		Method
10. Deliver goods	yes	

Figure 8

Start at the lowest level of supervision and work back up the organisation. The pattern of every meeting is the same:

- Show first that you understand the existing system (using the physical layout chart, activity list and flow diagram), that you know the volumes precisely and recognise the problem areas. During this explanation you will be gaining the manager's confidence that your fact finding is thorough and that your conclusions will be based on sound arguments.

- Now verify your business understanding by going through the Purposes you have identified and by discussing in depth those you have eliminated (either because of Method or because they are not necessary).

- Try to establish where he sees problem areas and record them against the appropriate Purpose. Make a note of his ideas for change – but also look for the reasons the ideas have not already been implemented (unworkable, suit him but not others, genuine inertia).

- Pay particular attention to gaps in management control – control of business effectiveness and profitability, control of performance. Add any new Purposes identified in this way.

- Prepare a brief meeting note covering the points he agreed and disagreed (not including your explanation of the existing system). Let him have the meeting note and an opportunity for reflection. You may need evidence of the points of agreement later. Pay particular attention to the points of disagreement; you may well be right but go over the arguments carefully, then be prepared to stick to your guns.

Phase 5 — Developing Solutions

Before developing any solutions, carry out a preliminary analysis:

- arrange the Purposes in a logical *time sequence.* Purposes which can be achieved at the same point in time can be arranged side by side.

- decide *where* it would be most logical to carry out each Purpose; *who* should do it? Can document movement be reduced or eliminated?

- what *information* is required to achieve the Purpose?

- *group together Purposes* which could logically be performed by the same person. Eliminate Purposes which can be achieved by reasonable amendments of some other Purpose.

Now it is time to consider solutions. The more experienced the analyst, the better the solutions he will consider and at this stage the more ideas the better.

Sketch out every solution (there must be more than one) in as much detail as possible, then write a one-page summary of each. Here is a list of points to check when evaluating the alternatives:

1. How well does it satisfy the business need? Are all the Purposes covered? Are the problem areas in the existing system truly removed (or have we just improved the areas which are working well already)? Is it flexible enough to cope with growth or change?

2. How well will it work in practice? Will it be reliable; how will it cope with hardware failure? Is it convenient to use? Is all computer input made through VDUs or transferred from another system (batch input adds substantially to hidden costs)? Does it have the right degree of security and privacy? How will the hardware and software be maintained?

3. What is the development risk? Does it involve unknown hardware and software? How complex is the solution (Data Base Management Systems, Networks and Real Time all add substantially to complexity and risk)? Will there be a long development timescale?

4. What is the impact on other systems? How will they interface and evolve? Is this one of a general class of problem solutions?

5. What will it all cost? Calculate the cost of equipment (depreciate capital purchases over five years; include installation costs and annual maintenance). Calculate the cost of development (depreciate over four years but make a good allowance for software maintenance and include contingency for risk areas). Calculate the annual running cost (consumables and perhaps accommodation, but not staffing).

6. What will be the relative savings? Cost staff differences (including overtime). Are there savings in equipment or time which can be quantified?

The evaluation process will help to crystallise the alternatives. Solutions which do not satisfy the business need reliably must be weeded out (however attractive they appear for other reasons). Evaluation difficulties arise because we are rarely comparing like with like. A computer package, for example, may be cheap but only solve part of the problem; a good long term solution may prove expensive in the short term — or depend on a degree of business growth that cannot be guaranteed.

It is not necessarily the job of the analyst to make the final decision. His job is to analyse the problem and present two or three viable solutions as fairly as he can. The final decision must be made by the business user executive once he has the relevant facts marshalled.

The role of the analyst is to develop the best possible system to support the business. Sometimes the analyst sees ways of improving that business — and here he is on dangerous ground. Any such ideas should be fed back to the supervising manager who will decide how to feed the ideas to the user executive.

CHECKLIST OF EVALUATION FACTORS

	Alternative Solution A	Alternative B
1. Meeting Business Needs – flexibility for growth and change		
2. Ease of Use – reliability – convenience – security – hardware maintenance – software maintenance		
3. Development Risk – degree of innovation – complexity – timescale		
4. System Interfaces – impact on other systems		
5. Costs – equipment – development – hardware maintenance – software maintenance – running costs		
6. Savings – equipment – staff		

Figure 9

Phase 6 — Writing the Survey Report

The Survey Report is the written evidence of the analyst's work. It is important that it is well written and presents the case for change clearly and objectively. The contents list should include typically:

1. INTRODUCTION
   - background to the report, the project team
   - (reference to) the terms of reference
   - a sentence commencing 'The purpose of this report is . . .'
2. SUMMARY OF RECOMMENDATIONS/CONCLUSIONS
   - only include this section in long reports

- make sure it is a balanced and fair summary

3. CURRENT SITUATION

   - show a grasp of essential features
   - record volumes
   - identify problem areas
   - do not, in general, include detailed fact finding charts

4. REQUIREMENTS OF NEW/IMPROVED SYSTEM

   - compile this section from agreed Purposes
   - distinguish essential from desirable

5. ALTERNATIVE SOLUTIONS CONSIDERED

   - summarise alternatives
   - rule out obvious non-starters, end up with two or three

6. EVALUATION OF ALTERNATIVES

   - tabulate alternatives against requirements
   - use the headings outlined earlier, i.e. satisfying the business need, practicability, development risk, impact on other systems, costs and savings — with sub-headings as appropriate to the problem

7. RECOMMENDED APPROACH

   - highlight the main differences
   - make a recommendation if you are sure

8. CONCLUSION

   - suggest what needs to be done next
   - request agreement to proceed by a date (at least two weeks ahead)

9. APPENDICES

   - terms of reference
   - relevant working papers (including cost calculations)

This report structure takes the reader through the argument, building up the picture as he goes. Check particularly that ideas do not emerge for the first time in the later sections. Consider also removing parts which are not followed through to the appendices. Resist publishing working papers (keep them as backup) and resist pushing technical detail (write the ideas down to be considered further in the Evaluation Stage).

Project Management Considerations

The Survey Stage needs to be carried out by a very experienced analyst. It will probably be done by the project manager himself – if not he will be deeply involved in its development. Either way there are very few people involved but their time needs to be carefully planned.

The six phases of the survey provide the planning base. As a guide the time required is about a fortnight for every ten staff (an average of one day per person) plus 20% management and supervision, 15% contingency. It is tempting to reduce the time allowed and the amount of detailed investigation – either because of assumed familiarity or time pressure. The project supervisor needs to be on sure ground before he cuts out any of the steps.

Part of the planning is concerned with getting the project off to the right start – in particular that the staff concerned are fully briefed on the nature and extent of the study. Here is a list of the points to be agreed in advance:

1. Advice to all staff who are likely to fall within the scope of the study:
   - of the presence of analysts within the departments
   - of the nature and extent of the study
2. Organisation chart – to assist in identifying functional responsibilities and relationships
3. Positive arrangements for access to senior managers in the departments
4. Availability on loan of sample 'live' documents necessary to support fact finding
5. Use of desk and office services in the area being studied for the duration of the study.

B The Evaluation Stage

The Survey is concerned with identifying the problem to be solved together with an examination of the main alternatives. Once a decision in principle has been taken on the way ahead the project moves into the detailed Evaluation Stage.

The purpose of the Evaluation Stage is to prepare a precise business specification of the problem (the Functional Description) and to develop the technical solution (the Design Strategy). In both cases we need to go into sufficient detail to be confident that the solution will be successful. It follows that in areas of innovation we may need to carry out a great deal of research before the project can move ahead to the Specification Stage.

It is during the Evaluation Stage that the problem area is divided into projects and *procedures.* We saw in earlier chapters the importance of correctly breaking the project into manageable activities. It is now that the process begins.

In an Order Processing/Stock Control application, the computer systems might be introduced in a series of projects:

Sales Ledger (Accounts Receivable)

Stock Control

Order Processing/Invoicing

Payroll

Purchase Ledger

Nominal Ledger and Reporting.

Each of these projects is made up of a number of *procedures*. For example, the Sales Ledger project would be divided into:

Customer File Enquiries and Maintenance

Entering Invoices and Credit Notes

Allocating Cash and Discounts

Sales Ledger Update and File Purge

Statement Printing and Aged Debtors' Report

File Conversion to new system.

It may take some time to get the procedure breakdown right. As a general rule, jobs carried out by different sections or in different time cycles should be separated.

The procedure which is underestimated tends to be File Maintenance. The procedure which is overlooked tends to be File Conversion. Not all the procedures are computer procedures. Some may be entirely manual, others will have a high manual content.

1. Functional Description

Figure 10 gives an overview of the contents of the Functional Description.

The Functional Description represents a complete statement of the business problem: it is the basis on which technical solutions will be developed so it is worth double checking that everything has been included.

Preparing an Invitation to Tender

It can also be used as the basis for a hardware proposal from a computer supplier. The hardware salesmen will be emphasising the good features of their equipment and possibly forcing your problem to fit their solution.

A clear statement of the business problem in advance, as in the Functional Description, helps the buyer to separate essential features from those which are either desirable or merely cosmetic.

When sending out an invitation to tender the Functional Description should be accompanied by a letter which covers the following points:

- Background information on XYZ Company
- Desire to install computer systems in areas P, Q, R
- Current situation, reference to Functional Descriptions of desired systems
- Information required from manufacturers:

  (a) Specification of equipment proposed and delivery

  (b) Full statement of costs (rental, purchase, extended use, trade-in values, consumables)

FUNCTIONAL DESCRIPTION

The Functional Description is one of the most important documents prepared during the development of a project. Its purpose is to present the design of the system together with costs, benefits and implementation plans, such that the user may understand, evaluate and formally agree the proposed system.

The contents list should be taken from:

1. OVERALL SUMMARY OF THE SYSTEM
   (intended to give any reader of the Functional Description who is not already familiar with the area, a general understanding of the system before encountering the detail — start by stating simply what the system does)

2. NARRATIVE DESCRIPTION OF EACH MAJOR ASPECT OF THE SYSTEM
   (procedure by procedure is preferable to input/processing/output; include interfaces with other systems)

3. OUTPUTS FROM THE SYSTEM
   (specimen of all printed outputs with details of frequency, principal uses, sequence, volumes, copies and distribution)

4. INPUTS TO THE SYSTEM
   (tabulation of all inputs and keyboard I/0 giving description, data content, layout, cut-off timings, volumes, source)

5. TABULATION OF MAIN FILES
   (description, data content, field sizes, sequence, origin of updating data, responsibility for maintenance, security and backup, volumes)

6. TABULATION OF MAJOR CODES USED
   (description, structure, source, maintenance)

7. CONTROL PRINCIPLES
   (controls, data validation, reconciliation, re-submission of errors)

8. VOLUMES, FILE SIZES AND RUN TIMINGS
   (current and future volumes)

9. DEPARTMENTAL RESPONSIBILITIES
   (control of input, file maintenance, output distribution, equipment)

10. FUTURE DEVELOPMENT OF THE SYSTEM
    (show how the system will cope with growth and change; state all known limitations of the system)

11. IMPLEMENTATION PLAN
    (implementation bar chart; budgets for remaining stages; user involvement; conversion strategy)

12. CONCLUSION
    (calling for agreement to proceed by a date at least 2-3 weeks away)

Figure 10

(c) Physical planning requirements

(d) Description of operating systems and software including numbers of existing users

(e) Details of how the equipment will process each application (with full timings). Where can a similar application be seen on this equipment?

(f) Pre-installation support (systems, programming, site inspection, machine time)

(g) Maintenance cover and standby arrangements proposed

(h) Special requirements (benchmark test, program conversion, file conversion).

By asking manufacturers to supply this information in sequence by a deadline it is easier to evaluate the proposals and to avoid supplementary questions. In any case, the number of competing suppliers should be limited to three or exceptionally four — otherwise the evaluation process becomes an end in itself.

Project Management Considerations

The structure of the Functional Description provides a framework for carrying out the evaluation. Start by breaking the project into procedures and drafting Section 2 (Narrative Description of Procedures). The section will probably have to be re-written later, but it provides the analyst with a starting point.

Now take each procedure in turn and draft the outputs (Section 3), inputs (Section 4) and file contents (Section 5) while working closely with affected user departments. It is then time to start looking at the design strategy. When the analyst has obtained user agreements which have been developed it is time to revise the sections already written and to complete the remaining sections.

The Functional Description is then subject to a formal Business Review before it is finalised.

A fair proportion of the Functional Description will be straightforward (an Invoicing system has to produce Invoices) and the design effort is a matter of preparing sensible output and input documents.

Management control reports, however, are likely to be much more demanding in an analytical sense. A real understanding of the way the organisation works is required before the analyst can arrive at the right level of reporting flexibility.

For preparing the FD, allow one week per average 'procedure' plus a week. Allow additional time for procedures which are particularly complex. Allow familiarisation time also if the analyst doing the FD was not the one who did the survey

Allow at least three weeks for user agreement — longer if it is a complex project — and use the time to carry out the Business Review.

2. Systems Design Strategy

The Functional Description is important because it states the business problem to be solved — and it provides an opportunity to verify that the right problem is being solved. The Design Strategy is equally important. Brilliant programming cannot make up for bad design but a good design can weather almost any number of problems of a minor nature (a bad program can just be re-written).

What then are the characteristics of good design and what are the guidelines to be used?

Good design can be recognised as being:

- *partitioned to reflect the business problem*. The closer the solution reflects the structure of the business problem, the better it is likely to cope with business change. In practice this means designing the system to reflect the business procedures and not striving to write generalised programs across procedures. In this way the solution will be flexible for growth and change.

- *efficient in its use of resources.* The resources involved are not restricted to the computer. Efficient use of business users is perhaps most important, sometimes efficient use of analysts and programmers (in getting a job working quickly) is more important than saving time on the computer.

- *secure, accurate and reliable.* Problem partitioning tends to improve security by isolating the problem areas.

- *simple to understand.* Complex solutions tend to be difficult to modify when there is business change. The simpler the solution the easier it is to comprehend and the more likely that problem areas are anticipated.

When beginning to design the technical solution it is helpful to examine separately:

- input streams

- mainfile updating

- output procedures and reporting

The approach is illustrated in Figure 11. The input streams cover static data for file maintenance, transactions for file updating and entries generated from within the system. All three types of input should be subject to controls with audit trails.

In principle, only one program should update each of the main files — which are then frozen for the remainder of the processing cycle. File updating is the most critical job in terms of accounting accuracy and control — and having only one place to look when things go wrong simplifies error detection and recovery.

The output procedures may read from any of the main files but may write to none directly. Generated Entries (which modify the main files) must be posted via the input stream, subject to all the input controls.

By looking at the design of a system in this way (input, updating, reporting) there are a number of benefits. First, we have got a way of approaching the design problem and can develop rules which suit the different needs of the three phases.

Second, we can achieve parallel project development. Once the files have been designed and frozen, different teams can work on different parts of the problem with confidence. The file contents and design are seen to be at the heart of the strategy.

Third, and perhaps most important, the strategy satisfies the principles of good design outlined earlier — simplicity, security, flexibility for growth and business partitioning — without sacrificing any efficiency.

It is not suggested that Systems Design is easy nor that it can be done without considerable thought. But the great majority of business applications fall into relatively few patterns which yield to general analysis. More time thus remains for designing for exceptional situations without developing every solution from first principles.

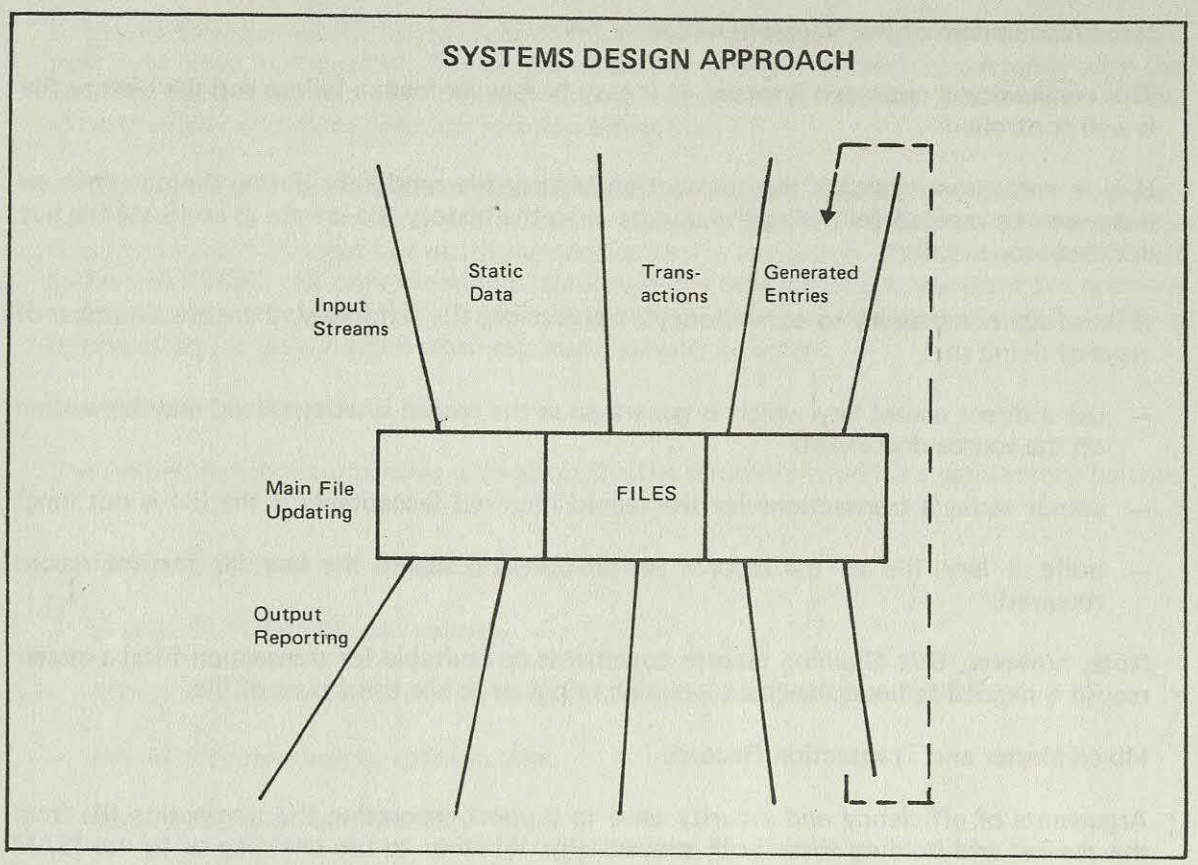


Figure 11

Selecting the File Structure

Most files fall logically into one of four classes:

- Main Files (such as Customer File, Product File) which are characterised by a relatively low volume of insertions and deletions.
- Transaction Files (such as Sales Ledger Transactions, intermediate transaction files) which are characterised by a high volume of insertions.
- Mixed Main File and Transactions (such as Customer File with Sales Ledger transactions).
- Files which need to be accessed by more than one key (including Data Bases).

These four classes of file require somewhat different treatment.

Main Files

Main Files need to be accessed both sequentially (e.g. end-of-month sales ledger, aged debtors' report) and randomly (e.g. names and addresses for order processing). The most common organisation therefore is Index Sequential (or ISAM for Index Sequential Access Method).

Transaction Files

The most common way of holding transaction files is sequentially. Records can be written sequentially as they occur and if necessary the file can be sorted prior to a conventional

father/son update of the transaction history file.

This conventional approach is secure, it is easy to recover from a failure and the history file is well controlled.

If it is necessary to access the transaction history file randomly during the day then an index can be created following the update – so the history file is held as an ISAM file but updated sequentially.

If it is further necessary to access today's transactions file randomly, there are a number of ways of doing so:

- use a direct access key which is generated as the record is written (and may be written on the source document)
- search today's transactions for the record required (acceptable if the file is not long)
- build a key file as the records are written and search the key file for the record required.

Note, however, that chaining records together is not suitable for transaction files; a master record is needed to head the chain – which brings us to the third class of file.

Mixed Master and Transaction Records

Arguments of efficiency and security tend to support separating the transaction file from the master and holding them both sequentially. Whether to use chaining or to use ISAM for both files is a matter for the systems designer with a given problem to solve. In general the chaining solution is likely to require more complex software and is less secure. With either solution the master file should contain a total of the individual transaction records for that master – to aid control and error recovery.

Multi-Key Files

Sometimes it is necessary to access records in two sequences or using two different keys. Examples of such a need would be:

- accessing vehicle records by registration number or chassis number
- accessing outstanding orders for a given customer or a given product
- accessing outstanding foreign exchange deals for a given customer or for a given currency on a particular day.

For on-line systems the problem is compounded by the need to respond to enquiries in either sequence in an acceptably short time (a few seconds typically). There are a number of different ways these problems might be tackled.

The simplest way would be to hold the records in as many sequences as is necessary. This is the approach usually adopted in a batch environment.

A second solution would be to hold the records in one sequence and to maintain an index in a second – known as an inverted file. Vehicle records, for example, might be held by registration number.

A third solution would be to hold the records in one sequence (the most used) and to chain them in another. For example, Outstanding Order records can be held in product sequence and chained to the customer master file.

A fourth solution would be to hold the records in the main sequence and to maintain summary totals in the other. The orders outstanding might be held by customer with the quantity outstanding for each product held as a total on the product master file – likely to be a simpler and more practical solution altogether.

A fifth solution is to use a Data Base Management System (DBMS). A DBMS is a generalised approach to file handling and is usually based on chained records or an inverted file. It is attractive in concept but usually unsatisfactory in execution. The reason is not difficult to find. A DBMS will only work satisfactorily if the data structures represent the problem accurately and efficiently. By the time the problem has been analysed enough to achieve its objective, the way is often clear to a much simpler solution.

Summary of File Structures

The system designer's objective is to select the file structure which is a satisfactory balance of:

– relevance of structure to the business problem

– simplicity, reliability and security

– efficient use of available resources (during enquiry, updating or reporting)

– satisfactory response or running time

– ease of operation and recovery

The Index Sequential Access Method is the basic structure on which most designs are based. The major design problem is the way to handle transaction files – partly because inserting new records in a file can quicky reduce efficiency to an unacceptable level; partly because the data records need to be accessed efficiently from more than one direction.

Guidelines for Designing Batch Input

We come now to the guidelines for designing the input, updating and reporting phases of the system.

Figure 12 shows diagrammatically a comprehensive approach to the design of a batch input suite of programs. Although most input systems should be on-line, there remain a substantial number of occasions when batch input is necessary. The design reflects many years' experience of what is needed in a secure and reliable input system and any designer should think carefully before taking short cuts.

Here are the main features of this design with their reasons.

There should be a single entry point for all transaction types, including static data, to allow simple and reliable operation. All records should be in card image format (80 characters) and all types of input media allowed (card, paper tape, magnetic tape, transmission).

Transaction vetting should be split from static data vetting. The static vet program is usually very large and has relatively few transactions – which also explains why the transaction vet comes first.

Input should be batched with provision for multiple input batches from multiple input centres in multiple shots. A simple one-user system can quickly get a second user (the auditors perhaps).

New batches are written sequentially on the transaction disc. Generated Entries are used

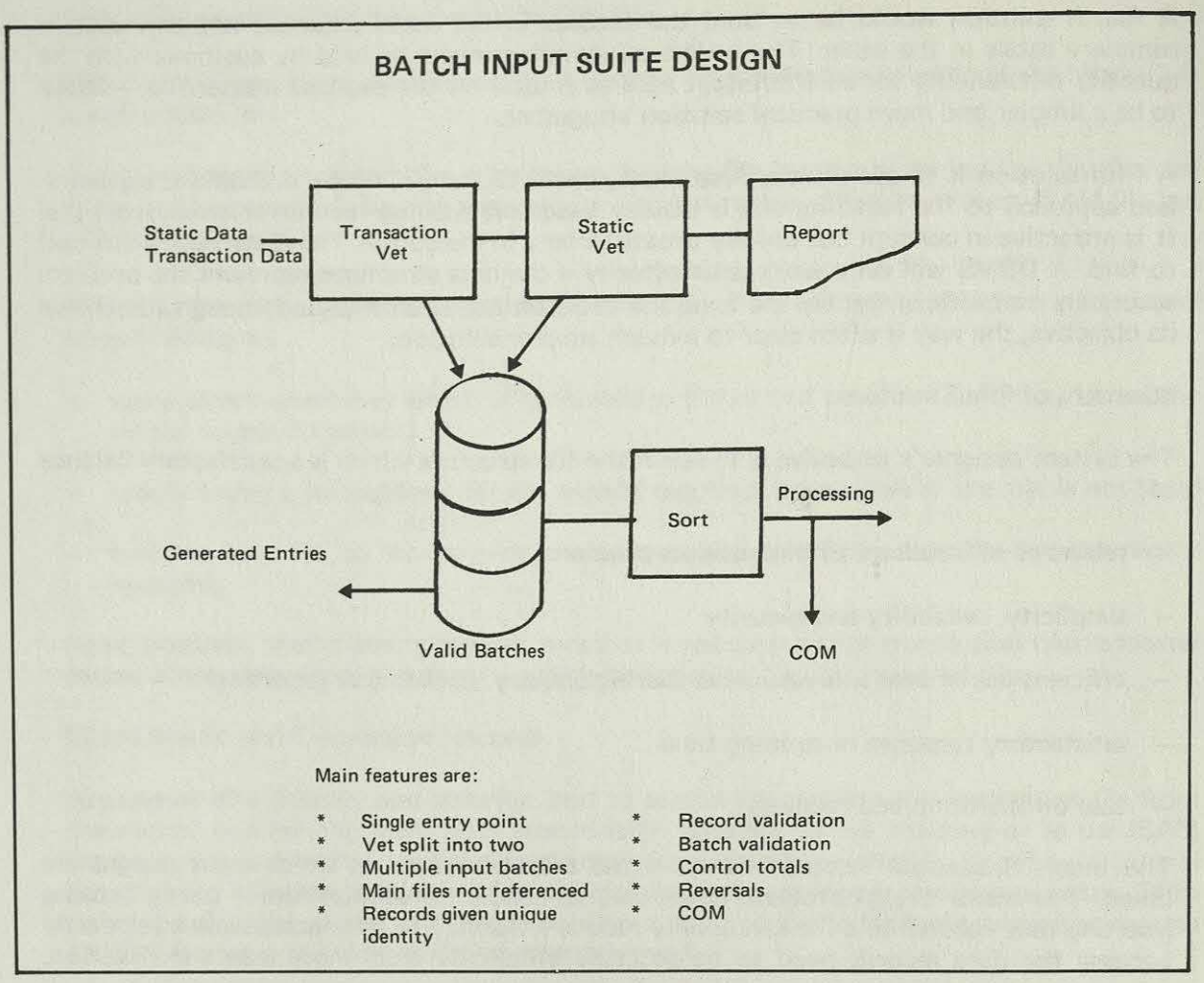


Figure 12

typically to initialise the transaction disc or they may enter the vet input stream. The transaction disc must have been previously cleared (not just overwritten with today's batches) to avoid restart problems.

In general main files are not referenced during the vet as there is not a great deal that can be done about rejections at this stage. A cycle delay would be imposed on transactions for newly opened accounts. On the other hand an on-line system needs to reference files during data entry so that the originator can correct errors before they get into the system.

Duplicate batch numbers should not be accepted and all batches must have today's run date. Written authority should be provided before a date override can be used.

Every input record should have a unique identity or be given one by the system — to allow for reversals and to provide an audit trail. Every record should be listed on input.

Individual records should be rejected if:

— a field is invalid

— the set is incomplete

— the set is out of sequence.

Validation of a record is quitted after three errors have been detected.

The whole batch is rejected only if:

- the batch header is unrecognisable
- the batch is a duplicate
- the batch date is invalid.

Control totals should include:

- record count by type, split between those accepted/rejected
- accumulation of value fields for accepted records
- hash totals of codes if there is no check digit verification (CDV).

Batch totals should be compared with controls in the batch header – but batches are not rejected because of a difference.

There should be provision for rejecting a complete input batch – which should be actioned before the sort of input transactions to posting sequence. There should be provision for rejecting individual records – actioned after the transaction sort. Control totals should be recalculated following the sort, when the original batch sequence has been lost (but records have retained their original reference numbers).

Where records are to be archived on microfiche the COM output should follow the sort – a sensible sequence for building the audit trail.

Guidelines for Designing an On-Line System

On-line is preferable to batch input because the terminal can be sited in the office or factory as an integral part of the workflow. With a batch input system the work leaves the natural work station and is processed by coders, punch and verifier operators and computer operators – all of whom add cost but no benefit.

On-line systems can do more than cut out the middlemen, however. They can reduce errors and complexity by validating the data entered in context. Customer orders can be checked against stock levels before the order is accepted; the customer's name and address can be checked against the order to make sure we have the right one; cash can be allocated to invoices and the customer's credit balance updated.

So on-line input is more than just data entry. To get the real benefits, main files must be on-line to the data entry programs and in many cases they will be updated in real time. Here then are the requirements of an on-line system:

1. Data should be validated in context.
2. All input should be proof listed in some way – journal entries would be listed, customer orders would be on an order confirmation, a name and address change would be on a change report.
3. Recovery from failure should be designed in – the need for re-keying should be avoided by logging the input and letting the computer reprocess the work after a failure.
4. On multiple VDU systems a common printer cannot, in practice, be run by the data entry programs.

Given these requirements, there are three ways of carrying out the functions of data entry

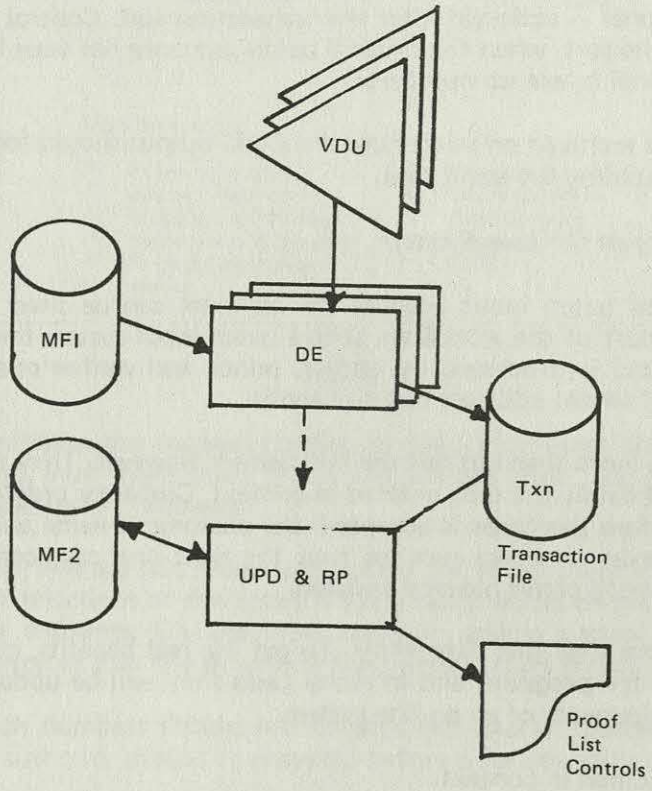
(DE), updating (UPD) and reporting (RP). The three ways can be described as:

- Real-time updating
- Pseudo real-time updating
- On-line input with batch updating

The real-time updating approach is well suited to main file maintenance. If transaction data is to be validated in context it must be possible to correct errors or to make changes with immediate effect. If a new customer is to be processed the sooner he is on the file the better. If an address or credit limit is wrong, it needs to be put right.

The Pseudo Real-Time Approach is illustrated in figure 13. Updating is carried out by a single program (instead of multiple VDUs) which picks up transactions soon after they are written. Updating is almost instantaneous, therefore, but is carried out at a single point.

PSEUDO REAL-TIME UPDATING



Transactions are validated and written to a transaction file. They are immediately picked up by a single program which does all the updating. Recovery is achieved by re-running the transactions through the update.

Figure 13

The purpose of this approach is to reduce potential queueing problems in the data entry programs and to split out work that can sensibly be delayed a few seconds. It would not help in an order processing system where the spread of file accesses is wide and the master files have to be read anyway. It applies in situations such as foreign exchange dealing where

currency positions and limits need to be updated from transactions – but the positions and limits files are not needed during data entry. The data entry programs are thus kept to a minimum and a potential bottleneck is removed.

The on-line input with batch updating approach recognises the problems associated with adding new transaction records to a file. It aims at combining the virtues of validation in context (an account balance field on the Master File Record is updated by the transaction) with secure file updating (the Transaction History File is updated on a father/son basis) and control (the account balance in the master record is checked against the total of the transaction records). The approach is well suited to applications such as sales ledger, purchase ledger and payroll where a transaction file is maintained in addition to the master file.

In practice an on-line system will use a combination of these three approaches depending on the requirements of the individual procedures.

Guidelines for File Updating

Freezing the design of the main files (data content, structure and access methods) is the key to smooth project development. The design should be complete before the specification stage begins. Any change thereafter should be written on a formal System Change Note and the implications thoroughly costed. Generally it is better not to make changes as they crop up – hitting a moving target is unreasonable to both the project team and the project.

Files divide the data entry and update from the reporting and business control side of the system. Here are the guidelines for updating these files:

1. Only one program should update each main file. Writing main files incorrectly, losing data and overwriting data is all too easy. By minimising the programs which write the main file, the scope for error is reduced and errors that do occur are easier to trace.

2. Transaction History files need to be purged from time to time. The history should be output to microfiche (COM) and not lost.

3. Apart from essential byproducts of the update, avoid extracting reports during the main update. Accounting needs are fairly static but reporting requirements are subject to change – so flexibility is obtained by separating the two functions.

4. More generally, areas subject to change should not be part of the main update. Where possible, complex areas should be taken into a 'back-end' phase as well.

5. Changes to a main file generated from within the system should circulate as a generated entry and be subject to normal input controls (batch totals, proof list, stops).

6. File descriptions (COBOL Data Division) should be set up on a catalogue and be copied into all programs. A degree of data independence is achieved and subsequent change can be accomplished more easily.

7. Files should carry control records containing identification, generation, date and control totals. The control totals should contain a count by type of record and totals of key fields.

8. Every time a file is read through or written, the controls must be recalculated and printed. On random access files special programs may have to be written to carry out the function on a regular basis. Generally special attention should be directed to random access files to ensure data integrity and to allow recovery from failure. A disc copy does not check the file controls.

9. A reference/parameter file should be used to control generations, and to contain dates, diary triggers, common tables and changing rates.

10. As far as possible business entities should be logically separated on file so that the risk of privacy and security violation is minimised.

Reporting Guidelines

Once the main files have been updated they can be exploited in a number of 'back-end' procedures. Examples of these back-end procedures are:

- Allocation of cash and discount to invoices
- Control of credit (aged debtors' report and statements)
- Allocation of newly received stock to outstanding orders
- Stock control (reordering and urging)
- Central Bank returns
- Payment of dividends to shareholders
- Management information.

The design strategy is to get the accounting entries into the system quickly and simply – and to keep the business procedures separate. Here are the guiding principles for the reporting phase:

1. The reporting suites follow the pattern of business procedures specified in the Functional Description.

2. Every procedure is a self-contained suite and may not reference another. If they have been divided correctly there will be no need for interdependence – a great help to the project manager.

3. The program suites may read any of the main files and may write to none. File changes are achieved by circulating generated entries back through the input cycle.

4. Extract files should be used to simplify complex main file structures – particularly for parameter driven reporting packages or when only some of the main file data is relevant. Holding extract files on disc allows a number of back-end suites to run in parallel.

5. Each suite will tend to consist of:

   - extract records required (with calculation)

     sort to desired sequence

   - print/output results (with calculation)

6. Printed reports should be uniform in appearance with standard headings for internal reports. Along the top of the report should appear:

   - program name, date and time of run
   - report name

– page number

Reports should end with 'END OF REPORT'.

Project Management Considerations

The end product of the design strategy is a set of procedure run charts as illustrated by figure 14. There is one chart for every procedure with an overview at the beginning.

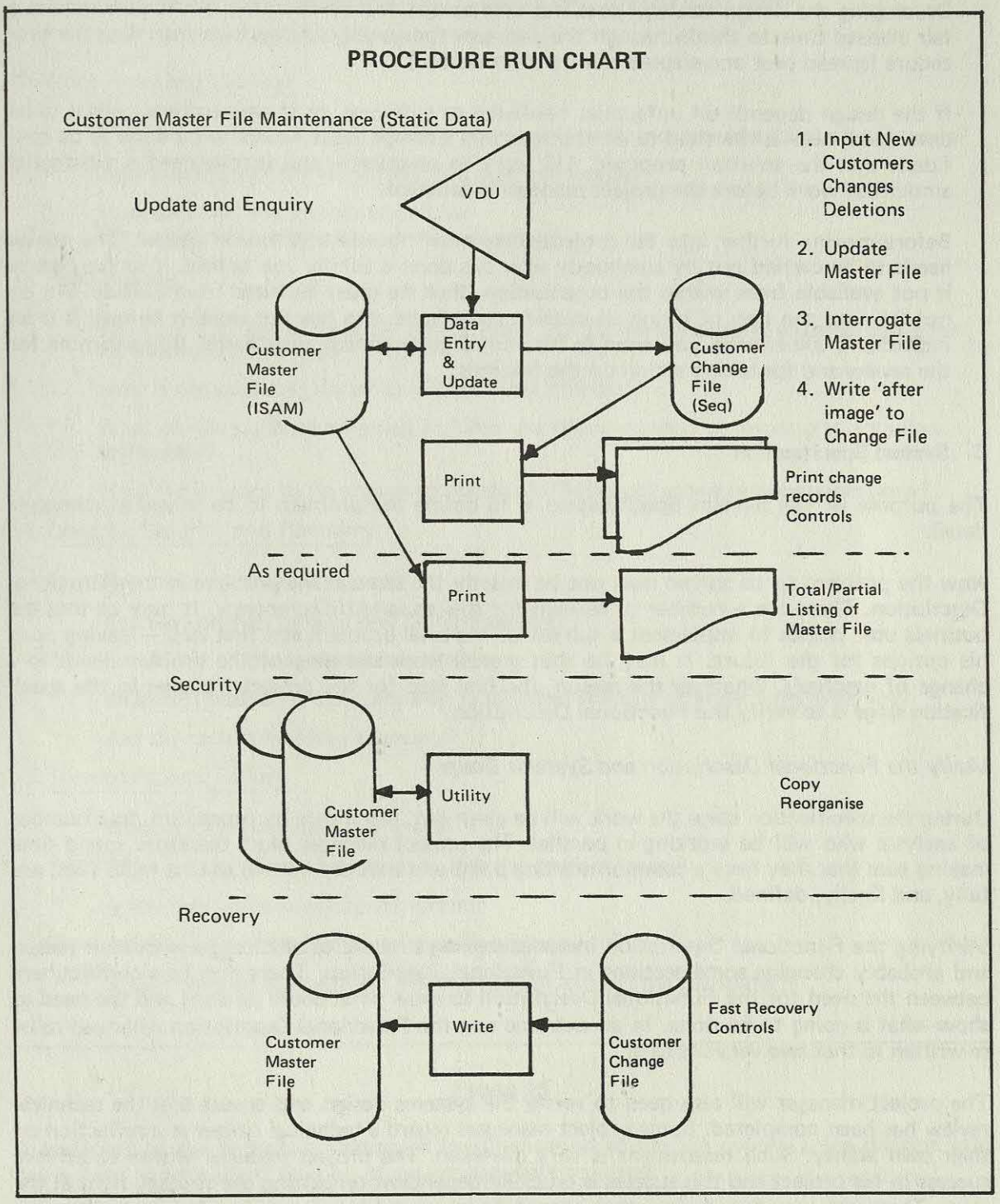


Figure 14

Alongside the individual programs there should be sufficient explanation to make the function of the program clear.

These procedure run charts may be included in the Functional Description as an appendix or be bound separately. If they are bound separately the reader will also need to have access to the file contents, structure, volumes and file sizes — sections 5 and 8 in the Functional Description, probably Section 2 as well.

Developing the design strategy does not take a great deal of mantime, but it does require a fair elapsed time to think through the problem thoroughly. Allow two man days per procedure (spread over one elapsed week per procedure).

If the design depends on unfamiliar hardware or software, or if new software needs to be developed, now is the time to do the research. Enough work needs to be done to be confident that the solution proposed will work in practice — and it may need a substantial amount of work before the project manager is satisfied.

Before moving further into the project *there must now be a technical review.* The review needs to be carried out by somebody who has done a similar job before. If such a person is not available from within the organisation, then he must be hired from outside. But do not fall into the trap of hiring an outside consultant who has not done it before; it is an individual's experience you need to hire not a good consultancy name. Build in time for the review and for taking action on the findings.

C System Specification

The purpose of the System Specification is to define the problem to be solved in complete detail.

Now the problem to be solved may not be exactly the same as the problem in the Functional Description. There are a number of reasons for this apparent discrepancy. It may be that the business user wishes to implement a sub-set of the total problem as a first step — leaving open his options for the future. It may be that greater understanding of the problem leads to a change of emphasis. Whatever the reason, the first step for the project manager in the specification stage is to verify the Functional Description.

Verify the Functional Description and Systems Design

During the specification stage the work will be given out, procedure by procedure, to a number of analysts who will be working in parallel. The project manager must therefore spend time making sure that they have a common starting point and that the interfaces (the main files) are fully, and finally, defined.

Verifying the Functional Description includes making a record of all changes with their reason and probably changing some sections in Functional Description. There may be a conflict here between the need for the Functional Description to show what could be done and the need to show what is going to be done. In an extreme case the Functional Description will need to be re-written so that two versions exist.

The project manager will also need to verify the systems design and ensure that the technical review has been completed. Some project managers regard a technical review as a reflection on their own ability. Such resentment is very mistaken. The project manager wishes to achieve success in his project and this success is crucially dependent on getting the strategy right at the start. There are very few designs that cannot be improved by discussion. Anyway, when the project is a success the project manager and team will get the credit — the fact that he sought a second opinion from time to time will add to, rather than detract from, his reputation. Figure 15 contains a checklist for a technical review.

CHECKLIST FOR THE TECHNICAL REVIEW

1. Preparatory Briefing

   * Is the Functional Description complete?
   * Is the Design complete?
   * How are the business needs likely to evolve over the next few years?
   * Why has this design been chosen rather than . . ıy other?

2. Work Flow and Timings

   * What are the timings of the runs and how will peak and period end jobs be fitted in?
   * How will the system work from the user's viewpoint?
   * How easy will the system be to use?
   * Where are the bottleneck and the risk areas?

3. File Strategy

   * What are the file accesses per transaction? Are the files organised to be efficient for transaction processing?
   * How is the updating done? Is it secure and efficient?
   * What are the patterns of enquiries? Are the files organised to respond to enquiries efficiently?
   * How is file space to be organised physically? How will space requirements grow?

4. Control, Security and Recovery

   * Are input controls adequate?
   * Are file controls present and comprehensive?
   * Is security of the system adequate?
   * What will happen if the computer fails? What standby arrangements are there?
   * Are the restart facilities adequate?

5. Risk of Project Failure

   * What are the hardware risk areas?
   * What are the areas of innovation (for this project team)?
   * Is the procedure breakdown correct?
   * Are the project budget and timescale realistic?
   * Which parts of the solution might be better left off the computer?

Figure 15

A key part of the technical strategy is the finalisation of the file contents, the division into procedures and the identification of system software required.

- The file contents need to be frozen as the interface between the procedures.
- The division into procedures allows parallel project development in identifiable steps.

– The system software (screen handling utilities, VDU conventions, file handling utilities, spoolers, common functions) needs to be identified now so that it can be ready, fully tested, by the time the application programming begins.

Once all the preparatory design work has been completed, the analysts can begin work on the systems specification.

Systems Specification

Preparing the systems specification is not merely a question of filling up a set of standard forms. It requires a high degree of problem solving ability as well. A poor analyst will define the main problem flow and add the exceptions as they occur to him, as something of an afterthought. A good analyst will build the exceptions into a more general solution to the problem.

Systems documentation can be an important aid to good design, but it is no substitute for analytical ability. We will therefore cover the principles of developing the systems specification before going into the documentation mechanics.

The steps in developing the systems specification are illustrated in figure 16.

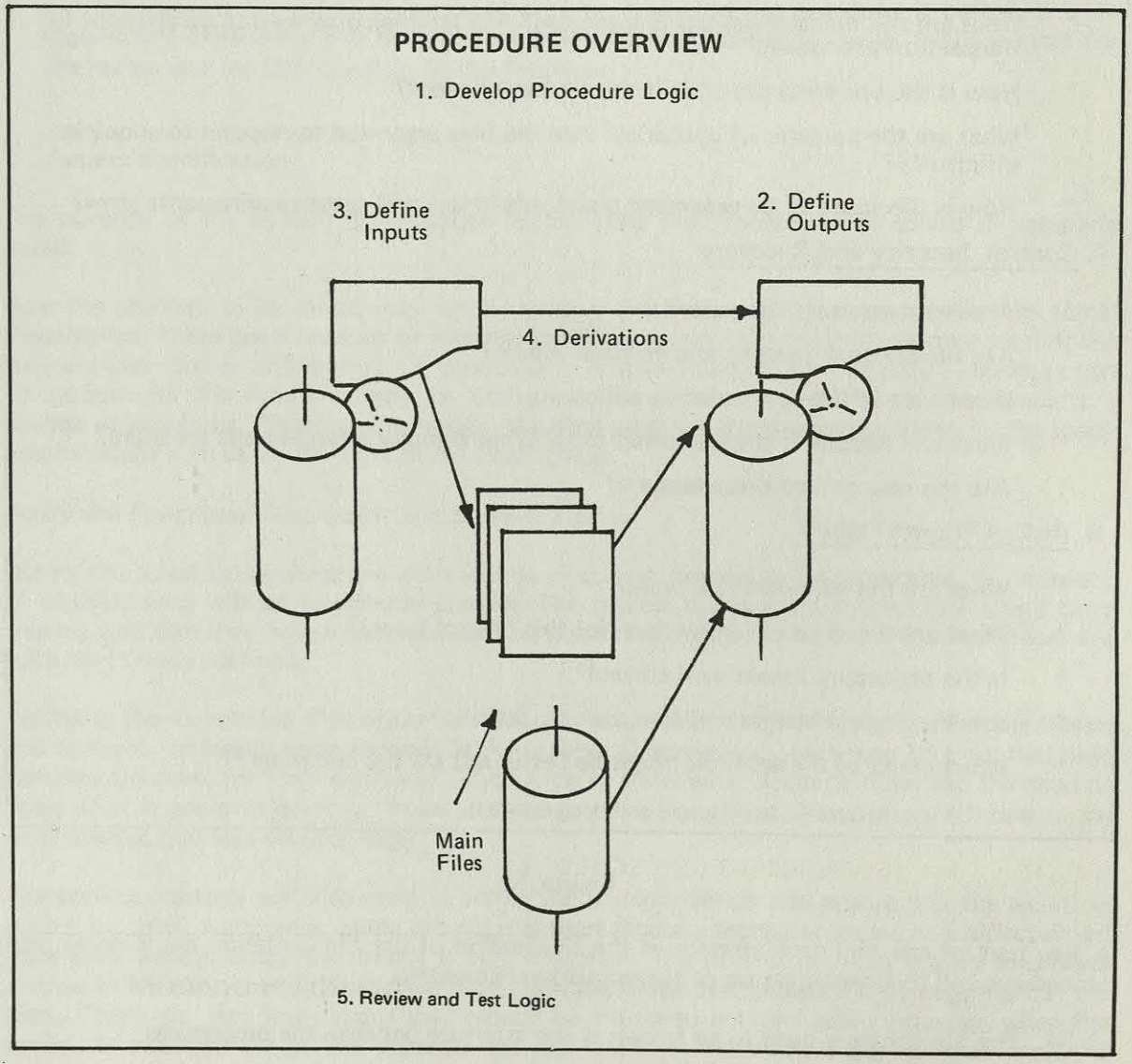


Figure 16

Developing the procedure logic consists of breaking the problem into a number of logical steps. The steps should be logical in the sense that they follow the natural problem sequence (whether the solution is to use a computer or manual methods). An example is shown in figure 17.

LIST OF TRANSACTIONS PROCEDURE

READ	SORTED REQUEST FILE
BUILD	TABLE OF REQUESTS
READ	WHOLE STOCK HISTORY FILE (transactions are held by portfolio within stock)
For every stock:	
SELECT	ALL TRANSACTIONS FOR REQUESTED PORTFOLIOS
REJECT	CONTRAS AND TRANSACTIONS OUTSIDE DATE RANGE
GET	CORRESPONDING STOCK RECORD
REJECT	TRANSACTIONS IF SECURITY NOT FULLY OR NOT PERMANENTLY PARTLY PAID
For selected transactions:	
IDENTIFY	TRANSACTIONS FOR AGI AND YIELD CALC
IDENTIFY	DIVIDEND FIGURE FOR AGI AND YIELD
CALCULATE	AGI & YIELD
BUILD	EXTRACT RECORD FOR SORT
---SORT	RECORDS TO PORTFOLIO SEQUENCE ----
READ	SORTED EXTRACT RECORDS
GET	CORRESPONDING PORTFOLIO RECORD
For each portfolio:	
PRINT	MAIN SCHEDULE – SALES
PRINT	CGT SUPPLEMENT – (IF REQUIRED)
PRINT	MAIN SCHEDULE – PURCHASES
PRINT	SUPPLEMENTARY SCHEDULE

Figure 17 EXAMPLE OF PROCESSING STEPS WITHIN A PROCEDURE

The next stage in the problem development is to expand the processing steps until they are fully defined. The analyst will have to show the layout of printed reports and VDU screens. He will also have to expand the processing rules until he has shown how every output field is derived.

He should tackle the work systematically defining first outputs and then inputs and intermediate files. The derivation rules which come next are written out on 'derivation sheets'.

Every file, every printed report, every screen layout and every field which is referred to in the derivation sheet must be cross-referenced. More generally, the rule is that every output field should be traced back to its source – either directly or through derivation sheets.

The analyst thus views the system specification from two directions. In terms of business function the processing steps and derivation sheets flow from top to bottom and from beginning to end. In terms of output field definitions the flow is from the final result back to the source.

The systems specification is thus well structured in a problem solving sense and completely defined in a technical sense.

The specification is also organised to suit the programmer, avoiding the need for a separate programming specification. In the situation where there is an intermediate file within the procedure (the Sort File in figure 8.4 for example) the final output is referenced back to the sort record (which is the input to the second program). The sort record is the output of the second program and its fields are referenced back to the prime input. The specification can thus be organised into the two programs.

User Role

During the preparation of the specification the analyst must work closely with the user representatives. Printed layouts and VDU screens need to be agreed as they are developed. Similarly the analyst will wish to agree the derivation rules with the users progressively – rather than waiting until the specification is complete and presenting them with a dauntingly large problem. Notice that structuring the processing steps in a business oriented way helps the analyst and user to concentrate on one aspect of the problem at a time.

The final task in preparing the specification is to review and logic test the specification with the user concerned. The analyst checks that the derivation rules for output fields are complete and unambiguous. He further reviews the processing steps to make sure that the steps give the reader a good understanding of the problem.

Meanwhile, the user representative assembles test conditions covering both general and special situations. The analyst and user then take the items of test data in turn and work them through the specification, confirming that all the conditions have been catered for. Rectifying mistakes later in the project is going to be expensive so time spent on logic checking at this point must not be begrudged.

Technical Role

In all this specification work little reference has been made yet to the involvement of programmers and technical specialists. In fact they have an active role in three areas:

- The technical review determines the technical strategy and thus allows the analyst to concentrate on the business problem during the specification.

- Technical advice is available to the analyst during his specification work – in particular the processing steps need to be confirmed.

- System software is being developed in parallel with the specification so that application programming will not be held up later.

Technical work of this nature requires a very experienced systems programmer. It is normal to appoint this 'strategic programmer' during the specification stage of the project. But it is unlikely that any other programmers will be needed yet, unless the system software is very substantial. We still need quality, not quantity.

Standard Documentation

A sound approach to systems documentation has the following advantages:

- method of communicating with users and programmers
- aid to good systems design (encouraging a problem oriented structure and avoiding omissions)
- permanent record which assists subsequent system maintenance
- record of progress in the project stage
- single system for analysts and programmers to work from.

There are, however, a number of traps to be avoided:

- excessive paperwork is an obstacle to understanding
- business needs should not be sacrificed to computer needs
- transcription before programming should be avoided
- bad documentation will not get maintained
- simple narrative is not enough; it is ambiguous and does not encourage completeness.

Project Management Considerations

The steps the analyst has to follow while developing the specification are the following:

1. Familiarisation with the problem area – remember that the analyst may be just starting on the project and working in an unfamiliar area. Allow two to three weeks for familiarisation, longer if the area is very technical.

2. Writing the specification – in addition to the procedure summary sheet there are outputs and inputs to be agreed, derivation rules to be developed and agreed. Allow two days for every output, input and file (one day for existing main files). Generally this allowance will cover time for preparing derivation sheets unless the procedure is very complex.

3. Review and logic testing – the user representative should be taken through the logic step by step. Allow two weeks for this process.

These time allowances assume that the design is complete, main file layouts have been finalised and the technical review has been completed. The time for this preparatory work should be estimated separately.

On a substantial project, several analysts may be working in parallel on the systems specification, taking a procedure each. In addition there is likely to be systems programming effort to set up standard functions – the nature of these functions is explained in Section III D.

Quality control of the work is essential. The project manager needs to be satisfied with:

1. The analyst's grasp of the business problem – in particular the logical division into processing steps (on which the program structure will depend).

2. The user representative's commitment to the completeness and accuracy of the specification.

D Programming

The aim of the project team is to develop systems which are robust, easy to maintain, flexible to business change and secure. We have seen that these aims are most likely to be achieved if the system and program structures mirror the business problem:

- the system is divided into business oriented procedures
- the data structures are organised to process routine operations, management control reports and ad hoc enquiries efficiently
- the processing steps on the procedure summary sheet reflect the logic of any good solution
- the program structure follows the processing rules.

If programs can reflect the business problem in this way, they are easier to understand and more likely to be responsive to business change. The programs are thus highly structured with the main modules corresponding to the systems analyst's processing steps.

Now the average productivity of programmers is disturbingly low. As a rule of thumb, it takes ten weeks to write and test a COBOL vet or update; four weeks to write and test a report program. The excellent programmer, however, aims at an order of magnitude improvement on these times; more like a program in a week or two.

Five days per program is a very high order of productivity. Clearly it is not achieved by writing every program as a new adventure. We need to give the application programmers a set of 'tools' so that they can concentrate on the business application (without having to re-invent standard logic).

If common functions can be developed for a suite of programs, the coding of the individual programs will be faster. If common functions can be developed for the complete department, the development time for every suite will be reduced. Using this approach, it follows that program maintenance will be easier and less dependent on the individual's knowledge of his idiosyncratic program.

The 'strategic programmer' has the job of organising the programming effort to achieve this sort of productivity. His role can be seen to have three main elements:

1. Understanding the Functional Description (business need) and the system design strategy.

2. Identifying and developing the tools (common functions) the application programmers will use.

3. Ensuring programs are soundly constructed and gross inefficiency is avoided.

Systems Design

In general the strategic programmer is involved in the project from the Functional Description stage. While his involvement is initially on a part time basis it is important that he is appointed and involved early. The division into procedures and division of procedures into processing steps need to be satisfactory in a technical sense as well as being business oriented.

The functions to be carried out during the preparation of the Functional Description, the Systems Design and the Technical Review have been explained in earlier chapters. The end result is that the strategic programmer:

- has a thorough knowledge of the requirements of the system and is clear on the analyst's intentions
- has agreed the functional description and the system design and resolved any hardware/software conflicts
- has prepared a detailed block diagram for the suite and checked that the procedure summary sheets correspond to the block diagram.

We will now look at the strategic programmer in his role as 'toolmaker'.

Developing Common Functions

Figure 18 contains a checklist which summarises the areas where common functions may be necessary. In addition to these functions, 'skeleton' programs are needed for the more common program types.

The job of the strategic programmer (as toolmaker) is to identify the common functions required and to make sure they are specified, coded, tested and documented in time for the application programming. Generally he will write the common functions himself. A key part of his job is making sure that the team is aware of them and that they get used.

Program Construction

Standards are no substitute for intelligent programming but they help programmers avoid known problem areas and improve program readability.

All programmers should have their programs checked to ensure they have followed the standards. In particular the programs are checked for:

- structure
- use of standard functions
- simplicity
- efficiency.

Project Management Considerations

During the programming stage the project manager should not be concerned with detailed coding. He should concentrate on ensuring that the problem solution is structured correctly at every level.

The system analyst should hand over a specification which is well structured and which does not contain endless exceptions. The strategic programmer should build the tools so that the application programmers can be highly productive once they start. The application program-

CHECK LIST OF COMMON FUNCTIONS

1. File and record descriptions, special file access routines.
2. System utilities (start up, close down, backup).
3. Field validation (e.g. CDV, numeric, date).
4. Field packing/unpacking (e.g. name and address, arithmetic fields).
5. Date conversion, calculation of days between dates.
6. Field editing (e.g. code display, floating asterisk, leading zero insertion.
7. VDU handling (e.g. characteristics, page mode screen handling).
8. Special peripheral handling (non-standard peripherals or use).
9. Complex arithmetic or logical functions (e.g. Gross Redemption Yield).
10. Use of parameter file for batch/generation control.
11. Use of special software (including operating system).
12. Conventions for operator response.
13. Programming conventions.

Common functions are the tools used by the application programmers. They need to be specified, coded, tested and documented before application programming can begin.

Figure 18

mers should have logically structured programs which reflect the systems specification and which will be robust enough to cope with change. Where applicable they should follow the standard program skeletons.

In short, the job of the project manager is all about quality control.

Time estimates depend very much on the quality of earlier work. It should be possible to write and test small to medium-sized programs in two to three weeks; substantial programs in four to six weeks – provided that the preparatory work has been carried out thoroughly. The actual rate will depend on the installation and the problems they face. Targets can only be set by measuring actual performance while striving constantly to raise standards.

E Program And System Testing

As we have seen, a computer project starts with a statement of the business needs (Functional Description) and is then developed in progressively greater levels of detail (Systems Specification and Procedures, Program Specification and Programs). It is the purpose of 'testing' to ensure that the communication between each level is verified and that the end product satisfies the business need. The concept is illustrated diagrammatically in figure 19.

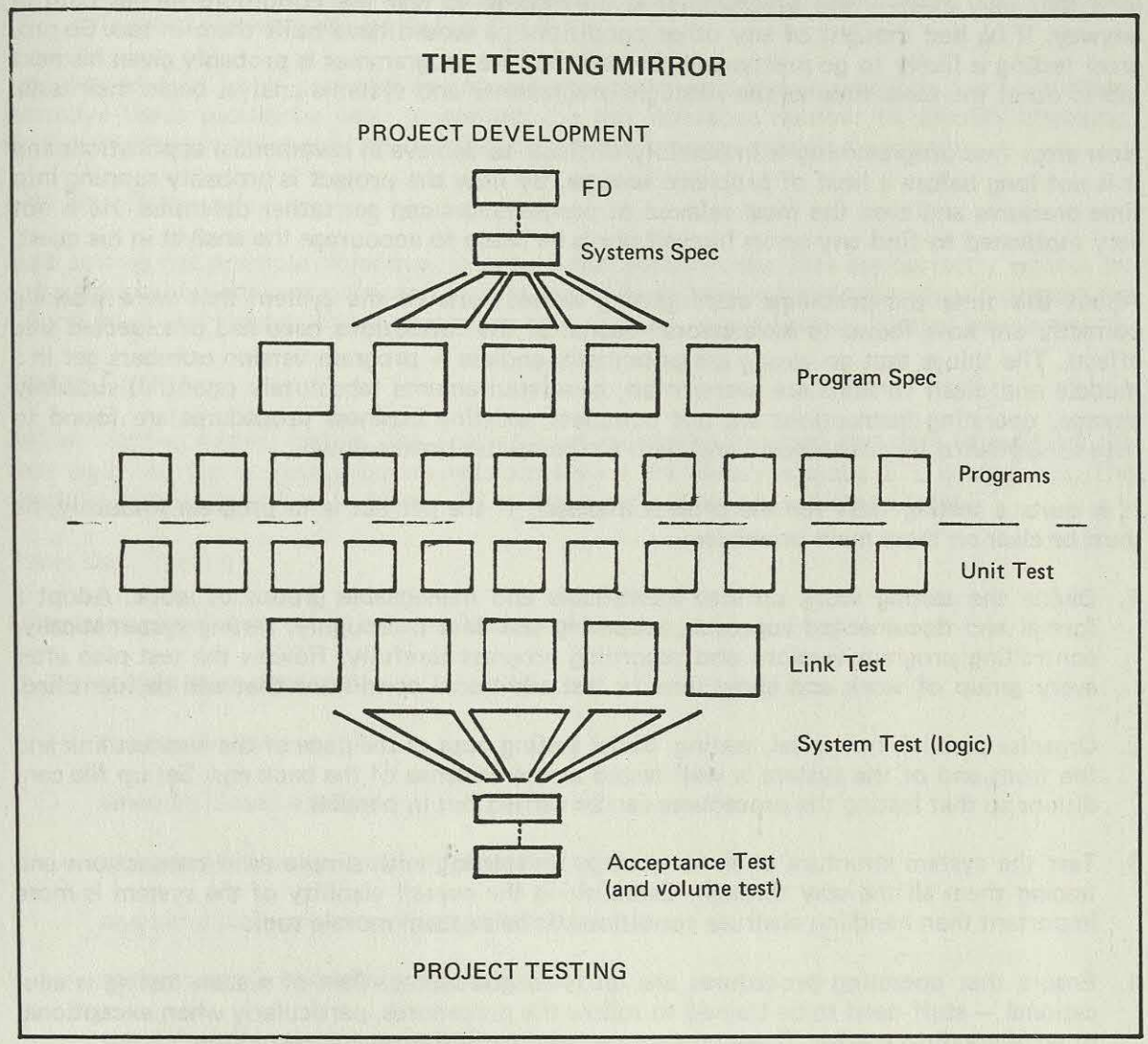


Figure 19

Unit testing is carried out by the programmer, whose aim is to ensure the program works as specified. Unit in this sense generally involves several modules which make up a program.

Link testing is carried out by the strategic programmer, whose aim is to ensure that the programs link together as designed. Link testing is also known as suite or integration testing.

Systems testing is carried out by somebody other than the programmer, usually the systems analyst, whose aim is to ensure that the procedure works according to the system specification. System testing includes date testing (advancing the calendar over year ends) and testing restart recovery.

Acceptance testing is carried out by the final users, whose aim is to ensure that the system

works within the business environment and satisfies their needs. Acceptance testing also includes volume testing (abnormally high and low) using live data.

On the face of it testing is straightforward enough. A testing plan is developed systematically to cover normal and abnormal conditions with good and bad data. The test data is passed through the system and any errors picked up and corrected.

In practice it is not quite so easy. Unit testing, for example, is unlikely to reveal the real problems that may exist – the programmer is only going to test the conditions he has built in anyway. If he had thought of any other conditions he would have built them in too. So program testing is likely to go pretty well. So well that the programmer is probably given his next job to do at the same time as the strategic programmer and systems analyst begin their tests.

Now error free programming is impossibly difficult to achieve in commercial applications and it is not long before a host of problems emerge. By now the project is probably running into time pressures and even the most relaxed of programmers can get rather defensive. He is not very motivated to find any errors himself nor is he likely to encourage the analyst in his quest.

About this time the problems start getting worse. Parts of the system that were working correctly are now found to have errors – some of the corrections have had unexpected side effects. The things that go wrong are potentially endless – program version numbers get in a muddle and clean versions are overwritten, new requirements (absolutely essential) suddenly emerge, operating instructions are not complete, existing business procedures are found to have fundamental inconsistencies, and then the computer breaks down.

It is quite a testing time for the project manager. If the project is to progress smoothly, he must be clear on these main principles:

1. Divide the testing work up into identifiable and manageable groups of work. Adopt a formal and documented approach, preparing test data thoroughly, testing systematically, controlling program versions and recording progress carefully. Review the test plan after every group of work and allow time to test additional conditions that will be identified.

2. Organise parallel, not serial, testing. Serial testing goes at the pace of the weakest link and the front end of the system is well tested at the expense of the back end. Set up file conditions so that testing the procedures can be carried out in parallel.

3. Test the system structure at an early stage by starting with simple valid transactions and tracing them all the way through. Establishing the overall viability of the system is more important than handling abstruse conditions (it helps team morale too).

4. Ensure that operating procedures are ready in good time. Part of system testing is educational – staff need to be trained to follow the procedures, particularly when exceptional or error conditions arise. These procedures need to be thoroughly tested too.

5. Organise the resources well in advance (machine time, media, test conditions, test files, test utilities) and make generous time allowances. After every group of tests, repeat selected tests from earlier groups to make sure that they have not been affected by subsequent changes.

Program Test Data

It is not possible to test every combination of the paths in a program, but the programmer should aim to test all conditions that he can visualise. In particular he should aim to:

- test every path at least once

- test boundary conditions on branches

- test all limiting values (explicit or implicit) to their limit, e.g. maximum number of parameters, print lines, entries in a table

- check that initialisation is correct after every processing level (and work areas cleared)

- check the first and last record are handled correctly on data files (read and written); check records with duplicate keys are processed correctly

- check file dumps of all files written against computed results (accuracy) and file specifications (format) using a *logical file dump.*

Test data should be as compact as possible consistent with creating realistic test conditions. Narrative fields should be used to contain the test reference number to simplify checking. Programs after the vet should assume that input data has been validated correctly.

Link Test Data

Link testing has a simple objective; to ensure that intermediate files are correctly written by one program and are correctly read by the next. The strategic programmer should prepare the minimum of test data to check that every field of every record is being handed over correctly.

System Testing

Before starting system testing, the analyst must re-run the program test data to establish the base position, the correct program versions are on the library and the JCL is complete. The checklist of test conditions in figure 20 is organised below into groups of tests.

Static Data Testing

Open, close, amend accounts using valid data. Also:

- reopen open accounts

- reclose closed accounts

- amend a closed account

- amend and close an account (same test run)

- open and close an account (same test run)

- make multiple amendments to accounts

- use all possible data entry methods.

Repeat using invalid data:

- pass transactions against closed and non-existent accounts

- check all numeric fields with alpha, blank, CDV, leading blanks, trailing blanks, embedded blanks, invalid dates

- check ranges both sides, near cut-off

- pass combinations of valid and invalid values, multiple errors

- check arithmetic fields for overflow, negative values, rounding, truncating, zero divide, alignment

SYSTEM TESTING PLAN

1. Static Data Testing

2. Transaction Data Testing (to update)
   - type (a)
   - type (b)

3. Procedure Testing (back end)
   - procedure (a)
   - procedure (b)
   - procedure (c)

4. Volume and Date Testing

5. Restart/Recovery Testing

Figure 20

- pass data set with sequence errors, omissions, duplicates
- pass batches with invalid dates, duplicates, missing headers, missing trailers, incorrect totals.

Procedure Testing

Set up the files to test each (back-end) procedure thoroughly, but do not set up impossible conditions on the files. The back-end programs have to assume the front-end programs have done their job. Carry out link testing with all related procedures within and outside the system.

Volume and Date Testing

Text extremes of volume and date changes:

- force continuation pages on all reports
- force multi-reel, multi-volume working (short tapes)
- pass high volumes of real data (as part of Acceptance testing)

- run two days as one (different run dates) with both being processed correctly
- process null requests (request with no transactions)
- advance the calendar over month end, February end, quarter end, year end, financial year end, tax year end
- run with no data at all.

Restart/Recovery Testing

Test the operational running of the system:

- flow of work and continuity of the suite
- accuracy of operating procedures
- restart following a checkpoint
- recovery following an input suite failure, update failure, reporting suite failure
- use of stationery, COM, transmission.

Acceptance Test Data

The purpose of the acceptance test is not for the user to satsify himself that the system works— he is not particularly concerned with the detail of how it works.

We want him to pass a wide variety of transactions against relatively few accounts so that he can check system performance — but bear in mind that he is not likely to be familiar with system testing procedures.

Ask him to:

- keep a note of the different transactions that occur over the next couple of weeks
- select a small representative sample of clients/accounts
- select a variety of the transactions that have occurred.

The project team should arrange to set up the files to their starting position (from the data provided by the user), to complete the input documents and to run the test.

When the test is successful it will be possible to get the user involved in more elaborate testing:

- filling in the input documents himself
- putting in more complex conditions, especially those he thinks will break the system
- testing month-end and cyclic conditions
- testing the most important customers/accounts/stocks
- high volume testing and checking.

It is most important that each step is thoroughly tested before going on to the next. Finding errors does not breed confidence and it must be clear that the project team expects to find and remove them.

Project Management Considerations

Programmers should be responsible for their own unit testing although it is up to the project manager to ensure that they are not held up for lack of resource. Similarly the strategic programmer is responsible for link testing.

The project manager has significantly more to organise during system and acceptance testing.

During system testing it is the responsibility of the project manager:

- to divide up the testing load and prepare a strategic plan, getting the analysts to prepare test data while the programs are being written
- to bring forward activities as far as possible into the end of program testing:
  * libraries correct and up to date
  * systems test data finalised and results calculated
  * test files initialised and set up to test procedures in parallel
  * 'incorrect' input correctly prepared
  * job stacks, discs and tapes prepared
  * validation tests completed
- to have a contingency plan for testing a subsystem if programming is running late overall
- to control new program versions, documentation, re-running of earlier tests
- to test the file conversion suite as thoroughly as the other procedures
- to include all non-standard or external interfaces – COM, EFT, data transmission, external tapes
- to create a permanent test pack for ongoing system maintenance
- to check actual running times against estimates and to act decisively if need be.

During acceptance testing it is the responsibility of the project manager:

- to help user managers to prepare test data (using the guidelines above), to give them clear instructions about what to check and how, to get them to accept procedures as they are completed
- to get user staff to log queries on official forms (informal reporting leads to an abundance of trivia)
- to ensure that test data includes abnormally high and low volumes (checking high volumes can be a daunting task – make sure a separate room is allocated with sufficient staff)
- to ensure the system has been tested over quarter end, year end, February end, financial year end and tax year end
- to ensure the system runs satisfactorily in the operating environment (forms, VDUs,

batch control, restart, standby, I/O messages, operating procedures, multi-reel files, labels and generations).

Include the most important clients/accounts/items in these trials so that you are quite sure they will be well served by the new system.

F Procedure Manuals

Among the most important parts of a new system are the procedure manuals. It is through manuals that the users learn to operate the system and to understand its capacity, so the quality of the system is often judged by the quality of the manual.

A problem we must accept is that manuals are required to serve more than one purpose:

- Instruction (frequently at both a high level and a detailed level)
- Reference (in case of absence of key staff or a non-routine event occurring)
- Control and review (management control, work assignment, auditing, documentation).

Happily if the first of these purposes is fully satisfied the others will be too; and, provided the job is tackled in the right way, writing a good manual need not be too difficult.

1. User Procedure Manuals

The purpose of this section is to provide guidelines so that procedure manuals can be prepared in a clear and uniform way.

Structure

We need to be clear on the purpose for which the manual is intended before deciding the structure. The objectives will probably be:

- to describe the function, limitations and operation of the system
- to provide each user department affected with the information needed to operate its area
- to give step-by-step instructions on how to use the system and for sending information to the system.

Notice three principles that emerge from these objectives:

1. It is necessary to give an overall appreciation of the system before going into detail.
2. The structure of the manual should recognise the organisation structure of the user – at least at section level.
3. The writing style must suit the reader.

Following these principles, the overall structure of the manual is deceptively simple:

- An Introduction which states the purpose of the manual and gives an overview of the complete system covered in the manual
- A section for every logical function
- Sections containing routines common to several functions.

The trick is to get the logical functions right. Usually there is a conflict between organising the manual to reflect the work done by a user department (e.g. Sales Office, Data Control) and dividing it by major functions (Order Processing, Sales Ledger). Should it be function within department or department within function?

Probably department within major function is right (a business will tend to be organised such that departments mirror the major functions anyway); but be prepared to produce the manual in two sequences if necessary. It must be possible to split a large manual into stand-alone sections — otherwise the size would make it daunting and inconvenient to use.

SPECIMEN CONTENTS OF A PROCEDURE MANUAL

INVESTMENT MANAGEMENT LTD

INVESTMENT ACCOUNTING SYSTEM
PROCEDURES MANUAL

CONTENTS

IAS PROCEDURES July 1975

Figure 21

Common routines and those likely to require frequent reference should be separated out from the main procedures. Examples of such routines are:

- VDU signing on and operating conventions
- Standard error messages
- Code Book
- Supervisor's operating instructions (minicomputer).

In general, try to avoid using appendices. An appendix means cross-referencing and looking in two places to find the explanation. Our aim is to allow the reader to find all the information required to do the job in one place. It follows that some instructions may need to be repeated in more than one place.

Detailed Structure

The heart of the procedure manual consists of sections for every logical procedure in the system. The procedures are likely to correspond to systems 'procedures' such as Customer File Maintenance, Processing Customer Orders, Cash Allocation (both manual and computer procedures are involved).

The detailed structure within the section depends on the task to be done. The five main types of task are:

1. Writing an Overview of the procedure
2. Filling up forms
3. Entering input through a VDU
4. Explaining VDU and printed output
5. Performing other clerical functions.

Procedure Overview

The overall system is described in the manual in progressively more detail. First it is broken down into procedures, then the procedures are broken down into functions, and so on, until we eventually reach the individual steps to be carried out. At every level the same approach is followed:

Start with an overview before going into detail.

So in the same way that the manual starts with an overview of the whole system, every procedure starts with a descriptive overview.

It may also be necessary to describe the flow of work, which is done using the procedure narrative style or flowchart.

Filling Up a Form

Provide a specimen of the form to be completed facing its completion instructions. The completion instructions should be in two parts; the purpose of the form and general instructions on its use, facing detailed instructions for completing every entry.

VDU Input

VDU input requires:

- the originating document (with specimen entries)
- the VDU layout (highlight/box operator entries)
- completion instructions.

Attach the originating document to the VDU layout (so that they fold out together) on the page facing the completion instructions. (Greater originality may be required if several almost identical forms use a single VDU layout.)

Explaining Printed Output

To describe output, provide a realistic specimen on the page facing the explanation. The standard headings for the explanation are shown in figure 22.

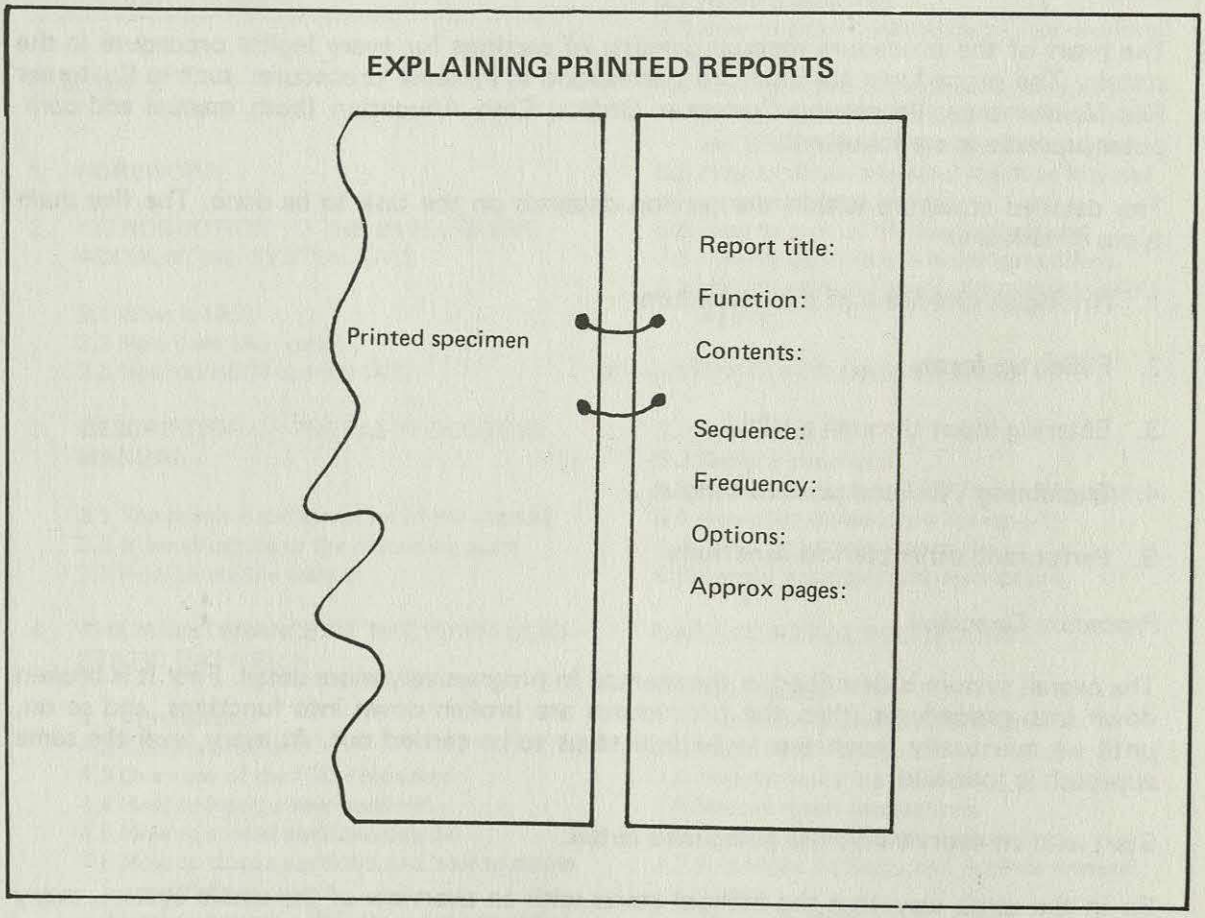


Figure 22

Describing Clerical Procedures

A specimen clerical procedure is illustrated in figure 23.

Note the way that the function (Accepting Savings Deposits) is described briefly then broken into, in this case, three routines. Every routine similarly commences with a descriptive sentence followed by the detailed steps. Write the steps as simple imperative statements.

EXAMPLE OF CLERICAL PROCEDURES

ACCEPTING SAVINGS DEPOSITS

Savings Deposits are accepted by counter clerks who must verify the passbook before giving the customer a receipt. The counter clerks also enter interest earned into the passbook.

7.1 VERIFICATION

You must verify four items on every deposit you receive:

- Verify the **amount** by counting the cash and cheques and comparing the counted amount with the amount written on the deposit slip.
- Ensure that all **cheques** are correctly made out and signed.
- Compare the **account number** on the deposit slip with the account number in the passbook. They must be the same.
- Read the **name and address** on the deposit slip and in the passbook to see that they are legible, and are the same.

If the customer does not have his passbook, verify the **account number** and **name and address** on the Customer Static Data (CSD) microfiche.

7.2 RECEIPTS

You must give a receipt for every deposit:

- Give a receipt in the passbook by writing the amount of the deposit in words, updating the balance column and stamping it with the date and your initials.
- If the page is full, enter the Carried Forward balance against Brought Forward balance at the top of the next page.
- If the customer does not have his passbook, give him a receipt by making a copy of the deposit slip. Stamp it with your 'duplicate' stamp, write in the amount and your first initial and name.

NAME
A/C No.

Date	Amount in words	Signature	Balance £
1.4.75	One hundred pounds	B/Fwd	100.00
1.7.75	Fifteen pounds deposit		115.00
1.7.75	Two pounds 25 interest to 30.6.75		117.25
		C/Fwd	

7.3 INTEREST

etc.

Figure 23

Note also that all the instructions within the function relate to the job of one person (the Counter Clerk). Do not mix the jobs of different people in one function.

Getting the Facts

Any new system is designed for its users. It is their system which they are going to run — not the designers'. Ideally they should write their own manual with expert help. If a Systems Analyst is the author he needs the user's full involvement to do the job properly.

Normally it will be the section head in the affected area who will be responsible for accepting the manual and getting it to work in practice. This is an opportunity for the analyst to work closely with him, to understand the practical problems of his job and to take account of existing customs and attitudes.

Key points to remember are:

- Consult at every stage (purpose, overall structure, detailed structure, first draft, final draft)
- Accept criticism
- If explanation is required, realise there is something missing from the manual
- Give public credit for other people's good ideas.

With the right attitude of interdependence and involvement comes a final bonus. When the system is introduced the section head is a powerful ally.

Sometimes the procedure manual is written by a technical writer who is not the original designer. This situation requires joint authorship of the manual. The writer needs to obtain the technical information from the systems analyst quickly, accurately and comprehensively.

One approach is to ask the systems analyst to provide the information required against standard headings along the lines of figure 24. It is up to the writer to sort the facts into the right sequence. If anything is not clear to him he must go back and ask.

Before a manual is finalised it goes through a careful checking process:

- Proof read it thoroughly; check spelling, cross-referencing, correct titles quoted, uniformity of style
- Check it with the systems analyst to make sure the facts are correct and all are included
- Check it with the project manager for content and uniformity of style
- Check it with the user section head to make sure his area is correct
- Check it with the supervising manager on the project.

One manual should be finished ahead of the rest in a system to become the standard on which the others are based.

2. **Computer Operating Procedures**
The new system when running needs to be supported by sound operating procedures. The operating procedures should be thought of in three distinct sections:

CHECKLIST FOR TECHNICAL AUTHORS

XYZ Draft Manual Procedures

Please prepare rough notes in the following format:

1. Procedure Title

2. Introduction to Procedure, how it fits in total framework

3. Object of Procedure, why it exists

4. Overview of inputs and outputs

5. Input details, for each input:

   – who uses it and when, timescale
   – how to fill it in, what not to fill in
   – any constraints, consequences
   – expected or connected outputs
   – next destination of input documents
   – filing requirements

6. Output details, for each output:

   – when and where should it arrive
   – what is shown
   – when is it produced, trigger
   – how is it requested
   – what to do with it
   – destination
   – filing requirements

7. Detailed list of all codes in use with values

Figure 24

– *Computer room procedures* (covering hardware, tape and disc libraries, error logging, security, maintenance and supplies).

– *Running supervisory programs* (start-up procedures, run control procedures and end-of-day procedures).

– *Running application programs* (system input, program execution, output distribution).

It follows that the first application on new hardware requires substantially more work than the second and subsequent applications – yet another reason for keeping it simple at the beginning.

Setting up good operating procedures relies on experience in the same way that systems analysis and technical design do. A member of the existing operations group should therefore be nominated to join the project team at an early stage. All project reports (from the survey onwards) need to be appraised by operations staff and their involvement increases progressively until it becomes full time over implementation.

During the implementation stage the job covers:

- designing and writing up the computer room and supervisory procedures
- bedding in the procedures
- training user and operations staff
- testing fully all aspects of the system (that are not application dependent).

If there is no existing operations group, expert help is needed from consultants who have themselves worked in this area.

The operating procedures manual should consist of the following sections:

Section 1 – Computer Room Procedures

Section 2 – Running Supervisory and System Programs

Section 3 – Running the programs for Application A

Section 4 – Running the programs for Application B

and so on.

The purpose of this chapter of the book is to provide a check list of the points to be covered with a brief explanation where necessary.

Computer Room Procedures

The computer room procedures cover all aspects of running the installation – except the running of programs. The headings to cover are:

1. Introduction
2. Hardware Operation
3. Tape Library Procedures
4. Disc Library Procedures
5. Error Logging and Recovery
6. Security
7. Computer Room Maintenance
8. Ordering Supplies
9. Controlling Program Versions
10. Using Backup Site

11. Control Section.

These headings are expanded into subheadings below:

1. Introduction. Give a statement of the purpose of this section of the manual, provide the hardware configuration, computer room layout diagram and reference to the manufacturer's manuals for detailed points.

2. Hardware Operation. Topics include:

- hardware power-up, with step-by-step instructions (including air conditioning) and simple checks if any step fails
- operating system start-up, with step-by-step instructions and console layout
- operating system close-down
- log-in, log-out procedure
- peripheral operating instructions
- peripheral switching (between more than one processor)
- system changeover (between more than one processor)
- logging system usage, particularly lost time
- power supply and action on failure
- air conditioning running times, operating instructions and action on failure.

3. Tape Library Procedures. Topics include:

- colour codes (showing type of use, e.g. production, testing, system, backup) and serial numbers
- documentation, covering tape history card, job record card and card showing tapes in security
- tape cleaning procedures (about every six months)
- transferring security tapes, releasing the reserve copy of the operating system master tapes.

4. Disc Library Procedures. Topics include:

- disc pack serial numbering, care in handling
- documentation, covering disc history card and location card (discs in security backup)
- disc inspection and cleaning procedures
- transferring security discs, releasing the reserve copy of the master disc.

5. Error Logging and Recovery. Topics include:

- importance of error logging and monitoring

- recording hardware errors
- recording software errors
- engineer liaison and callout procedure, signing the engineer's activity report
- preventive maintenance cycle
- error recovery procedure, automatic system recovery, manual system recovery, recovery on the same system, recovery on second processor.

6. Security. Topics include:

- computer room security, restricted access to computer room and to building (after hours)
- system security, accounts and passwords
- file security, running backup programs (two security copies), storing up to four cycles at security backup site
- fire drill.

7. Computer Room Maintenance. Topics include:

- daily schedule (tape drives, printers)
- weekly schedule (filters on processor and disc drives, cleaning VDUs and tape capstans)
- environmental conditions, controlling dust, need for tidyness
- computer room cleaning, three-monthly jobs, six-monthly environmental check.

8. Ordering Supplies. Topics include:

- ordering computer room supplies (tapes, discs, ribbons, printer loops, continuous stationery, cleaning materials, reflective markers etc)
- ordering office stationery and supplies
- stock list with minimum stock levels and re-order sizes
- re-order history (so that stock levels can be revised if necessary).

9. Controlling Program Versions. Topics include:

- change of program version form from maintenance controller
- procedure for updating programs
- holding current and previous source versions
- taking a full backup after updating program versions.

10. Using Backup Site. Topics include:

- deciding when to use backup

- getting authorisation and notifying affected people/organisations
- checklist of items needed for backup
- special operating procedures required at backup site
- supervision required
- checklist of 'disaster procedures' following total loss of computer site.

11. Control Section. A mainframe computer is likely to be organised with a control section separated from the computer room. The control section is responsible for:

- Job preparation (JCL, documentation)
- Job control (progress chasing, user liaison)
- Tape/disc library (file control)
- Scheduling
- Output handling and distribution
- Releasing program versions.

In a small installation the scheduling function (which must not be overlooked) is carried out by the supervisor.

Running Supervisory Programs

The second section of the operating manual is concerned with running programs which are not dependent on any one application. The headings to cover are:

1. Introduction
2. Start-of-Day Procedures
3. Run Control and Error Recovery
4. End-of-Day Procedures
5. System Utilities.

These headings are expanded below:

1. Introduction. Give a statement of the purpose of this section of the manual, including details of normal operating times with allowance for start of day, end of day and peak loads.

2. Start-of-Day Procedures. Topics include:

- starting up on-line systems, with dates and other parameters
- allocating file space to transaction files
- initiating transaction archiving programs (logger).

An example of the way instructions should be organised is given in figure 25.

SPECIMEN OPERATING INSTRUCTIONS

3.5 TRANSACTION ARCHIVING (TX)

LOGGER is a program which runs detached and which performs the following functions:

i. Accepts a message from a message queue and decodes it.

ii. Uses the decoded information to access a Transaction File record on disc, perform certain consistency checks, and then write out a magnetic tape image of the record.

iii. Outputs logging information to the system console.

3.5.1. Operating Instructions

LOGGER can only be run via the SUPER program which will either:

(a) Pass messages to LOGGER if it is already running, or
(b) Chain to LOGGER to initiate.

The program is initiated by SUPER by typing TX to the PROGRAM? question. This causes the question:

FUNCTION?

to be displayed, to which any of the following responses may be typed:

H – Prints a help message

I – Initiate Logger

T – Terminate Logger

Q – Enter an item onto the queue

S – Suspend Logger etc

Note the way the function is described with an overview before going into detail

Figure 25

3. Run Control and Error Recovery. Topics include:

– input control, correct authorisation of jobs, special requests

– output control, checking print quality, output distribution (including output tapes/discs/security tapes)

– need for security to be maintained

- need for errors to be logged systematically
- action on hardware/software failure
- action on VDU/line failure
- running transaction file recovery programs, re-archiving transaction files.

4. End-of-Day Procedures. Topics include:

- running batch programs when on-line input is complete
- distributing batch control sheets
- despatching output tapes
- running security backup programs.

5. SystemUtilities. Describe here system utilities, including those run occasionally, e.g. running error logging programs, system usage statistics, disc copying utilities, simple diagnostic programs.

Running Application Programs

Procedures for running application programs fall into two categories – operating VDUs and operating the computer. In general, instruction on operating VDUs will be in the user's procedure manuals. Instructions on operating the computer are described in this section of the manual.

Operating instructions require the following points to be covered:

1. System Input Procedures. Topics include:

- the way the work reaches the computer
- input media and control documents for every type of media
- job requests and authorised signatories
- names of user representatives to contact in case of difficulty
- volumes, cut-off timings, input control.

2. Key punching instructions. Topics include:

- specimen documents
- key punch instructions
- batching rules
- controls.

3. System flowchart. Topics include:

- overall flowchart
- restart points

- interfaces with other systems
- frequency of runs
- elapsed times
- names of duty programmers.

4. Program Operating Instructions. Topics include:

- brief description with program function and operating instructions
- block diagram with resources needed (memory, peripherals, files, stationery)
- JCL listing
- recovery procedures
- programmer.

5. System Utilities. Describe here system utilities, including those run occasionally,

- output media, numbers of copies
- guillotine and decollator requirements
- volumes and deadlines
- distribution and transport
- people to contact if output is late.

6. Tape and Disc Security. Topics include:

- master files and generations
- saving transaction files
- security storage
- tape and disc usage.

7. Current List of Contacts. Bring together an up-to-date list of contacts with their telephone numbers, e.g. user input, duty programmers, system supervisor, output contacts.

Project Management Considerations

There is a lot of ground to be covered in the operating procedure manuals so the project manager must get the work started as early as possible. The draft manual should be prepared at least a month before the parallel run date. The time is needed to give operations management the time to review the contents; also so that the procedures can be checked out during system testing. Quality control of this nature is required even if operations staff are involved in its preparation (as they must be).

Here are some of the points for the project manager to consider:

- A new computer needs to be run correctly from day one – even if day one is only program testing. Correct operating procedures are needed with file backup, security

copies taken to remote locations, error monitoring etc. So make sure that these procedures are ready in time.

- System and program changes are likely to cause procedure changes as well. Control procedure manuals as well as programs.

- Make sure that punching documents are checked out by the punch room supervisor before they are shown to users.

- Make sure that the manual is *used* during system testing and that the examples will work (i.e. no errors in them).

- Carry out benchmark tests as soon as possible and identify areas where the programs run slowly or do not flow continuously. Decide what to do about any problems.

- Remember to train everybody in operations – there may be several shifts to train as well as the schedulers. Get early experience communicated between them (use an incident log for the first few weeks' running).

- Hardware gets in grooves, so anticipate head alignment problems, problems with new tapes and discs, generally greater downtime with a new system (high correlation with little justification).

Generally treat operations as you treat your user; they need to be kept on your side. Operations also must *accept* the system before it goes live.

The time to prepare the manuals will depend to a large extent on the work that exists already. Allow a month per section for the first time that section has been written, two weeks or so for repeats which are similar.

G Implementation

The implementation stage of the project covers the activities required to introduce the new system and to shut down the old. In many cases it means changing every detail of the way people work. Even on a medium sized project, it is a difficult problem with heavy penalties (in social and business terms) for getting it wrong. The way to a smooth implementation is to break the work into discrete and manageable activities (once again). By starting early on these activities the problem can be caught unawares (so to speak) and before the team get frightened by the enormity of it all.

All the various project threads are brought together during the implementation stage and the project manager's main job is *making things happen.* All the loose ends have to be tied up; equipment delivery and commissioning, acceptance testing and user training, external supplies and external services.

An important concept is the Holding Point. It is the moment when all the preparatory work has been completed and the decision can be taken to go live. The project team can decide on the holding date; only the final user can decide on the implementation date.

While implementation is the final project stage, it is important to begin the activities as early as possible. Equipment procurement can start as soon as the decision to go ahead has been made (following the Evaluation Stage). Similarly, other activities can and should be brought forward as early as is sensible. Every job postponed adds disproportionately to the problems at the end when time has often run out.

Project Management Considerations

During the implementation stage the project manager begins to move the system closer and closer to the final user. It is their system which they are going to run and which requires their total commitment.

The user involvement is at several levels: at a senior level to review the internal organisation structure in the light of the new systems, at a more junior management level to carry out the system testing, write procedures and train staff. At least one user must be assigned full time on these tasks.

Some line managers are reluctant to get involved in detailed procedures as they feel that such work should be delegated. Of course, some can be delegated, but sound procedures are a key result area for line managers. A little time taken getting them right at the beginning brings accumulative benefits – fewer daily problems, better level of service, lower manpower levels and less senior management time lost. The project manager must involve his own senior management in the education process if he feels that his user executive is not persuaded of the importance of this function.

Before going into detailed planning the next step is to develop and agree the implementation strategy. To go live on all aspects of the project at the same time is likely to cause too much strain on the new system and its users. The project manager should therefore seek ways of implementing the project in easy stages.

Progressive implementation can be achieved by starting with one section, with one ledger or with one geographic location. Allow time for the first area to settle in before plunging into the second and subsequent areas. Do not be trapped into saying 'We've nearly got it right so we can move on'; make sure the foundation is firm before building further.

Now that the user involvement is clear and the implementation strategy agreed, the project manager can get down to preparing the detailed implementation plan, identifying the activities and all the tasks to be done as early as possible. Taking account of the resources available, he prepares an action list for all the activities (see Figure 26).

This action list can be prepared following a 'brainstorming' meeting held with key members of the project team, users and computer operations. As part of the planning, the project manager works out the *holding point,* the date on which a decision to proceed to implementation can be made.

Equipment and External Factors

Equipment must be specified, ordered, delivered and fully installed before parallel running can commence. Similarly all external agents need to complete all their agreed contributions before the start of parallel running. As external agents are difficult to control, there need to be generous time allowances built in.

Equipment to be considered includes:

- hardware
- terminals, lines, modems, switches
- cabinets, racks, guillotining and decollating accessories
- air conditioning
- computer room environment and power supply

A SPECIMEN ACTION LIST (FOR THE EQUIPMENT AND EXTERNAL FACTORS ACTIVITY) PREPARED FOLLOWING AN IMPLEMENTATION PLANNING MEETING

Equipment and External Factors

	Action	Task Completion Date	Responsibility
1.	Consider supply of operator aids e.g. racks etc	1.7.77	PJC
2.	Arrange binders for daily printout	1.7.77	PJC
3.	Arrange for storage of printout	1.7.77	PJC
4.	Arrange set of tapes for HSUTM with coloured seals (black)	1.7.77.	PJC
5.	Arrange storage facilities for HSUTM tapes	1.7.77	PJC
6.	Arrange security copies of HSUTM tapes and storage of same	1.7.77	PJC
7.	Organise transport of tape to L-ACS on regular basis (Kendals ?)	5.8.77	PJC
8.	Make PDP Supervisor aware of increased usage of single-part stationery	1.7.77	PJC
9.	Ensure two Lynwood terminals are delivered and installed – liaise with DEC and Ken Burt	25.7.77.	PJC
10.	Ensure Multiplexor delivered and installed or make alternative back-up arrangements	25.7.77	PJC
11.	Ensure long-line drivers are installed or make alternative arrangements	5.8.77	PJC
12.	Arrange for L-ACS punching to cease but retain as back-up	12.8.77	PJC
13.	Liaise with L-ACS Data Prep. Dept. re changes to catalogued procedures	27.7.77.	PJC
14.	Agree timetable of events on PDP re EOD and EOP with PDP System Supervisor	29.7.77	PJC

Figure 26

- terminal operator environment and cabling
- stationery, forms, tapes, discs and cards

- standby hardware.

External agencies include:

- equipment suppliers and Post Office
- electrical/building contractors
- consultants/contract staff
- Customs
- proprietary software
- COM bureau
- punching bureaux
- computer bureaux
- printers (stationery and forms design)
- taxis
- security site
- other service departments within the organisation
- trade organisations and services.

The project manager must progress chase all equipment and supplies, take nothing for granted and plan on the basis that it will be late, and will not work for some time when it does arrive.

He must also make sure that all external agencies know what they are expected to do, by when, and involve them in all full system tests.

Interfaces with Related Systems

Few systems are installed without affecting some other aspects of the business. The purpose of this activity is formally to think through interfaces with related systems and to ensure that the planning procedures and testing are at least as thorough as for the main system.

Here are the points to consider:

- Look for changes to timing and content of:
  * input procedures
  * input control
  * media links to other systems
  * output procedures
  * user department responsibilities
- Develop comprehensive test data

- Make sure the users of both systems realise the implications of the interface, especially where these affect system status or timings

- Include standby systems and standby system testing.

Remember that changes to a file content in a related system may mean quite extensive changes to all programs using that file and this consequently increases your testing load.

File Conversion

The purpose of file conversion is to convert existing files, manual and/or computer, to those of the new system's format.

The size of file conversion often warrants treating the activity as a sub-project. Therefore regard it as requiring as much control as an average project.

Here are the points to consider:

- Prepare detailed conversion procedures, times, workflow, for user resources and management.

- Begin conversion early – procedures, programs and manuals for conversion should be completed well before the start of conversion.

- Freeze the files to be converted and convert them to reflect the status as at the day of freezing.

- Control all amendments to files after freeze date so that the converted files can be updated before volume testing and parallel running with these alterations.

- Control all numeric fields and carry out record counts before and after conversion, and at every intermediate step – if the number of existing records is not known or cannot be determined accurately, develop other reconciliation procedures.

- Use the conversion programs when creating files for volume testing and parallel running – live conversion is not allowed to be the only run of the conversion suite.

- All files must be converted before parallel running can commence.

Before finalising the plan, the project manager needs to check the availability and accuracy of source data for himself. He may need to allow additional time for a file clean-up before conversion starts. Generally a file clean-up should not take place during the live conversion (there is too much else to do).

Allow about one month to convert each file (manual conversion) and convert only one at a time. This means that if there are two files to be converted in this way you will have to start at least two months before the holding date – so examine them carefully for a clean-up six months before the holding date.

It is sensible to let users maintain converted files on a daily basis for some time before final implementation to get used to operating the new procedures.

Holding Point

The Holding Point is an event rather than an activity – but rather an important event. Its purpose is to make a formal assessment of each implementation activity and to decide upon a realistic live date that can be agreed with the user.

The following conditions must be satisfied in order to decide upon the start of parallel running and the publication of a live date:

- System logic testing is complete (run just one more test pass using the object libraries to make sure) – comprehensive test packs exist for each procedure.
- Acceptance testing is complete and signed off.
- All manual procedures are complete.
- Operating procedures are complete and accepted.
- All equipment is installed and working; external agencies have been tested.
- Interfaces with related systems have been tested.
- All files are converted and up to date or machine conversion has been fully tested.
- User organisational changes are agreed and training material is ready.
- Security copies of all files, programs and documents have been stored.
- Run-time estimates are satisfactory – or, if unsatisfactory, user and operations agreement to continue has been obtained.
- All essential program amendments are complete and tested – these program versions will now be frozen except for errors found in parallel running. All other amendments will be handled as maintenance post-implementation.

The holding point is the last occasion when the project can be held up so ensure that everything is checked and ready before moving into parallel running.

User Organisation and Training

To take advantage of the new system some organisational changes are likely to be necessary within user areas – as well as changes to job descriptions and workflow. The purpose of this activity is to ensure that the organisational changes are agreed, training needs identified, training prepared and eventually given.

The training sessions should be given by the user managers attached to the team, after detailed instruction from the project team.

Here are the jobs to be done:

- Make sure the overall workflow in the new system is documented.
- Concentrate on areas which have been changed and on totally new functions (data control for example).
- Separate supervisor training (especially when operating minicomputers) from the applications training. Only train staff in any detail on the parts of the system they are going to operate.
- Train as late as practicable, before live date.
- On large systems, carry out a live running simulation as part of the training process. Even on small systems get the trainees to make dummy entries (and process them efficiently).

- During dress rehearsals and simulations make sure all logistics are tied up with the users – delivery times agreed, desks laid out and labelled.

- Set up a procedure for maintaining procedure manuals once the system is 'live'.

Users may 'dodge' training due to pressure of current work, so lay on more sessions as they become available.

Parallel Running and Going Live

Parallel running and 'going live' has one purpose – to ensure that the transition from the existing system to the developed system is achieved with the minimum amount of disruption to the user.

Parallel running is the term used to describe two different, but related, activities:

- Running the new system on old (but genuine) data for several cycles, checking the output of every cycle against the original output.

- Running the new system in parallel with the old in a live situation.

The former is essential. The latter is desirable, but is not always possible. On a large system with a daily cycle, doing everything twice but differently may *reduce* the chances of success rather than improve them.

Here are the points to consider:

- Phase the implementation whenever possible. Introduce one main file at a time or one section at a time. Give the first section time to bed in before going on to the next.

- After the testing stage is complete, the full system test should be repeated to ensure that all amendments are satisfactorily included.

- Plan early which files need to be frozen for parallel running, take security copies, balance input and files early.

- Include all related systems in the parallel running. Involve all third parties. Obtain additional clerical support to help over the parallel running and conversion stages.

- Make extra machine time available at critical times by seeing if other users can reduce their requirements.

- If it can go wrong, it will; so test everything. Nothing must be done for the first time when going live.

- On going live, discover the source of all conversion errors, however small. Re-run conversion now if you are in any doubt about the effect of the error discovered.

- Organise:

  * *input* batching, timing, liaison, balancing, error correction and responsibilities

  * *output* sorting, batching, timing, checking, re-running procedures and responsibilities

- Keep input under control. It is very difficult to run two systems in parallel (mainly because they each have their own rejections and generated entries) so plan it, write procedures and test carefully.

- Run the new system first and distribute the output to the user while running the old system.

Aim to go live as at a Thursday night – it gives time to recover over the weekend in the event of problems. Avoid going live at month end, year end or any time-critical date. Avoid holiday dates, work load peaks and Mondays. Arrange for all affected project and user staff to be available for the weekend if called upon.

Old System Shutdown

A formal decision is required to shut down the old system. It does not just stop, but needs to be brought to an orderly halt.

Here are the points to consider:

- Determine how long the old system will run with the new in a 'live parallel' period. When should it finish?
- Consult with the auditors and establish.
  * which reports need to be produced on the last day of running the old system (e.g. copies of files in old and new state)
  * which files need to be retained and for how long from the old system
  * how long the programs should be retained on the source library
  * what archives, photocopies, microfiche need to be created from the old system.
- Ensure that all procedure manuals and input forms relating to the old system are withdrawn on the agreed date for closing down the old system.
- Determine what action will be taken if the *new system* fails during the 'live parallel' run.
- Have a contingency plan available if the *old system* fails during the 'live parallel' run.
- Establish with the users which will be the master system during 'live parallel' running.

Do not try and run incompatible systems in parallel – shut the old down and open the new simultaneously.

Post-implementation Support

The purpose of post-implementation support is to ensure that the live system is carefully nursed over the first days and weeks of its running so that problems (if any) can be quickly rectified.

- Since the system will not yet be in maintenance, ensure that all queries are noted on the correct documents. System errors will have priority at this time. Make sure that each error correction is fully tested, using the standard test pack for the system.
- Changes to the system will fall into two categories:
  * system or program errors (E) – the system cannot run
  * system amendments (A) – the system can run

SYSTEM AMENDMENT REQUEST		Project No
System	Procedure	Job No
Date of Request	Originating Manager	Program Version

AMENDMENT/ERROR DETAILS

Attach specimen output etc. What is main purpose? Who else is affected? Explain desirability. Quantify benefits where possible.

Date required:

Programs/Documents affected, details of work to be done, test plan

AUTHORISATION

The above work should be charged to

Director

Date

Priority High / Average

Costs	Estimated	Actual
Analyst Programmer M/c time External		
Total		
Scheduled completion date		

Test Output Accepted Manager: Date:	Date user manual changes sent	Date program released

Ensure that both types are clearly identified and agreed with the user representative.

- Use standard Amendment Request Forms from the start to record any changes.
- Phase out direct project support after a week or so of live running – only then will the user accept real responsibility for running his system.
- Arrange for a project team member to be available in the user area to answer queries during the first few weeks of live running.
- The project team should be responsible for the system for not longer than three months. The system should then be handed over to maintenance.

The system is almost certain to have some teething troubles as users start to appreciate the new facilities and perhaps get used to codes. Consider having a member of the project team working with user data control, checking coding and resolving difficulties (after they have got stuck).

IV. SYSTEM MAINTENANCE

Once the new system is live its importance changes. Testing and development are over and the business relies on the new system for day-to-day running. Any program changes can have disastrous side effects – which need to be tightly controlled. Programmers therefore must not be allowed easy access to their own production programs – a concept which is difficult to get across ('It's only a one-line change' is on the list of famous last words).

A realistic way of tackling the problem is to appoint a Maintenance Controller. It is usually a function rather than a full time job but a function which brings discipline into a difficult area. The Maintenance Controller is responsible for:

- accepting new systems from development into maintenance
- maintaining a register of maintenance jobs, and preparing monthly summaries
- releasing new and amended programs into production
- maintaining security copies of documentation
- maintaining the programmer overnight call-out roster
- maintaining an up-to-date list of common subroutines.

While the maintenance controller is likely to be familiar with one or two systems, he is unlikely to be familiar with them all. So for every production system, a System Supervisor should be appointed. The system supervisor is responsible for:

- analysing and agreeing amendment requests with users, estimating their cost and duration
- scheduling and supervising the work of maintenance programmers (errors get highest priority, amendments are carried out on a first come, first served basis – subject to any user set priorities)
- getting amendments tested and accepted by users, then handing them over to the maintenance controller for release
- acting on system failures (advising users, management services management, getting the problem rectified and the system restored, retesting the change).

Maintenance Control

Explaining the elements of the maintenance controller's job will show how he and the system supervisor interact.

Accepting New Systems

The acceptance procedure is designed to ensure that the project team complete system development tidily. The maintenance controller ensures that the documentation is complete. The documentation includes:

- user manual
- operations manual

- program documentation
- security program listings
- program test procedures and test files.

Maintenance Register

An amendment request, when costed and authorised, is given a serial number by the maintenance controller. The amendment is entered into a day book with space for noting when it is put into work, and when it is completed and accepted by the user.

Once a month the maintenance controller prepares a register showing the jobs completed in the month, the work in progress and the jobs not yet started. The maintenance register provides a useful summary of the position for both the user executive responsible and the head of the management services department.

The user executive is able to see the work that is going on and can step in if he feels it is not justified. The head of management services can see that the work is being progressed and that no programmer has too many jobs half completed or waiting to be signed off.

Releasing New and Amended Programs

When testing is completed the programmer should:

- complete the System Amendment Request with details of the work done
- document the changes
- provide details of changes, or a new source
- provide a new program listing (if not part of the library update)
- get user agreement (through the system supervisor) that the amendment has been tested to his satisfaction
- include full program documentation (if it is a new program).

The maintenance controller then updates the source and object libraries and advises the user that the new program version is ready. A release date is agreed with the user and operations (normally on Thursday night or Friday so that any unexpected problems can be resolved over the week end) and the maintenance controller arranges for the new version to be released into production.

Security Documentation

A copy of the program documentation, source listings and test files is held near the computer, a second copy is held by the maintenance controller in security. The copy by the machine is needed if programmers are required in an emergency. The security copy must be held in a different location – conveniently close to the analysts and programmers if they are on a separate site.

The maintenance controller's copy contains the latest copy of the source listing with a complete history of library changes. For mainframe systems, three generations of the source library tapes are held – on minicomputers there is a daily disc dump but in addition programs should be dumped following any change.

Call-out Roster

Production systems which are run overnight may need overnight maintenance cover.

Priority 1 on the call-out roster means that the duty shift leader will telephone immediately the system fails and cannot be restarted. He telephones the programmer who is top of the duty list for that week.

Priority 2 means that the shift leader will not make the call until 7.00 a.m. The programmer is expected to start work on the error as soon as he can get in.

If the shift leader cannot contact the first programmer on the list he calls the next – and so on. He calls the system supervisor only if he cannot raise any of the programmers.

The maintenance controller makes sure that operations have an up-to-date list of duty programmers with their telephone numbers and ensures that extra cover is available at holiday times.

Common Subroutines and Copy Libraries

In developing a new system a number of utilities and standard routines are needed. When the system is completed the strategic programmer hands over to the maintenance controller full documentation on these subroutines – so that they can be used by subsequent projects.

The maintenance controller maintains a register of these subroutines and circulates it to project staff periodically – and when it is changed substantially.

Testing Procedures

Production failures usually have to be corrected quickly so that production running can be resumed. Testing the 'fix' can only be rudimentary and needs to be followed up by an adequate suite test. If the programmer was called out in the middle of the night he has probably gone home to bed by the time the full test can be run. Before going home he must make a point of writing down everything he has done including tests performed and further action required.

The system supervisor then needs to decide how much testing should be done. His starting point is the standard test procedure for the system (see Figure 28).

Generally a full suite test should be carried out if:

- more than one program is involved
- any update program is involved
- any data may have been corrupted.

Maintenance Programming

Maintenance programming is an integral part of the job of programming. In its own way it is challenging and interesting (more customer contact than many systems analysts have, a greater need for business awareness than writing new programs needs) but too much of it can get dispiriting.

Ideally new programmers should be trained on new programs before they are asked to maintain existing suites. Even if they enjoy the maintenance challenge, some variety is required so that they see that they are making progress in their profession.

SPECIMEN TESTING PROCEDURE

TESTING PROCEDURES (BAS)

PROGRAM	TESTS (if no new data)	TEST (with new data)	REPORTS REQUIRED
BACC1100	Process whole test data pack.		Test – Vet List
	Compare Vet Lists	Add new data to pack. Process whole test pack: a) Using old version b) Using new version Compare Vet List Compare BANVET files	Day 6 – Vet List
BACC1200	Process BANVET test file. Compare output with previous test results to ensure amendment is properly reflected.		Summary of Reversals
	Compare BANKTRAN files	Add new data to test pack. Process BACC1100 and BACC1200 a) Using old version BACC1200 b) Using new version BACC1200 Compare output from each.	Summary of Reversals
BACC1300	Process BANKREFT test file.		
	Compare output files	Add new data to test pack. Process BACC1100 a) Using old version BACC1300 b) Using new version BACC1300 File compare (from (a) and (b)) – BANKREFT) – BANKTRAN) – BANKCURR) files – BANKREFD)	

Figure 28

Technical Review

The introduction of computer systems in business is far from being an exact science. So it is not surprising that there are few projects that cannot be improved (with the benefit of hindsight). A post-implementation review is therefore both necessary and likely to yield substantial benefits. Improvements can be expected in running time, ease of operation, security, error recovery, volume of output and user satisfaction.

There are two traps to avoid, however. The first is the temptation to carry out the review too soon. The new system needs time to bed in so that the *permanent* deficiencies can be identified. Carry out the review too early and implementation problems can distort the true picture. Generally the review should be carried out nine to eighteen months after live date.

The second trap is the danger that the review is regarded as some sort of witch-hunt. Happily enough factors will have changed to allow everybody to be objective (and not defensive) provided that the review team are given the right lead.

Objectives for the review might then be:

- 'given the current situation, what aspects of the design would we have done differently?'
- 'how far is it desirable to make changes to rectify design errors?'

So far the technical review has been described as a single study. In practice it falls into three distinct areas – and may have three different review teams. The three areas are:

- Operational Running
- User Satisfaction
- Technical Design.

The three reviews should probably take place in the order stated. The review of technical design should follow the review of user satisfaction so that any improvements recommended take account of the user's current awareness. Similarly the review of user satisfaction should follow the review (and improvement) of operational running – operating problems may again give a false picture. So, although there will always be overlap between the three areas, there is purpose behind the sequence recommended.

Operational Running

The review of operational running will only make sense if it is carried out by someone experienced in computer operations. Headings for the review are likely to be:

1. Ease of operation; a smooth and natural flow of the work.
2. Improving running time; not so much program efficiency as reducing the elapsed time.
3. Flexibility in case of problems; steps to minimise the impact of hardware and software malfunction (additional restart points, copy input tape etc).
4. Control and security.
5. Operator training and procedures manuals; how clear, how comprehensive, how up to date?
6. Resolving special problems which are system dependent (e.g. data links, COM, interfaces with other systems).

Against every heading the team should report in a consistent way on:

- current situation
- problems identified
- solutions proposed
- likely costs
- implications (if any).

Figure 29 shows the way such a report might be written.

SPECIMEN FROM A REVIEW OF OPERATIONAL RUNNING

4.5 File Security

Two aspects of the file security were studied:

- The security of the tape/disc file and back-up procedures.
- The security of the information held on the files.

4.5.1 Current Situation

The significant files (not work files) in the Banking system are organised on a 5 day cycle of which two cycles are stored in Beech Street. Additional protection is provided by the normal hardware/software of the computer to stop overwriting of files.

The security of the information held on the files is provided by isolating the 3rd floor of Gordon House with locked doors and, within that, a lockable tape library, although COM files are released to Lowndes-Ajax.

4.5.2 Problems Identified

The problems identified below, whilst applicable to Banking, are also applicable to any system operating a day number cycle of tapes. Problems identified were:

- In the event of a long weekend or public holiday (say Easter), the file back-up is degraded because certain generations are skipped; eg the last working day before Easter is day 4 and the first after is day 2. If during that Tuesday update to a file re-creation is necessary, only one previous generation (day 3) is available.

Figure 29

User Satisfaction

The review of user satisfaction should be carried out by two people; one team member should be knowledgeable about the business systems, the other about the technical design of the computer system. The two skills are needed so that essential, desirable and practical improvements can be identified separately.

Review headings are likely to be:

1. Input cycle; design of input documents, use of VDUs, delays, errors and inefficiency.
2. Output cycle; reliability of meeting deadlines, quality of output.
3. Volume of paper; number of copies and their use, alternative output methods.
4. Presentation of reports; convenience in use, suitability for purpose, use of exception reporting, frequency.
5. Control and security; meeting the standards described in chapter 9.
6. Organisation and training; revised organisation structure, procedures manuals, staff training.
7. Achievement of benefits; measured against original system objectives.
8. Special situations which are system dependent.

Technical Design

The technical design review should be carried out by two people. One should be the system supervisor (with a detailed knowledge of the files and programs); the other should be a senior technical person with a good working knowledge of the business.

Review headings should follow the following lines:

1. Program and system maintenance; review of experience to date and a look to the future.
2. Program and system documentation; review of current situation.
3. Design review — input phase; against the standards in chapter 7.
4. Design review — update phase; also against standards.
5. Design review — reporting phase; also against standards.
6. File structures; looking to the future.
7. Special situations which are system dependent; e.g. interfaces with linked systems, adding new branches, converting system to new computer.

The review team must avoid the danger of seeing everything that is bad about the system — and recommending a complete re-write. The re-writing route is certain to be time consuming, costly and without much guarantee that it will be better. Generally it is possible to evolve towards a more satisfactory system in stages with less disruption and less risk. There is an analogy here with local authorities in the UK in the mid-70s. They discovered the hard way that knocking down old houses and building new ones in their place was not the answer to housing problems; it was simpler, faster and cheaper to improve what already existed (the quality was often better too).

Another dilemma for the review team is how far old systems should be made to conform to current standards. As a guide, changes should only be made if there is a clear benefit. Documentation can be improved progressively whenever a program comes up for maintenance; on the other hand problems of control and security should be tackled with urgency.

Project Management Considerations

The technical reviews are an opportunity to consolidate the past and to learn valuable lessons for the future. They will be done best by the most experienced and competent staff in the installation.

As a guide the time required to carry out a comprehensive review of a major system is:

Operational Running	15 days
User Satisfaction	30 days
Technical Design	30 days

Quite small amounts of time compared with the benefits they tend to generate.

V. POSTSCRIPT

In the wealth of detail associated with a computer project it is all too easy to lose sight of essentials. Here are the essential points for the project manager (and his masters) to concentrate on.

1. Get the project off to a good start with clear terms of reference and the right leadership.

2. Before every stage identify the activities carefully.

3. Plan the work and get the right resources to carry it out. Do not be afraid of exercising positive control.

4. Take extra care at these review points:

   — functional description

   — design review

   — implementation plan

   — post-implementation review

Be prepared to take a little longer to get it right. Remember the project manager who said, 'Why is it that we never have time to get it right — so we get it wrong three times?'.

Abstract

Report Series No. 8

Project Management

by Hamish Donaldson

June 1978

Many systems projects fall short of the expectations of both end users and management services managers.

While the manifestations of failure or partial success are usually technical ones in many cases the underlying causes are concerned with the management of the project.

This report sets out to explain a set of rules for managing projects which are known to work well in practice. They are applied to both large and small projects, using a variety of equipment.

These rules are largely concerned with the structuring of the project – both to minimise management 'interference' and to minimise the penalty to be paid for failure in any one area.

The content of the report is detailed, and embraces a number of technical issues which are encountered in system development projects, since the effective management of computer projects does require an understanding of these matters.

The Butler Cox Foundation

The Butler Cox Foundation is a research group which examines major developments in its field — computers, telecommunications, and office automation — on behalf of subscribing members. It provides a set of 'eyes and ears' on the world for the systems departments of some of Europe's largest concerns.

The Foundation collects its information in Europe and the US, where it has offices through its associated company. It transmits its findings to members in three main ways.

- as regular *written reports,* giving detailed findings and substantiating evidence.

- through *management conferences,* stressing the policy implications of the subjects studied for management services directors and their senior colleagues.

- through *professional and technical seminars,* where the members' own specialist managers and technicians can meet with the Foundation research teams to review their findings in depth.

The Foundation is controlled by a Management Board upon which the members are represented. Its responsibilities include the selection of topics for research, and approval of the Foundation's annual report and accounts, showing how the subscribed research funds have been employed.