Project Estimating

BUTLER COX
P,E,P

PEP Paper 16, October 1990

BUTLER COX

P,E,P

Project Estimating

PEP Paper 16, October 1990
by Nigel Saker

Nigel Saker

Nigel Saker is a senior consultant with Butler Cox in London, where he specialises in project management and the specification of user requirements. He has 20 years of experience in software design and management.

During his time with Butler Cox, he has carried out several PEP assessments, and he researched and wrote PEP Paper 13, *Software Testing*. He also conducted research for the Butler Cox Foundation Position Paper, *Legal Protection for Computer Systems*. He has been involved in a large consulting assignment for a major US bank, developing specifications for the market-data delivery system for its new dealing room.

Prior to joining Butler Cox, Nigel Saker was a project manager with Aregon International, responsible for the design and installation of dealing-room systems in five major banks in London, New York, and Oslo. Earlier, he spent six years with Logica, advising on the selection of equipment, designing user interfaces for highly interactive systems, and managing turnkey systems development and installation. His early career was with the Meteorological Office.

Nigel Saker has an MA in mathematics from Cambridge University and an MSc in fluid dynamics from the University of Sussex.

Published by Butler Cox plc
Butler Cox House
12 Bloomsbury Square
London WC1A 2LL
England

Printed in Great Britain by Flexiprint Ltd., Lancing, Sussex.

BUTLER COX

P,E,P

Project Estimating

PEP Paper 16, October 1990
by Nigel Saker

Contents

Chapter 1

A more systematic approach is required

Both the systems department and the customer need estimates

The main reason for developing any computer system is that it should benefit the business. The benefit could derive from increasing efficiency, from gaining an edge over competitors, or from complying with legislation. Whatever the objective, the customer needs estimates of the cost and timescale for developing a proposed system in order to be able to assess its net benefit. The systems department also needs these estimates in order to decide whether it has adequate resources to meet its deadlines. Constraints on resources may limit the speed at which the system can be developed, and reduce the benefits that might be expected to derive from it.

The customer authorises the project to proceed on the basis of the estimates, and expects that the system will be delivered more or less within the agreed timescale and budget. If, during the course of the project, it becomes apparent that the budget will be exceeded by a significant amount, the organisation could be faced with the difficult choice either of continuing to fund the development, with the risk that it may be throwing good money after bad, or of cancelling the project and writing off the whole investment.

Even the best-managed projects will sometimes overrun

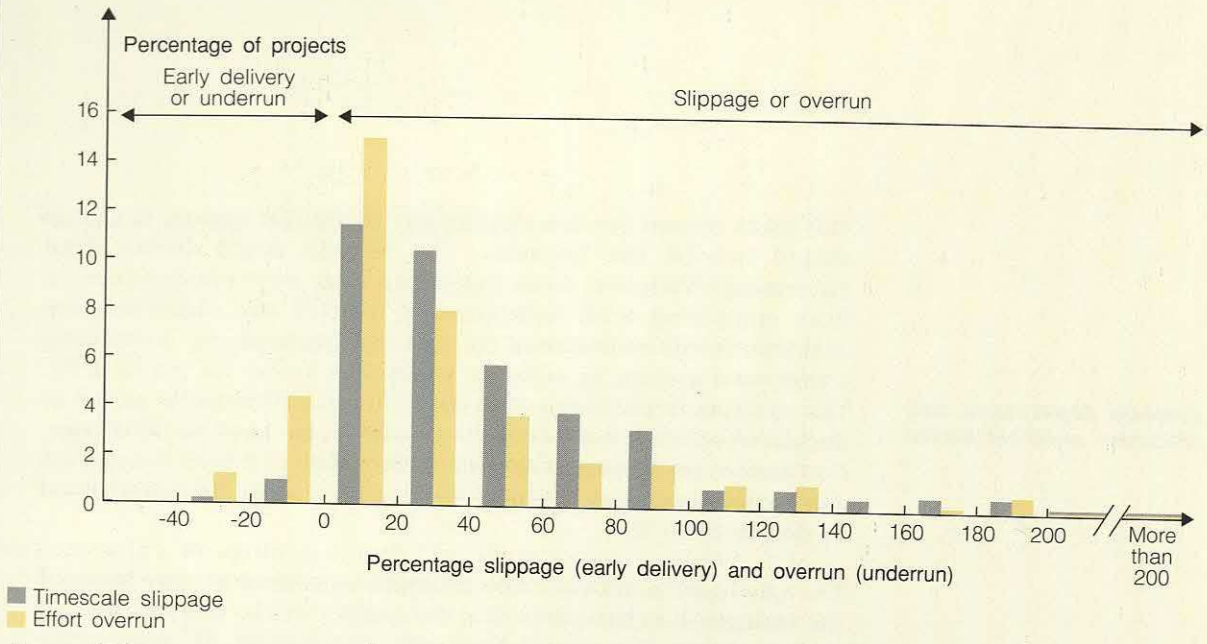
Since most systems development projects involve some elements of technical or commercial risk, and since the estimating process is, itself, fraught with difficulties, budget overruns will occur from time to time, even on the best-managed projects. In this paper, we present a practical approach to estimating that is within the capability and budgets of most systems departments to implement, and that will ensure that estimates are as accurate as possible.

SLIPPAGES AND OVERRUNS ARE A COMMON FEATURE OF DEVELOPMENT PROJECTS

Our analysis of data submitted for PEP assessments reveals that about 40 per cent of projects exceed their estimated cost or timescale for the main-build stage. Typically, timescale slippages and effort overruns range from 30 to 40 per cent, but there are also some very large variations, as shown overleaf in Figure 1.1. It should be noted that this data represents slippages and overruns based on estimates made at the start of the main-build stage. If data had been recorded for estimates made at the start of projects, it is likely that slippages and overruns would have been significantly greater.

If those involved in producing estimates learned from experience, one would expect as many overestimates as underestimates. In fact, only about 2 per cent of projects in the PEP database were delivered in less than the estimated time, and less than 10 per cent were delivered with less than the estimated effort. This

Figure 1.1 Slippages and overruns are a common feature of development projects



(Source: PEP database)

indicates either that organisations produce estimates that are consistently too optimistic, or that projects are not well controlled, or both. Whatever the reason, it is clear that the net benefits gained from nearly half the projects undertaken by PEP members are less than expected, and in some cases, the development might never have been started if its costs and timescales had been known at the outset.

THE ESTIMATING PROCESS IS FRAUGHT WITH DIFFICULTIES

The experience of the vast majority of PEP members indicates that it is difficult to make accurate estimates of the time and effort involved in a systems development project. There are three main reasons for this — the nature of estimating is, itself, often misunderstood, estimating is inherently imprecise, and there are no estimating techniques that will be universally applicable.

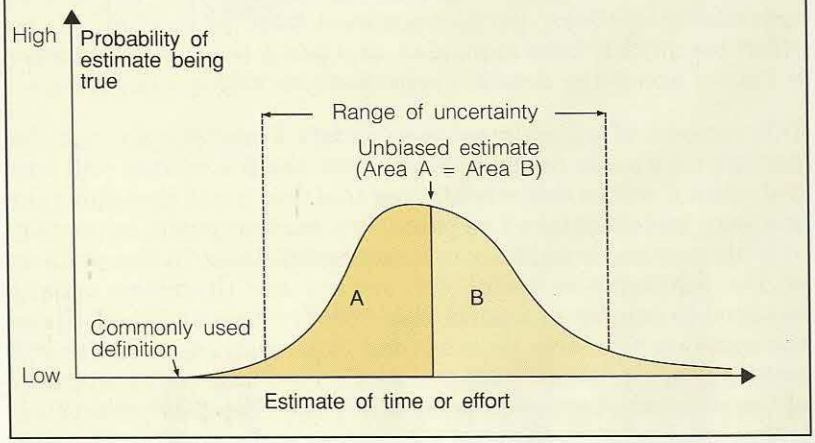
THE NATURE OF ESTIMATES IS OFTEN MISUNDERSTOOD

The data on slippages and overruns illustrated in Figure 1.1 suggests that many PEP members perceive an estimate as being 'the most optimistic prediction that has a non-zero probability of coming true'. According to Tom DeMarco, a well known writer on the subject of estimating, this is a common, but misguided, definition of an estimate. An alternative view, held by many developers, is that an estimate is 'the highest figure likely to be accepted'; this approach to estimating is likely to be adopted where funding is sought for a research type of project.

A more useful concept of an estimate is that it consists of a predicted value, and a range of uncertainty about that predicted value, as shown in Figure 1.2. An unbiased estimate is one that is equally likely to be exceeded or undershot by the actual value.

An unbiased estimate is equally likely to be exceeded or undershot

Figure 1.2 An estimate consists of a predicted value and a range of uncertainty about that predicted value



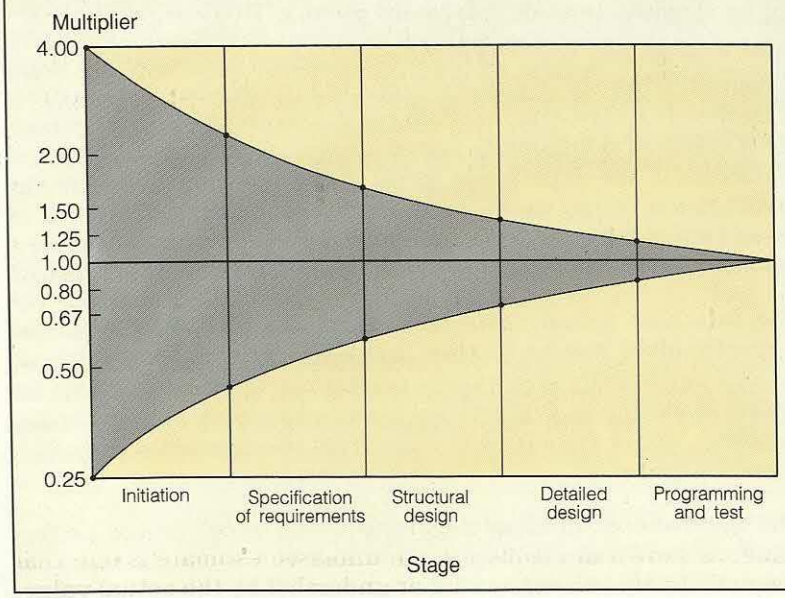
The range of uncertainty will define a range of values within which the eventual outcome might be expected to lie, say, 90 per cent of the time.

The range of uncertainty of an estimate diminishes as the development proceeds

The range of uncertainty of an unbiased estimate varies according to the stage of the development project at which the estimate is made, as shown in Figure 1.3. At the very early stages of development, there is a wide range of uncertainty, because little is known about the system other than a general statement of the

Figure 1.3 The range of uncertainty varies according to the stage of the project at which the estimate is made

Unbiased estimates made at each stage of the project would lie on the horizontal line. The expected range of uncertainty is defined by multiplying the unbiased estimate by the values indicated on the Y-axis. Thus, at the beginning of the initiation stage, the range is from a quarter of the unbiased estimate to four times this figure.



requirements. At this point, an estimator may predict, for example, that 12 man-years of effort will be required, but it would not be surprising if the actual effort were between three and 48 man-years. At later stages in the project, the amount of uncertainty declines, partly because a large proportion of the effort has already been expended, and partly because much more is known about the detailed requirements and system design.

This concept of an estimate may satisfy a statistician, but the customer normally needs to know how much a system will cost and when it will be delivered. Being told that it will probably take one year, but could take two years, or even four years, or possibly only six months, is unlikely to inspire confidence in the abilities of the development team. Developers are therefore usually required to commit to a particular delivery date and cost. Given the common pressures on costs and timescales, developers will usually propose targets that are within the range of uncertainty of the estimates, but towards the lower end. The inevitable result is that most projects overrun the budget and are delivered late. Customers accuse the developers of being inefficient, and developers blame the customers for setting unrealistic objectives. This situation is of benefit neither to the customers, nor to the developers, nor to the organisation as a whole.

SOFTWARE ESTIMATING IS INHERENTLY IMPRECISE

One of the main problems in estimating development costs is that most new systems include some technical features that are new to the organisation's development staff, and there is therefore no experience on which to base those parts of the estimates affected by the new features. Where a development project uses new hardware or software products, there will be little experience within the industry as a whole, and certainly no standard measures of productivity that will help an organisation to assess how the new products will affect estimates within its own development department.

The time and effort required for apparently similar tasks can vary considerably

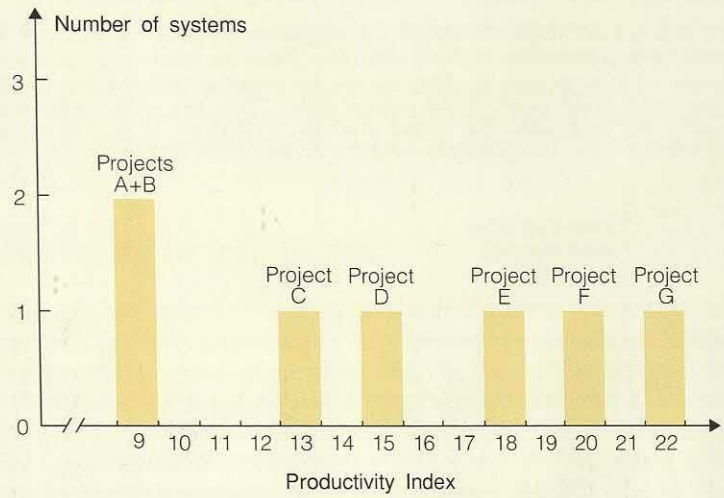
Furthermore, the time and effort required for apparently similar tasks can vary considerably. Often, the variability arises because development standards are not consistently applied. For example, some systems may undergo months of rigorous testing before being accepted, whereas others are subjected to only minimal testing. A rigorously tested system could easily absorb twice the development effort of a minimally tested system. In addition, there are variations in the experience levels of the team members, in the involvement of the users, and in management practices. All of these factors increase the variability in productivity, and hence, development time and effort, between different project teams.

Actual costs and effort are often not reported correctly

The fact that actual costs and effort are often not reported correctly gives rise to further problems. Indeed, many organisations unwittingly give experienced project managers scope for hiding overruns in budgets such as 'administration' or 'maintenance', or for transferring costs from overspending to underspending projects.

The net result of all these uncertainties on development project costs and timescales is illustrated in Figure 1.4, which shows the wide variability of productivities (as measured by the productivity

Figure 1.4 Marked variations in productivity are not uncommon within a single organisation



(Source: PEP database)

index, or PI value) on a sample of seven systems developed by one organisation. This is by no means the most extreme case in our database, although many PEP members are achieving much more consistent PI values than this.

Consider what would happen if the organisation whose PI values are illustrated in Figure 1.4 were to base its estimates on the assumptions that all its projects are developed at the same productivity, and that it could estimate the size of its systems accurately. Compared with the effort and time actually used, the estimates would have the levels of accuracy shown in Figure 1.5, overleaf. The solid lines through the origins of these graphs represent perfect estimates, and the dotted lines either side represent estimates with a level of accuracy of plus or minus 25 per cent. Only two of the seven projects fall within this range for estimated effort, and only one for estimated time. Unless this organisation were to be more consistent about the way in which it develops systems, it would be unlikely to improve its estimating skills beyond the level illustrated in Figure 1.5. For this organisation, and for many others in a similar position, the first steps in improving the accuracy of estimates are to understand the reasons for such variability in the development environment and to remove as much of it as possible.

To improve estimates, it is essential to remove as much variability as possible from the development environment

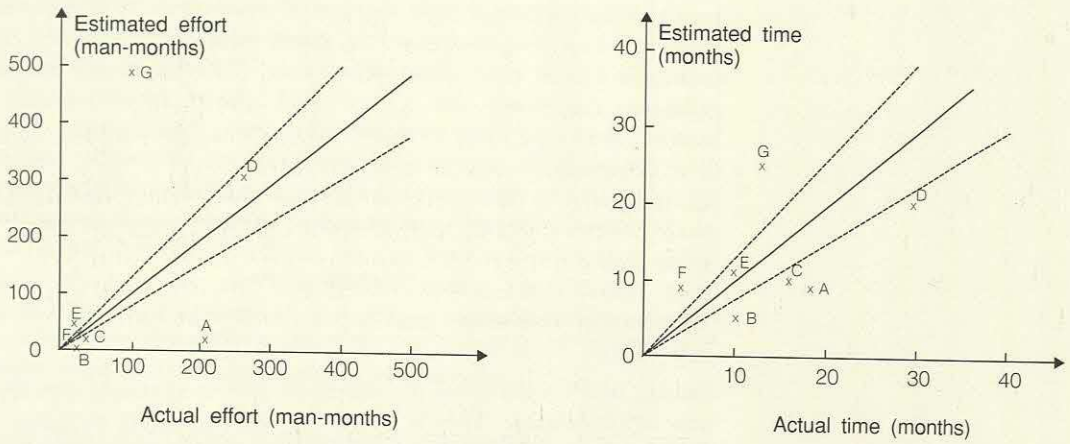
NO ESTIMATING TECHNIQUE WILL BE UNIVERSALLY APPLICABLE

Some of the early theoretical work on estimating was based on the assumption that there were universal formulae that would apply to any development project, and that would produce accurate estimates. Those who derived these formulae, or 'models', were attempting to define a relationship between the size of the system, usually measured in terms of the number of lines of code, and the effort and time required to develop it. Many of them showed a surprising lack of scientific rigour in their

Much of the early theoretical work on estimating was not scientifically rigorous

Figure 1.5 To improve the accuracy of estimates, it is important to remove as much variability as possible from the development environment

If the organisation whose PI values are illustrated in Figure 1.4 were not aware of the variability in its development projects, it might naturally assume that all projects are developed with about the same productivity and therefore estimate an average level of effort or time for projects, depending on their size. Plotting the estimated effort and time actually used would produce the graphs shown. Only two of the estimates of effort and one of the estimates of time fall within the boundaries that define a range of accuracy of plus or minus 25 per cent.



approach. Most of the models were derived from measurements of a small set of projects of a similar nature, and were not tested on projects developed in other organisations. When others have attempted to verify the models, the positions have been as inaccurate as those shown in Figure 1.5.

These results are not surprising in the light of the earlier discussion. Each organisation has its own way of developing systems, and its staff have different levels of skills and experience from those in other organisations. It is unreasonable to expect that the resulting differences in productivities between organisations could be modelled by a single formula.

The theoretical studies have shown, however, that the formulae can provide reasonable estimates of development time and effort for a particular organisation, provided the models are calibrated to take account of that organisation's productivity. The calibration cannot be done just once; it will need to be modified when new languages, computers, operating systems, design methods, CASE tools, and development methods are introduced. Some of these changes may have very significant effects on systems development productivity, and it must be recognised that the accuracy of estimates is likely to diminish for a time after a major change in development practices is introduced. As experience is gained with the changed development methods, estimating accuracy should rise to its previous levels, provided the estimating models are recalibrated to take account of the new environment.

Estimating formulae can be useful if the models are calibrated to an organisation's own development practices . . .

Even if the development methods used do not change much over time, the people in the organisation will. Some leave, some join,

. . . and to the productivity of its development staff

and those who stay become more experienced and may undergo various forms of training. There is a reluctance to measure the performance levels of individual developers, except in very general terms, so it will not usually be possible to assess directly the effect of these changes on overall development productivity. Nevertheless, the systems development manager needs to be aware that such changes may be taking place, and to measure their effects on the development process for all projects — for example, by keeping track of the changes in the Productivity Index.

STRUCTURE OF THE PAPER

Some of the organisations we surveyed during the course of our research for this paper have recognised the problems discussed above, and have taken steps to address them. The advice we give in this paper is therefore not merely theoretical, but has been put into practice by some organisations with positive results. By being more systematic in their approach to project estimating, systems development managers can effect quite dramatic improvements without necessarily making a large investment of time or resources.

In Chapter 2, we argue that estimates can be improved only if a programme is initiated to introduce more consistent estimating procedures. One person, or a small group of people, should be appointed to ensure that the nature of estimates is clearly understood, both by developers and their customers, that consistent estimating procedures are defined, that measurements of past projects are made and stored so that they can be used as a basis for calibrating estimating techniques, and that the accuracy of the estimates is constantly monitored.

The accuracy of estimates may be improved further by the selective use of techniques and models, not all of which are currently in widespread use. The various types of techniques available are discussed in Chapter 3. No one estimating technique will be more accurate in all circumstances than any other — each has different strengths and weaknesses. We explain the basis on which each technique is founded and discuss where the use of each would be most appropriate.

While these techniques have some features in common, each also has unique features. To ensure that an appropriate set of techniques is available to the development department, we provide guidelines in Chapter 4 on selecting those that are best suited to an individual organisation and to a particular stage of the development life cycle, choosing tools to support them where appropriate, and calibrating the tools to a particular development environment.

RESEARCH SOURCES

We carried out a review of the published literature on estimating, which revealed that while considerable effort has been expended on developing models of the systems development process, very little has been put into validating the accuracy of the predictions produced by them. We also consulted our colleagues at the

Cranfield IT Institute, who have studied recent developments in the field, and offer a training seminar on estimating. For information on the practical application of estimating techniques, we held detailed discussions with several PEP members. We should like to offer our special thanks to Ann Eldred of the FI Group for providing particularly valuable information on the successful implementation of estimating techniques within her organisation, and to Paul Rook, an independent consultant and member of the Esprit Mermaid project team, who provided information on the state of the art in using function-point techniques for estimating purposes. In the course of the research for this paper, we visited the suppliers of the most widely used estimating tools in the United Kingdom and attended demonstrations, in order to understand the facilities that they provide.

Chapter 2

Allocating responsibility for improving the estimating process

In Chapter 1, we identified some of the factors that make estimating so difficult and so inexact. One of the main problems is the variability in development productivity, both between different organisations and on different projects within the same organisation. Systems development departments can improve the accuracy of their estimates by measuring the productivities of their own development projects, by attempting to reduce the variability in the development environment, and by making allowances for the effect on productivity of the factors whose variability cannot be reduced.

Good estimating requires that data on past projects should be collected

Estimates can therefore be improved only if data about past development projects is collected, and if consistent development procedures are used. Collecting the data and ensuring that the procedures are consistent would normally be part of the responsibilities of a nominated individual or a small group of people. Throughout the rest of this report, we shall, for convenience, refer to this as the estimating group, even if it consists of only one part-time person.

Staff should be assigned to the estimating group for a maximum of six months

Some organisations have assigned staff to work full time in the estimating group, but experience has shown that after about six months, the accuracy of the estimates produced by an individual decreases. Our advice is that staff should be assigned to work full time in the estimating group for no more than six months at a time. It may be preferable, even for large systems departments, to have a panel of estimators, of about project-manager grade, who can be called upon to make estimates as and when required, but whose main job is developing systems. The benefits of this are that they maintain contact with the problems that occur in real projects and that they have an opportunity to use estimating techniques on a regular basis.

The main tasks of the estimating group are:

- To educate development staff and customers about the nature of estimates.
- To define estimating procedures.
- To collect data about projects to provide the basic input for estimating techniques.
- To measure the accuracy of the estimates.

EDUCATE DEVELOPMENT STAFF AND CUSTOMERS ABOUT THE NATURE OF ESTIMATES

We have seen that while exceeding estimated time and effort on a project is a common occurrence, it is unusual for a project to be finished in less time or with less effort than estimated. In many cases, estimates are undoubtedly too optimistic. Sometimes,

however, estimates are exceeded because they are not true estimates, but targets set for the developers, or chosen by a desire to win the business.

The first essential task of the estimating group is to educate both the developers and the customers so they understand that an estimate is an unbiased assessment of how long and how much effort it is likely to take a team to develop a particular system. A true estimate is not influenced by external factors such as personal ambition on the part of the project manager, or a desire to ensure that the project is authorised to proceed. The degree of uncertainty associated with an estimate must also be clearly understood — the uncertainty reflects the inherent and irreducible variability in the development process as much as the imprecision in the estimating process.

Everyone involved should understand why there is uncertainty about every estimate

Specifying the range of uncertainty in an estimate may appear to give the project manager an excuse for exceeding the estimate. To remove any suspicions that he is building unnecessary slack into the estimate, the project manager should involve the customer fully in the estimating process, should be open about the progress of the project by informing him of problems as they occur, and should involve him in replanning (and re-estimating) the project when necessary. As an organisation's estimating ability improves, some projects will continue to exceed their estimates, but an equal number should be delivered under estimate. As a consequence, customers should begin to realise that the estimates are genuinely unbiased, and should begin to make allowances in their own budgeting for the uncertainty of the estimates.

The customer should be involved in the estimating process

DEFINE ESTIMATING PROCEDURES

Any two people asked to produce an estimate for the development of the same system are likely to produce two very different answers. Part of the reason for the difference will be that each has made different assumptions about the activities required to produce the system, and part will be that each has different ideas about how long each task ought to take. Both may be right about some aspects of the estimates, but there is no way of telling who is right about which aspects. In order to ensure that, as far as possible, estimates are consistently produced, and are not dependent on the prejudices and experiences of particular individuals, it is essential that a set of procedures is defined for estimating, the essential components of which are described below.

Estimates must not depend on the prejudices of particular individuals

A STANDARD SET OF ACTIVITIES

To eliminate the variability caused by developing systems in different ways, estimates need to be related to a standard set of stages, and to activities within stages. Different sets of activities may be used for different types of projects, such as large or small projects, or for projects that use a prototyping approach. Most organisations are already using methods based on a standard development life cycle, such as that shown in Figure 2.1. The estimating procedure should ensure that all estimates are based on the activities for each stage of the life cycle.

Figure 2.1 The estimating procedures should take account of all activities at each stage of the development life cycle

Preliminary survey
Determine scope and objectives of stage
Investigate problem and determine need
.
.
.

Feasibility study
Determine requirements
Identify and evaluate solutions
.
.
.

Systems analysis
Investigate current business operation
Establish user requirements
.
.
.

Business design
Design local process
Produce outline system test plan
.
.
.

Technical design
Identify programs
Produce database definitions
.
.
.

Coding
Design, code, and test program A
Design, code, and test program B
.
.
.

System testing
Perform system tests
Review results of system tests
.
.
.

Acceptance testing
Perform acceptance tests
Review results of acceptance tests
.
.
.

Implementation
Perform take-on and conversion
Cut over to operational running
.
.
.

Post-implementation review
Review operational system
Obtain sign-off from users
.
.
.

GUIDELINES ON WHEN TO PRODUCE ESTIMATES

Estimates should not be produced just once at the start of a project, but should be updated at specific points in the development life cycle. The ends of the stages are natural points at which to revise estimates. Organisations may not wish to re-estimate formally at the end of every stage, but the estimating procedures must define the points at which it should be done. As a minimum, new estimates should be produced at the end of the feasibility study, and at the end of the requirements specification. It would also be advisable to undertake a re-estimating exercise before starting coding, since a large proportion of the effort is typically consumed between the start of coding and the start of live operation.

Estimates should be updated at specific points in the development life cycle

The end-of-stage estimates are formal exercises that should include people not participating in the day-to-day activities of the project as well as those directly involved. In addition to these formal estimates, the project manager should be making his own re-assessments of the estimates on a regular basis.

RESPONSIBILITIES FOR PRODUCING ESTIMATES

The procedures should define who is responsible for producing estimates. It may not necessarily be the same person or group for each formal estimating exercise. We suggest that the estimating group should be responsible for producing the first estimate for all projects, and all subsequent re-estimates on larger projects. On smaller projects, the project manager could be responsible for end-of-stage re-estimates, with provision for a full re-estimating procedure being carried out by the estimating group if the estimates change by more than a specified amount.

Ultimate responsibility for devising effort and timescale plans should, however, lie with the project manager. All the PEP members we spoke to in the course of the research for this paper emphasised the importance of the project manager's setting his own targets so that he feels committed to meeting them. Most PEP members who have set up a separate estimating group require that differences between formal estimates and a project manager's targets should be investigated, and authorisation for the project to proceed should be given only when there is a satisfactory resolution of the different views.

Ultimate responsibility for devising effort and time-scale plans rests with the project manager

SPECIFICATION OF THE TECHNIQUES TO BE USED

There are many estimating techniques, as we discuss in Chapter 3. The standards should specify which techniques to use at each estimating point in the project. The input data required for each technique should be defined and a procedure given for collecting it. For example, several techniques use function points as an estimate of the size of a system, so the procedures should give guidelines on how to count function points, and provide a standard form or computer program that can be used to calculate them.

For each estimating technique, the data and the procedure for collecting it should be specified

A STANDARD SET OF FORMS

In producing estimates, a large number of factors needs to be considered, particularly when constructing 'bottom-up' estimates.

In order to be sure that no factor is omitted, it is advisable to use a set of forms for building up estimates. An example of part of such a set of forms is shown in Figure 2.2. These forms are used both for estimating and for collecting measurements of the development process.

Figure 2.2 In order to be sure that no factor is omitted, it is advisable to use a set of forms for building up estimates

An example of part of a set of forms is shown below.

Project name:	Change record no.:

System size

Object occurrence counts:			
Task no.	Deliverable	Count	No.
3.1	Options	Feasible options	
3.7	Approved option outline	Online events (Event catalogue)	
		Batch events (Event catalogue)	
		Other function-list entries (Online)	
		Other function-list entries (Batch)	
Function-point count:		Approved option outline	

Work effort

Task no.	Task name	Man-hours (estimated)	Man-hours (actual)
3.1	Identify options		
3.2	Produce high-level design		
3.3	Produce system test strategy		
3.4	Provide development plan(s)		
3.5	Cost each option		
3.6	Compare for risks and benefits		
3.7	Present to obtain approval		
	Total man-hours all standard tasks:		

Total stage effort:			
Total man-hours all tasks for stage (excluding overheads)		Estimated	Actual
Skills distribution: (enter split of total man-hours across applicable resource categories)	Project manager		
	Systems analyst(s)		
	System designer(s)		
	Programmer(s)		
	Technical reviewers (outside project team)		
	Others (specify)		
Start date of development stage			
End date of development stage			
Total duration in working days			

Project leader signature:	Date:

(Source: Adapted from Legal & General)

COLLECT PROJECT DATA TO PROVIDE THE BASIC INPUT FOR ESTIMATING TECHNIQUES

All estimating techniques require a knowledge of the past. It is therefore necessary to collect data about past projects in order to be able to make accurate estimates. However, as we discussed in Chapter 1, each organisation develops systems differently, and the conclusions drawn from analysing the data collected by one organisation may not be applicable to another. Each organisation must therefore collect and analyse its own measurement data on past projects.

Each organisation should collect and analyse its own measurement data on past projects

Collecting the data is not a time-consuming task, but before data collection can start, the estimating group must make two important decisions — which data to collect, and how to collect it consistently. The set of data collected will be determined by the estimating techniques that are used. As a minimum, however, data must be collected about the size of each system, and about the time and effort required to develop it. The time and effort data should be broken down at least to the stage level, and where possible, to the activity level. A sample set of data that should be collected for each project is shown in Figure 2.3.

The meaning of each data item collected must be consistent from project to project, so clear definitions are required for each item. A common area of concern among PEP members is the precise definition of a line of code. The definition used for the purpose of PEP assessments is given in the Data Collection Manual, and runs to about two pages. Many other definitions are possible, but none is necessarily 'correct' or better than any other. The important point is that an organisation should find a definition that suits its style of development and use it consistently. Of course, we encourage PEP members to use the definition given in the Data Collection Manual because this enables us to compare the productivities of different organisations.

The meaning of each data item must be clearly defined

Unfortunately, development methods and tools change with time. Data about Assembler-based projects collected 10 or 15 years ago would have little relevance to producing estimates for projects based on the use of CASE tools or expert systems. The project data being collected must therefore be kept under review to ensure that it is relevant for estimating in the current development environment.

The project data being collected does not necessarily have to be held in a computer 'database'. In fact, one of the organisations that we talked to in our research, and that had the most advanced estimating procedures of those we encountered, holds its historical data in paper files. Another PEP member holds data about 100 projects in a Lotus 1-2-3 spreadsheet. Lotus may not be the ideal package for storing a database, but it was supplied 'free' within the organisation, and was therefore a tempting choice for an estimating group with a small budget. There are many other PC-based database packages that would also be suitable for holding project data. For the purposes of PEP assessments, we use the PADS PC-based metrics database management system supplied by QSM. This provides both data capture and storage facilities, and the graphical analysis techniques used in the PEP assessment reports.

It is not essential to hold project data in a computer database

Figure 2.3 The set of data collected for each project will be determined by the estimating techniques that are used, but is likely to include the items listed below

General system information

Project name
Project manager
Brief description
Planned and actual start date
Planned and actual finish date
Planned and actual effort
Main build elapsed time
Total project elapsed time
Development type — new/modification/package/other

Programming statistics

For each programming language:
— Number of new lines of code
— Number of re-used but modified lines of code
— Number of new programs written
— Number of programs modified
Lines of code written for data dictionary
Total resulting system size

Effort statistics

A breakdown of effort and time into the activities defined by the life cycle
Within each activity, effort broken down by grade of staff
Peak manpower for each stage

Non-programming costs

Machine-time by stage of project
Cost of software specifically purchased for the project
Cost of hardware specifically purchased for the project

Estimates produced during project

For each estimate produced at the end of a stage:
— Estimated elapsed time
— Estimated effort
— Estimated program size
— Estimated function-point count

Development environment

Staff turnover
Ability of staff related to required skills
Use of standards/methods
Use of tools
Turnaround times
Hardware reliability
Effort required to develop conversion programs
Difficulty of agreeing on requirements with users
Number of user departments involved in agreeing on requirements

MEASURE THE ACCURACY OF THE ESTIMATES

It is essential to be able to measure the accuracy of each estimate

The prime objective of the estimating group is to improve the accuracy of systems development estimates. To determine whether improvements are being made, it is essential to be able to measure the accuracy of each estimate. The common measures of estimating accuracy are slippage and overrun. For both time and effort, these are defined as the difference between the actual and estimated values, divided by the estimated value. In PEP assessments, the measures apply only to the main-build stage of projects. Since we recommend, here, that re-estimates are made

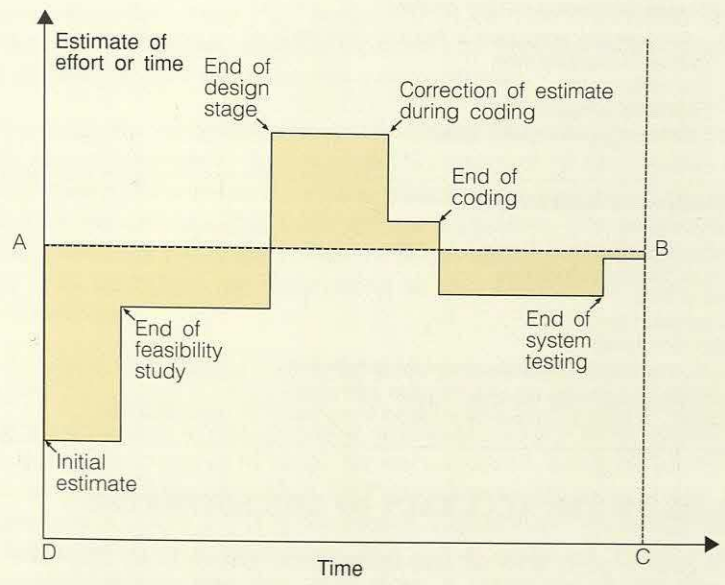
at the end of most stages during a project, these single measures of slippage and overrun do not reflect the accuracy of estimating throughout the project.

A visual method of assessing estimating accuracy is to plot the actual slippage and overrun for the estimates made at each stage of a project on a diagram similar to the one illustrated earlier in Figure 1.3. Perfect estimates at each stage would lie on the horizontal line, with a value of 1.00. Most projects ought to lie within the darker shaded area. A high percentage falling outside this area indicates poor estimating performance.

Another measure of estimating accuracy is the Estimating Quality Factor (EQF), which was described by Tom DeMarco in 1982, but which is not yet in widespread use. The measure, which is defined in Figure 2.4, is easy to construct and gives not only a single number that represents the accuracy of a series of estimates, but also a graphical representation showing how the estimates have changed during the course of a project. The use of this measure will encourage realistic re-estimates to be made as soon as possible after a deviation becomes apparent. The measure results in a value between zero and infinity, where a large EQF corresponds to a good estimate. DeMarco suggests that a value of 10 is a reasonable target to aim for.

Use of the Estimating Quality Factor will encourage the project manager to re-estimate as soon as a deviation becomes apparent

Figure 2.4 The Estimating Quality Factor gives not only a single number, which represents the accuracy of a series of estimates, but also a graphical representation showing how the estimates have changed during the course of a project



The line A-B represents the actual total effort or time used.

$$EQF = \frac{\text{Area ABCD}}{\text{Darker shaded area}}$$

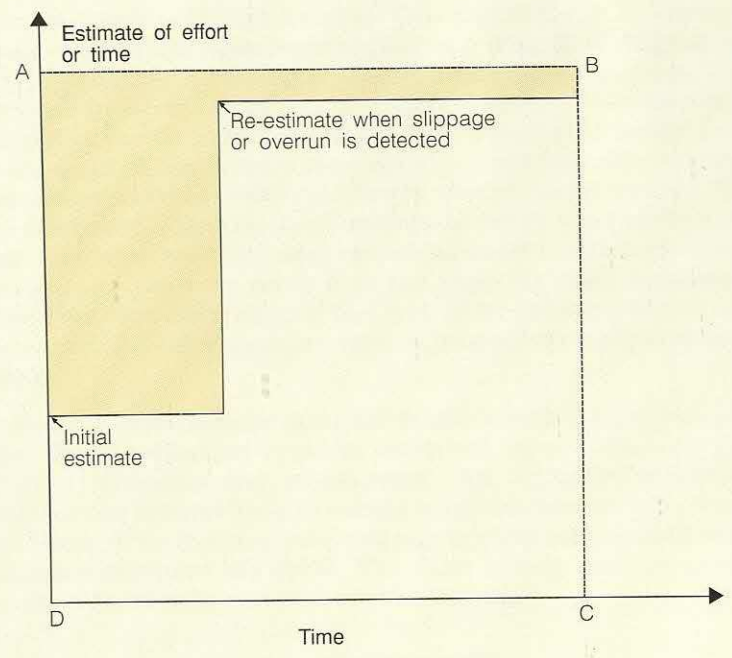
Poor estimates result in a low value for EQF; good estimates result in a high value. An EQF of 10 or higher is good.

In the example shown in Figure 2.4, the initial estimate was much lower than the actual value, but this was partially corrected at the end of the feasibility study. At the end of the design stage, a further correction was made to the estimate, but this turned out to be an over-estimate. Three further corrections to the estimate were made before the end of the project. The final total effort could not, of course, be known until the end of the project. Once this value is known, the line A-B can be drawn on the graph. The darker shaded area then represents the cumulative variance of the sequence of estimates. A perfect estimate would have been at point A initially, and would never have varied. There would then be no darker shaded area, giving an EQF value of infinity. One advantage of the EQF measure is that it equally penalises over- or under-estimates, thereby discouraging a tendency to over-estimate to be on the safe side.

The Estimating Quality Factor penalises over- and under-estimates equally

Some estimators argue that a good estimate can turn out to be inaccurate because of poor project control. However, if a project is deviating significantly from the original estimates, re-estimates should be made as soon as the slippage or overrun is detected. Figure 2.5 shows the graph of estimates for a project in such a situation. Prompt action to correct the estimate minimises the shaded area, and hence, increases the EQF. The EQF measure

Figure 2.5 **If a project is deviating significantly from the original estimate, re-estimates should be made as soon as the slippage or overrun is detected, in order to increase the Estimating Quality Factor**



The line A-B represents the actual total effort or time used.

$$EQF = \frac{\text{Area ABCD}}{\text{Darker shaded area}}$$

therefore rewards both good estimating and good project monitoring and prompt re-estimating.

The EQF measure can be calculated in a way to reflect values that are important to the organisation. For example, if an organisation places a high value on achieving good estimates as early as possible, it would be possible to weight the components of the shaded area, so that estimates produced near the start of a project have a greater influence on the score than those at the end.

The Estimating Quality Factor can be calculated to reflect values that are important to the organisation

The accuracy of estimates of the costs and timescales of development projects might be further improved by selecting and using appropriate estimating techniques. None will, of course, be universally applicable to all organisations or for all stages of a development project. The types of techniques available to the systems development manager and indications of their particular strengths and limitations are discussed in the next chapter.

Chapter 3

Choosing appropriate estimating techniques

All estimating techniques start with a measure of the system's size

There are many estimating techniques, ranging from 'guessing' to complex mathematical models of the development process, that can be used to calculate the time and effort required to develop a system. All of them require the estimator to start with a measure of the system's size. Currently, lines of code is the most widely used measure, although an increasingly used measure is the inherent amount of functionality delivered by a system. There are at least three variants on this measure — function points (devised by A J Albrecht while at IBM), feature points (devised by T Capers Jones), and Mark II function points (devised by C Symons of Nolan Norton). All of these variants involve calculating weighted sums of the number of features such as files, transactions, and processing steps. The specification of weights and the identification of features depends on the estimator's subjective judgement. Two different people are likely to arrive at significantly different values if asked to estimate a system's size in these terms. However, if an organisation is developing systems of a similar type, it will be able to develop its own rules, and could set up a group of people who specialise in counting function points. These measures should ensure consistency for its own systems.

A third measure of a system's size is based on design characteristics. One such measure, which was proposed by Tom DeMarco, is System Bang (or design weight). This measure can be derived when a system has been represented as a data flow diagram of the type produced by Yourdon's structured systems analysis. The technique involves constructing weighted sums of various items depicted in the data flow diagrams. The weights are subjective, and although DeMarco has some suggestions as to suitable weights, he recommends that each organisation should derive its own. We have not heard of any rigorous attempt to verify whether System Bang produces accurate and useful measures, although some organisations are now using it as the basis for their estimating techniques. One advantage of this type of technique is that it can be automated if CASE tools are used to produce structured system designs.

No one technique is more accurate in all circumstances than any other

No one technique is more accurate in all circumstances than any other. Each technique requires different input data, and has different strengths and weaknesses. An organisation should therefore not restrict itself to using a single technique, but should use a range of techniques, selecting appropriate ones at each stage in the development life cycle. The most widely used types are discussed further in the rest of this chapter.

THE WIDEBAND DELPHI TECHNIQUE

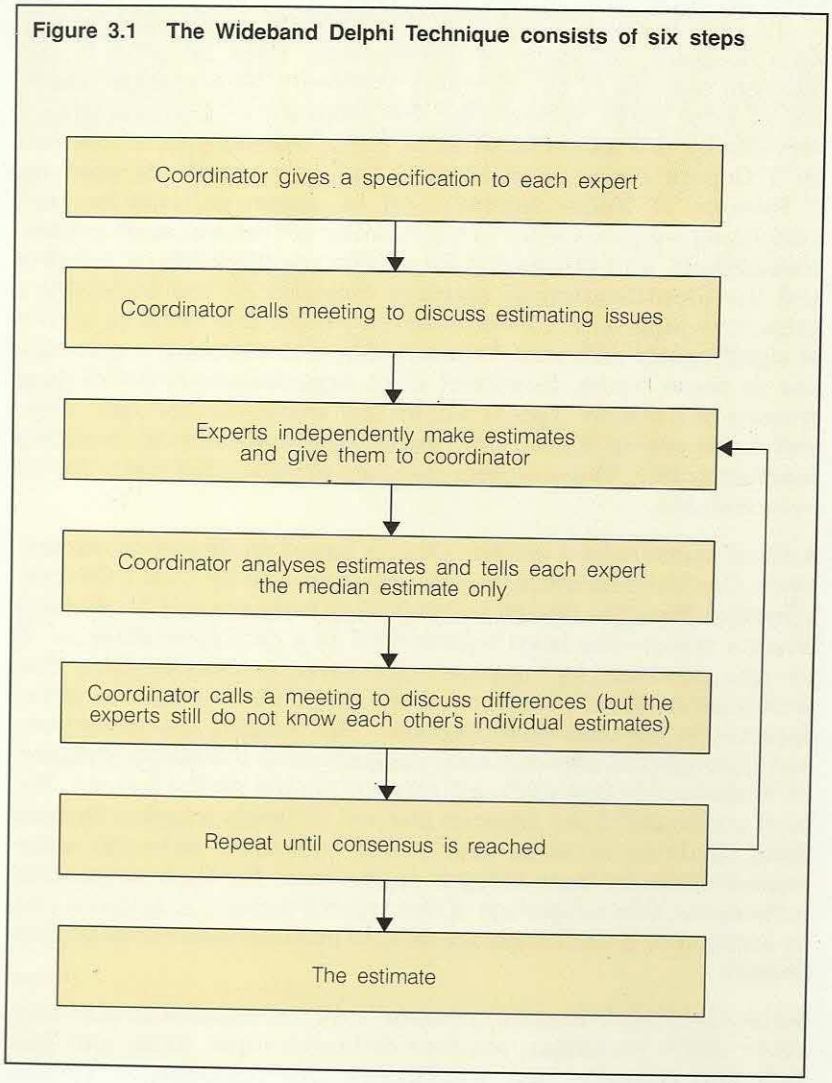
With all techniques, and particularly modelling techniques, subjective assessments (of size, complexity, skill, or other factors pertaining to a particular environment) will have a significant

effect on the resulting estimate. It would therefore be unwise to rely on the judgement of a single individual, so we recommend that a group of people be involved in any estimating exercise. This will help to reduce the biases, incomplete recall of facts, and errors to which individuals are prone.

The Wideband Delphi Technique can be used to obtain the best combined estimate from a group of people

The Wideband Delphi Technique can be used to obtain the best combined estimate from a group of people. This technique was developed by Barry Boehm, a well known writer and speaker on software metrics, and consists of six steps, which are described in Figure 3.1.

Figure 3.1 The Wideband Delphi Technique consists of six steps



A systems development department may not wish to implement this technique in the full manner described in Figure 3.1. However, if it uses groups of people to prepare estimates, as we recommend, it should try to implement the main feature of the Wideband Delpi Technique, which requires that the estimators write down their estimates independently, and do not initially divulge them to other members of the group. This avoids the problem of round-table discussions, where the views of the estimator with the strongest personality tend to dominate.

ESTIMATING BY ANALOGY

Estimating by analogy involves matching the characteristics of the proposed project to those of similar previous projects. This can, of course, be done only if there is a documented history of the organisation's previous projects. The larger the history, the more likely it is that similar projects will be found.

Several factors must be taken into account when selecting previous projects. It is important that the same development life cycle was used so that activities can be compared on a consistent basis. The type of application should also be similar so that, for example, a database project is compared with a database project, and preferably with one that used the same database management system and the same query language. The sizes of the projects should be similar. The relationship between size and effort or timescale is not linear, and simple extrapolations will lead to significant inaccuracies in the resulting estimates. If there are no projects of a similar size in the database, an alternative estimating technique should be used.

All organisations should have estimating by analogy as one of their techniques

All organisations should have estimating by analogy as one of their basic techniques, since by its nature, it reflects the conditions of the organisation's particular development environment. It does not, however, seem to be in widespread use among PEP members, probably because few yet have relevant information about a sufficiently large number of past projects.

THE TOP-DOWN ESTIMATING TECHNIQUE

The starting point of the top-down technique is an estimate of the total effort and time required to complete one or more stages of a project. These total values are then broken down into stage-by-stage estimates, and sometimes into activity estimates. The method of doing this breakdown is to allocate a fixed percentage of the total effort to each stage. A similar breakdown is often used to allocate the proportions of each grade of staff to each stage. In both cases, the percentages depend on the programming language used. An example of the effort breakdown used by one PEP member is shown in Figure 3.2, overleaf.

In top-down estimating, a fixed percentage of total effort is allocated to each stage

The main benefit of the top-down technique is that it does not require a detailed knowledge of the system design. It can therefore be used at early stages of the project, and should be used as a cross-check on the bottom-up technique. The top-down technique is used in many of the commercially available estimating tools, which are described in the next chapter.

THE BOTTOM-UP ESTIMATING TECHNIQUE

Bottom-up estimating starts from a detailed breakdown of the activities required to develop a system

The bottom-up estimating technique starts from a detailed breakdown of the activities required to develop a system. Each activity is individually estimated, and the total estimated cost and timescale are derived from these individual estimates. Each activity should be expected to take no more than a few days. If any activity is expected to take more than about two weeks, it should be broken down into smaller activities because progress monitoring, and hence, project control is more difficult if activities take longer than this.

Figure 3.2 With the top-down estimating technique, total time and total effort are broken down into stage-by-stage estimates

The table gives guidelines for the proportion of the total project effort likely to be spent in each stage. The column headed PL/1 shows the figures for a typical project using PL/1 or a similar third-generation programming language. The column headed FOCUS shows the corresponding figures for a typical development using FOCUS or a similar fourth-generation language. The column headed Telon shows the corresponding figures for a typical development using Telon or a similar application generator. The figures quoted are for the same size system, and hence give an indication of the reduction in effort that results from using a fourth-generation language or an application generator.

	Relative effort					
	New system			Enhancement system		
Stage	PL/1	FOCUS	Telon	PL/1	FOCUS	Telon
Preliminary survey	1	1	1	1	1	1
Feasibility study	1	1	1	1	1	1
System analysis	10	9	10	5	5	5
Business design	12	12	12	6	6	6
Technical design	9	3	9	9	6	9
Programming	27	7	21	28	23	22
Implementation planning	9	9	9	9	9	9
System testing	15	9	15	23	15	23
Acceptance testing	11	5	11	11	8	11
Implementation	5	5	5	7	7	7
Total	**100**	**61**	**94**	**100**	**81**	**94**

(Source: Lloyd's of London)

The bottom-up technique should be used when the next one or two stages of a project are being planned in detail. If a bottom-up estimate cannot be made, it may indicate that the current stage has not been completed to a sufficiently detailed level.

A disadvantage of bottom-up estimates is that they do not easily take account of the large increases in effort that result from shortening project timescales. (This effect is discussed in more detail later in this chapter.) Even though bottom-up estimates may appear to include everything that could contribute to the total effort, they must be complemented by other techniques to ensure that the effects of timescale compression have been correctly assessed, and also to provide an independent cross-check on the realism of the estimates.

MATHEMATICAL MODELS

Mathematical models for estimating relate size to the rate at which lines of code are produced

One of the most widely used estimating techniques is a mathematical model of the relationship between the characteristics of a system and its development environment, and the time and effort required to develop it. In its simplest form, such a model relates size to the rate at which lines of code are produced — for example, 20 Cobol statements per man-day.

The aim of a model is to specify a formula that takes account of the most important factors that determine the costs and timescales of developing a system. A model is, in effect, a formalised version of estimating by analogy and must be tailored to each organisation's style of development to give accurate results.

As we discussed at the beginning of this chapter, all estimating techniques require an estimate to be made of the size of the system, and modelling techniques are no exception. Given an estimate of the size, the formula can then be used to calculate the likely effort and time required to develop the system. An array of correction factors can be applied to the basic formula to take account of the development team's productivity, the project's complexity, and the development environment. Some of the models include a correction factor to allow for the effect of compressing timescales. These form a particular class of models, known as constraint models, which are described in the next section.

Many of the models that have been developed can be reduced to the same basic formula, but with different parameters. This formula, together with examples of particular models, is shown in Figure 3.3. This figure shows the wide range of effort and time estimates produced by the different models for an 'average' PEP

Figure 3.3 The basic formulæ for calculating effort and time are used by several estimating models

Effort equation:

$$\text{Effort} = A.(\text{Size})^b$$

where Effort is measured in man-months, and Size in thousands of lines of code. A, b are constants determined by the particular model.

Schedule equation:

$$\text{Time} = C.(\text{Effort})^d$$

where Time is measured in months, and Effort is measured in man-months. C, d are constants, determined by the particular model.

Some of the constants proposed for these formulæ are given in the table below, in increasing order of 'b'.

The right-hand columns show the effort, time, and cost that would be estimated by each model for a program of 50,000 lines of code, and assuming a cost of £3,000 per man-month. For comparison, estimates obtained from Putnam's model for average PI and MBI are shown at the bottom of the table.

Model	A	b	C	d	Effort (man-months)	Time (months)	Cost (£ thousand)
Watson-Felix (IBM)	5.2	0.91	2.47	0.35	183	15.3	548
Nelson (Software Development Corporation)	4.9	0.98	3.04	0.36	227	21.4	680
Freburger-Basili	1.48	1.02	4.38	0.25	80	13.1	240
COCOMO — organic mode	2.4	1.05	2.5	0.38	146	16.6	438
Herd	5.3	1.06	—	—	335	—	1,005
COCOMO — semi-detached mode	3.0	1.12	2.5	0.35	240	17.0	719
Frederic	2.43	1.18	—	—	246	—	737
COCOMO — embedded mode	3.6	1.20	2.5	0.32	394	16.9	1,180
Phister	0.99	1.275	—	—	145	—	435
Jones	1.0	1.40	—	—	239	—	717
Watson-Felix	1.12	1.43	—	—	301	—	903
Halstead	0.70	1.50	—	—	247	—	742
Values from the Putnam model with PI = 15, MBI = 3					84	10.7	252

project. Other models using linear relationships, look-up tables, and other mathematical functions, have also been developed. The estimates produced by QSM's SLIM tool, which is based on Putnam's model and uses a mathematical function known as a Rayleigh curve, are included in the figure for comparison.

There is wide variability in models' parameters and in the resulting estimates

The important point to note from the list of models in Figure 3.3 is the variability of the parameters and the resulting estimates. Each model was derived by statistical analysis of the size, effort, and time data for a set of projects from within the same organisation, or for the same general type of project, and represents the best fit with that particular set of projects. The differences between the models is a consequence of the variability in the development environments between the sets of projects. It does not indicate that some of the models must be 'wrong', or, indeed, better than others.

The choice of model must be based on its fit with an organisation's own development environment

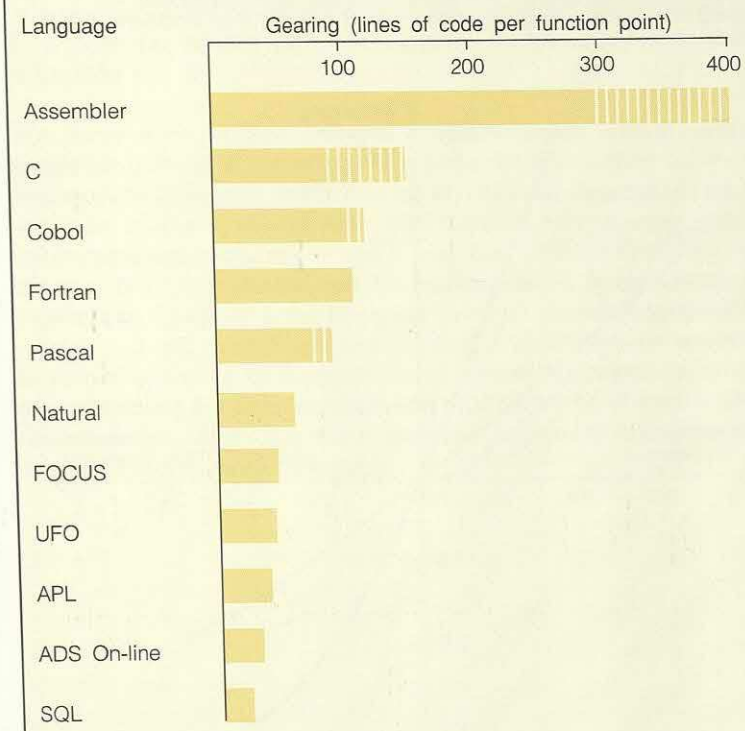
It can be seen from this brief description of the models that there is a wide variety to choose from, but that the choice of model must be based on its fit with an organisation's own development environment. In choosing a model, an organisation must analyse its own historic data to decide which is the most suitable. Several PEP members have done this by 'replaying' historic data through the models to see which gives the most consistently accurate estimates. Alternatively, if sufficient historic data is available, PEP members might carry out a statistical analysis to determine the parameters for the basic formula that best matches their data, or even to determine a different formula.

A function-point count may be converted to an estimate of effort by one of two methods

The models described above use lines of code as their measure of size. Many organisations, however, are now counting function points rather than lines of code. A function-point count may be converted to an estimate of effort by one of two methods:

- The first method is for a systems development department to analyse its own projects to determine a productivity rate in function points per man-month, and to use this value directly. Some PEP members are already using this approach. It is likely to produce more accurate estimates than lines-of-code-based models for development activities such as specification, documentation, and training, whose scope is determined by functionality rather than program size. This approach is, however, likely to be too simplistic for estimating main-build time and effort because it takes no account of the effect of time pressure, or of the relationship between the rate of function-point delivery and size of project. A new estimating tool (Before You Leap Mark II) incorporates a model that directly converts Mark II function points into estimates of effort and time for the main-build and other stages of the life cycle. The details of the model on which the tool is based have not, however, been released.
- The second method converts function points to lines of code. Typical conversion factors for some of the more common programming languages are shown in Figure 3.4. (A more complete list can be found in PEP Paper 12, Figure 1.5.) These factors are used to convert a function-point count to a lines-of-code count, which can then be input to one of the models described earlier.

Figure 3.4 **For each programming language, there is some consistency between the number of lines of code and the number of function points**



CONSTRAINT MODELS

Estimates should be adjusted to take account of timescale or staffing constraints

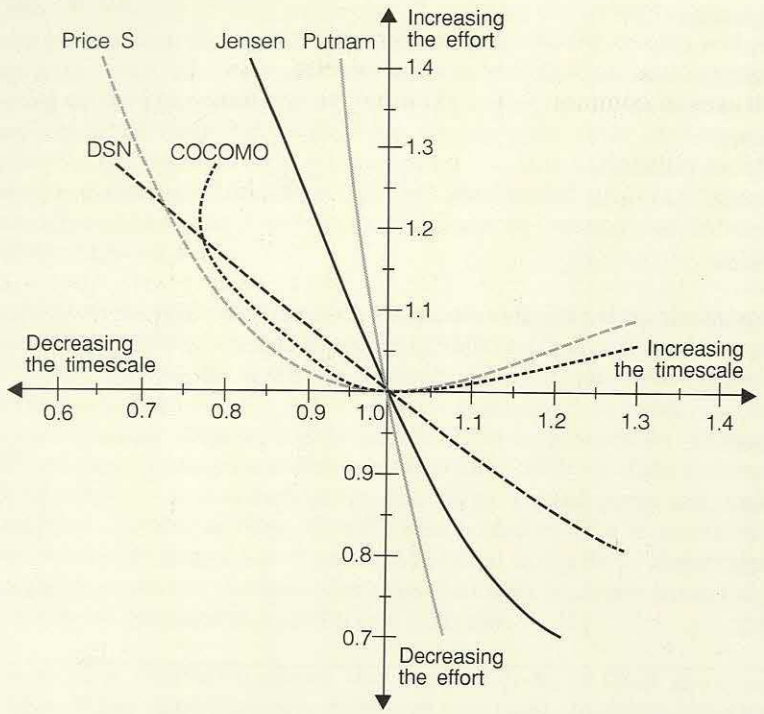
The estimates for effort and time for a particular size of system produced by all of the models discussed so far are based on the assumption that there are no constraints either on the development schedule or on the peak size of the project team. Often, however, a system needs to be developed within a particular timescale, or with a limited number of staff. To derive an estimate under these conditions, it is not sufficient merely to compress or expand the same amount of effort into a shorter or longer time. Several researchers have investigated the nature of the trade-off between time and effort for a given size of system. They have all found that shortening the timescale leads to a disproportionate increase in effort, and that there is a minimum time below which it is impossible to develop the system, no matter how many people are assigned to its development. One of the best known descriptions of this effect is given in the book, *The Mythical Man-Month,* by Frederick P Brooks (published in 1982 by Addison-Wesley).

There is agreement that effort increases as the timescale is shortened . . .

Several models, known as constraint models, have been developed in an attempt to calculate the effects of timescale constraints on manpower, and *vice versa*. A comparison of some of these models is shown in Figure 3.5, overleaf. PEP members should be familiar with the Putnam model shown in this figure. All models are in agreement that effort increases as the timescale is shortened, although there are considerable variations in the magnitude of such effects. There is less agreement on the effect of lengthening

Figure 3.5 Several models have been developed in an effort to calculate the effects of constraints on estimates

All models are in agreement that effort increases as the timescale is shortened, but there is less agreement about the effect of lengthening the timescale.



The numbers on the axes are the values for effort and timescale relative to the estimated values assuming no constraints.

the timescales, with some models showing that effort increases, and others that it decreases. Data collected by PEP indicates that there appears to be an optimum development time, which for projects of between 20,000 and 50,000 lines of code, is about nine months. Beyond this time, there is a tendency for effort to increase, because staff turnover increases and the requirements change.

. . . but less agreement about the effect of lengthening timescales

As with the mathematical models described earlier, the reason that different researchers have found different relationships is that each organisation responds differently to constraints. It is difficult to determine the exact relationships for any particular organisation, because controlled experiments with the same system being developed under different time or staffing constraints are not practical. The relationship between time compression and effort is, however, very important since, as Figure 3.5 shows, all the models agree that shortening the timescale increases the effort significantly.

A constraint model should form an essential part of every organisation's estimating 'toolkit'. It is particularly important to use one in the initial planning stages of a project. Developers are often under pressure to shorten timescales, and simultaneously to reduce costs. The use of a constraint model, particularly if

supported by a software tool, allows developers to discuss with their customers the various options for striking a balance between functionality, delivery date, and cost. The customer can then decide whether it is worth paying a premium for early delivery of a system, and the development manager can avoid being placed in a position where he is committed to an impossible delivery schedule.

We have seen in this chapter that there are many techniques available to help in the estimating process. While they have certain features in common — for example, they all depend on an initial measure of the system's size, and they all rely to some extent on subjective assessments — each also has unique features, which means that they should not be implemented in isolation. As we explain in Chapter 4, techniques need to be implemented in the context of an overall project-control framework to ensure that tools are selected to support them where appropriate, that they are calibrated for the development environment of each individual organisation, and that they provide adequate coverage of the whole development life cycle.

Chapter 4

Selecting and implementing the techniques

In the previous chapter, we described several techniques that can be used for estimating the effort and time required to develop a system. These techniques cannot be used in isolation, but must form part of a consistent estimating and project-control framework. In particular, those involved in implementing estimating techniques must choose a set of techniques to ensure that the most appropriate one is always available at any particular stage of a development project — no one technique provides the best results at every stage of the systems development life cycle. Once the techniques have been chosen, consideration can be given to the merits of using software tools to support them. Of course, since no technique can be expected to model the characteristics of every organisation's development environment, the tools will always have to be calibrated before they can be used.

USE DIFFERENT TECHNIQUES AT DIFFERENT STAGES IN THE LIFE CYCLE

In Chapter 2, we recommended that estimates should be revised at the end of each stage in the development life cycle. The information available at the end of successive stages increases as the project progresses. At the beginning of a project, very little is known about the required system, apart from the general requirements. As the project proceeds, the business requirements are identified in detail, a technical design is completed, and code is produced. To produce the most accurate estimates at each stage, it is essential to use all the information that is available, and the most appropriate estimating techniques. We indicate below the possible combinations of techniques for each stage. These combinations are not exhaustive; circumstances may dictate that other techniques be used instead.

The information available for estimating purposes at the end of successive stages increases as the project progresses

Before a project is authorised, the cost/benefit analysis will require that estimates be made of the total cost and the delivery date for the system. These estimates form the basis of the decision about whether to proceed with the proposed system. Sometimes, these estimates may be produced before the start of the feasibility study, but they should always be produced or revised at the end of that stage. When preparing the total cost and timescale estimates, it is usually advisable to examine the effects of altering the timescales or of limiting staff numbers. The customer can then choose a timescale that maximises the net benefit from the development and the project manager can plan how to obtain additional staff, if this is necessary. The techniques available for producing the cost and time estimates at this stage are mainly analogy and expert judgement (using, for example, the Wideband Delphi Technique). The overall estimates produced from this process can be used to derive cost estimates and staffing profiles for individual stages, using the top-down technique, which could

Estimates of the total cost and delivery date should be produced or revised at the end of the feasibility stage

then be fed into a constraint model (together with the size of the system) to investigate the impacts of different timescales and staffing levels.

At the end of the business-requirements stage, modelling techniques are appropriate

At the end of the business-requirements stage, enough information will be available to use modelling techniques based on counts of function points. An estimate of the number of lines of code can also be made at this point, based on a preliminary technical design, or by converting the function-point count. A lines-of-code-based model can then be used to produce overall estimates. If data flow diagrams have been produced, techniques such as System Bang can be used. The top-down technique will provide a useful cross-check at this stage. If, for example, experience shows that the effort required to develop any project as far as the end of the business-requirements stage is typically 20 per cent of the total effort, the remaining effort can be estimated as being four times that which has already been spent.

The completion of the detailed technical design is the next point at which significant new information is available. This information can be used to make more accurate estimates of both function points or lines of code, which can then be fed into a model to give estimates of time and effort for the remaining stages of the project. At this point, it is particularly important to use constraint models again, to ensure that the planned staffing levels are cost-effective and that the planned timescale is achievable.

At the testing stage, estimates are made by analogy with previous projects

Before embarking on the system and acceptance testing stages, some organisations need estimates of how long the testing will take. Others are constrained by the fact that the system has to go live on a certain date, and the estimate, if any is made at all, is likely to be of the reliability that will be achieved by that date, rather than of the effort that it will take to complete the system tests. The amount of system testing depends very much on the views of a particular organisation. The main methods of estimating at this stage are therefore by analogy with previous projects, and a bottom-up estimate, composed of estimates for each individual test or group of tests. The top-down technique can give a useful cross-check, based on the knowledge that system testing takes, say, 10 per cent of the total development effort.

More than one estimate should always be produced, and any discrepancies resolved

At the end of each stage of development, the project manager needs to produce detailed plans for the next stage. These should usually be produced using bottom-up estimating. On its own, this is not sufficient, however, since no account is taken of the effects of time compression. Bottom-up estimates for the next stage should therefore always be compared with an estimate produced from a model or from the application of the top-down technique. Any discrepancies between the different estimates should be understood and resolved.

The application of these techniques throughout the development life cycle is shown in Figure 4.1, overleaf. The life cycle shown is the traditional sequential development. Many organisations now use other life cycles, such as prototyping and iterative development, and the details of the techniques that are most appropriate at various stages would need to be developed by each organisation for its own environment. The two key principles, however —

Figure 4.1 Different estimating techniques are appropriate for different stages of the development life cycle

The ticks indicate the techniques that may be used at the end of each stage to produce estimates for subsequent stages. There are therefore no ticks in the system and acceptance test column, because at the end of this stage, the project is complete.

Technique	Stage of the life cycle					
	Initiation	Feasibility study	Business design	Technical design	Code and unit test	System and acceptance test
Wideband Delphi	✓✓	✓✓	✓			
Analogy	✓✓	✓✓	✓	✓	✓✓	
Mathematical models (size specified in function points)		✓	✓✓	✓✓		
Mathematical models (size specified in lines of code)			✓	✓✓		
Mathematical models (size specified in design metrics)			✓	✓		
Constraint models		✓✓	✓✓	✓✓		
Top-down	✓	✓✓	✓	✓	✓	
Bottom-up		✓	✓	✓✓	✓✓	

✓ Useful
✓✓ Very useful

choosing the most appropriate techniques at each stage, and using more than one technique whenever estimates are produced — remain valid.

CONSIDER USING SOFTWARE TOOLS TO SUPPORT THE TECHNIQUES

At least 15 different software tools for estimating are commercially available in the United Kingdom. Many of them implement one or other of the mathematical or constraint models and other estimating techniques described in Chapter 3. The main facilities provided by these tools are shown in Figure 4.2. Most of the available tools can be calibrated to make them consistent with the unique characteristics of each organisation's development environment. Without such calibration, an estimating tool will provide only limited benefits.

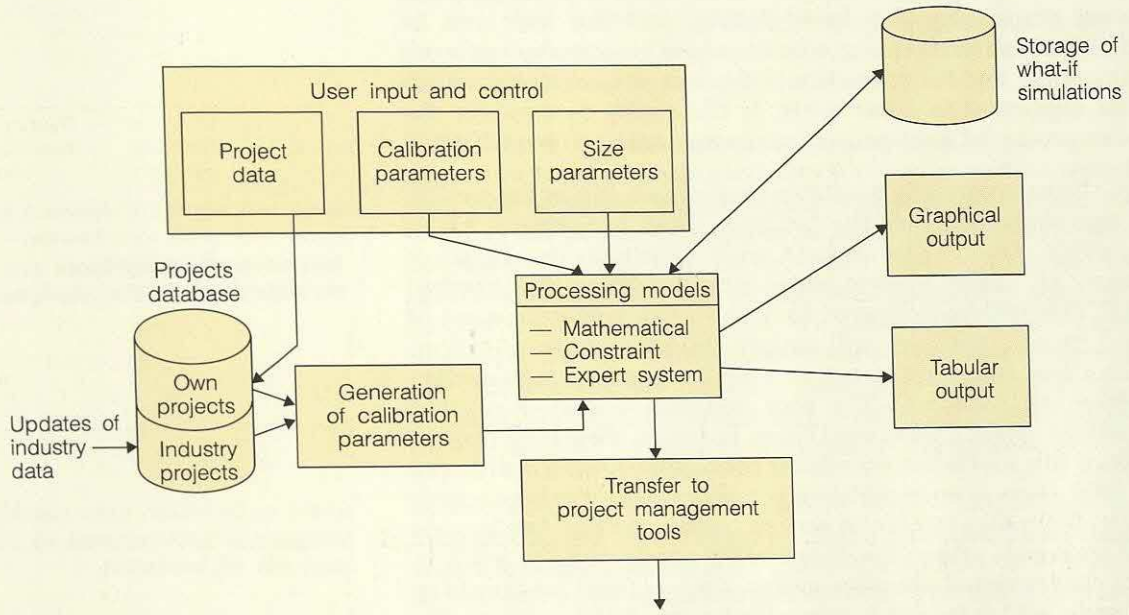
Without calibration, an estimating tool will provide only limited benefits

Before selecting a tool, it is essential to decide precisely which estimating techniques and models best suit the organisation. Only when these techniques have been selected and the procedures for estimating have been defined should consideration be given to the possibility of using a software tool. It is vital that the tools should be selected to support the technique, and not the other way around.

Tools should always be selected to support a technique, not vice versa

Nevertheless, it is perfectly possible to produce good estimates without the use of a tool, so an organisation should not purchase a tool just because it feels it ought to have one. As we have seen, many of the models are based on a simple formula, and can be run on a pocket scientific calculator. The software tools do not

Figure 4.2 Software tools to support estimating techniques provide a wide range of facilities



therefore provide a quicker or more accurate means of estimating than could be achieved manually at a fraction of the cost.

It is also possible to develop a tool in-house — Anglian Water, in conjunction with Barbara Kitchenham of the National Computing Centre, is doing just this because none of the available tools provided the kind of support that was suitable for Anglian Water's estimating methods. We have been told that this tool may be marketed commercially later in 1990.

Tools ensure a consistent approach to estimating

The main benefit of estimating tools is that they ensure a consistent approach to estimating. Most of them ask the user a sequence of questions to ascertain various characteristics of the system and the development environment. The same life-cycle model is used each time, although with many tools, the user is given the opportunity to delete unwanted activities or to insert new ones. A tool will also ensure that the calculations are done correctly each time, although the user must be careful to enter data correctly and be aware of the large differences in the estimates that can sometimes result from small changes in the input data.

Most of the commonly available tools are PC-based

Most of the commonly available tools are PC-based, and use the various features of standard PC interfaces, such as pull-down menus, colour displays, and windows. This gives them the appearance of being easy to use, although it is arguable whether these features are really necessary to input the relatively small number of data items that most of the tools require. Most of them also provide report-generation facilities, which can be used to represent the estimates, resources, and timescales both as tables and graphically in the form of pie charts, histograms, and cumulative-effort graphs.

One of the main benefits of the user interface and graphical reports would seem to be their ability to facilitate communication between developers and users. Results of 'what-if' modelling can be shown graphically and immediately, and the user can be directly involved in deciding what trade-offs to make between cost, timescale, and functionality. A feature of some tools, which could be important to some users, is the ability to transfer the estimates produced into project-planning tools, such as PMW.

Newer estimating tools are based on expert-system techniques

Some of the commercially available tools are shown in Figure 4.3, which also shows some of the facilities listed in Figure 4.2 that they provide. One of the newest tools shown in the figure is Estimator, an expert-system-based product from BIS Applied Systems. This will be delivered to a new customer with a set of standard 'rules'. The user will supply the answers to questions based on these rules, and using its inference engine, the system will derive an estimate. The tool is being designed so that organisations can add their own rules, based on their own project data. Since this tool is not scheduled for release until towards the end of 1990, there is no experience yet of how it works in practice. However, it appears to offer a way of 'remembering' the lessons learned from past projects without being constrained to work in terms of conventional sizing measures, such as function points or lines of code, which themselves need to be estimated, particularly in the early stages of the development.

Figure 4.3 The software estimating tools available in the United Kingdom provide a range of features

Name of tool	Supplier	Input size parameters			Processing model				Own project data-base	Graph-ical output
		Lines of code	Function points	Other[1]	Mathe-matical model	Con-straint model	Expert system	Other model[2]		
Before You Leap	Strategic Systems Technology	✓	✓		✓	✓				✓
Before You Leap Mark II	Strategic Systems Technology	✓	✓		✓	✓			✓	✓
Bridge	Hoskyns		✓		✓			✓		
CA-Estimacs	Computer Associates			✓	✓	✓				✓
Estimator	BIS Applied Systems		✓	✓			✓		✓	
GECOMO	GEC Marconi Software Systems	✓			✓	✓				✓
PC-COMO	Microcase Ltd.	✓			✓	✓			✓	
Price S	Price Systems Europe	✓		✓		✓		✓	✓	
Putnam	QSM	✓	✓		✓	✓			✓	✓

[1] Size determined from characteristics of the development, such as the complexity of the organisational involvement and geographic factors.

[2] Details of the method used by the model to construct estimates are not published.

CALIBRATE TOOLS FOR THE DEVELOPMENT ENVIRONMENT

We emphasised in Chapter 3 that any model must be calibrated with reference to an organisation's previous projects (and hence,

the tools that support it) before it can produce useful estimates. If there has not been sufficient time to build up a record of past projects, the standard default values supplied with the tools can be used, but they are unlikely to give very accurate results. In these circumstances, it is even more important than normal to use more than one estimating technique.

The possible settings for most parameters have not been derived by rigorous statistical analysis

Many of the models contain large numbers of parameters that adjust the basic estimates to allow for factors such as complexity, skill, and reliability requirements. The possible settings for most of these parameters appear to be have been derived by 'expert assessment', rather than by rigorous statistical analysis. Research has shown, however, that setting these parameters requires a high level of subjective judgement. We believe that a better approach is to apply a single multiplying factor to the basic estimates, with the factor being set for each class of project (database, maintenance, prototyping, and so on). This factor effectively replaces the 'A' and 'C' parameters defined in Figure 3.3.

Problems with calibration often relate to interpreting the inputs required by the tool

Some of the potential problems with calibration relate to interpreting the inputs required by the tool. The CA-Estimacs tool, which is being used, or is on trial, by a number of PEP members, appears to have problems in this area. The tool asks the user a series of questions such as, ''what is the complexity of the organisational involvement?'' The user has to select a reply from values representing a range, from simple to complex. The tool is indirectly trying to build up a function-point-related score, but both the meaning of the question, and how best to answer it are difficult to determine objectively, and different responses can result in significant differences in the estimates provided by the tool. The experience of PEP members is that significant amounts of effort need to be put into the task of understanding this particular tool. However, CA-Estimacs is unlikely to be unique in this respect, as a significant effort is required to calibrate all the tools, and to learn how best to interpret the results.

Tools must be recalibrated regularly

It is important that the estimating group regularly recalibrates the tools. In most organisations, the development environment is continuously changing, and the systems development department should be continually improving its productivity. It is not necessary to wait until the end of a project to collect data for calibration purposes. The end of each stage should produce measurements that can be used to update the historical database of projects. Waiting until the end of a project will probably add at least a year's delay to the process, whereas the need for most organisations is to collect as much data as quickly as possible.

Chapter 5

Improving the process of estimating

It is undoubtedly difficult to produce accurate estimates of the costs and timescales for systems development projects because the process is inherently imprecise and not well understood. Figure 5.1 lists the basic ingredients of a more rigorous approach to estimating, which will provide PEP members with a more realistic basis on which to make plans for the business.

Figure 5.1 Action checklist

Nominate someone to be responsible for improving estimates in the organisation.
Build up a group of part-time estimators.
Educate users to understand that estimates have a range of uncertainty, and that the uncertainty diminishes as the project progresses.
Use a range of estimating techniques, choosing ones that are appropriate for each stage of the development life cycle.
Choose a model that fits with the way in which projects are developed.
Use constraint models to assess the effects of timescale compression and staff constraints.
Consider the use of software tools to support the chosen estimating techniques and models.
Collect data about past projects.
Use this data for estimating by analogy, and for calibrating models and the tools that support them.

As organisations come to rely more and more on systems to support their business, it is imperative that the systems development department and its customers work together to produce reliable estimates, and to set achievable and cost-effective targets for systems development. The guidelines in this paper should provide the basis for better performance by the systems development department, and more realistic expectations on the part of customers, thereby avoiding the 'blame culture' that prevails in many organisations.

BUTLER COX

P,E,P

Butler Cox

Butler Cox is an independent, international consulting company specialising in areas relating to information technology.

The company offers a unique blend of high-level commercial perspective and in-depth technical expertise, a capability which in recent years has been put to the service of many of the world's largest and most successful organisations.

Butler Cox provides a range of consulting services both to organisations that are major users of information technology and to suppliers of information technology products.

Consulting for Users
Supporting clients in establishing the right opportunities for the use of information technology, selecting appropriate equipment and software, and managing its introduction and development.

Consulting for Suppliers
Supporting major information technology and telecommunications suppliers in assessing opportunities, formulating market strategies, and completing acquisitions and mergers.

Foundation
The Foundation is a service for senior managers responsible for information management in major enterprises. It provides insights and guidance to help them to manage information systems and technology more effectively for the benefit of their organisation.

Education
The Cranfield IT Institute, a wholly owned subsidiary of the Butler Cox Group, educates systems specialists, IT managers, line managers, and professionals to understand more fully how to apply and use today's technology.

P E P

The Butler Cox Productivity Enhancement Programme (PEP) is a participative service whose goal is to improve productivity in application systems development.

It provides practical help to systems development managers and identifies the specific problems that prevent them from using their development resources effectively. At the same time, the programme keeps these managers abreast of the latest thinking and experience of experts and practitioners in the field.

The programme consists of individual guidance for each subscriber in the form of a productivity assessment, and also publications and forum meetings common to all subscribers.

Productivity Assessment
Each subscribing organisation receives a confidential management assessment of its systems development productivity. The assessment is based on a comparison of key development data from selected subscriber projects against a large comprehensive database. It is presented in a detailed report and subscribers are briefed at a meeting with Butler Cox specialists.

Meetings
Each quarterly PEP forum meeting focuses on the issues highlighted in the previous PEP Paper. The meetings give participants the opportunity to discuss the topic in detail and to exchange views with managers from other member organisations.

PEP Papers
Four PEP Papers are produced each year. They concentrate on specific aspects of system development productivity and offer practical advice based on recent research and experience. The topics are selected to reflect the concerns of the members while maintaining a balance between management and technical issues.

Previous PEP Papers
4 Requirements Definition: The Key to System Development Productivity
5 Managing Productivity in Systems Development
6 Managing Contemporary System Development Methods
7 Influence on Productivity of Staff Personality and Team Working
8 Managing Software Maintenance
9 Quality Assurance in Systems Development
10 Making Effective Use of Modern Development Tools
11 Organising the Systems Development Department
12 Trends in Systems Development Among PEP Members
13 Software Testing
14 Software Quality Measurement
15 Application Packages
16 Project Estimating

Forthcoming PEP Papers
Leading and Motivating Development Teams
Managing Small Projects
Involving Users in Systems Development
The Impact of CASE

Butler Cox plc
Butler Cox House, 12 Bloomsbury Square,
London WC1A 2LL, England
☎ (071) 831 0101, Telex 8813717 BUTCOX G
Fax (071) 831 6250

Belgium and the Netherlands
Butler Cox Benelux bv
Prins Hendriklaan 52,
1075 BE Amsterdam, The Netherlands
☎ (020) 75 51 11, Fax (020) 75 53 31

France
Butler Cox SARL
Tour Akzo, 164 Rue Ambroise Croizat,
93204 St Denis-Cédex 1, France
☎ (1) 48.20.61.64, Télécopieur (1) 48.20.72.58

Germany (FR), Austria, and Switzerland
Butler Cox GmbH
Richard-Wagner-Str. 13, 8000 München 2, Germany
☎ (089) 5 23 40 01, Fax (089) 5 23 35 15

Australia and New Zealand
Mr J Cooper
Butler Cox Foundation
Level 10, 70 Pitt Street, Sydney, NSW 2000, Australia
☎ (02) 223 6922, Fax (02) 223 6997

Finland
TT-Innovation Oy
Meritullinkatu 33, SF-00170 Helsinki, Finland
☎ (90) 135 1533, Fax (90) 135 2985

Ireland
SD Consulting
72 Merrion Square, Dublin 2, Ireland
☎ (01) 766088/762501, Telex 31077 EI,
Fax (01) 767945

Italy
RSO Futura Srl
Via Leopardi 1, 20123 Milano, Italy
☎ (02) 720 00 583, Fax (02) 806 800

Scandinavia
Butler Cox Foundation Scandinavia AB
Jungfrudansen 21, Box 4040, 171 04 Solna, Sweden
☎ (08) 730 03 00, Fax (08) 730 15 67

Spain and Portugal
T Network SA
Núñez Morgado 3-6°b, 28036 Madrid, Spain
☎ (91) 733 9866, Fax (91) 733 9910