Trends in Systems Development Among PEP Members

BUTLER COX
P,E,P

PEP Paper 12, December 1989

BUTLER COX

P,E,P

Trends in Systems Development Among PEP Members

PEP Paper 12, December 1989

Chris Woodward

Chris Woodward

Chris Woodward is a senior consultant with Butler Cox in London, with 20 years of experience in the management of information systems. He is responsible for running the productivity assessment service provided as part of PEP. He was the author of PEP Paper 7, *Influence on Productivity of Staff Personality and Team Working,* and of a report entitled *Managing the Human Aspects of Change,* published by the Butler Cox Foundation.

During his time with Butler Cox, he has also carried out a wide range of consulting assignments. Recent projects in which he has been involved include technology and strategy studies in the field of office systems, organisation studies, investigations leading to the selection of hardware and software, and work on the human aspects of systems development.

Prior to joining Butler Cox as one of the company's founders in 1977, Chris Woodward worked for four years with Citibank as a project manager for the European management services function. Earlier, with a subsidiary of Joseph Lucas, he was responsible for development projects. His early career was as a systems engineer with ICL.

He has a BSc honours degree in electrical engineering from Imperial College, London University.

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

Trends in Systems Development Among PEP Members

PEP Paper 12, December 1989

Chris Woodward

Contents

Chapter 1

The analysis of the database of PEP projects

This report contains the results of an extensive analysis of the project data held in the PEP database, carried out in the summer of 1989. At the time of our analysis, the PEP database contained details of 344 sizeable systems development projects, accumulated over a three-year period since the inception of PEP in 1986. The data relates to the projects submitted by PEP members for assessment, which is an important component of the PEP service.

The purpose of the analysis was to identify trends in, and correlations between, the various measures used to describe the projects. The results are presented in a way that will enable individual PEP members to benefit from the collective experience of all members. No single PEP member has submitted data for more than 16 projects. This report provides an overview based on analysing over 20 times that number.

The analysis has been very revealing, and we believe that PEP members will be surprised at some of the results. We should stress two important qualifications, however. First, given the large number of variables, the data is not always statistically significant. Second, it has not always been possible to separate cause from effect in the correlations.

MANAGEMENT ACTION CHECKLIST

The main findings of our analysis of the project data stored in the PEP database are summarised in Figure 1.1, overleaf. It shows the effect on productivity, as measured by the Productivity Index (PI), and on system quality, as measured by software error rates, of key variables such as project size, type of development environment, whether methods and techniques are used, and the working environment. The details of the analyses that led to these findings are set out in the later chapters. To provide a context for the findings, we present the following checklist of the management actions that are suggested by the results of our analysis. We believe that taking these actions will help PEP members to become more effective in managing the systems development process. (The checklist introduces certain terms — such as language gearing and MBI — that are explained later in this chapter.)

KEEP PROJECTS SMALL, DESPITE THE PRODUCTIVITY PENALTY

The average size of projects added to the PEP database has been decreasing, while average manning levels have remained steady. The decreasing size of projects brings some crucial benefits, such as reduced development time, reduced staff turnover, fewer requirements changes, and less risk of overrunning time or effort estimates. However, smaller projects have lower PIs. PEP members should therefore ensure that smaller projects are manned at appropriate levels and that the techniques and methods used are appropriate for the size of project.

Figure 1.1 Summary of main findings

The table identifies the systems development variables that are associated with high and low PIs and software error rates.*

Systems development variable	High PI (+) Low PI (–)	Low error rate (+) High error rate (–)
Project characteristics		
Decreasing project size	–	+ / – –
New development projects	+	
Enhancement/maintenance projects	–	
Projects exceed time estimates	–	
High level of requirements changes	– –	
Development environment — hardware		
IBM 43XX	– –	– – / –
IBM 309X		+ + / + +
IBM compatibles		+ + / + +
ICL mainframes	– –	– – / – –
Other mainframes	+ +	– / +
Minicomputers	–	– – / –
PCs	+ +	+ / –
Development environment – software		
Fourth-generation languages	+	
Screen painters	+ +	– / +
Report writers	– –	– – / – –
Enquiry generators	– –	+ / –
Data dictionaries	– –	– / –
DL/1	–	+ / + +
Techniques		
Walkthroughs	+	+ / +
Structured analysis	–	+ / –
Structured design	–	– / –
Structured programming	+	
Data analysis	–	– / –
Formal development methods	–	– / – –
Tools		
Analyst workbenches	–	– / + +
Programmer workbenches	+	– – / – –
Testing tools	–	– / –
Staff factors		
High turnover	– –	
High levels of interruptions/noise		– /
Inadequate man-management	–	– /
Concerns about team structure		– /
Lack of recognition		– /
Experience with similar system	+	
Experience with similar computer	+	
Experience with using methods	– –	
Experience with using software aids	–	
Experience of project managers	– –	

*Two error rates are indicated. The first relates to errors at integration and system testing; the second relates to errors in the first month of operation.

The software error rates for small projects are higher than the average, particularly for maintenance projects. PEP members should ensure that staff are trained to understand and apply practices that encourage high technical quality. (Quality assurance in systems development was the subject of PEP Paper 9.)

ADOPT TECHNIQUES, METHODS, AND TOOLS THAT WILL MAKE ENHANCEMENT AND MAINTENANCE EFFORT MORE PRODUCTIVE

The PIs for enhancement projects are generally lower than those for other projects. Yet there are some PEP members, particularly those who use programmer workbenches, who are achieving high PIs for enhancement projects. PEP members need to examine their practices and standards carefully, particularly in terms of their effect on overall life-cycle development and maintenance costs, with the aim of identifying actions that will reduce these costs. This means that productivity must be measured throughout the whole systems life cycle, so that better decisions can be made about when to enhance and maintain existing systems and when to redevelop them.

MAKE MAXIMUM USE OF PERSONAL COMPUTERS

PC projects, although few in number in the PEP database, have very high PIs as well as high language gearing (which means that fewer lines of code are required to achieve a given purpose). On the other hand, some mainframe environments (ICL in particular) are associated with lower-than-average language gearing. PEP members should therefore ensure that their technical architectures and strategies encourage the use of PCs, particularly for cooperative-processing applications.

BEWARE OF PRODUCTIVITY AND QUALITY PROBLEMS ARISING FROM MISUSE OF TECHNIQUES AND TOOLS

The use of modern structured techniques and modern tools is often associated with projects with lower-than-average PIs and higher-than-average error rates. PEP members should ensure that the techniques and tools used are well matched to the needs of developers, that staff are properly trained and supported in their application, and that the techniques and tools are not a substitute for human endeavour.

BE PARTICULARLY CAREFUL ABOUT QUALITY PROBLEMS IN MINICOMPUTER ENVIRONMENTS

Minicomputer applications appear to have higher-than-average error rates. PEP members undertaking minicomputer developments should be particularly careful to ensure that good quality practices are applied to these environments.

PERSIST WITH FOURTH-GENERATION LANGUAGES

Some projects developed with fourth-generation languages have high PIs and high rates of delivering functionality (measured as function points per man-month). This is particularly evident where these languages make use of screen-painting aids. PEP members should make maximum use of such languages, consistent with their circumstances.

The extent to which fourth-generation languages are used has not changed significantly over the period covered by the analysis. Although PEP members are making good use of them on new developments, the need to maintain existing third-generation-language applications offsets the productivity gains obtained by using fourth-generation languages. PEP members should therefore maximise the opportunities to redevelop applications, or parts of them, in new languages, particularly code-generating languages.

MAKE USE OF STAFF WITH EXPERIENCE OF SIMILAR SYSTEMS

Higher PIs are achieved when the staff (in-house or external) working on a project have experience of similar systems. However, there is also a need to motivate staff by providing them with opportunities to work in new development areas. One way to balance these conflicting pressures is for systems development managers to use staff with relevant experience to review the work of the project team at critical stages in the project.

ADOPT TECHNIQUES THAT IMPROVE ESTIMATING ACCURACY

Most projects continue to exceed time and effort estimates by relatively large amounts. Estimates of the effort required are exceeded for two main reasons — underestimating the size of the project, and overestimating the team's ability to develop the required system. PEP members should adopt techniques such as function-point analysis to help size applications in the early stages of development. Tools should be acquired or developed to obtain more precision and consistency in estimating, and in calibrating estimating performance.

MAKE FULL USE OF WALKTHROUGHS AND INSPECTIONS

Error rates are significantly reduced when formal walkthroughs and inspections are used. PEP members who are not using these techniques should apply them wherever possible.

AVOID RAPID MANPOWER BUILDUP

The consequences, in terms of increased effort and cost, of raising the Manpower Buildup Index (MBI) have been noted in previous PEP papers. High MBIs mean increasing the manpower effort while shortening the timescale of a project. However, there is a need to balance the effort and the elapsed time required to complete a project. As much as 50 per cent of development effort can be squandered by adding more staff in order to reduce the elapsed time. Undertaking projects on tighter-than-necessary timescales merely adds to the total manpower effort needed. Both systems development managers and user managers should understand the implications of time pressure on effort and cost, and plan more cost-effective timescales.

AVOID NEEDLESS INTERRUPTION, HIGH NOISE LEVELS, AND INADEQUATE MAN MANAGEMENT

To ensure below-average error rates, PEP members should create working environments that enable staff to concentrate on their

work without needless interruption by others and by high noise levels. This is particularly important with larger projects — above, say, 40,000 lines of code.

There is also strong evidence that project productivity and quality are adversely affected by project managers who have inadequate man-management skills. PEP members should therefore seek to increase the sensitivity of their project managers to 'people' factors and to improve their ability to deal with them. Many of the existing systems staff may lack the personality characteristics that would make them good man-managers. PEP members should rectify this situation through recruiting and training.

FOCUS MANAGEMENT ATTENTION ON REDUCING REQUIREMENTS CHANGES AND STAFF TURNOVER

Not surprisingly, high levels of requirements changes are associated with decreasing PIs. PEP members should therefore manage the change process effectively, and adopt techniques and tools that will make it easier to implement the changes.

High levels of staff turnover are also associated with decreasing PI. Staff turnover can be reduced by providing staff with opportunities to work in different development environments. PEP members should also avoid lengthy projects — those with a main-build stage of nine months or more. They generally suffer high rates of staff turnover.

BASIS OF THE FINDINGS

Projects have to meet certain criteria to be selected for analysis

Our findings are based on a detailed analysis of the data relating to 344 sizeable systems development projects stored in the PEP database. To be included in the analysis, each project had to meet certain criteria. The criteria are a minimum of 4,000 source lines of code, peak manning of at least two in the main-build stage (to ensure that they were team, as opposed to individual, undertakings), a main-build time of at least four months, and a main-build effort of at least 10 man-months. The mid-point of the main-build stage also had to fall between the first quarter of 1985 and the last quarter of 1988. However, the trends over time are based on the subset of projects whose main-build mid-point fell between the second quarter of 1986 and the first quarter of 1988. About two-thirds of the projects (219 in total) fell into this category, and we believe that they provide a more reliable basis for trends over time than the complete set of 344 projects.

The source of the projects reflects the PEP membership — about 85 per cent were provided by UK organisations, and the remainder by Benelux organisations. In total, they span seven main industry sectors — insurance, engineering, banking, utilities, government, food production, and retailing. Most of the non-UK projects were provided by the engineering and banking sectors, so the analyses of those sectors reflect performance on a multinational scale.

Most of the projects are developments of new systems, but some are enhancements — often major ones, involving a high proportion of new code — to existing projects.

For each project, we have collected and stored in the database two types of data: *core* and *supplementary*:

— The core data consists of project size, amount of new code, elapsed time, effort, manning levels, time overlap between the functional-design and main-build stages, cost, error-rates, time overrun, and effort overrun. Most of this data relates to the main-build stage of the systems life cycle, although some of it is also collected from the other three stages of the life cycle, as shown in Figure 1.2.

— The supplementary data includes the nature of the techniques and tools used, an estimate of the percentage change in requirements that occurs once the main-build stage has started, information about the staff engaged on the project, such as their level of expertise, and data about the working environment. We have been unable to collect all types of supplementary data as completely as the core data, so that analysis of some of this data is less reliable.

We have used the core and supplementary data to analyse the projects in terms of six key variables:

Six key variables have been analysed

— Size, usually measured in thousands of lines of code.

— Timing of the project, in terms of the quarter during which the mid-point of the main-build stage was reached.

Figure 1.2 Key project data

Core data is collected for PEP project assessments at each of four stages in the life cycle.

	Unit of measure	Stage 1: Feasibility study	Stage 2: Functional design	Stage 3: Main build	Stage 4: Operations and maintenance
Project size	Lines of code			✓	
New code	Lines of code			✓	
Time	Months	✓	✓	✓	✓
Effort	Man-months	✓	✓	✓	✓
Peak manning	Number of staff		✓	✓	
Time overlap	Months		✓(1)	✓(1)	
Cost	Dollars			✓	
Software errors	Number of errors			✓	✓
Time overrun	Months			✓	
Effort overrun	Man-months			✓	
Project constraints	(2)			✓	

(1) Time overlap refers to the period of overlap between the functional-design and main-build stages.

(2) Project constraints specify any time, cost, and peak-manning constraints imposed on the project.

- The programming language used.
- The computer environment used.
- The types of techniques and tools used.
- The industry sector of the PEP member.

The PEP project database forms a subset of a larger database, PADS (Productivity Analysis Database System). PADS, which is proprietary to US-based Quantitative Software Management (QSM) Inc, defines the information to be collected in the database, and provides a supporting set of software, key measures, and assessment procedures.

The PADS database has been in use for longer than the PEP database, and is therefore larger, containing details of about 1,600 projects that are currently used for deriving trends. One thousand of these are business applications. The PEP projects are, however, mainly European in origin and are, on average, more recent than PADS projects. The data about individual PEP projects is also more extensive than the PADS project data.

PEP and PADS projects have been compared, where appropriate

Nonetheless, it can be useful to compare the trends in PEP and PADS projects. A convenient way of depicting trends is by means of the least-squares best-fit trend line that can be drawn through the data when it is plotted against project size, together with the parallel lines representing plus one and minus one standard deviation from the best-fit line. For the PADS data, we refer to these lines as PADS trend lines.

KEY MEASURES

There are two key measures used in analysing PEP projects — the Productivity Index (PI) and the Manpower Buildup Index (MBI). Both are calculated (by the PADS software) from three project parameters — size, elapsed time, and manpower effort (the latter two relating specifically to the main-build stage). There are three further measures of significance — language gearing, function-point delivery rate, and software error rate.

PRODUCTIVITY INDEX (PI)

The PI of a project is a measure of the productivity achieved at the main-build stage by the development team in producing applications. It is not a measure of the value or functionality delivered to the business by the application.

The PI is calculated from a project's size, manpower effort, and elapsed time by using an empirical formula developed by QSM Inc. The formula, known as the software equation, generates a Productivity Measure, PM. In practice, the value of PM ranges widely between projects, typically from around 3,000 to 50,000 and more.

The PI is used in preference to the PM simply to make the range of numbers less unwieldy. The two are related in a non-linear way, so that a PM range of 3,000 to 240,000 is converted to PI values ranging from about 7 to 25 (see Figure 1.3, overleaf).

The average PEP project has a PI of about 15

The average PI of PEP projects in 1988 was about 15. PI values below 15 imply lower-than-average productivity; above 15, they

Figure 1.3 Productivity Index (PI)

The PI is a measure of a project team's efficiency. It is derived from an empirical formula (QSM's 'software equation'), which defines a parameter called PM, the Productivity Measure:

$$PM = \frac{Size}{(Effort/B)^{1/3} \times (Time)^{4/3}}$$

Where:

- Size is the number of source statements.
- Effort is in man-years.
- Time is the duration of the main-build stage in years.
- B is a staff skills factor that takes account of the point in the systems life cycle at which peak manning occurs. It varies with project size, from 0.16 for small projects of around 5,000 lines of code, to 0.39 for projects exceeding 70,000 lines of code.

The PI is derived from the PM, using the following conversion table:

PM	PI	PM	PI
754	1	17,711	14
987	2	21,892	15
1,220	3	28,657	16
1,597	4	35,422	17
1,974	5	46,368	18
2,584	6	57,314	19
3,194	7	75,025	20
4,181	8	92,736	21
5,168	9	121,393	22
6,765	10	150,050	23
8,362	11	196,418	24
10,946	12	242,786	25
13,530	13		

(Source: QSM Inc)

imply higher-than-average productivity. It is important to note that, because of its non-linear nature, small changes in PI value imply big shifts in team performance.

Consider, for example, a typical PEP project of 40,000 lines of code, with the main-build stage taking 10 months. At a PI of 15, the effort works out to be 60 man-months. At a PI of 14, the project takes a month longer and the manpower effort rises by 30 per cent. At a PI of 16, the project takes a month less and manpower effort drops by about 30 per cent. Thus, a one point movement in PI from around the average of 15 represents a productivity change of about 30 per cent.

MANPOWER BUILDUP INDEX (MBI)

MBI measures the effect of project time compression or expansion

The software equation also takes account of the effects of compressing or extending the project timescale. When the timescale is compressed, the total manpower effort is increased substantially. This happens because the timescale is often compressed by carrying out, concurrently, tasks that would usually be done sequentially. In turn, this means that more staff are working on the project at any one time, which means that there are more paths of communication between team members, more opportunities for errors to arise and to remain undetected, and a greater management overhead.

The effect of time compression (and expansion) is represented by a measure called the Manpower Buildup Index (MBI). As with the PI, the MBI is expressed as a simple integer value, or level, ranging

between one and six (see Figure 1.4). Level 1 represents a slow staff buildup. Projects with an MBI of one take the longest, but require the least effort. Usually, low MBI values are associated with projects that are subject to staffing constraints. Level 6 represents the opposite end of the spectrum — the 'throw people at it' approach. On projects of this type, many tasks are carried out concurrently, with virtually no constraints on money or the number of staff. For a given size and PI, projects with an MBI of 6 usually take the shortest time to develop, but require the most manpower effort.

In general, MBI values from one to three indicate below-average rates of manpower buildup; values of between four and six indicate above-average rates.

Consider, again, a typical PEP project of 40,000 lines of code and a PI of 15. An MBI value of three leads to a main-build duration of 10 months and effort of 60 man-months. Reducing the MBI to one means extending the duration to 12 months, but effort falls to only 25 man-months. Raising the MBI to five saves time by reducing the duration to eight months, but the effort nearly doubles to 115 man-months.

Low MBI values reduce project manpower effort and increase project duration. The disadvantage of projects with low MBIs is that the extended timescales mean that there is a greater chance of the requirements changing before a project is completed, and that it is often more difficult to keep staff motivated.

The MBI measure can be used by systems development managers, when they are planning projects, to assess whether a project can realistically be completed in a given time. High MBI values identified at the planning stage point to potential problems and high risks. A few systems development departments can consistently achieve above-average PIs under considerable time pressures, but they are a small minority.

Figure 1.4 Manpower Buildup Index (MBI)

The MBI is a measure of manpower buildup. It is derived from an empirical formula defining a parameter called MM, the Manpower Buildup Measure:

$$MM = \frac{Effort}{B \times Time^3}$$

Where:

- Effort is in man-years.
- Time is the duration of the main-build stage in years.
- B is the same staff skills factor as for the PI.

The MBI is derived from the MM, using the following conversion table:

MM	MBI
7.3	1
14.7	2
26.9	3
55.0	4
89.0	5
233.0	6

(Source: QSM Inc)

THREE FURTHER MEASURES OF SIGNIFICANCE

The PI measures the efficiency of a project team. It is an important measure for systems development managers interested in assessing the internal efficiency of their departments. A second, equally important, measure is departmental *effectiveness*, which is concerned with the *functionality* delivered to the business, per unit of effort.

Internal efficiency is analogous to the fitness of a cyclist, which determines the effort that will be put into pushing the pedals. What really matters, however, is the distance the cycle travels for the effort that is put in, and this is determined by the gears on the cycle. The cyclist may not be at peak fitness, but a high gear will enable him to travel, say, 10 times the distance for a given effort. High-level languages are analogous to high gears; the higher the *language gearing*, the fewer the number of lines of code that will be required to produce a given level of system functionality.

This does not mean, however, that programming languages with the highest language gearing should always be used. Just as trying to cycle uphill in an inappropriate high gear will result in significantly slower speed, so a failure to match the language to the application can result in significantly more effort being used.

Language gearing is the number of function points per thousand lines of code

The best known unit of measure of system functionality is the function point. We define language gearing as the number of function points generated, on average, by 1,000 lines of code.

Many PEP members do not calculate the number of function points for the projects submitted to the PEP database. We have therefore estimated the function-point count for a project by multiplying the number of thousands of lines of code by the appropriate language gearing, using the values shown in Figure 1.5. (The language-gearing values shown in the figure are derived from research carried out by Software Productivity Research, Inc of Cambridge, Massachusetts.) The average number of source lines of code needed to generate one function point works out to be 70 for all the projects in the PEP database, but the range varies widely from as few as 10 or less (high language gearing) up to 200 or more (low language gearing).

Knowing the effort required to develop a project and the language gearings for the programming languages used, it is possible to calculate the *function-point delivery rate*, expressed as function points per man-month.

Another significant indicator of the performance of a development team is the technical quality of the systems produced, measured in terms of the *software error rate*. A software error is a mistake or omission in the code, causing the code to deviate from the specification. (Software errors are therefore not caused by differences between the specification and the requirements of the business.) The software error rate is defined as the number of software errors per thousand lines of code.

The software error rate is influenced significantly by the PI and MBI of a project. High PI is often, but not always, associated with low software error rates. High MBI is often associated with high software error rates.

Figure 1.5 Language gearing

The list shows the language gearing, expressed in terms of the number of function points per thousand lines of code, for the high-level languages used in PEP projects.

Language	Language gearing	Language	Language gearing
Acumen	35	Guest	35
ADF	50	Ideal	35
ADS/Online	50	Keyplus	25
Algol	10	Lotus	100
APL	35	M204	35
Application Factory	50	Magna8	35
Application Master	35	Mantis	70
APS	60	Mapper	18
Artemis	33	Mark IV	25
Ask	30	MFS	25
Basic (Compiled)	13	Natural	18
C	8	Nomad	25
CA-Earl	35	Pascal	11
CBAS	13	PL/1	13
CLI	25	PLDS	30
Clipper	25	PPL	25
Cobol	10	QMF	70
CSP	35	Quickbuild	35
Culprit	65	Quiz	70
Data	25	Rally	35
Dataflex	25	Ramis	25
Datatrieve	50	Rapidgen	35
dBASE	30	RDB	25
DCL	6	RPG	17
DDL	35	SAS	30
EAL	35	SIR	35
Easytrieve	65	SQL	70
Enform	50	Sybol	14
FCS	25	Telon	70
Filetab	17	TIG	10
FMS	20	Transact	35
Focus	25	UFO	30
Fortran	10	Whip	10
Gener/ol	70	Wizard	35

(Source: Software Productivity Research, Inc)

On average, 85 per cent of software errors are corrected before the start of integration and system testing, and 95 per cent are corrected by the point of first operational capability. The software error rates both at integration and system testing and in the first month of live operation are therefore strong indicators of technical quality. Below-average error rates indicate good technical quality; above-average rates indicate poor technical quality.

STRUCTURE OF THE REPORT

To simplify the presentation of the material, most of the chapters in this report focus on project productivity, as measured by the PI, and technical quality, as measured by software error rates.

In Chapter 2, we look first at the size characteristics of projects, then at the impact of size on productivity and technical quality. We pay particular attention to smaller projects because of their relatively low PIs and high software error rates. This poorer performance is largely due to the fact that a higher proportion of smaller projects is concerned with maintenance and enhancement.

We turn our attention in Chapter 3 to the impact of project timescale on productivity and quality, and analyse the extent to which projects are overrunning time and effort estimates. PEP members are probably incurring 50 per cent more effort than is strictly necessary because of unnecessarily high rates of manpower buildup. Poor estimating is commonplace (because of the inability to size applications accurately).

In Chapter 4, we examine the effect on productivity and quality of the development environment, particularly the type of computer and programming language used. PC projects appear to have significantly higher PIs than other types of projects, yet the low number of PC projects suggests that many PEP members are failing to capitalise on them.

Chapter 5 is concerned with whether particular types of techniques and tools have a significant impact on productivity and quality. In general, they do not appear to have a significant impact, and there is evidence to suggest that their use may even adversely affect productivity and quality. One or two types, however, do seem to be providing some distinct benefits. This applies particularly to projects using formal walkthroughs, which have significantly lower errors rates and marginally better PIs than those that do not use this technique.

In Chapter 6, we discuss the impacts of the working environment and management behaviour. Staff are increasingly sensitive to these factors as projects increase in size. Deteriorating staff attitudes appear to be more directly related to an increase in software errors than to falling PIs.

Finally, in Chapter 7, we examine the productivity and quality being achieved according to the industry sector in which PEP members are operating. The analysis is tentative, because the number of PEP members and the number of projects in some sectors is relatively small. Nevertheless, this analysis confirms some expectations, such as the severe time pressures experienced by projects in the insurance industry.

Chapter 2

The effect of project size on productivity and quality

Projects recorded in the PEP database are mainly developments of new systems, and the trend is for projects to get smaller. At first sight, this may appear to be an encouraging trend, because smaller projects involve less risk and lower staff turnover. However, our analysis of the PEP database shows that smaller projects have lower PIs, higher MBIs, and higher error rates.

Although, on average, the PIs of PEP projects are close to international averages, the variation is wide. Closer inspection reveals that enhancement projects, mostly developed with traditional programming languages, have lower-than-average PIs, and new systems developments, often with a significant contribution from fourth-generation languages, have better-than-average PIs.

THE TREND IS FOR PROJECTS TO GET SMALLER

The projects submitted by PEP members average 41,000 lines of code, and are mainly for the development of new systems. During the three years that the programme has been in operation, the size of PEP projects has decreased significantly, as has the proportion of new development projects.

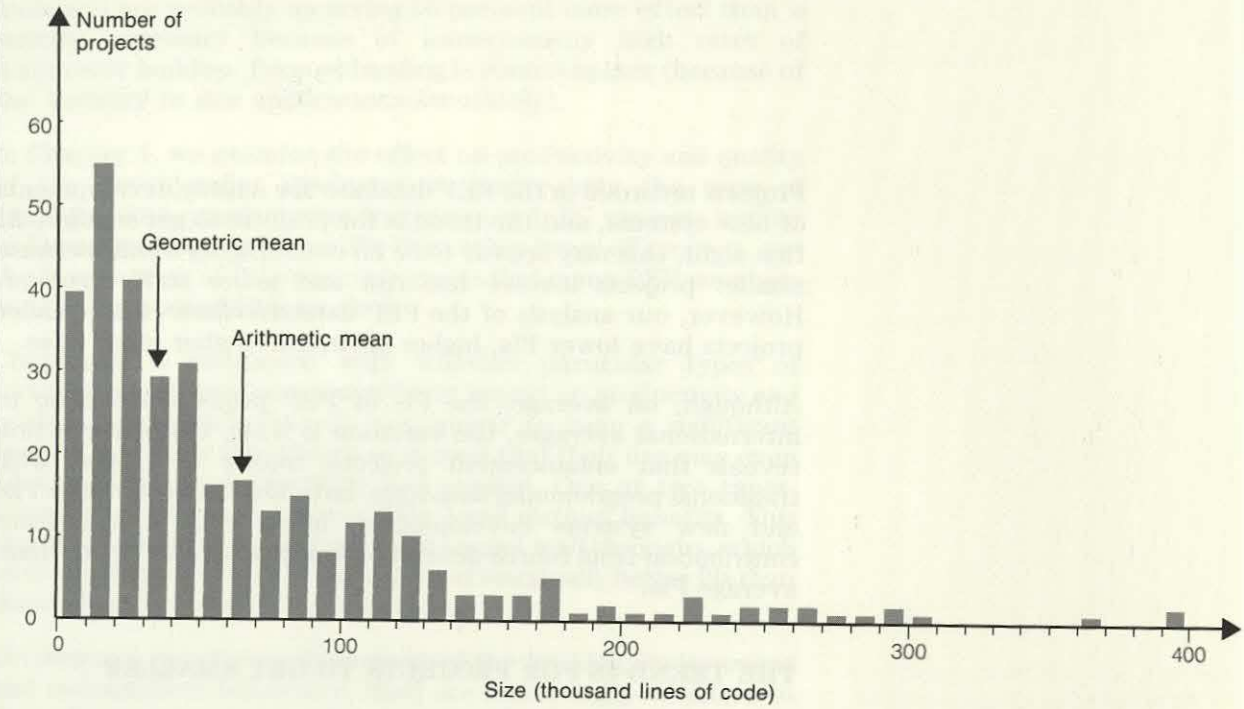
PEP PROJECTS AVERAGE 41,000 LINES OF CODE

Most PEP projects are small or medium-sized

PEP projects vary widely in size between about 5,000 and 400,000 lines of code. As Figure 2.1, overleaf, demonstrates, the distribution of project sizes is skewed significantly towards the lower end of the scale, with only a small number of very large projects. However, the criteria used to select the projects for analysis exclude projects with particular combinations of PI and MBI, particularly for projects below about 20,000 lines of code. This means that the small projects in the database are a subset of the total number of small projects undertaken by PEP members. There is also a relatively small number of large projects, above 200,000 lines of code.

The arithmetic mean of the projects is about 69,000 lines of code, and the geometric mean about 41,000 lines of code. (In this chart, and many other of the charts shown in this paper, we have shown both the arithmetic mean — the conventional average — and the geometric mean. The geometric mean of a population of n numbers is calculated as the nth root of the product of the n numbers. Its advantage over the arithmetic mean — which is calculated by dividing the sum of the n numbers by n — is that it is not distorted by a small number of very large projects. The geometric mean therefore gives a better indication of the average in a population highly skewed towards the lower values. Unless all the numbers in the population are equal, the geometric mean

Figure 2.1 The distribution of PEP projects, by size, is skewed significantly towards the lower end of the scale



(Source: PEP database)

will always be less than the arithmetic mean.) Measured by function points, PEP projects range in size from 20 to nearly 7,000. The geometric mean is about 600 function points and the arithmetic mean about 1,000.

THE CHARACTERISTICS OF PEP PROJECTS HAVE BEEN CHANGING

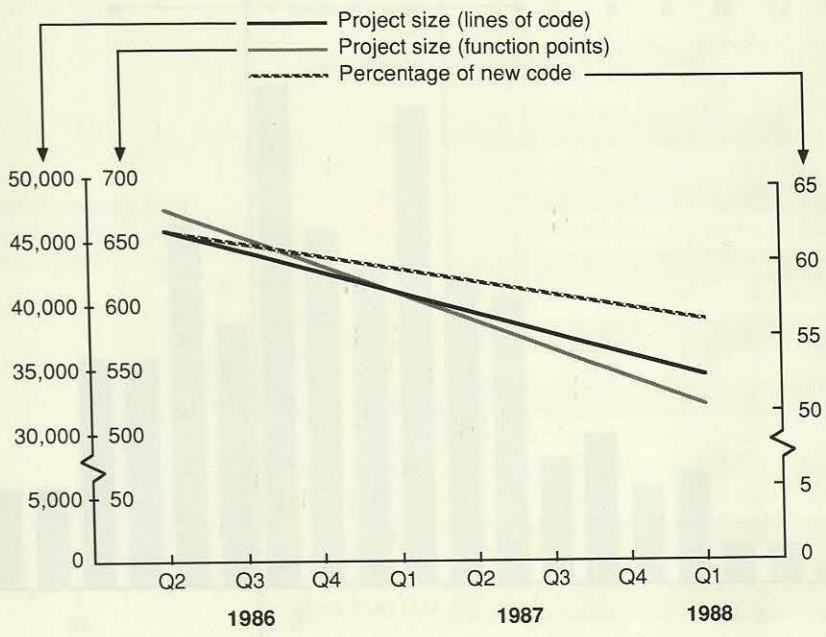
The basic characteristics of PEP projects have been changing (see Figure 2.2). Project size has decreased from about 45,000 to 35,000 lines of code over the period of our analysis. The downward trend in lines of code is closely matched by a similar trend in function points, from nearly 700 to just above 500 over the period. This indicates that the average language gearing of PEP projects has remained broadly level over the period of analysis.

The proportion of new code in projects has also decreased (although to a lesser extent than the decreases in project size), indicating that more enhancement projects have been submitted to PEP as time has gone on. Over the period of analysis, the proportion of new code has fallen from just over 60 per cent to just over 55 per cent.

SMALLER PROJECTS HAVE LOWER PIs

The variation in productivity, as measured by the Productivity Index, among PEP projects is wide. Some projects have very

Figure 2.2 Project characteristics change with time



(Source: PEP database)

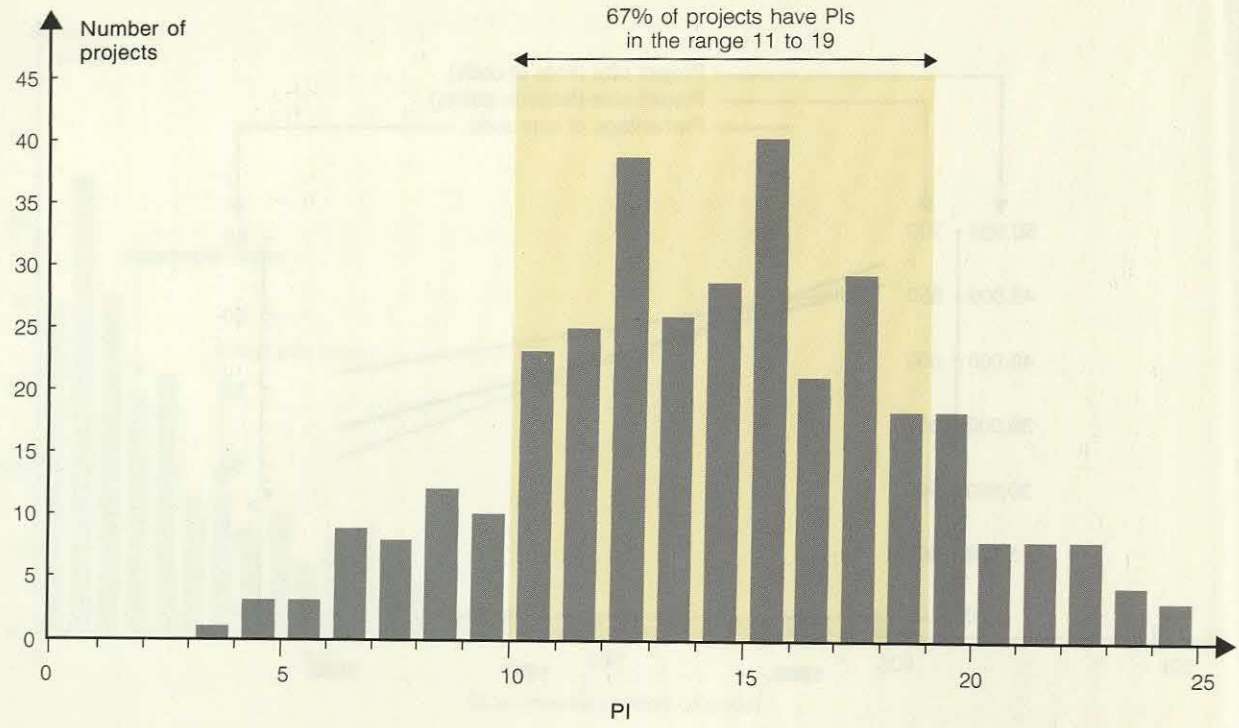
high PIs, including some where a significant proportion of the code was written in a fourth-generation language. Some projects, particularly some small enhancement projects, have very low PIs.

Two-thirds of PEP projects have PIs in the range of 11 to 19

The PIs of PEP projects range from 4 to 25 and are distributed approximately normally around a mean of 15 (see Figure 2.3, overleaf). The standard deviation is four, which means that two-thirds of the projects have PIs in the range of 11 to 19. Bearing in mind that a one-point increase in PI represents a reduction in effort of between 25 and 30 per cent, an increase in PI from 11 to 19 represents a reduction in effort of over 90 per cent.

Figure 2.3 shows that there are two primary peaks, at PIs of 13 and 16, and a secondary peak at 18. The projects with PIs of 13 average 34,000 lines of code, but in every other respect (for example, language gearing), they are close to the average. The projects with PIs of 16 are close to the overall average size for PEP projects, but have a slightly greater content of new code. They also have a significantly higher language gearing (about 60 per cent more than average) due to the greater use of fourth-generation languages, indicating that some PEP members are achieving good process productivity with such languages. (We examine the impact of languages on productivity in more detail in Chapter 4.) The projects with PIs of 18 are distinctly larger, averaging 81,000 lines of code. They are all new developments, and their language gearing is close to the average for all PEP projects.

Figure 2.3 The PIs of PEP projects are distributed approximately normally around a mean of 15



(Source: PEP database)

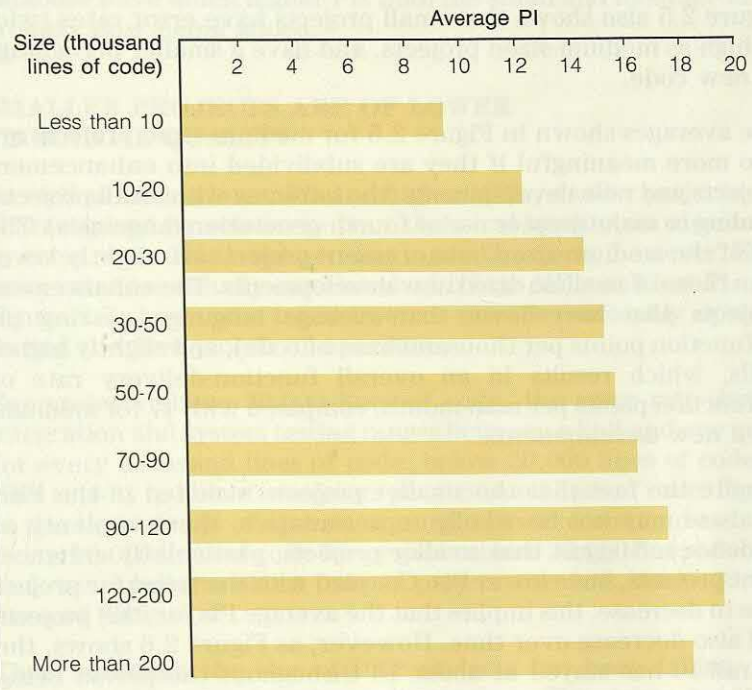
From this analysis, it appears that smaller projects have lower PIs, and this is particularly pronounced for projects below about 20,000 lines of code (see Figure 2.4). The characteristic of smaller projects having lower PIs is also demonstrated in the PADS trend lines. Taking, for example, the trend line for project size versus main-build effort for any one value of MBI, projects tend to be positioned above PADS averages at the lower end of the size scale, and below average at the higher end of the scale. With normally distributed PIs, projects would be equally distributed about the averages.

Small projects have different characteristics from other projects

Small projects, defined as those with fewer than 20,000 lines of code, differ from medium-sized and large projects in other respects, as Figure 2.5 shows. The lower PIs and higher MBIs of small projects mean that more effort is required to deliver a given amount of functionality. (These projects also use proportionally more effort in the earlier stages of feasibility study and functional design.)

However, small projects are essentially of two types — they are either small enhancements in traditional languages (and therefore have low language gearing), or new developments in fourth-generation languages (and therefore have high language gearing). This means that the average rate of delivering functionality (nine function points per man-month) hides some wide variations. The small projects with high language gearing (defined as more than twice the overall PEP average of 14) deliver function points at an average rate of 17 per man-month, which is better than the PEP average (13). Those with low language gearing (defined as

Figure 2.4 PI increases as project size increases



(Source: PEP database)

Figure 2.5 The performance measures of small, medium, and large projects differ significantly

	Small projects (fewer than 20,000 lines of code)	Medium projects (20,000 to 120,000 lines of code)	Large projects (more than 120,000 lines of code)
Average size (lines of code)	11,000	51,000	191,000
Average PI	11.2	15.6	18.8
Average MBI	3.5	2.6	2.5
Language gearing	16.0	14.0	12.0
Function points per man-month	9.0	14.0	20.0
New code (%)	43.0	62.0	73.0
Errors per thousand lines of code	1.45	0.76	0.90
First month's errors per thousand lines of code	0.32	0.15	0.10

(Source: PEP database)

less than two-thirds of the overall PEP average) deliver function points at an average rate of just four per man-month.

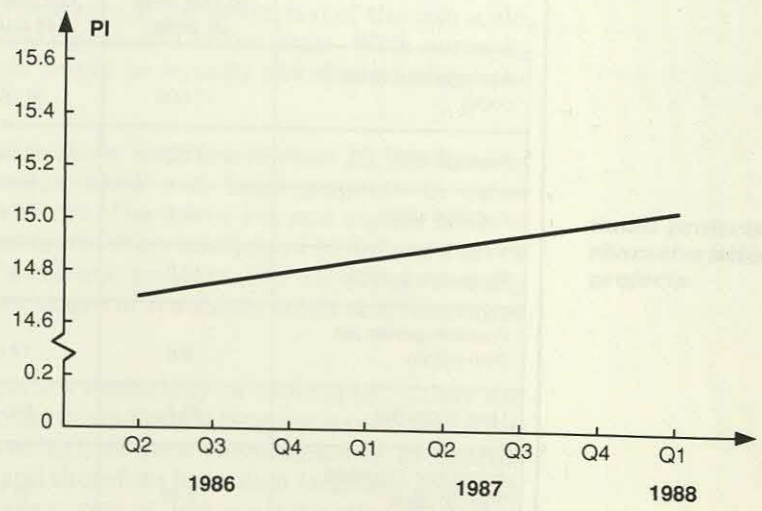
Figure 2.5 also shows that small projects have error rates twice as high as medium-sized projects, and have a smaller percentage of new code.

The averages shown in Figure 2.5 for medium-sized projects are also more meaningful if they are subdivided into enhancement projects and new developments (the latter, as with small projects, tending to make greater use of fourth-generation languages). The PIs of the medium-sized enhancement projects are slightly lower than those of medium-sized new developments. The enhancement projects also have lower-than-average language gearing (at 12 function points per thousand lines of code), and slightly higher MBIs, which results in an overall function-delivery rate of 10 function points per man-month, compared with 17 for medium-sized new developments.

Enhancement projects have poorer performance than new developments

Despite the fact that the smaller projects included in the PEP database may not be wholly representative, there is plenty of evidence to suggest that smaller projects, particularly enhancement projects, have lower PIs. Coupled with the trend for project sizes to decrease, this implies that the average PIs for PEP projects will also decrease over time. However, as Figure 2.6 shows, the overall PI has stayed at about 15 throughout the period being analysed, and further research is required to explain this apparent paradox. It could be, for example, that the smaller projects added more recently to the PEP database have higher PIs than earlier

Figure 2.6 PI is increasing only slightly and averages about 15



(Source: PEP database)

small projects. Equally, it could be explained by the fact that the decreasing proportion of large projects now being added to the database have much higher PIs than the small and medium-sized projects now being added.

SMALLER PROJECTS ARE OF LOWER TECHNICAL QUALITY

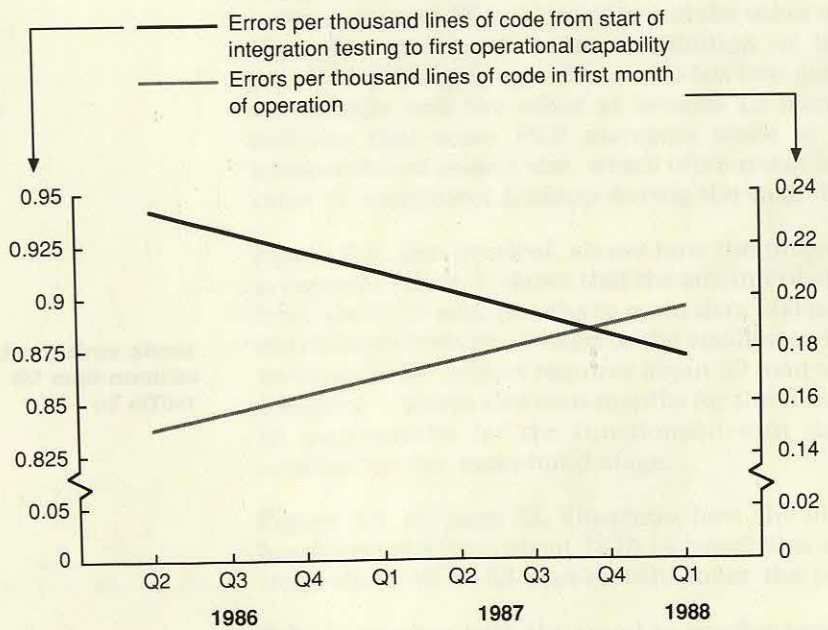
High error rates usually correlate with high MBIs and low PIs

As well as having lower PIs and higher MBIs, smaller projects are characterised by relatively poor technical quality, as measured by error rates. This is not peculiar to small projects, however. High error rates are usually correlated with high MBIs and low PIs. As Figure 2.7 shows, the trend in error rates is noticeably upward during the first month of operation.

For projects above 20,000 lines of code, the error rate during integration and system testing ranges between a half and one error for every thousand lines of code; below 20,000 lines of code, it rises rapidly to above two errors per thousand lines of code. Errors in the first month of operation range between one and two for every 10,000 lines of code for projects above 30,000 lines of code, but rise rapidly to about four as project size falls below 30,000 lines of code.

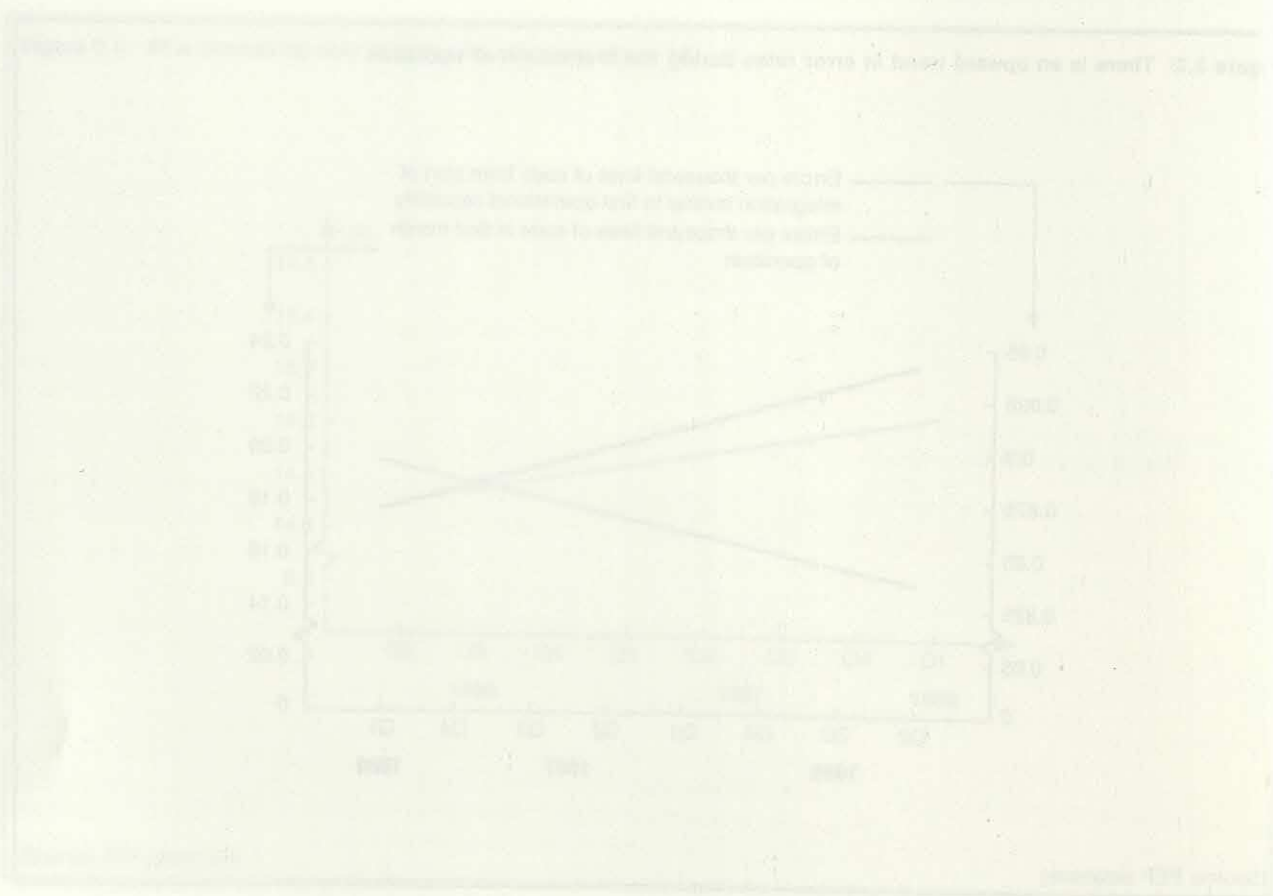
Thus, although the size of PEP projects has been falling, the number of errors has been increasing. The error rate for integration and system testing, during the period of analysis, has increased from about 0.7 to 1.2 errors per thousand lines of code.

Figure 2.7 There is an upward trend in error rates during the first month of operation



(Source: PEP database)

The number of errors in the first month of operation also shows a clear upward trend over the period, from about 0.14 to 0.25 errors per thousand lines of code, representing an increase of more than 50 per cent in the error rate and a consequent reduction in reliability. However, the trend to smaller projects also means that projects are taking less time to develop. In the next chapter, we consider the effect of timescale on productivity and quality.



Chapter 3

The effect of timescale on productivity and quality

Although the development time for PEP projects is decreasing as their size decreases, they are being subjected to more time pressure, and this is reflected in the upward trend in MBI. This trend is indicative of the increased effort required for projects, and the reduced rates of producing code and delivering functionality. Encouragingly, however, the amounts by which projects overrun time and effort estimates are decreasing, although most projects still experience some slippage.

Although some of the increase in MBI can be accounted for by the criteria used to select projects for analysis, part also seems to be associated with the way in which PEP members staff smaller projects; there appears to be a fixed level of staffing for smaller projects, which is not adjusted in line with project size.

PROJECTS ARE TAKING LESS TIME

The 'average' PEP project takes 17 months to complete ...

Three-quarters of PEP projects last two years or less, with the remaining quarter taking up to nine years to complete. Nearly a third of projects take between six months and a year to complete. As Figure 3.1, overleaf, shows, the 'average' project takes 17 months — three months for the feasibility-study stage, five months for the functional-design stage, and 10 months for the main-build stage, which overlaps the previous stage by one month. There appear to be two peaks in the distribution of project time, one centred around 12 to 14 months and the other around 21 months. This corresponds with the distribution of time taken in the functional-design stage, which also has two peaks, one at around six months and the other at around 12 months. These peaks indicate that some PEP members work to fixed timescales, irrespective of project size, which often result in higher and costly rates of manpower buildup during the main-build stage.

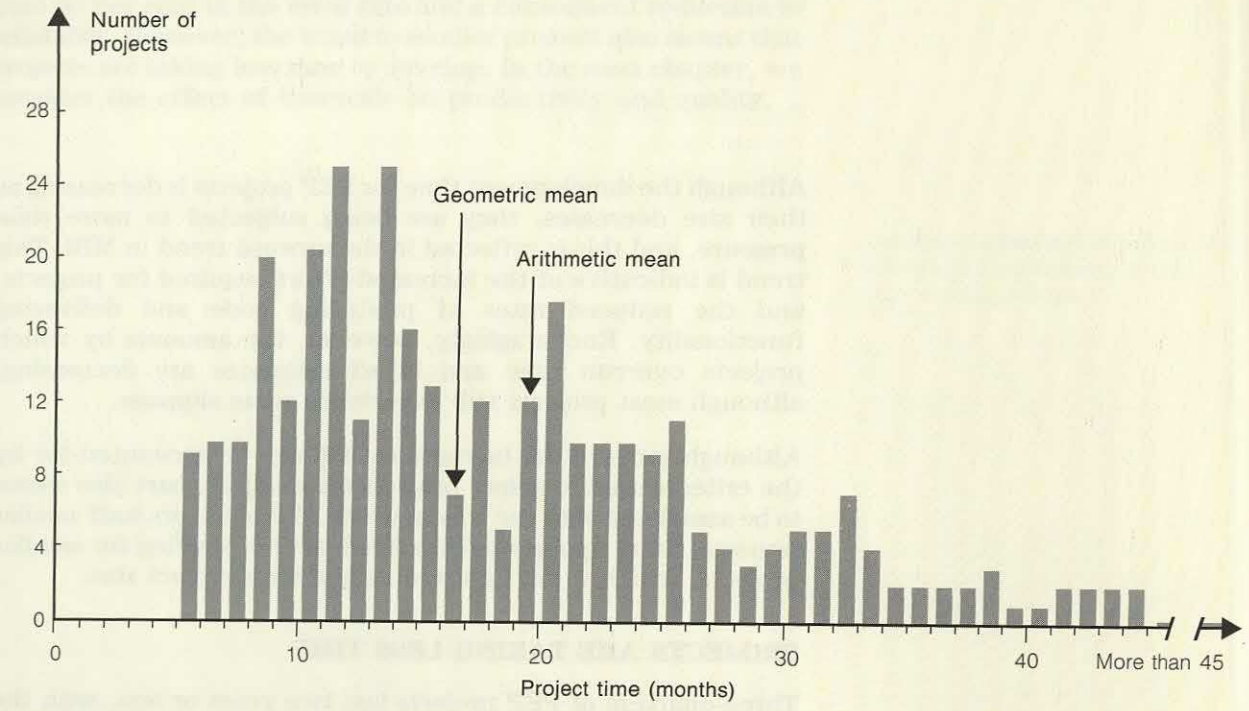
... and requires about 60 man-months of effort

Figure 3.2, also overleaf, shows how the projects are distributed in terms of effort. It shows that the amount of effort ranges widely from about 10 man-months to more than 200 man-years, with the distribution heavily skewed to the smaller end of the range. The 'average' PEP project requires about 60 man-months of effort to complete — about five man-months for the feasibility-study stage, 10 man-months for the functional-design stage, and 45 man-months for the main-build stage.

Figure 3.3, on page 23, illustrates how the duration of projects has decreased from about 18 to 14 months on average, and effort from about 58 to 53 man-months, over the period of analysis.

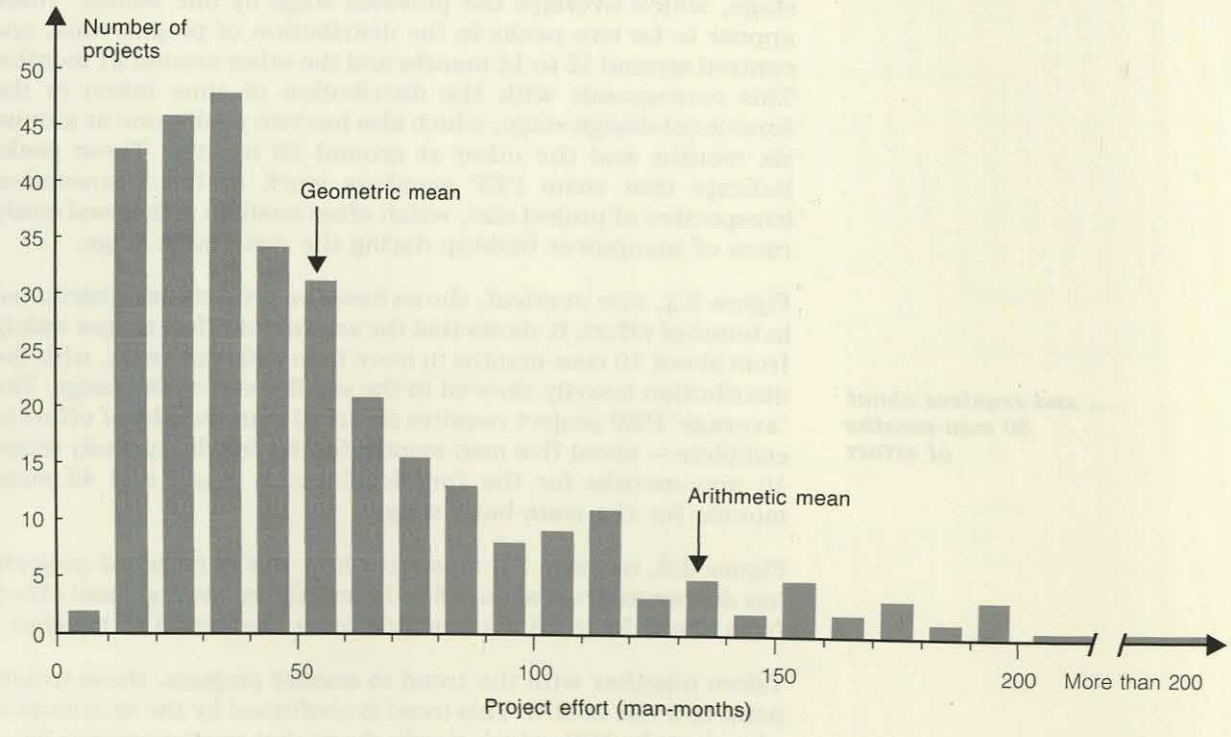
Taken together with the trend to smaller projects, these trends point to a rise in MBI. This trend is confirmed by the distribution of projects by MBI, which clearly shows that smaller projects have higher MBIs (see Figure 3.4, on page 23).

Figure 3.1 There are two peaks in the distribution of project time



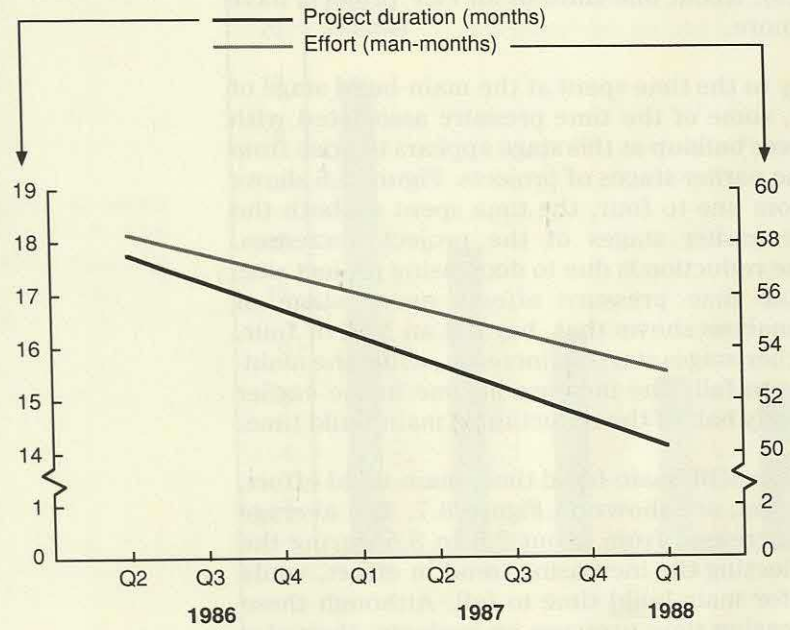
(Source: PEP database)

Figure 3.2 Project effort on PEP projects ranges from less than 10 to more than 1,000 man-months



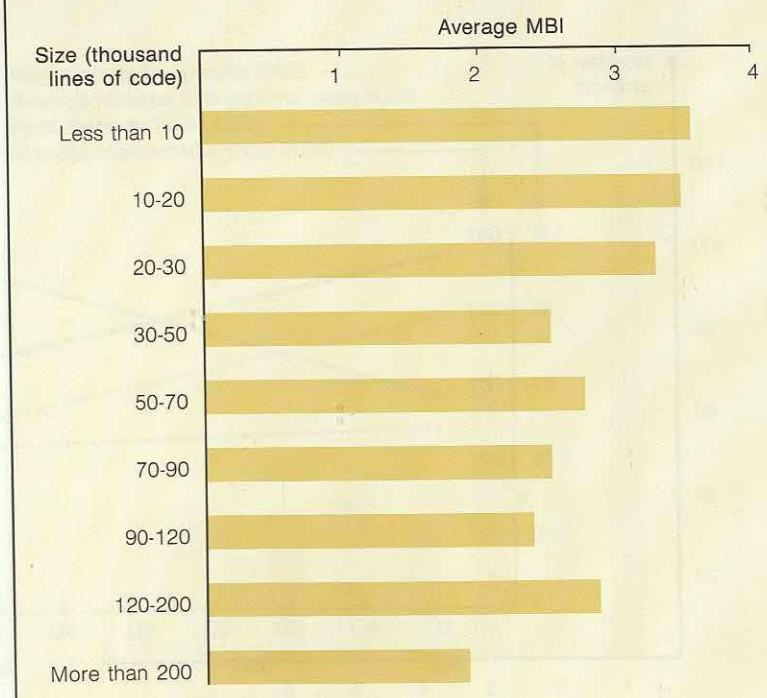
(Source: PEP database)

Figure 3.3 Duration and effort are falling with time



(Source: PEP database)

Figure 3.4 Smaller projects have higher MBIs than larger projects



(Source: PEP database)

MANPOWER BUILDUP RATES ARE INCREASING

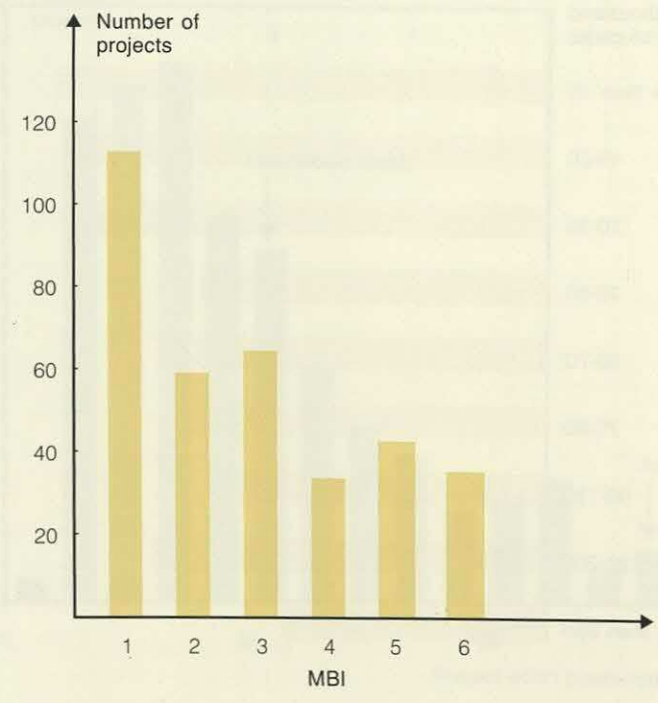
The average MBI is less than three

The distribution of MBIs for PEP projects is skewed towards the lower end of the range, with the average lying between two and three (see Figure 3.5). About one-third of all PEP projects have an MBI of four or more.

The MBI relates only to the time spent at the main-build stage of a project. However, some of the time pressure associated with fast rates of manpower buildup at this stage appears to arise from the time spent on the earlier stages of projects. Figure 3.6 shows that as MBI rises from one to four, the time spent on both the main-build and the earlier stages of the project decreases. Although some of the reduction is due to decreasing project size, it is also clear that time pressure affects every stage of development. Our analysis shows that, beyond an MBI of four, time spent in the earlier stages starts to increase, while the main-build time continues to fall. The increase in time in the earlier stages represents nearly half of the reduction in main-build time.

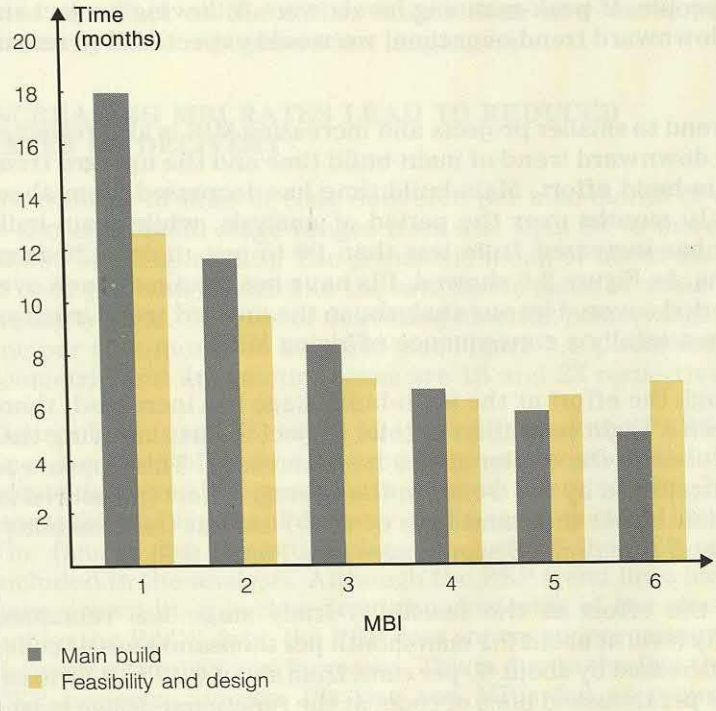
The trends over time in MBI, main-build time, main-build effort, and peak-manning level, are shown in Figure 3.7. The average MBI over time has increased from about 2.5 to 3.5 during the period analysed, reflecting the increasing trend in effort, while the trend has been for main-build time to fall. Although these trends indicate increasing time pressure on projects, they also reflect the trend to reduced project size (smaller projects have slightly higher MBIs, as illustrated in Figure 3.4).

Figure 3.5 The distribution of MBIs for PEP projects is skewed towards the lower end of the range



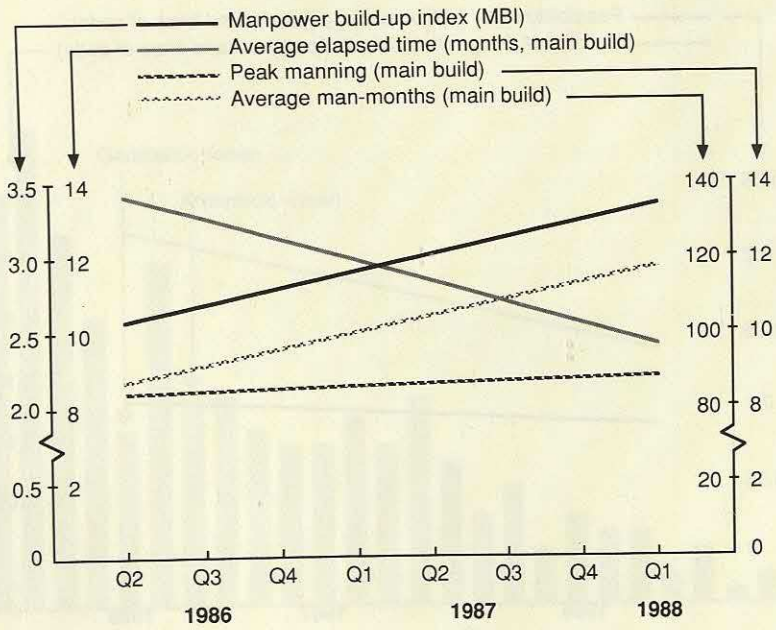
(Source: PEP database)

Figure 3.6 As MBI increases, the time spent on the main-build stage of a project decreases



(Source: PEP database)

Figure 3.7 Project time and effort trends



(Source: PEP database)

The increasing trend in MBI and decreasing trend in project size are also reflected in the trend in peak-manning level during the main-build stage. Figure 3.7 shows the average peak-manning level to have remained virtually constant at between eight and nine people. If peak-manning levels were following project size on a downward trend over time, we would expect MBI to remain level.

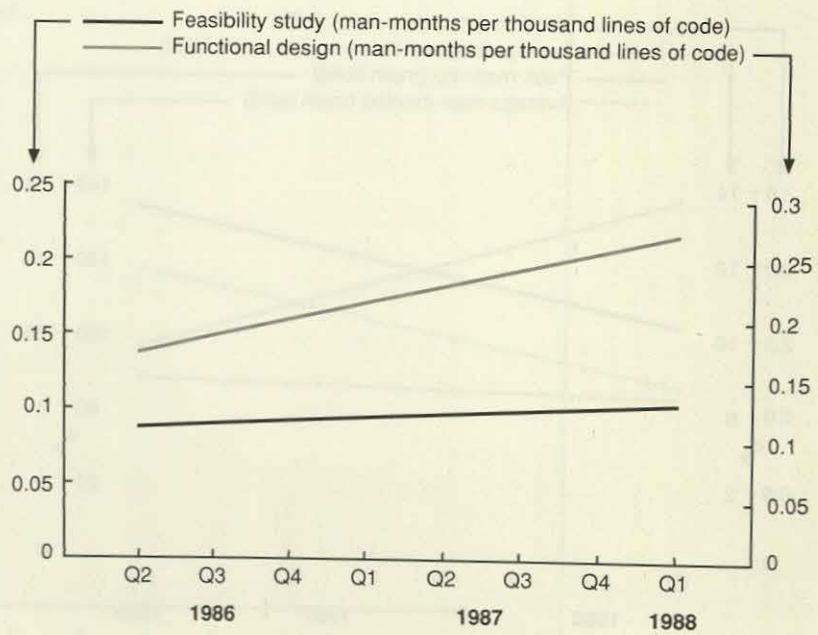
The trend to smaller projects and increasing MBI is also reflected in the downward trend of main-build time and the upward trend of main-build effort. Main-build time has decreased from about 14 to 10 months over the period of analysis, while main-build effort has increased from less than 90 to just under 120 man-months. As Figure 2.6 showed, PIs have not changed much over the period covered by our analysis, so the upward trend in effort is almost totally a consequence of rising MBI.

There is a downward trend in main-build time, and an upward trend in main-build effort

Although the effort at the main-build stage has increased, there has been a slight reduction in total project effort, implying that effort used in the earlier stages has decreased. This appears to be contradicted by the trends in the average effort (measured as man-months per thousand lines of code) used at the feasibility-study and functional-design stages (see Figure 3.8).

While the effort at the feasibility-study stage has remained virtually level at about 0.1 man-month per thousand lines of code, it has increased by about 60 per cent, from about 0.17 to 0.27 man-months per thousand lines of code, at the functional-design stage.

Figure 3.8 Effort per thousand lines of code at the feasibility-study stage has been essentially level, but has nearly doubled at the functional-design stage



(Source: PEP database)

However, this increase is a consequence of the trend to smaller projects, which use more effort per thousand lines of code at the earlier stages than larger projects. Surprisingly, enhancement projects (primarily in third-generation languages) have higher rates of effort at the earlier stages than new developments (primarily in fourth-generation languages).

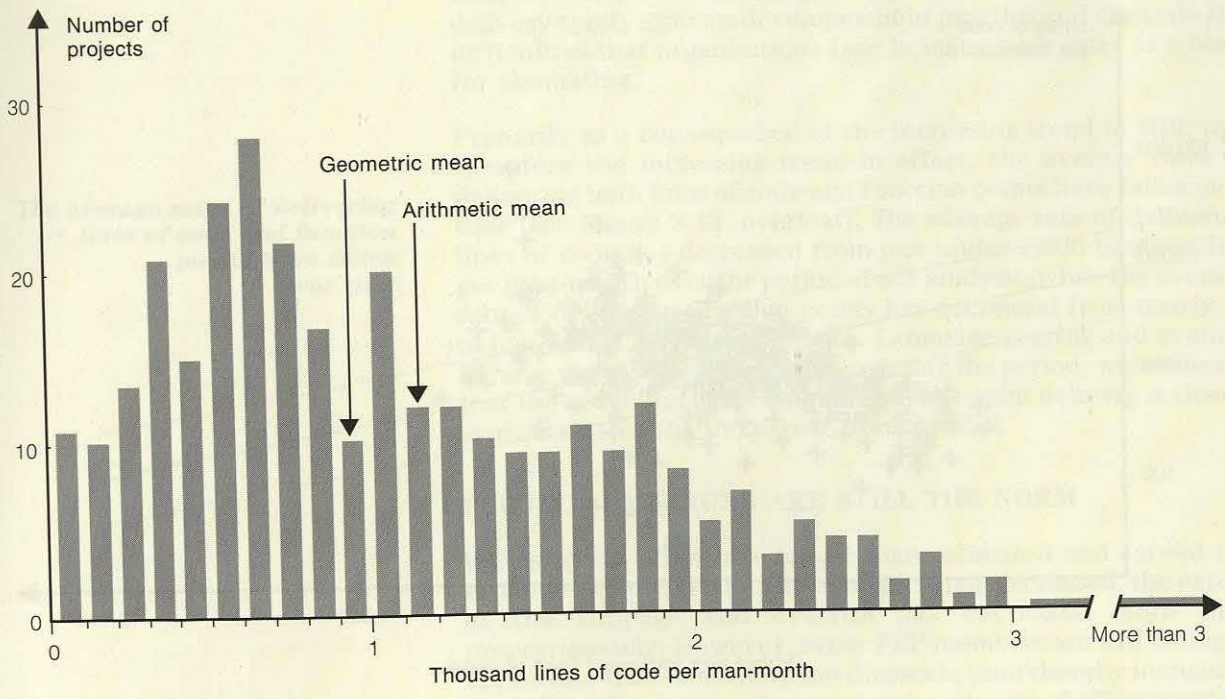
INCREASING MBI RATES LEAD TO REDUCED RATES OF DELIVERY

The rate of delivering function points ranges from one per man-month to over 300

The number of lines of code delivered per man-month of effort at the main-build stage ranges from less than 50 to more than 10,000 (see Figure 3.9). The geometric mean of about 950 lines of code per man-month and the arithmetic mean of about 1,150 are fairly close. The rate of delivering function points ranges from one per man-month to over 300 (see Figure 3.10, overleaf). The geometric and arithmetic means are 13 and 23 respectively.

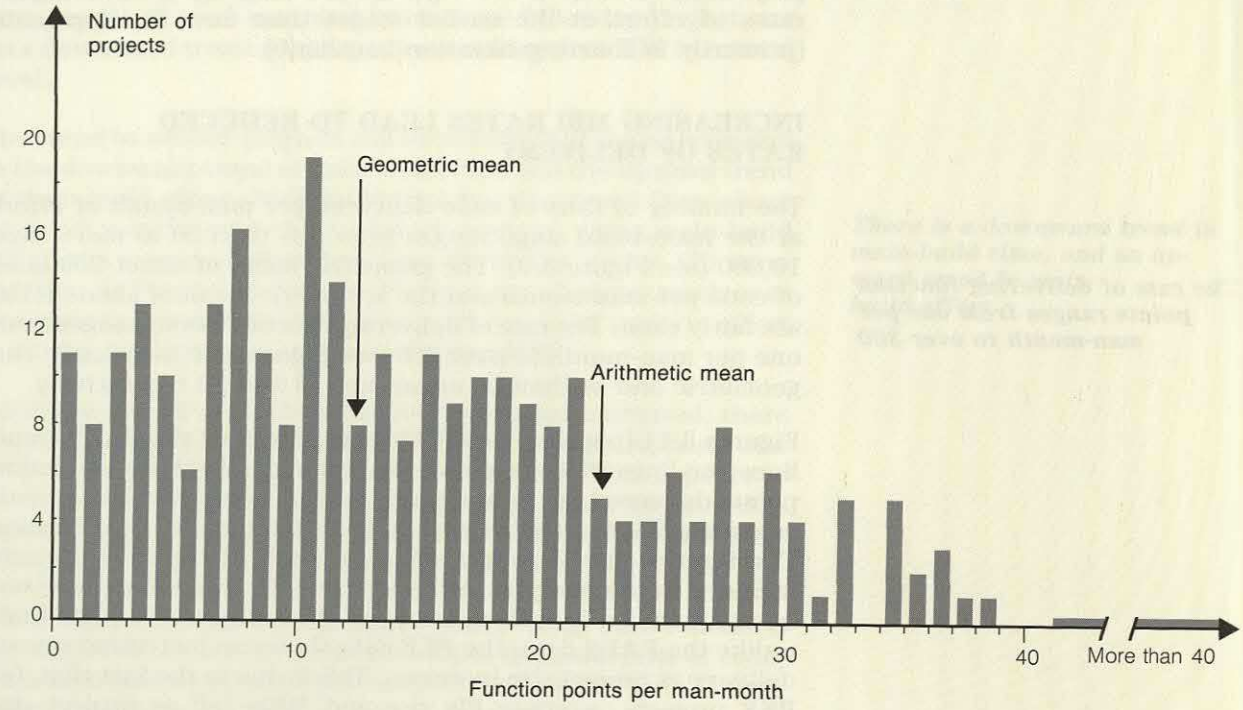
Figures 3.11 (page 28) and 3.12 (page 29) show the PADS trend lines for lines of code delivered per man-month and function points delivered per man-month, related to project size. In both cases, the PADS rates of delivery decline as project size increases. The figures also show the plots for each of the PEP projects included in the analysis. Although the PEP trend lines have not been drawn in, it is clear from the clustering of the plots that, unlike the PADS data, the PEP data shows an increasing rate of delivery as project size increases. This is due to the fact that, for PEP projects, average PIs rise and MBIs fall as project size increases.

Figure 3.9 The number of lines of code delivered per man-month of main-build effort ranges from less than 50 to more than 10,000



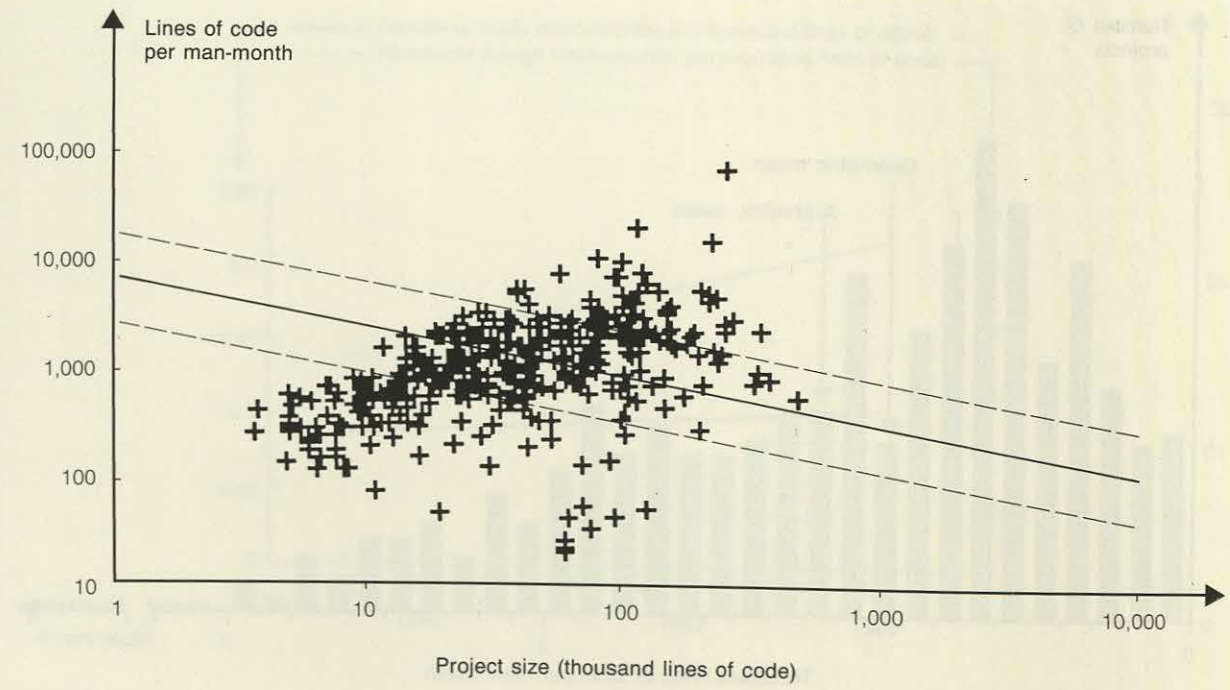
(Source: PEP database)

Figure 3.10 The rate of delivering function points ranges from one per man-month to more than 300



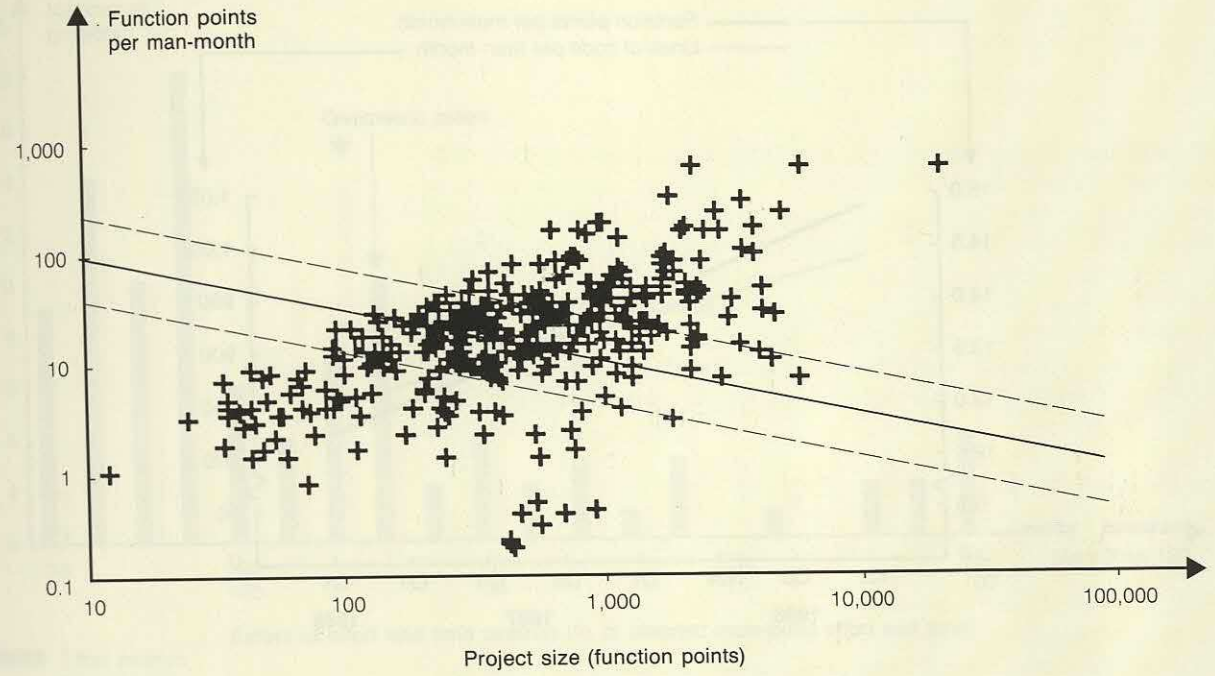
(Source: PEP database)

Figure 3.11 Unlike PADS, the rate at which lines of code are delivered for PEP projects increases as project size increases



(Source: PADS and PEP database)

Figure 3.12 Unlike PADS, the rate at which function points are delivered for PEP projects increases as project size increases



(Source: PADS and PEP database)

The wide variations in the rates at which lines of code and function points are delivered is due to the combined effects of varying PIs and MBIs (and, in the case of the function-point-delivery rate, to language gearing). The wide-ranging rates of delivery typify systems development in practice and illustrate the difficulties that organisations face in using such rates as a basis for estimating.

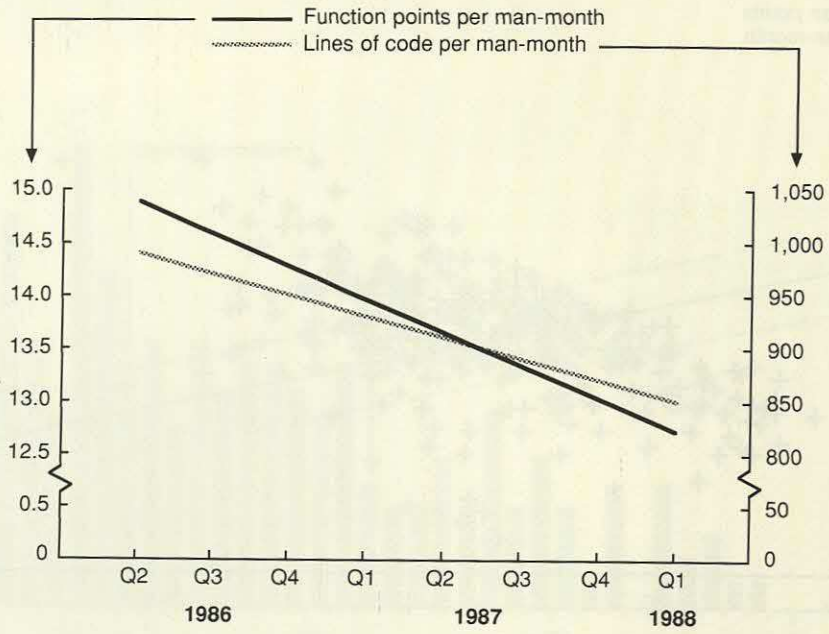
The average rates of delivering lines of code and function points have fallen over time

Primarily as a consequence of the increasing trend in MBI, and therefore the increasing trend in effort, the average rates of delivering both lines of code and function points have fallen over time (see Figure 3.13, overleaf). The average rate of delivering lines of code has decreased from just under 1,000 to about 850 per man-month over the period of our analysis, while the average rate of delivering function points has decreased from nearly 15 to just under 13 per man-month. Language gearing and average PI have not changed significantly during the period, which means that the reduction in the rate of function-point delivery is clearly attributable to the increasing trend in MBI.

PROJECT OVERRUNS ARE STILL THE NORM

PEP projects often take longer than estimated and exceed the estimated effort. As the size of projects has decreased, the extent of the slippage and overrun has decreased more than proportionately. However, many PEP members are still failing to appreciate that shortening the timescale (and thereby increasing the MBI) will substantially increase the amount of effort required to complete a project.

Figure 3.13 The average rates of delivery of lines of code and function points have both fallen over time



(Source: PEP database)

MORE THAN A THIRD OF PROJECTS EXCEED BUDGET OR ARE LATE

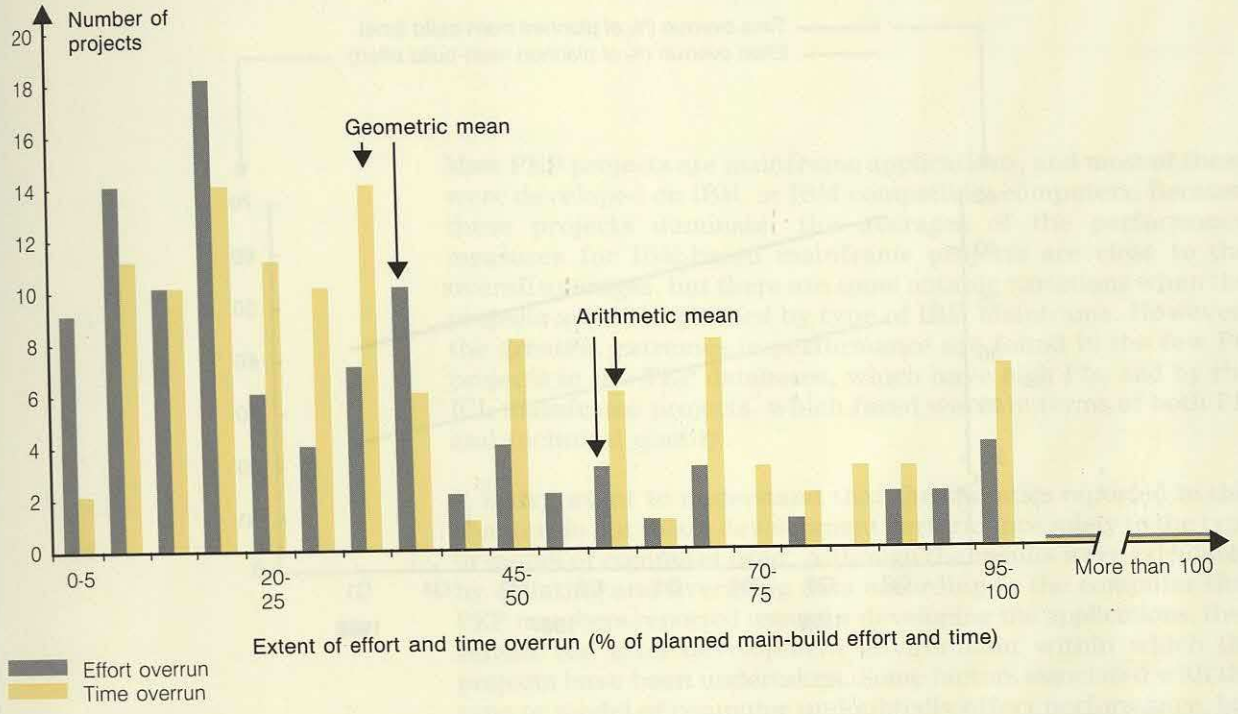
About 35 per cent of PEP projects exceed the estimated effort. Overruns range from under 5 per cent to over 400 per cent, but on average, are about 37 per cent of planned main-build effort (see Figure 3.14). The projects that overrun on effort are about 10,000 lines of code larger than the overall PEP average. The average PI for these projects is the same as it is for other PEP projects of the same size, while the average MBI is slightly above average. Although some of the overrun may be due to time pressure, most seems to be due to estimating problems.

About 40 per cent of PEP projects take longer than estimated

About 40 per cent of PEP projects take longer than estimated. As Figure 3.14 shows, the slippage ranges from under 5 per cent to over 200 per cent, but on average, is about 32 per cent of planned main-build time. The average size of the projects that exceed the planned time is also about 10,000 lines of code larger than the overall PEP average. Compared with other PEP projects of the same size, the average MBI for these projects is slightly lower, and the average PI is about a half point lower. It is likely, too, that underestimating the size of developments has contributed to the time overruns.

Surprisingly, the use of project-management tools does not seem to improve the situation. Thirty-five per cent of PEP projects have made use of them. These projects are, on average, little different from the remainder (they have slightly lower PIs, and marginally

Figure 3.14 The average extent of effort overrun on PEP projects is 37 per cent of planned main-build effort, and the extent of time overrun is 32 per cent of planned main-build time



(Source: PEP database)

greater time and effort overruns). The PMW project management tool is the most popular, being used on 20 per cent of PEP projects. A higher-than-average proportion of these projects have time and effort overruns (50 per cent in each case, compared with the overall PEP averages of 40 per cent and 35 per cent respectively), and the extent of the overruns is greater than the overall PEP averages (45 per cent and 35 per cent, compared with 40 per cent and 30 per cent respectively). Although PMW does not seem to help project managers reduce time and effort overruns, it may improve the accuracy with which such overruns are reported and recorded.

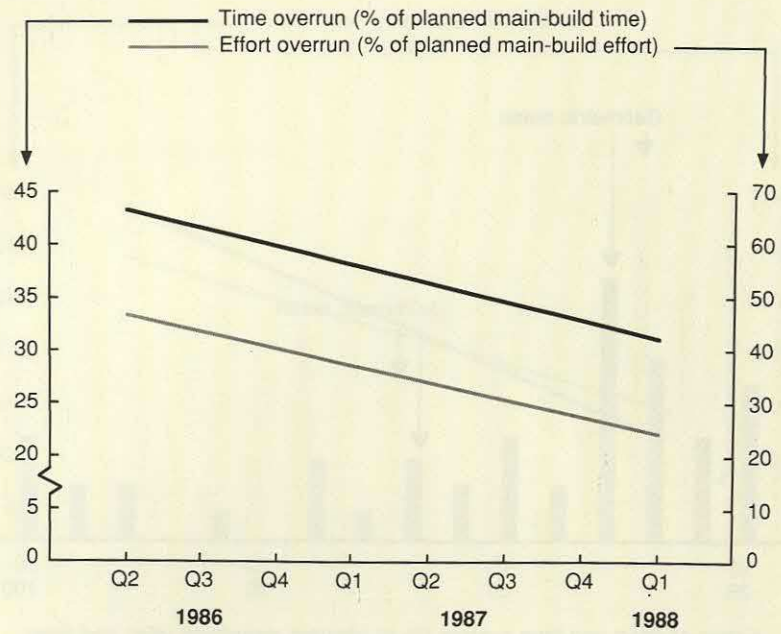
THE TREND IS FOR OVERRUNS TO DECREASE

The extent of time and cost overruns has decreased at a faster rate than project size has decreased. The time and effort required for small projects are clearly easier to estimate than for large projects, and small projects are easier to manage. Figure 3.15, overleaf, illustrates that average time overruns, expressed as a percentage of planned main-build time, have decreased from about 45 per cent to 30 per cent over the period of analysis. Average effort overrun has almost halved, from about 47 per cent to 24 per cent.

MBIs SHOULD BE REDUCED TO OBTAIN FURTHER IMPROVEMENTS

Increasing a project's MBI results in considerable additional effort (and, hence, costs). A very fast rate of manpower buildup (an MBI

Figure 3.15 Project characteristics change with time



(Source: PEP database)

of six) typically represents a seven- to eight-fold increase in manpower effort over a slow rate of manpower buildup (an MBI of one).

High MBIs are not often justified by the needs of the business

Based on our experience with PEP assessments, we believe that high MBIs have been justified by business needs in less than half of the projects examined. Higher-than-necessary MBIs result in effort that is 50 per cent higher than necessary across PEP members' projects. If this is representative of industry as a whole, the implications are very significant. In particular, it suggests that the performance of systems development departments could be dramatically improved by optimising the relationship between the elapsed time and the effort required to develop a system.

Having addressed the basic dimensions of project size and time, and their relationships with productivity and quality, we are now in a position to examine the performance of projects according to the computing environment and programming languages used.

Chapter 4

The impact of development environment and language on productivity and quality

Most PEP projects are mainframe applications, and most of these were developed on IBM, or IBM-compatible, computers. Because these projects dominate, the averages of the performance measures for IBM-based mainframe projects are close to the overall averages, but there are some notable variations when the projects are distinguished by type of IBM mainframe. However, the greatest extremes in performance are found in the few PC projects in the PEP databases, which have high PIs, and by the ICL-mainframe projects, which fared worst in terms of both PIs and technical quality.

It is important to understand that the statistics reported in this chapter do not relate development performance solely to the type or model of computer used. Although the results were calculated by collating and averaging data according to the computer that PEP members reported using for developing the applications, they reflect the total development environment within which the projects have been undertaken. Some factors associated with the type or model of computer undoubtedly affect performance, but there are many more influences associated with the wider environment — programming language in particular.

On average, each project uses two programming languages

On average, two languages are used for each PEP project. Most of the code for PEP projects is written in Cobol, with a wide variety of other languages being used for the remainder. Over 40 fourth-generation languages were identified, although they accounted for only about 10 per cent of all the PEP code. Fourth-generation languages, however, made a significant contribution to function delivery. Some projects using fourth-generation languages performed well, both in terms of PI and of function-delivery rate.

IBM MAINFRAMES AND COBOL DOMINATE THE DEVELOPMENT ENVIRONMENT

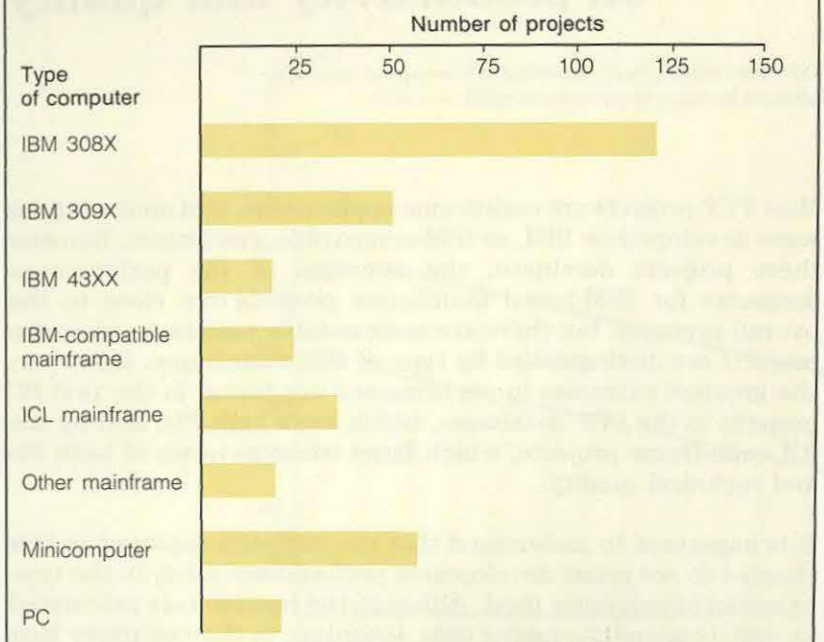
About 77 per cent of PEP projects are developments for mainframes, 17 per cent are minicomputer developments, and 6 per cent are PC developments. Cobol accounts for 62 per cent of all the code in PEP projects.

IBM ACCOUNTS FOR THE VAST MAJORITY OF MAINFRAME PROJECTS

Eighty per cent of mainframe projects were for IBM or IBM-compatible mainframes

The distribution of projects is shown overleaf in Figure 4.1. Eighty per cent of the mainframe projects were for IBM, or IBM-compatible, mainframes. Of the IBM mainframe projects in the PEP database, 65 per cent are based on IBM 308X mainframes, 25 per cent on IBM 309X mainframes, and 10 per cent on IBM 43XX mainframes.

Figure 4.1 PEP projects are developed for a variety of computers



(Source: PEP database)

About 85 per cent of all PEP projects are online applications and we were able to identify the transaction-processing software for about 60 per cent of these projects. Of the IBM mainframe projects (about 85 per cent of those with identifiable transaction-processing software), two-thirds used CICS and about 30 per cent used IMS/DC. The use of non-IBM transaction-processing software on IBM mainframes was very limited.

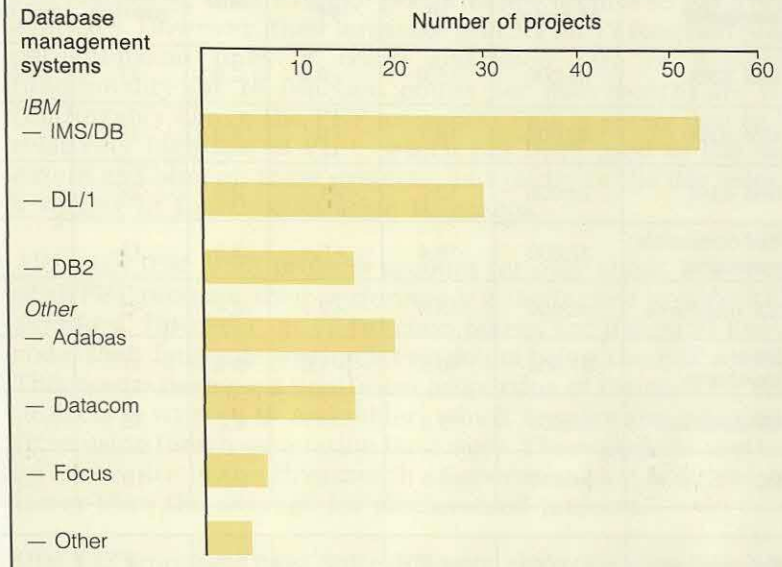
About 75 per cent of all PEP projects use a database management system, and we were able to identify the database software for 60 per cent of these projects. Two-thirds of the projects with identifiable database software were IBM mainframe applications. The distribution of projects by database management system is shown in Figure 4.2. Two-thirds of them used IBM products — IMS/DB, DL/1, and DB2. Non-IBM database management systems are being used to a greater extent than non-IBM transaction-processing systems software, with Adabas the most commonly used on PEP projects.

COBOL IS THE MOST COMMONLY USED LANGUAGE

Eighty per cent of the code in PEP projects is written in third-generation languages. Excluding variants of the same language, 24 third-generation languages are identifiable. The main language is Cobol, which accounts for 62 per cent of all the code in PEP projects, followed by PL/1 (13 per cent), and RPG (3 per cent). Fourth-generation languages account for 10 per cent of the code, low-level languages for 4 per cent, and job-control languages for the remainder. Third-generation languages account for more than 60 per cent of the delivered functionality (Cobol 46 per cent, PL/1 12 per cent, and RPG 3 per cent).

Cobol accounts for 62 per cent of all the code in PEP projects

Figure 4.2 IBM's database management systems are the most widely used



(Source: PEP database)

PRODUCTIVITY VARIES MARKEDLY ACCORDING TO THE TYPE OF COMPUTER

The few PC projects in the PEP database have high PIs, minicomputer projects perform above the average in terms of function points delivered per man-month, and those using ICL mainframes perform below average. Projects for mainframes other than ICL or IBM have high PIs, and ICL projects suffer from extremely low delivery of function points per man-month. As noted above, IBM mainframe projects dominate the database and therefore reflect overall PEP averages. The details of the differing performance characteristics are given in Figure 4.3, overleaf, which highlights the areas where performance is significantly different from the PEP averages.

PC PROJECTS HAVE HIGH PIs AND HIGH LANGUAGE GEARING

The rate of delivery of function points on PC projects is more than twice the PEP average

Only 6 per cent of PEP projects are PC developments. These projects are distinguished by having high PIs and high language gearing (19, compared with the PEP average of 14). At 16, the average PI for PC projects is more than one point better than the average for other projects of the same size. Language gearing, at 19 function points per thousand lines of code, is about 35 per cent above average and about twice the rate for an average Cobol program. The combination of high PI and high language gearing gives PC projects an average rate of delivery of 28 function points per man-month of development effort, which is more than double the PEP average (13).

Figure 4.3 Project characteristics differ according to the type of computer used

Type of computer	Size (lines of code)	Average PI	Relative PI[1]	MBI	Language gearing	Function points per man-month	Relative error rate 1 (%)[2]	Relative error rate 2 (%)[2]
IBM 308X	44,000	15.0	−0.2	2.8	17	16	−5	−15
IBM 309X	32,000	14.7	+0.1	2.8	12	12	−30	−50
IBM 43XX	24,000	13.2	−1.3	2.9	22	17	+60	+5
IBM-compatible mainframe	37,000	15.4	+0.4	3.0	13	14	−45	−45
ICL mainframe	40,000	13.0	−2.1	3.4	11	4	+75	+200
Other mainframe	37,000	16.6	+1.6	3.5	11	13	+45	−30
Minicomputer	55,000	15.3	−0.7	2.2	13	17	+40	+15
PC	34,000	16.0	+1.1	2.9	19	28	−20	+15
Pep average	41,000	14.9	[1]	2.8	14	13	[2]	[2]

[1] The figures shown are relative to the expected PI for PEP projects of the average size for the type of computer. Thus, IBM 308X projects have an average PI that is 0.2 below the expected PI for projects of 44,000 lines of code.

[2] The figures shown are percentages relative to the expected error rates for PEP projects of the average size for the type of computer. Thus, IBM 308X projects have 5 per cent fewer errors between integration and system testing and first operational capability, and 15 per cent fewer errors in the first month of operation than expected for projects of 44,000 lines of code.

MINICOMPUTER PROJECTS DELIVER A HIGHER-THAN-AVERAGE NUMBER OF FUNCTION POINTS PER MAN-MONTH

Minicomputer projects account for 17 per cent of all PEP projects. At 55,000 lines of code, their average size is the largest among the computer groupings. Their low MBI (2.2, compared with the PEP average of 2.8) compensates for the below-average PIs and language gearing, resulting in a delivery rate of 17 function points per man-month, which is comfortably above the PEP average (13).

ICL MAINFRAME PROJECTS FARE WORST

About 11 per cent of PEP projects are ICL mainframe applications. These projects have PIs about two points lower than other PEP projects of a similar size. They also have high MBIs, low language gearing, and a low rate of delivery of function points per man-month (just four, compared with the PEP average of 13). The high MBIs (which indicate severe time pressures) of these projects are, however, significantly affected by the MBIs of one PEP member, who has contributed about 40 per cent of all the ICL projects.

The low rate of function delivery per man-month is not surprising, because the projects have suffered from the combined effects of low PIs, higher-than-average MBIs, and below-average language gearing. The ICL projects also have error rates significantly higher than the average for similar-sized projects.

THE PRODUCTIVITY OF IBM MAINFRAME PROJECTS VARIES WIDELY

PL/1 and fourth-generation languages help 308X projects to have higher-than-average rates of delivering function points

Because IBM 308X projects form by far the largest group in the PEP database, their average performance is close to the overall averages. However, their language gearing (at 17 function points per thousand lines of code) and their rate of delivering functionality (at 16 function points per man-month) are both comfortably above the PEP averages. This is partly due to the relatively high use of PL/1, which has been used to the same extent as Cobol on these projects, and partly to the use made of a variety of fourth-generation languages.

Although IBM 309X projects account for only about 14 per cent of all PEP projects, their performance is fairly close to overall PEP averages. However, at 12 function points per thousand lines of code, their language gearing is two points below the PEP average. This occurs because a significant proportion of the code for these projects is written in Assembler, which negates any gains made from using fourth-generation languages. These projects also have an error rate in the first month of operation that is 50 per cent lower than the average for similar-sized projects.

IBM 43XX projects have quite different characteristics from other IBM mainframe projects. At 24,000 lines of code, they are smaller than for any other group of projects, and their average PI is more than one point below that for projects of a similar size. However, language gearing, at 22 function points per thousand lines of code, is the highest of any group, largely due to the fact that these projects have a high proportion of code written in Guest and Ideal. The high language gearing compensates for the lower-than-average PIs and accounts for their good rate of delivering functionality — 17 function points per man-month.

The IBM mainframe projects that use IMS/DB had average PIs and language gearing, and a near-average function-delivery rate. On the other hand, IBM mainframe projects that use DL/1 had PIs over half a point lower than other projects of a similar size. However, they had a high language gearing (20 function points per thousand lines of code), resulting in a high rate of delivering functionality — 19 function points per man-month, which is nearly 50 per cent above the PEP average.

OTHER MAINFRAME PROJECTS HAVE HIGH PIs AND HIGH MBIs

Among mainframe projects, the highest PIs are achieved by projects for mainframes other than ICL or IBM. However, below-average language gearing and high MBIs (indicating severe time pressure) result in only average rates of function delivery, in terms of function points per man-month. Error rates at integration and system testing for these projects are also 45 per cent above the average for similar-sized projects, but are 30 per cent below average during the first month of operation.

FOURTH-GENERATION LANGUAGES PROMISE IMPROVED PRODUCTIVITY

Although fourth-generation languages account for only 10 per cent of the code in PEP projects, they are used to provide as much

as 30 per cent of delivered functionality. Furthermore, although there are great variations in the PIs of PEP projects that use fourth-generation languages, some members are achieving very high PIs through the use of these languages.

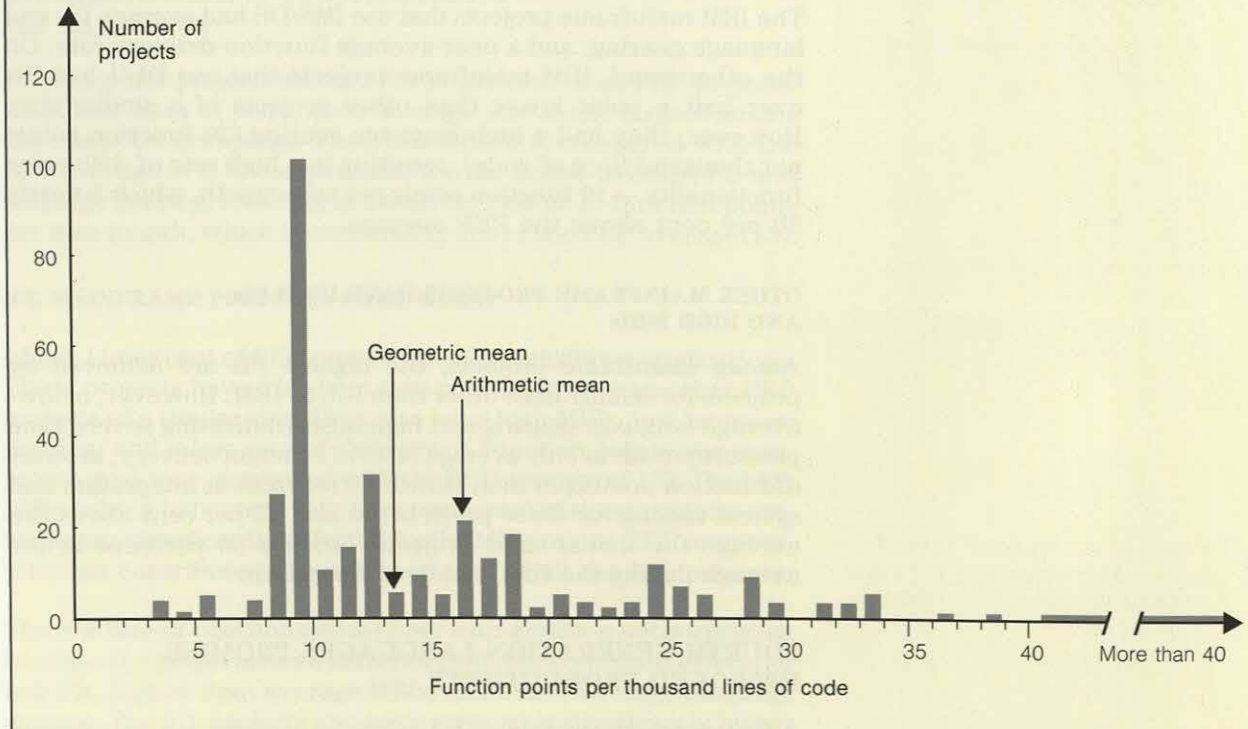
No single fourth-generation language is dominant

Forty-four different fourth-generation languages were notified by PEP members, but none of them is dominant in the way that Cobol is for third-generation languages. Natural (which comprises 3 per cent of the total code) is the most widely used, followed by ADF, Gener/ol, and Guest (1 per cent each of total code). However, the picture changes when the contribution of fourth-generation languages to delivered functionality is analysed. Natural and Telon each contributes 4 per cent of the functionality of PEP projects, Gener/ol about 3 per cent, and SQL and ADF 2 per cent each. The differences are, of course, accounted for by the different language gearings of fourth-generation languages.

Figure 4.4 shows the distribution of the language gearing of PEP projects, which varies from four to over 60. The geometric and arithmetic means are about 14 and 17 respectively. There is a pronounced peak at around 10 function points per thousand lines of code, and several other minor peaks. The principal peak is associated with projects written mainly in Cobol. The minor peaks coincide with the use of PL/1 and RPG, and the more widely used fourth-generation languages, such as Natural.

Although the high language gearing of fourth-generation languages leads to good function-delivery rates, the PIs of projects using these languages vary widely. Figure 4.5, overleaf, shows

Figure 4.4 The language gearing of PEP projects varies from four to over 60



(Source: PEP database)

the average PIs of projects developed with the leading fourth-generation languages. It also shows, for each language, the difference in the PIs from the average PEP PIs for similar-sized projects. The figure also shows equivalent data for the leading third-generation languages and for Assembler.

Fourth-generation languages are often used in conjunction with Cobol

Few projects are developed just with a fourth-generation language, however. Code written in fourth-generation languages is widely scattered among PEP projects, and is often found in conjunction with Cobol code. The projects for which the average PIs are shown in Figure 4.5 are those with a fourth-generation language as the primary or secondary language. Because of the small number of projects using each of the different languages, the data shown in the figure can be taken only as an initial indicator of the performance of the fourth-generation languages.

Projects using Mantis, Natural, and Telon have average PIs that are better by one or more points than those for similar-sized projects. Natural and Mantis are fairly well established languages and development staff are likely to have become quite skilled in their use. Telon, a Cobol code generator, is a more recent language, so it is encouraging that the PIs of projects developed with Telon are already about one point higher than the average PI for projects of a similar size.

Interestingly, Figure 4.5, overleaf, shows that the highest PI (and the highest positive difference) is achieved with RPG, a third-generation language. The lowest PI is achieved with a fourth-generation language (Ideal). This highlights the difficulty of measuring overall development performance in terms of lines of code produced, rather than in terms of functionality delivered. The rate at which functionality is delivered depends to a large extent on the language gearing of the particular programming language.

For example, Figure 4.5 shows that the average PIs of projects using ADF, Gener/ol, Ideal, and UFO are between one and two PI points lower than the average for similar-sized projects. However, when language gearing is taken into account, only UFO projects are below the overall PEP average of 13 function points per man-month. (The good function-delivery rate for ADF projects is also due to the low MBI of these projects.) The implication is that if PEP members can raise their PIs for projects using fourth-generation languages to the PEP average, they will usually be able to increase their function-delivery rate to four times that of typical Cobol projects.

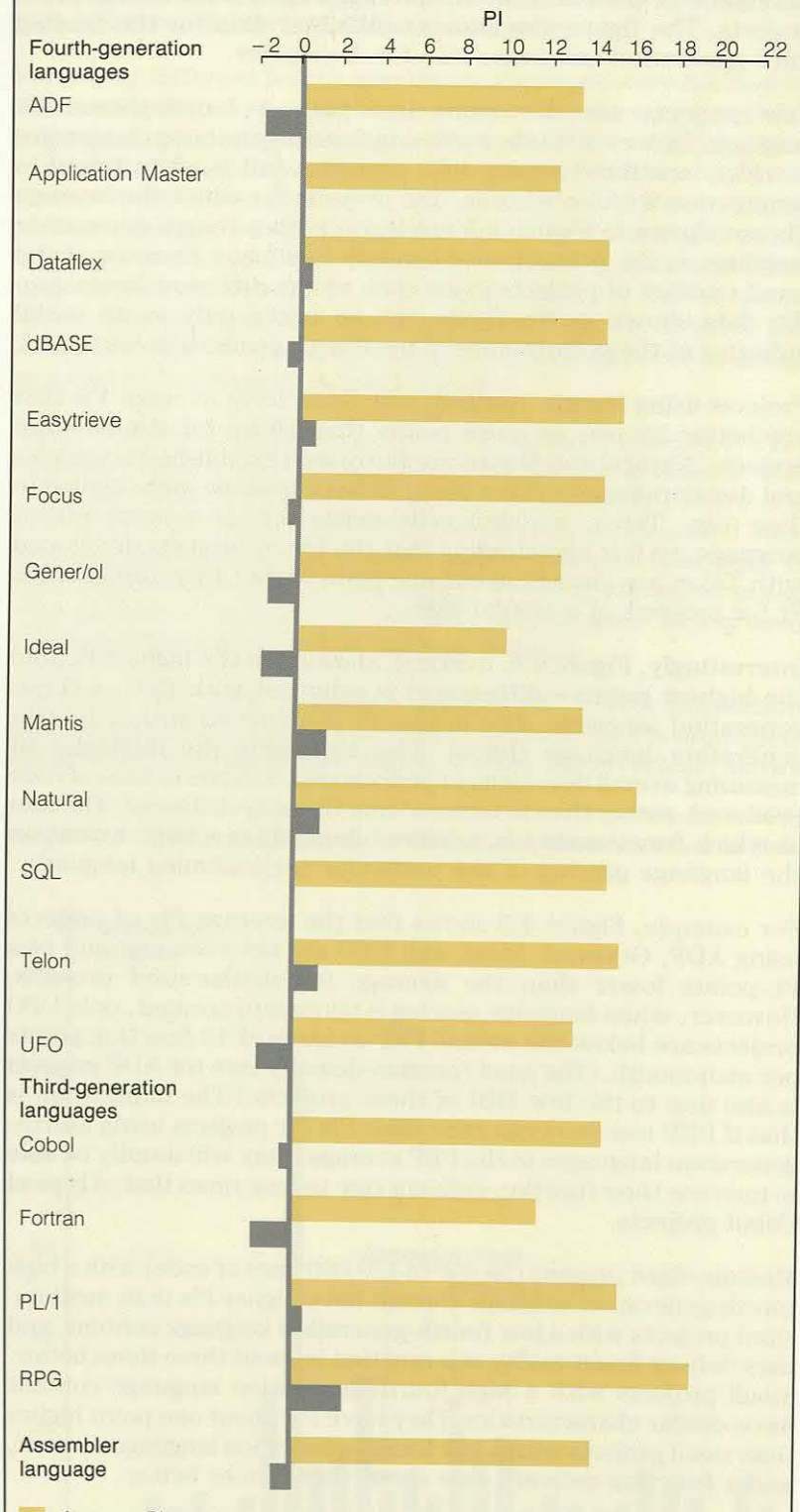
Raising PIs for fourth-generation-language projects promises large increases in the function-delivery rate

Medium-sized projects (20,000 to 120,000 lines of code) with a high fourth-generation language content have higher PIs than medium-sized projects with a low fourth-generation language content, and they deliver functionality at a rate that is about three times better. Small projects with a high fourth-generation language content have similar characteristics. They have PIs about one point higher than small projects with a low fourth-generation language content, and a function-delivery rate about three times better.

TECHNICAL QUALITY VARIES ACROSS DEVELOPMENT ENVIRONMENTS

As Figure 4.3 showed, error rates vary widely across the development environments. With the exception of 43XX projects,

Figure 4.5 The PIs of projects using different types of programming language vary widely



(Source: PEP database)

IBM and IBM-compatible mainframes have the lowest error rates. ICL projects are notable for very high error rates in integration and system testing, and for high error rates in the first month of operation.

MOST IBM AND IBM-COMPATIBLE MAINFRAME PROJECTS HAVE LOW ERROR RATES

Unlike other IBM and IBM-compatible mainframe projects, 43XX projects have high error rates, particularly in integration and system testing, where the error rate is 60 per cent higher than the average for similar-sized projects. However, during the first month of operation, the error rate for these projects is close to the average. This suggests that the IBM 43XX development environment does not encourage high-quality original work by project staff.

About 6 per cent of PEP projects are developed on IBM-compatible mainframes. These projects have very low error rates, both during integration and system testing, and in the first month of operation. Over 60 per cent of these projects are for banks, however, which probably explains the low error rates. Operational reliability of applications is fundamental to much of a bank's business.

The quality of projects using DL/1 appears to be above average

IBM mainframe projects using DL/1 have an error rate during integration and system testing that is 35 per cent lower than the average of projects for a similar size, and more than 50 per centlower during the first month of operation. This suggests that the quality of the original development work is above average for projects using DL/1.

The error rate during integration and system testing for projects using IMS/DB is about 25 per cent higher than the average for similar-sized projects, but the error rate during the first month of operation is 30 per cent below average. Thus, while the original development work may not have been at the same level of quality as for DL/1 projects, the integration and system testing of IMS/DB projects was effective and resulted in above-average reliability.

OTHER MAINFRAME PROJECTS HAVE HIGH ERROR RATES

ICL mainframe projects have error rates during integration and system testing that are 75 per cent above the average for projects of a similar size. The high level of errors continues into the first month of operation, where the error rate is three times higher than the average. (These error rates are, however, significantly affected by the projects of one PEP member.)

Although projects for mainframes other than ICL or IBM have the highest PIs, they also have a higher-than-average error rate during integration and system testing. However, their error rate during the first month of operation is 40 per cent below the average for similar-sized projects.

MINICOMPUTER PROJECTS HAVE HIGHER ERROR RATES THAN AVERAGE

Minicomputer projects have an error rate during integration and system testing that is 40 per cent above the average for

similar-sized projects. The error rate is still above average during the first month of operation, but only by 15 per cent, suggesting that minicomputer development environments do not encourage staff to produce high-quality original work.

Having analysed project performance by computing environment and language, we now turn our attention to the impact that techniques, methods, and tools have on productivity and technical quality.

Chapter 5

The impact of techniques, methods, and tools on productivity and quality

From our analysis of the PEP database, it appears that little benefit, in terms of higher PIs and better technical quality, is currently being obtained by using techniques, methods, and tools. However, until we are able to build into PEP a measure of overall quality expressed in user and business terms (a subject for PEP research in 1990), it would be very unwise to condemn their use on the basis of our findings. It is highly likely that factors other than those we can identify from the existing data contribute to the poor performance of projects using techniques, methods, and tools. In particular, providing inadequate staff training and using techniques or tools in an inappropriate way are likely to lead to lower PIs and to poorer technical quality.

Although projects using techniques and tools often have lower PIs than average, the exception is tools that automatically generate code to support transaction processing. The use of techniques and tools is also frequently associated with reduced technical quality, but projects that use formal walkthroughs are a notable exception. Such projects have fewer errors in integration and system testing, and in the first month of operation.

USE OF TECHNIQUES, METHODS, AND TOOLS LEADS TO LOWER PIs

Testing tools were used on 40 per cent of PEP projects

The most frequently mentioned techniques, methods, and tools were those that helped in the testing process, being used on more than 40 per cent of PEP projects. Other types of tools were used in a small minority of PEP projects. The use of structured techniques and formal development methods was reported in no more than a third of PEP projects. With some notable exceptions, these projects have lower-than-average PIs, as do the projects that use supporting tools such as analyst or programmer workbenches. Figure 5.1, overleaf, shows how the PIs of projects using the best known techniques, methods, and tools differ from the average PIs of similar-sized projects.

TECHNIQUES

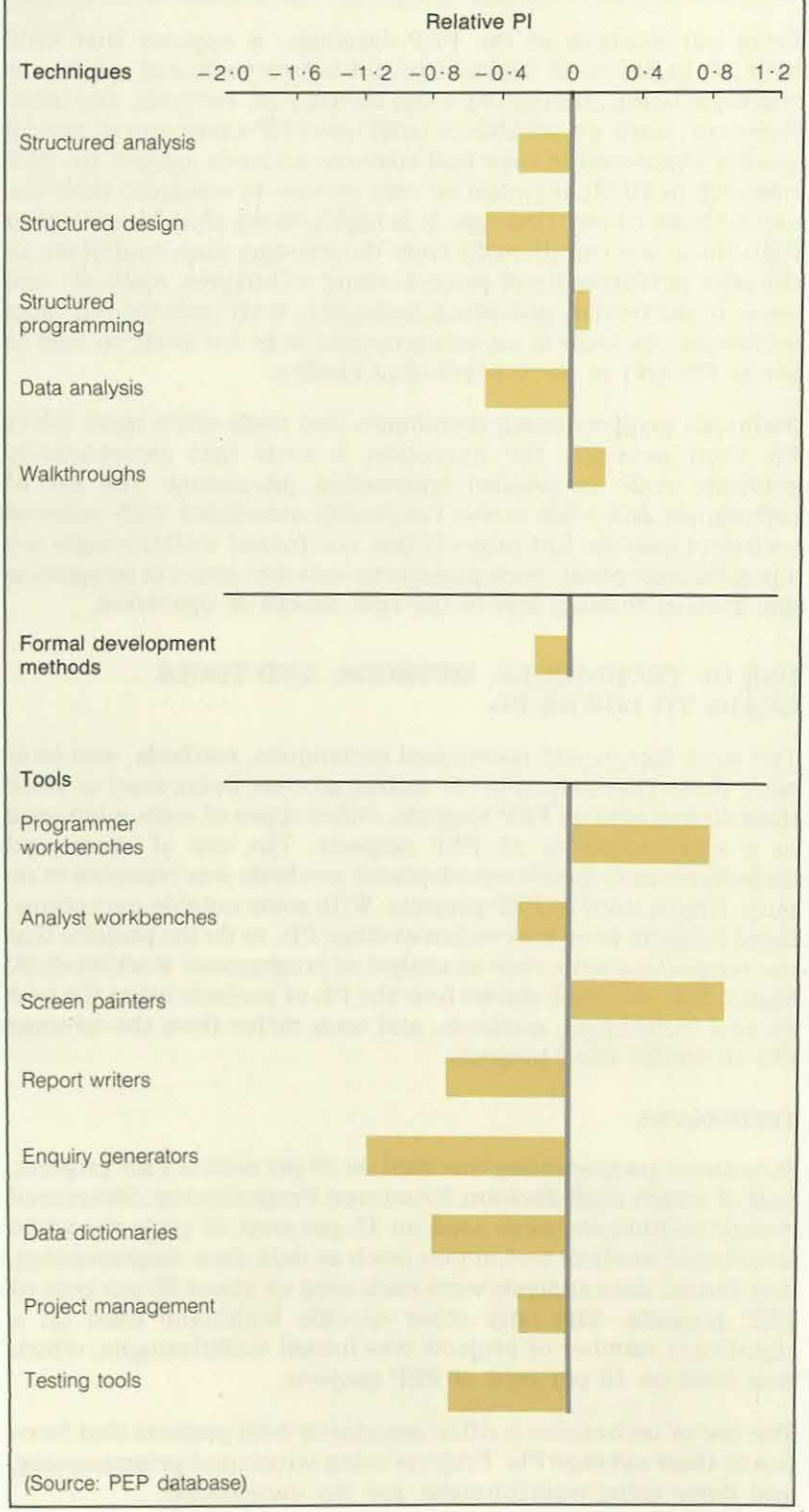
Structured programming was used on 30 per cent of PEP projects, half of which used Jackson Structured Programming. Structured design techniques were used on 15 per cent of projects, while structured analysis techniques (such as data-flow diagramming), and formal data analysis were each used on about 20 per cent of PEP projects. The only other specific technique used on a significant number of projects was formal walkthroughs, which was used on 16 per cent of PEP projects.

Structured programming and walkthroughs are associated with above-average PIs

The use of techniques is often associated with projects that have lower-than-average PIs. Projects using structured programming, and those using walkthroughs, are the exceptions.

Figure 5.1 Most PEP projects that use techniques, methods, and tools have lower-than-average PIs

Each chart shows the difference in PI relative to the average PI for all PEP projects of a similar size. Thus, projects using structured analysis techniques have an average PI that is 0.3 lower than the average PI for similar-sized projects.



(Source: PEP database)

Structured analysis: Projects using structured analysis have slightly lower-than-average PIs, slightly higher-than-average MBIs, and more software errors. (The sample size was small, however.) The use of fourth-generation languages was marginally lower than average, resulting in a rate of function delivery of 10 function points per man-month, about 30 per cent below average. The use of structured-analysis techniques may, of course, also contribute to a better fit of the developed systems to business needs, but the data currently stored about PEP projects does not allow us to measure this.

Structured design: At 57,000 lines of code, structured-design projects were larger than average and had language gearing that was just below average. This is not surprising, because such techniques are more likely to be used for large applications developed in traditional languages. PIs were half a point lower than the average for projects of a similar size, and the average rate of function delivery, at eight function points per man-month, was more than 40 per cent below average. These measures do not necessarily imply that the use of structured-design techniques reduces development performance. The main benefit of structured design is likely to come from easier maintenance. However, the existing PEP data does not allow us to measure improvements in the maintainability of systems.

Structured programming: Projects that used structured programming had PIs slightly higher than those that did not, and performed close to average in all other respects. Projects using Jackson Structured Programming had an average PI about half a point higher than those using other structured programming techniques. Staff using this technique will probably have been doing so for many years, and the skills will be well established. The high level of skill will, to some extent, account for the better-than-average PIs of these projects.

Data analysis: Projects using formal data analysis were larger than average — 47,000 lines of code — and had average language gearing. PIs were half a point lower than the average for similar-sized projects, resulting in an average function-delivery rate of 10 function points per man-month.

Formal walkthroughs: The PIs of projects using formal walkthroughs are slightly higher than the average for projects of a similar size, and about half a PI point higher than those not using formal walkthroughs.

FORMAL DEVELOPMENT METHODS

Just over a quarter of PEP projects used formal development methods. (By formal development methods, we mean the use of clearly defined approaches to, and practices for, systems development.) Such methods are typified by, but not exclusively the province of, proprietary methods. Although formal methods are, by definition, based on structured techniques, the use of such techniques is not limited to formal methods, and was considered separately in our analysis.

SSADM and Method/1 are the most commonly used proprietary development methods

No one proprietary product was mentioned in more than 5 per cent of PEP projects. The two most frequently mentioned were SSADM (5 per cent) and Method/1 (4 per cent).

As Figure 5.1 shows, the use of formal development methods results in a small reduction in PI. It is, of course, possible that their use may result in a better fit of the developed systems to business needs, but the existing PEP data does not allow us to measure this. Use of formal development methods appears to correspond with greater use of fourth-generation languages, because these projects had slightly higher language gearing than average, at 17 function points per thousand lines of code. Slightly higher-than-average MBIs have, however, resulted in a delivered-function rate that is below average — 12, compared with 13, function points per man-month.

TOOLS

Seven per cent of the projects used an analyst workbench, such as Excelerator and Auto-Mate, and about 10 per cent used a programmer workbench, such as Maestro. Nearly half of the latter projects were carried out by a large government-sector organisation.

Data dictionaries were used on nearly 25 per cent of projects, of which more than a quarter used Datamanager.

The use of code- and function-generation aids, such as screen painters, and report and enquiry generators, were also reported by sizeable proportions of PEP members. Screen painters automatically generate code to support transaction-processing applications, from screens that are designed interactively. Such aids were used on about 20 per cent of PEP projects. (Since 85 per cent of all PEP projects are categorised as online applications, for which screen painters would normally be appropriate, only one out of four suitable projects made use of screen painters.) Fewer projects used report and enquiry generators — about 10 per cent and 12.5 per cent respectively.

Testing tools were used on over 40 per cent of projects. The most popular were Intertest (7 per cent of projects), CEDF, Abendaid, and Batch Terminal Simulator (5 per cent), and Xpediter (4 per cent).

Projects that used tools generally had lower PIs than those that did not

In general, projects where tools were used had lower PIs than those where they were not, by nearly one point. The exceptions were projects using programmer workbenches and those using transaction screen painters.

Programmer workbenches: The projects using programmer workbenches are quite distinctive — they are usually enhancement or maintenance projects of large systems written in traditional languages, and have good PIs that are achieved under severe time pressures (in other words, they have high MBIs). Their new-code content is lower than average and they had a low language gearing — 10 function points per thousand lines of code, compared with the average of 14. Their high MBIs mean that they delivered functionality at the very low rate of four function points per man-month — the PEP average is 13. Since most enhancement or maintenance projects in the PEP database have significantly lower PIs, the relatively good PIs of projects using programmer workbenches is encouraging.

Analyst workbenches: The small proportion of projects using analyst workbenches were distinguished by being developments of new, smaller-than-average systems — about 28,000 lines of code. They were also characterised by their higher-than-average fourth-generation-language content — which resulted in an above-average language gearing of 19 function points per thousand lines of code. The function-delivery rate for these projects, at 19 function points per man-month, was also above average. Otherwise, their performance was close to average. On average, the PIs were slightly below the average for similar-sized projects, with the smaller projects having lower PIs than larger ones. Like formal development methods, analyst workbenches may also help to produce systems that are a better fit to business needs, but again, the PEP data does not at present enable us to measure this.

Screen painters: Screen painters are usually associated with fourth-generation languages, particularly code generators such as Telon. This is reflected in the fairly high language gearing of 17 function points per thousand lines of code for projects using screen painters. The average size of the projects — 49,000 lines of code — is above the overall average. PIs are nearly one point above average for the size of projects and nearly 1.5 points higher than those projects not using screen painters. This translates into at least a 25 per cent reduction in effort, and as much as 40 per cent. The high PIs and high language gearing meant that these projects delivered about 21 function points per man-month.

The use of screen painters is associated with significant reductions in effort

Report writers: The projects that used report writers are larger than average — 56,000 lines of code — and have near-average language gearing. The PIs are lower than the average for the size of project, by over half a point. These projects delivered functionality at a low rate of six function points per man-month, owing to their slightly higher-than-average MBIs.

Enquiry generators: The average size and language gearing of the projects that used enquiry generators are close to the averages. Their PIs are, however, more than one point below average for this size of the project, and the function delivery rate of nine function points per man-month is nearly 40 per cent below the average.

Projects using data dictionaries have below-average PIs

Data dictionaries: Projects using data dictionaries also perform below average, having an average PI nearly one point lower than the average for projects of a similar size. Although we could expect their use to lead to some reduction in the PI, it is not possible to attribute all of the poorer performance directly to their use. Projects where Datamanager was used fared slightly better than projects using other data dictionaries, having an average PI only about half a point lower than average for the size of the project. Many systems development managers will, of course, be happy to tolerate lower PIs for projects using data dictionaries, because of the resulting improvements in ease of maintenance.

Project-management tools: Project-management tools were used on 35 per cent of PEP projects. These projects had an average PI slightly lower than the average for projects of a similar size.

Testing tools: Testing tools are used on about 40 per cent of PEP projects. These projects have lower PIs (by more than half a point) than projects of a similar size. The main reasons for using such tools, however, is to improve the technical quality of systems.

MOST TECHNIQUES, METHODS, AND TOOLS ARE ASSOCIATED WITH LOWER QUALITY

The use of techniques, methods, and tools also seems to have an adverse effect on the technical quality of projects. The main exception is the walkthrough technique, as Figure 5.2 shows.

TECHNIQUES

Structured analysis: Error rates are available for about half of the projects using structured-analysis techniques. Error rates in integration and system testing were lower than average, but higher than average in the first month of operation.

Structured design: Error rates are available for about two-thirds of the projects using structured-design techniques. Error rates were higher than average in integration and system testing, and in the first month of operation, both by about 40 per cent.

Structured programming: Error rates, both at integration and system testing and in the first month of operation, for projects that use structured-programming techniques are almost exactly the same as the average for similar-sized projects.

Data analysis: Error rates are available for about half of the projects using formal data analysis. While the error rate is close to the average in integration and system testing, it is higher than average in the first month of operation.

Walkthroughs: The only technique that results in consistently lower error rates is formal walkthroughs (including inspections). The error rate at integration and system testing for projects using this technique was about 35 per cent below the average for similar-sized projects, and about 10 per cent below average in the first month of operation. This is a very encouraging result because it implies that the lower error rate at integration and system testing was due to inherently higher quality, not to insufficient integration and system testing.

Projects using walkthroughs have below-average error rates

FORMAL DEVELOPMENT METHODS

The use of formal development methods has not resulted in improved technical quality, as measured by number of software errors. Projects using formal development methods had error rates that were over 20 per cent higher than average during integration and system testing, and 55 above average in the first month of operation.

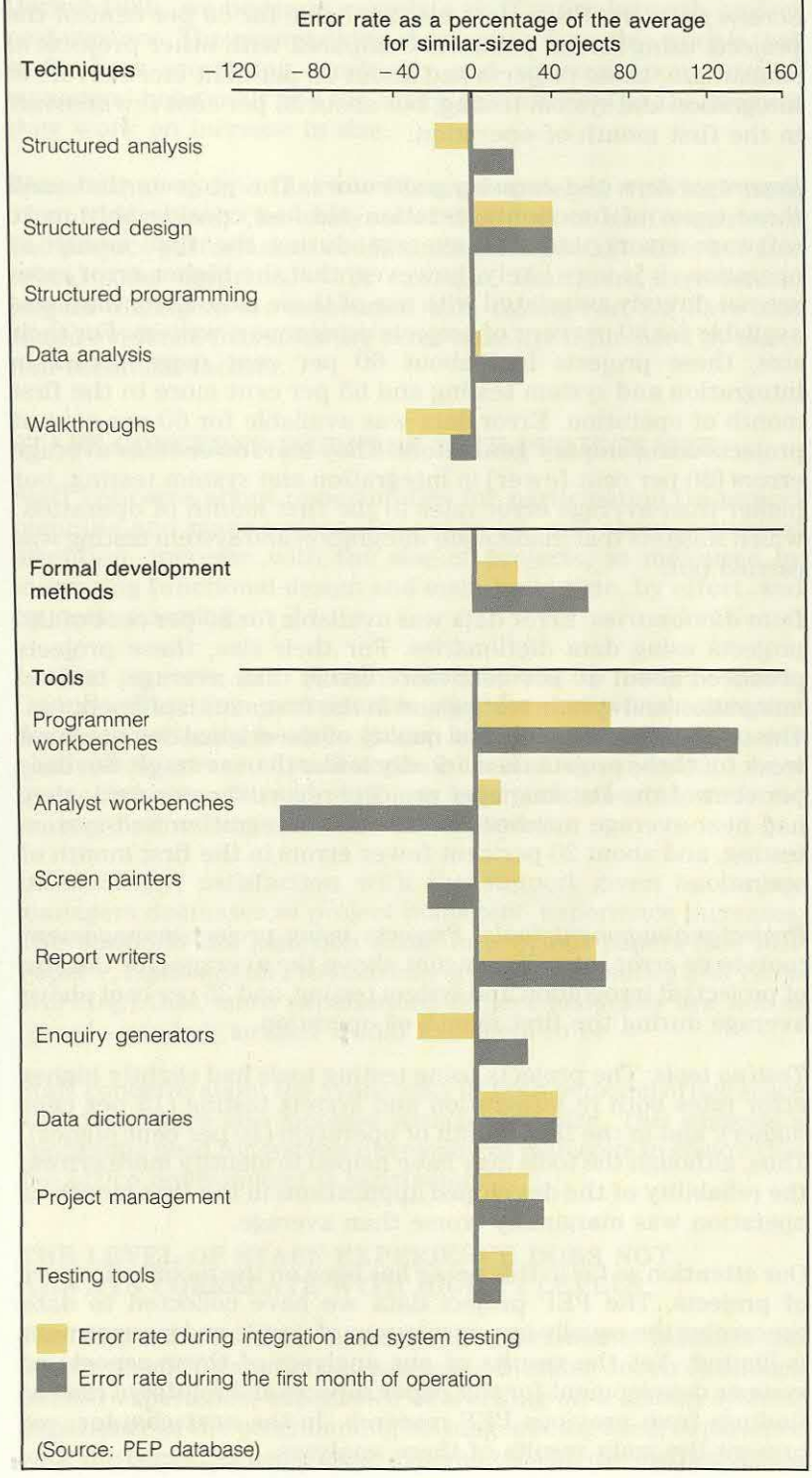
The use of formal development methods has not resulted in improved technical quality

TOOLS

Programmer workbenches: Error rates were available for about two-thirds of the projects using programmer workbenches. The error rates were significantly higher than average both in integration and system testing (70 per cent higher), and in the first month of operation (140 per cent higher). These high error rates

Figure 5.2 Most PEP projects that use techniques, methods, and tools have higher-than-average error rates

Each chart shows the difference in error rate relative to the average error rate for all PEP projects of a similar size. Thus, projects using structured analysis techniques have an average error rate during systems and integration testing 15 per cent lower than the average error rate for similar-sized projects, and an average error rate during the first month of operation 20 per cent higher than the average rate for similar-sized projects.



(Source: PEP database)

are due, however, to the very high error rates reported by one PEP member.)

Analyst workbenches: Error levels for those projects using analyst workbenches are slightly higher (10 per cent) than average in integration and system testing, but substantially lower (100 per cent) than average during the first month of operation. However, error data for the first month of operation was available only for a quarter of the projects using analyst workbenches.

Screen painters: Error data was available for 60 per cent of the projects using screen painters. Compared with other projects of similar size, these projects had about 20 per cent more errors in integration and system testing, but about 25 per cent fewer errors in the first month of operation.

Report writers and enquiry generators: The projects that used these types of function-generation aid had considerably more software errors than the average during the first month of operation. It is very likely, however, that the higher error rates are not directly associated with use of these aids. Error data was available for 60 per cent of projects using report writers. For their size, these projects had about 60 per cent more errors in integration and system testing and 65 per cent more in the first month of operation. Error data was available for 60 per cent of projects using enquiry generators. They had fewer-than-average errors (30 per cent fewer) in integration and system testing, but higher-than-average error rates in the first month of operation, which suggests that inadequate integration and system testing was carried out.

Data dictionaries: Error data was available for 60 per cent of the projects using data dictionaries. For their size, these projects produced about 40 per cent more errors than average, both in integration and system testing and in the first month of operation. This implies that the technical quality of the original development work for these projects is markedly lower than average. Seventy per cent of the Datamanager projects reported error data; they had near-average numbers of errors in integration and system testing, and about 20 per cent fewer errors in the first month of operation.

Projects using data dictionaries have above-average error rates

Project-management tools: Projects using project-management tools have error rates 20 per cent above the average (for the size of project) at integration and system testing, and 25 per cent above average during the first month of operation.

Testing tools: The projects using testing tools had slightly higher error rates both in integration and system testing (15 per cent higher), and in the first month of operation (10 per cent higher). Thus, although the tools may have helped to identify more errors, the reliability of the developed applications in the first month of operation was marginally worse than average.

Our attention so far in this paper has been on the technical aspect of projects. The PEP project data we have collected to date concerning the equally important areas of people and management is limited. Yet the results of our analyses of these aspects of systems development for this paper support and reinforce critical findings from previous PEP research. In the next chapter, we present the main results of those analyses.

Chapter 6

The impact of working environment and staff management on productivity and quality

During 1989, we began to correlate staff attitudes with project performance. The most striking observation from this work is that staff show increasing sensitivity and concern about project managers' behaviour and the office environment as the projects they work on increase in size.

We continue to find that low PIs are correlated with high rates of staff turnover. Our analyses of the level of staff experience and project performance confirm some long-held beliefs but also reveal some surprises — for example, that more experienced project managers are associated with falling PIs. We have also found evidence that software error rates are influenced by many non-technical factors.

STAFF CONCERNS INCREASE WITH PROJECT SIZE

Staff concerns about opportunities for participation (in project planning and management, for example), and about individual attention, increase with the size of projects, as measured by increasing functional-design and main-build time, by effort, and by peak manning.

There is no correlation between increasing staff concerns and PI levels

Larger projects are usually undertaken by larger teams. Opportunities for participation and for individual attention decrease as teams become larger, unless project managers are particularly aware of staff needs in these areas and act to meet them. However, although staff may have such concerns, we found no correlation between them and PI levels.

Interestingly, satisfaction with the support given by project managers decreases as project managers' experience increases. This supports the assertion made in previous papers (see PEP Paper 7, *Influence on Productivity of Staff Personality and Team Working*) that more experienced project managers are not as 'people-minded' as staff would like them to be.

Staff concerns about the availability of both personal office space and desk space also increase with project size. This suggests a continuing need for systems development managers to ensure that the office environment is satisfactory.

THE LEVEL OF STAFF EXPERIENCE DOES NOT ALWAYS CORRELATE WITH HIGH PI LEVELS

The data submitted for normal PEP assessments includes the length of staff experience, categorised under seven headings: overall experience, experience of working on a similar system, experience of the programming languages being used, experience with the computer being used, experience with the methods being

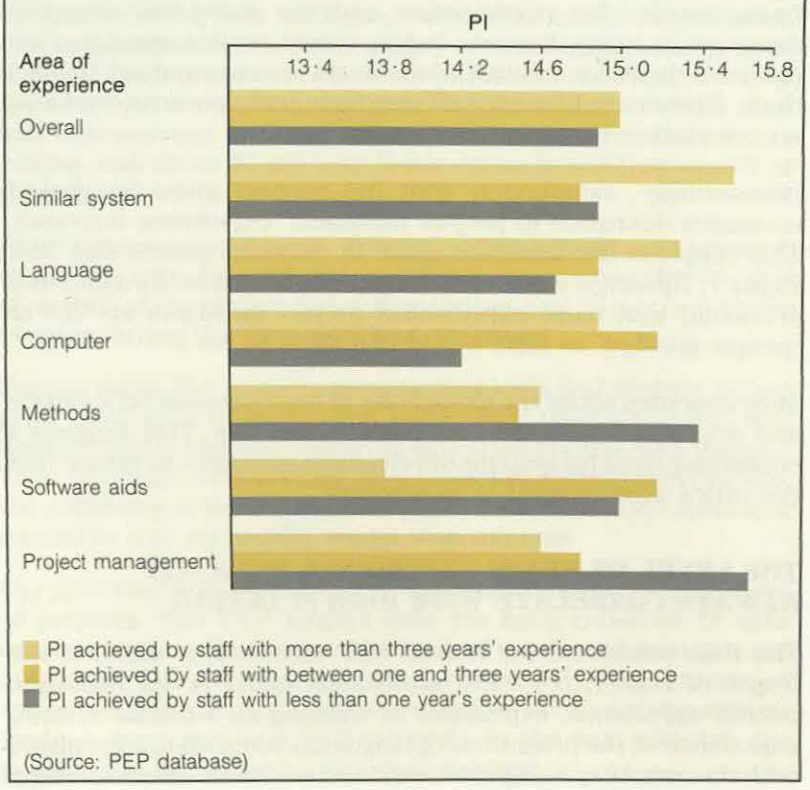
used, experience with the software aids being used, and the length of experience of the project manager(s). For each of these areas, staff are divided into three categories: those with less than one year's experience, those with between one and three years' experience, and those with more than three years' experience. Figure 6.1 summarises the average PIs of projects according to these experience categories. Figure 6.2 shows how the PIs vary from the average, taking account of project size. A high level of experience with similar systems is associated with high absolute and relative PIs. A high level of experience of using software aids and of the project-management team is associated with low absolute and relative PIs.

EXPERIENCE WITH SIMILAR PROJECTS TENDS TO INCREASE PRODUCTIVITY

The average PI for projects involving people with more than three years' experience with similar systems, at 15.6, is higher than the PEP average, and is over half a point higher than the average PI for other PEP projects of a similar size. The number of projects in this category is, however, fairly small, and the average should therefore be viewed with caution.

The results suggest that staff with a high level of experience of developing similar systems are more productive. If this finding continues to be supported by subsequent PEP data, it will indicate that this type of experience has a greater effect on PI than, say, language skill. Although PIs (absolute and relative) do gradually improve as the length of language experience increases, the

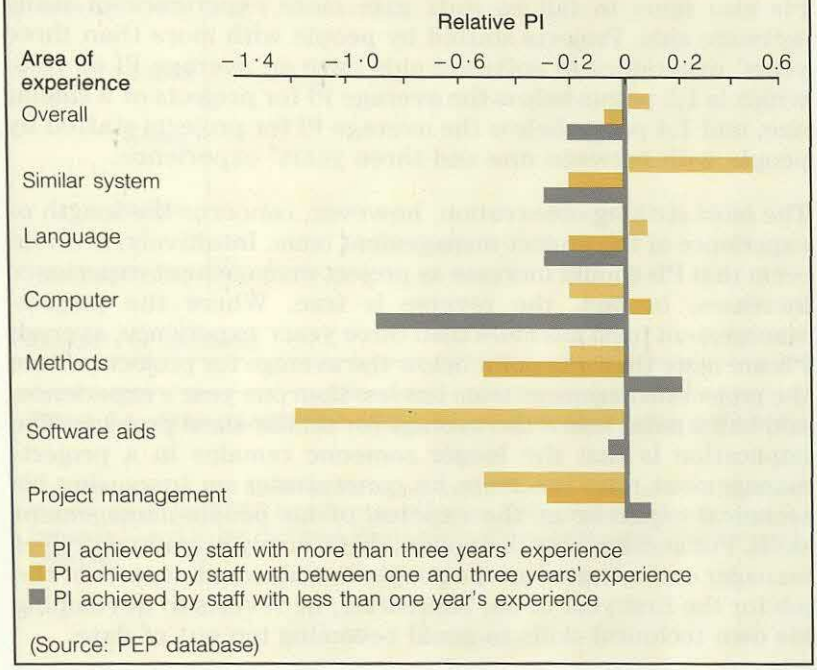
Figure 6.1 PIs vary considerably according to the experience of staff



(Source: PEP database)

Figure 6.2 Previous experience with similar systems results in higher PIs

The figure shows how the PIs achieved by staff with differing lengths of experience differ from the average PI for all PEP projects of a similar size. Thus, staff with more than three years' experience of working on similar systems achieve PIs half a point higher than the average for similar-sized systems.



(Source: PEP database)

improvement is not as marked as it is for increasing experience with similar systems.

Figure 6.1 shows that the average PI for projects where the staff have less than one year's experience of the computer being used is, at 14.2, one point lower than for projects where staff have between one and three years' experience, and 0.7 lower than for projects where the staff have more than three years' experience. This pattern is repeated in Figure 6.2, which shows the PIs of these projects relative to the average PIs of projects of similar size. The results are consistent with what would be expected. Most of the staff involved in the main-build stage of a project have to work quite closely with the technology. A lack of experience with systems software, for example, can seriously affect their performance.

EXPERIENCE OF USING METHODS AND SOFTWARE AIDS AND OF THE PROJECT-MANAGEMENT TEAM TENDS TO REDUCE PI

The average PI for projects staffed by people with less than one year's experience of using methods is about one point better than the average PI for projects staffed by people with more experience (although in the case of projects staffed by people with more than three years' experience of methods, the sample size is fairly small, and the average should therefore be viewed with caution). The projects staffed by people with the least experience of methods also have slightly higher average PIs than similar-sized projects.

This suggests that staff with the least experience of using methods may get the most benefit from them. As experience increases, methods appear to become more of a hindrance than a help. It is unclear whether this is due to inherent weaknesses in the methods, or whether staff prefer to work in a way that allows them more personal discretion as their experience increases.

PIs also seem to fall as staff gain more experience of using software aids. Projects staffed by people with more than three years' experience of software aids have an average PI of 13.8, which is 1.2 points below the average PI for projects of a similar size, and 1.4 points below the average PI for projects staffed by people with between one and three years' experience.

PIs decrease as length of project-management experience increases

The most striking observation, however, concerns the length of experience of the project-management team. Intuitively, it would seem that PIs should increase as project-management experience increases. In fact, the reverse is true. Where the project-management team has more than three years' experience, average PIs are more than one point below the average for projects where the project-management team has less than one year's experience, and half a point below the average for similar-sized projects. The implication is that the longer someone remains in a project-management role, the more he concentrates on increasing his technical expertise at the expense of his people-management skills. Put another way, it appears that a newly promoted project manager concentrates on the people-management aspects of the job for the first year or so. Thereafter, he reverts to developing his own technical skills to avoid becoming too out of date.

STAFF TURNOVER AND REQUIREMENTS CHANGES REDUCE PIs

It will come as no surprise to PEP members to hear that our analysis confirmed that high staff-turnover rates and high levels of requirements changes markedly reduce PIs.

STAFF TURNOVER

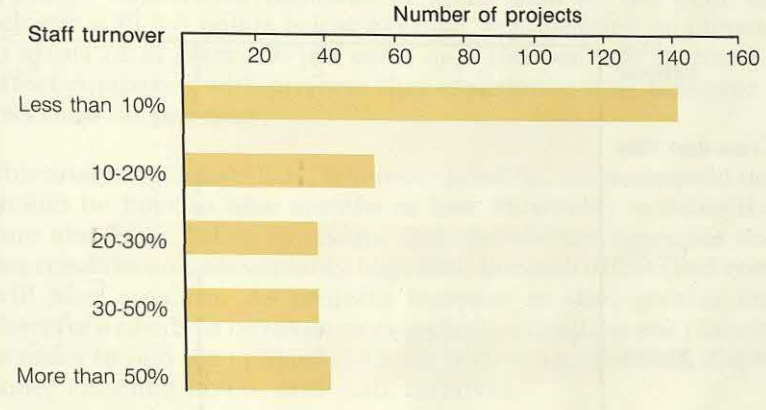
On average, the staff-turnover rate on PEP projects is between 10 and 20 per cent. Nearly 40 per cent of the projects have staff turnover in excess of 20 per cent. The distribution of projects by staff-turnover rate is shown in Figure 6.3.

Staff turnover in excess of 50 per cent increases effort by 60 per cent or more

Figure 6.4 shows how the PIs for projects with different levels of staff turnover compare with the average PIs for projects of a similar size. The figure clearly shows that higher rates of staff turnover adversely affect PIs. Staff turnover in excess of 20 per cent is associated with projects whose PIs are below average for the size of project, with the difference reaching 2.5 points when turnover exceeds 50 per cent. This means that staff turnover in excess of 50 per cent results in an increase in manpower effort of at least 60 per cent.

High staff turnover is also associated with projects that are more stretched out over time, which means that they have low MBIs. The average is confirmed by the distributions of MBIs, and of average duration of the main-build stage, by staff-turnover rate, which are shown in Figures 6.5 and 6.6, on page 56.

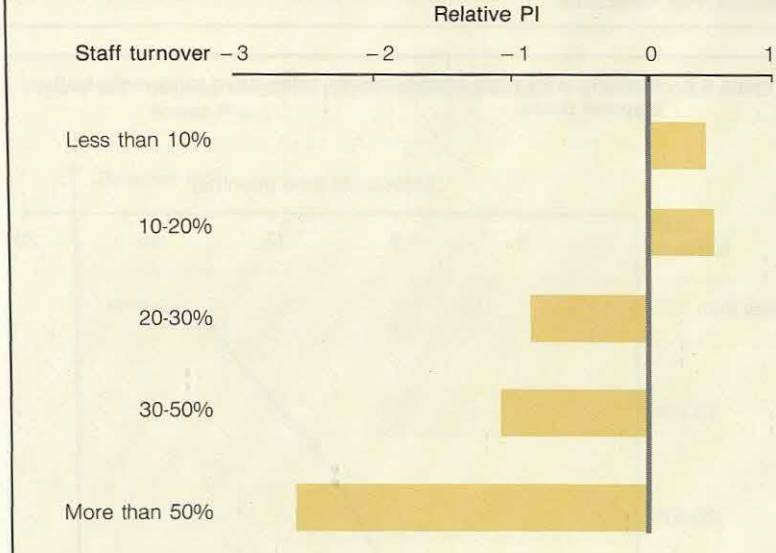
Figure 6.3 Average staff turnover on PEP projects is between 10 and 20 per cent



(Source: PEP database)

Figure 6.4 Projects with high staff-turnover rates have below-average PIs

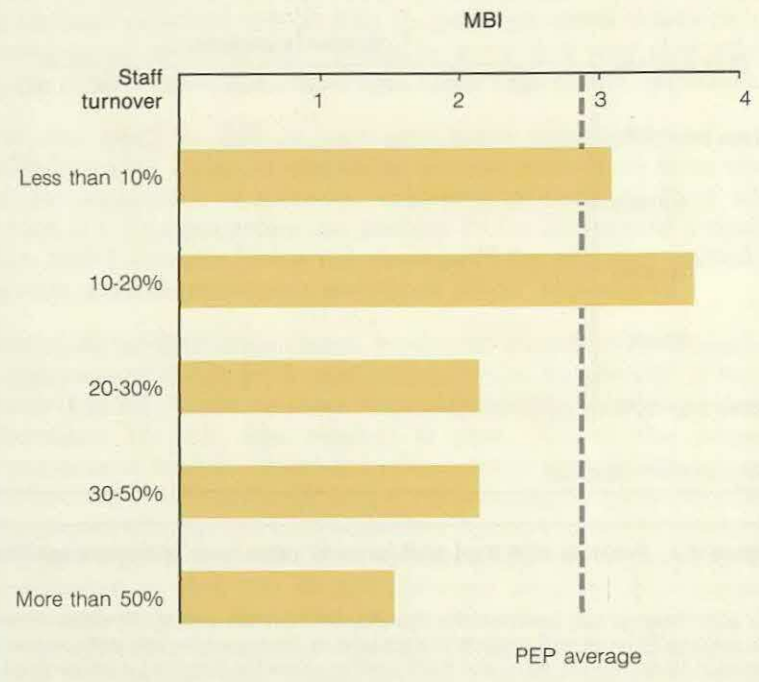
For each level of staff turnover, the figure shows how the average PI differs from the average PI for all PEP projects of a similar size. Thus, projects with staff turnover between 10 and 20 per cent have PIs 0.5 points above the average for similar-sized projects.



(Source: PEP database)

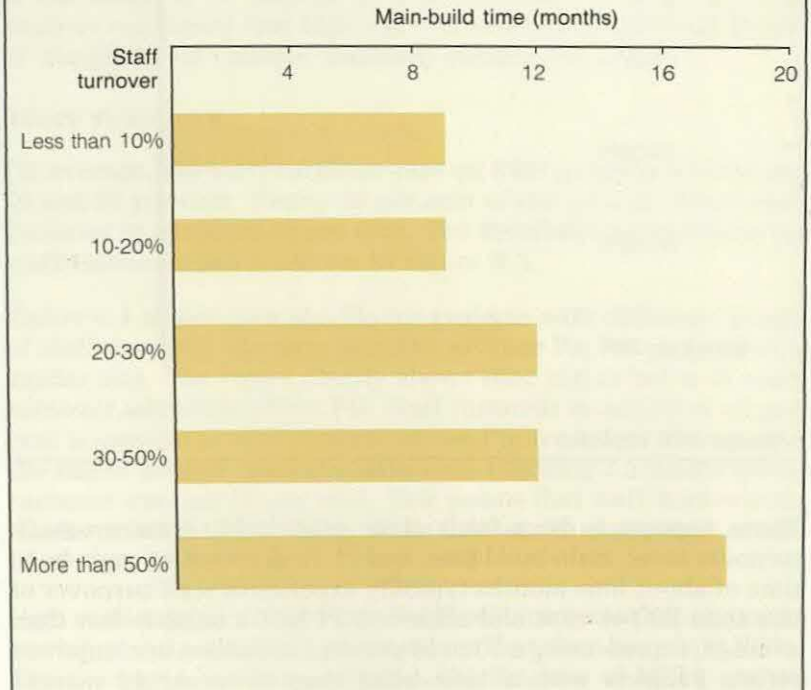
There appears to be a fairly close relationship between staff-turnover level, main-build time, and PI. Projects with a main-build time of about nine months typically experience staff turnover of less than 20 per cent and achieve a PI half a point better than average, representing a 10 to 15 per cent reduction in manpower effort. Projects with a main-build time of about 12 months typically experience staff turnover of between 20 and 50 per cent and achieve a PI one point below average, representing a 20 to

Figure 6.5 Projects with high staff-turnover rates have low MBIs



(Source: PEP database)

Figure 6.6 Projects with high staff-turnover rates have long main-build elapsed times



(Source: PEP database)

25 per cent increase in effort, and a 30 to 40 per cent increase in effort compared with projects with less than 20 per cent staff turnover. Projects with a main-build time of about 18 months typically experience turnover of more than 50 per cent and achieve a PI 2.5 points below average, representing an increase in effort of at least 100 per cent, or a 150 per cent increase in effort compared with projects that experience staff turnover of less than 20 per cent.

If possible, main-build time should be no longer than nine months

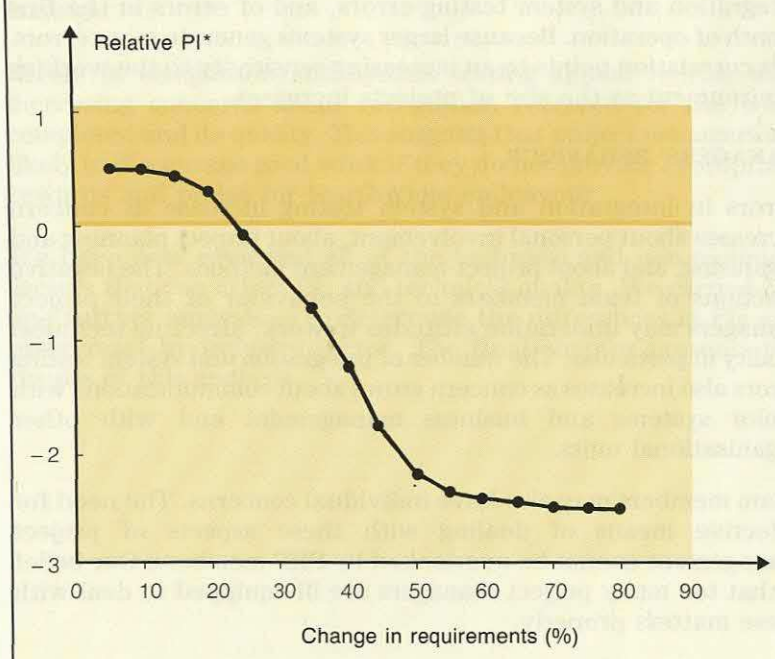
This analysis suggests that, wherever possible, the main-build time should be kept to nine months or less. However, in doing this, care should be taken to ensure that the shorter timescale does not result in an unacceptably high MBI, because effort (and costs) will also escalate. As projects increase in size, greater care therefore needs to be taken over project estimating and planning in order to find the optimum balance between project size, elapsed time, manning levels, and staff turnover.

REQUIREMENTS CHANGES

Increasing levels of requirements changes are associated with falling PIs

PEP members are asked to estimate the percentage change in requirements that occurs after the start of the main-build stage of a project. Our analysis shows that the requirements are changed for more than 80 per cent of the projects in the PEP database, with the average estimated change being about 20 per cent. As Figure 6.7 illustrates, there is a clear indication that increasing levels of requirements changes are associated with falling PIs. Above about 20 per cent, PIs start to fall below average, and by

Figure 6.7 Higher levels of requirements changes are associated with lower PIs



* Each point shows how the average PI for projects with a given level of requirements changes differs from the average PI of all projects of a similar size.

(Source: PEP database)

the time the level of requirements changes exceeds 50 per cent, the PI is over two points below average, representing an increase in effort of between 40 and 50 per cent.

The way in which requirements changes are managed and incorporated into existing code therefore has a significant impact on PI. The lower PIs of enhancement and maintenance projects is another indication of the difficulties that systems developers have in changing software.

TECHNICAL QUALITY IS ADVERSELY AFFECTED BY MANY NON-TECHNICAL FACTORS

We found many correlations between the concerns of staff and the number of software errors experienced by projects. In particular, these correlations point to the need for staff to be able to work without needless interruption, and for noise levels to be kept within acceptable limits. They also suggest that certain aspects of project managers' behaviour, concerns about team structure, and lack of personal responsibility and recognition may also increase error levels.

INTERRUPTIONS AND HIGH NOISE LEVELS

Concerns about noise increase as time pressures increase

Concern about being able to work without needless interruption rises as the number of errors in integration and system testing increases, and as the overlap between the functional-design and main-build stages increases. Increasing overlap is often associated with larger undertakings and with projects with high MBIs, where staff are more likely to have to work under more severe time constraints.

Concerns about noise levels increase with rising numbers of integration and system testing errors, and of errors in the first month of operation. Because larger systems generate more errors, this correlation points to an increasing sensitivity to the working environment as the size of projects increases.

MANAGERS' BEHAVIOUR

A negative response by staff to their project manager may affect the technical quality of their work

Errors in integration and system testing increase as concern increases about personal involvement, about project planning and organising, and about project-management methods. The negative reactions of team members to the behaviour of their project managers may undermine attitudes to work, affecting technical quality in particular. The number of integration and system testing errors also increases as concern grows about communications with senior systems and business management and with other organisational units.

Team members may also have individual concerns. The need for effective means of dealing with these aspects of project management cannot be overlooked by PEP members. Our belief is that too many project managers are ill-equipped to deal with these matters properly.

CONCERNS ABOUT TEAM STRUCTURE

Our analysis showed that there is a correlation between an increasing number of errors in integration and system testing, and

growing concern about team structures and skills in managing people. Staff are more sensitive to these concerns on larger projects, where interpersonal communication is likely to be more of a problem.

Concerns about team structure also increase as the overlap between the functional-design and main-build stages increases. Increased overlap, which is more often experienced by larger projects and by projects with tight deadlines, is associated with decreasing satisfaction with team size and with deteriorating relationships between team members. Increasing the overlap between the functional-design and main-build stages normally requires the team members working on functional design to be more closely involved with those carrying out the detailed design and programming. Increased sensitivity to team structure and relationships between team members are therefore more likely, and this seems to be borne out by the correlations we have found.

LACK OF PERSONAL RESPONSIBILITY AND RECOGNITION

Increasing numbers of errors in integration and system testing is also correlated with a growing concern about freedom and independence, which is one of the basic dimensions of a job that contributes to its potential to motivate. (In PEP Paper 7, we described the motivating potential of jobs and the basic job dimensions.) Ensuring that an individual's need for personal discretion and responsibility are met will certainly encourage better quality of work. IT staff, who usually have high needs for growth, are less likely to respond to a working environment where they are told exactly what to do and how to do it, and more inclined to want to learn for themselves. Achieving the right balance in managing such staff is no easy matter for a project manager.

Errors increase as concern about recognition for good work increases

Errors in integration and system testing appear to rise with increasing concerns about recognition received for the work completed and its quality. This suggests that project managers are likely to discourage good work if they do not provide appropriate rewards and praise for worthwhile endeavour.

We have now reviewed all of the technical and non-technical factors that can affect PIs and technical quality. We carried out one further analysis — to determine the differences in PIs and error rates by industry sector. The final chapter presents the results of this analysis.

Chapter 7

Productivity and quality by industry sector

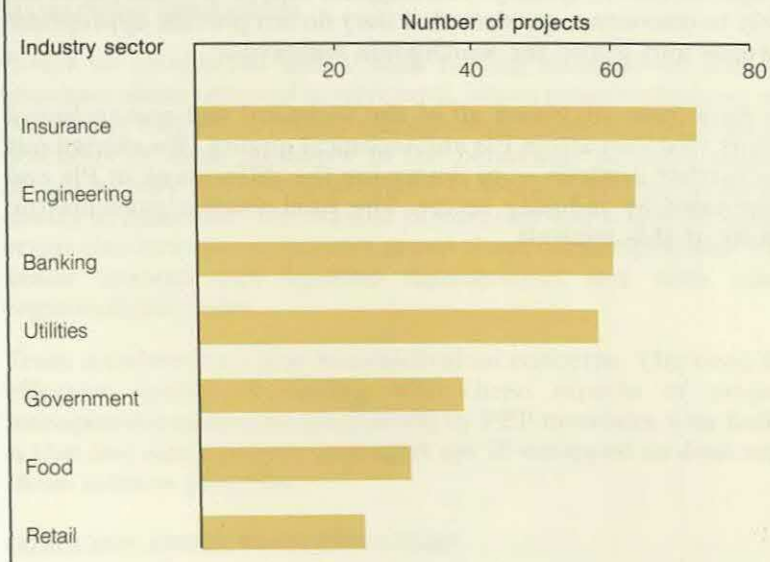
From the outset of PEP, members have wanted to know how they compare with other organisations in both the same and different industry sectors. It has not been possible to answer this question until now, when the PEP database has grown sufficiently to provide reasonable sample sizes. In this chapter, we present the results of our analysis of the PEP database by industry sector. There is no best-performing sector. Those that rate best on one measure do less well on another.

There is no best-performing sector

At the time of our analysis in the summer of 1989, the 344 projects in the database spanned seven main industry sectors (see Figure 7.1):

- *Insurance*: 72 projects, 66 of which were submitted by UK insurance companies.
- *Engineering:* 63 projects. This sector also includes electronics and oil companies, but there are too few projects in the latter two categories to justify separate analysis. Ten PEP members contributed the projects for this sector.
- *Banking:* 60 projects, about one-third of which were submitted by non-UK organisations.
- *Utilities:* 58 projects, submitted by seven PEP members.

Figure 7.1 PEP projects span seven main industry sectors



(Source: PEP database)

— *Government* (both central and local): 38 projects, about a third of which have been submitted by one organisation, which inevitably introduces a heavy bias.

— *Food producers*: 29 projects submitted by only three PEP members. The results therefore have to be treated with caution.

— *Retail:* 24 projects, submitted by six members.

The characteristics of projects in each sector are different

Projects in each of the sectors can be compared in terms of the six main measures — project size, PI, MBI, language gearing, function points delivered per man-month, and error rate. Figure 7.2 sets out the characteristics for projects by industry sector, and highlights the areas where they differ significantly from the PEP averages:

— Average *project sizes* range from 69,000 lines of code for food producers to about 33,000 lines of code for the insurance industry. Food producers' projects have a very high new-code content, which accounts for the fact that their projects are nearly 70 per cent larger than the PEP average. Insurance-industry projects are about 20 per cent below the PEP average of 41,000 lines of code.

— Taking account of the size of the projects, more than one-and-a-half PI points separate the top-performing sectors — insurance and food, from the lowest performing sector — utilities.

— However, projects submitted by utility companies have the lowest average MBI. The differences are very significant — 0.7

Figure 7.2 The characteristics of PEP projects vary by industry sector

Industry sector	Project size	PI	Relative PI[1]	MBI	Language gearing	Function points per man-month	Relative error rate 1 (%)[2]	Relative error rate 2 (%)[2]
Insurance	33,000	15.5	+0.6	3.4	16	16	+25	−15
Engineering	45,000	14.9	−0.3	2.6	14	15	−40	−15
Banking	37,000	14.1	−0.9	3.0	11	8	0	−20
Utilities	34,000	13.9	−1.0	2.1	15	16	−5	−25
Government	58,000	14.8	−0.8	3.1	15	9	+250	+160
Food	69,000	16.6	+0.6	2.5	17	21	+55	+50
Retail	42,000	15.6	+0.5	3.3	13	10	0	+5
PEP average	41,000	14.9	[1]	2.8	14	13	[2]	[2]

[1] The figures shown are relative to the expected PI for PEP projects of the average size for the industry sector. Thus, insurance industry projects have an average PI that is 0.6 above the expected PI for projects of 33,000 lines of code.

[2] The figures shown are percentages relative to the expected error rates for PEP projects of the average size for the industry sector. Thus, insurance industry projects have 25 per cent more errors between integration and system testing and first operational capability, and 15 per cent fewer errors in the first month of operation than expected for projects of 33,000 lines of code.

below the overall PEP average, and 1.3 below the insurance-sector projects (the high MBI of insurance projects is caused by the severe time pressures that have resulted from business changes and increasing competition).

- While food producers' projects, with a language gearing of 17, appear to have used fourth-generation languages to the greatest extent, the banking sector, with a language gearing of 11, has used them the least. Banks have submitted a larger proportion of enhancement projects, which is an indication of the maturity of their base of installed applications, and explains, in part, their limited use of fourth-generation languages.
- *Function-delivery rates* range from 21 function points per man-month for the food producers' projects to eight function points per man-month for the banking sector. The function-delivery rates are affected by PI, MBI, and language gearing. Thus, for example, the utility sector's above-average function-delivery rate of 16 function points per man-month, despite a low PI, is the result of slightly better-than-average language gearing and low MBI.
- Government-sector projects have error rates that are substantially higher than average for the size of project. (The data is, however, significantly influenced by one PEP member, whose projects are mainly enhancements to very large systems.) The high PIs of food producers' projects are offset somewhat by the higher-than-average error rates. Engineering-industry projects have the lowest error rates, having the smallest number of errors at the integration and system testing stage and below-average error rates in the first month of operation.

INSURANCE

Insurance-industry projects have relatively high MBIs

The most noticeable characteristic of insurance-industry projects is the relatively high average MBI — more than half a point higher than average. This difference represents about 25 per cent more effort than average. The high average MBI is no doubt associated with the significant amount of change that the UK insurance industry has experienced in the last few years, brought about by new legislation and increased competition.

At an average of 15.5, the PIs for insurance-industry projects vary more widely than in any other sector — a standard deviation of five points, compared with four for all PEP projects. This means that two-thirds of insurance-industry projects have PIs in the range 10.5 to 20.5. The average PI of insurance-industry projects is just over half a point higher than the PEP average for projects of the same size. This difference usually represents a reduction in effort of about 15 per cent. However, taking account of both the higher MBI and the higher PI, the insurance industry expends about 10 per cent more effort than average, owing to the tight timescale in which projects have to be developed.

The average size of insurance-industry projects is 33,000 lines of code, which is smaller than the PEP average. The language gearing of 16 function points per thousand lines of code, which is 14 per cent higher than the PEP average, is mainly a consequence of the

higher use of PL/1, rather than Cobol. The net effect of the higher-than-average PI and MBI, and near-average language gearing, is a function-point delivery rate of about 16 function points per man-month, which is comfortably above the PEP average (13).

ENGINEERING

Engineering-industry projects have low error rates

The projects submitted by engineering companies have an average PI that is the same as the overall PEP average, and an average MBI that is slightly lower than the PEP average. At an average of 45,000 lines of code, the projects are about 10 per cent larger than the PEP average. They have average language gearing, and a function-delivery rate of 15 function points per man-month, compared with the PEP average of 13.

The projects are notable for their lower error rates in integration and system testing. This does not result in below-average reliability, however — the average number of errors in the first month of operation is also less than the PEP average. This suggests that the original development work was of above-average quality.

BANKING

Banking projects are notable for their low language gearing, low function-delivery rate, and low new-code content. At an average of 37,000 lines of code, they are also a little smaller than the PEP average.

Banks have the least new code of all industry sectors

Language gearing, at 11 function points per thousand lines of code, is the lowest for any industry sector. This reflects the relatively limited use of fourth-generation languages in banking projects. The amount of new code is also the lowest of all the industry sectors. The low new-code content and low language gearing are typical of organisations with significant investments in well-established systems, where a high proportion of the work is maintenance and enhancement.

Banking projects have an average PI of 14.1, which is nearly one point below the PEP average for projects of a similar size. Variations in PI are very high — a standard deviation of five, implying that 67 per cent of banking projects have PIs in the range nine to 19. The average MBI of three is near to the PEP average. All these factors result in a function-delivery rate of about eight function points per man-month, the lowest average for any industry sector.

Nearly 20 per cent of the banking-sector projects in the PEP database are minicomputer projects, with an average size of 27,000 lines of code, and an average PI about two points below the overall PEP average. These projects have made a significant contribution to the banking sector's measures being below the PEP averages.

UTILITIES

Utility-company projects have the lowest average PIs of any sector

With an average PI of just below 14, utility-company projects have the lowest average PIs of any sector. The average PI is also one point below the average for projects of a similar size, which is the largest variance of any sector. At 2.1, the projects also have the

lowest average MBI. Language gearing of 15 function points per thousand lines of code is close to the overall PEP average, however. The low MBI compensates for the low PIs, resulting in an average rate of delivering function of 16 function points per man-month, which is above the overall PEP average (13).

GOVERNMENT

The averages, particularly for MBI and software error rates, for the projects submitted by PEP members in the government sector, are influenced significantly by one organisation. The average MBI of 3.1 is slightly higher than the overall PEP average. However, the average MBI for the projects submitted by the one organisation is about 4.5, more than double that of the remaining government-sector projects.

Error rates for projects in the government sector are substantially above average

The error rates are substantially above average (3.5 times higher than average at integration and system testing, and 2.6 times higher in the first month of operation), and particularly so for the projects submitted by the largest government-sector contributor. This member's projects are mainly enhancements to very large applications.

Government-sector projects have an average PI of 14.8, which is 0.8 below the PEP average for similar-sized projects. The near-average language gearing of 15 function points per thousand lines of code is not sufficient to offset the high average MBI and lower-than-average PI. The result is a low rate of delivering functionality — an average of nine function points per man-month, scarcely better than that achieved by the banking sector.

FOOD PRODUCERS

An average language gearing of 17 function points per thousand lines of code for food-producers' projects is the highest for any industry sector. This is a consequence of the greater use made of fourth-generation languages, and languages such as RPG. About 25 per cent of the code for this project is in fourth-generation languages, compared with about 10 per cent for PEP as a whole.

The projects submitted by food producers are mostly new developments, and at an average of 69,000 lines of code, are the largest of any sector. The average PI is 16.6, which is above the PEP average and more than half a point higher than the average for projects of a similar size. Moreover, the average MBI at 2.5 is 0.3 lower than the PEP average.

Food producers' projects deliver function at the highest rate of all industry sectors

The combined effect of high PI, lower-than-average MBI, and high language gearing results in food producers' projects delivering function at an average rate of about 21 function points per man-month, the highest rate of any industry sector.

About 40 per cent of food producers' projects are minicomputer developments averaging 87,000 lines of code. Unlike the minicomputer projects in the banking sector, all of these projects perform very well. Their average PI is 19, which is about 2.5 points higher than the average for this size of project.

Despite the high average PI and below-average MBI, the food producers' projects have higher-than-average error rates both at integration and system testing and in the first month of operation.

RETAIL

Projects submitted by retailers have above-average MBIs

The most striking characteristic of the projects submitted by retailers is the above-average MBI, which at 3.3, is half a point above the PEP average. At 15.6, the retailers' projects have slightly higher-than-average PIs. This PI is also about half a point above the average for the size of projects submitted. As in the insurance sector, the net effect of the higher MBI and higher PI is about 10 per cent more effort per thousand lines of code than average.

The average size of retailers' projects, at 42,000 lines of code, is near to the overall PEP average, while language gearing, at 13 function points per thousand lines of code, is just below average. The projects make lower-than-average use of fourth-generation languages — less than 10 per cent of code. This is probably due to the lower proportion of new code in the projects, and has the effect of lowering the average rate of delivering functionality to about 10 function points per man-month, about 25 per cent below the PEP average of 13.

This concludes our first analysis of the PEP database. We have identified the trends in the main measures used in PEP, and the correlations between the variables. We believe that these findings will in turn help PEP members to identify their systems development practices that help or hinder productivity and quality. However, our analysis has not been able to identify the cause and effect relationships that lead to the correlations. This will be the subject of future PEP research.

BUTLER COX

P,E,P

Butler Cox

Butler Cox is an independent international consulting group specialising in the application of information technology within commerce, industry and government.

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.

The services provided include:

Consulting for Users
Guiding and giving practical support to organisations trying to exploit technology effectively and sensibly.

Consulting for Suppliers
Guiding suppliers towards market opportunities and their exploitation.

The Butler Cox Foundation
Keeping major organisations abreast of developments and their implications.

Multiclient Studies
Surveying markets, their driving forces and potential development.

Public Reports
Analysing trends and experience in specific areas of widespread concern.

PEP

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
1 Managing User Involvement in Systems Development
2 Computer-Aided Software Engineering (CASE)
3 Planning and Managing Systems Development
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

Forthcoming PEP Papers
Software Testing
Software Quality Measurement
Selecting Application Packages
Project Estimating and Control

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) 6 75 51 11, Fax (020) 6 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, 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, New Zealand and South-east Asia
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
Sinikalliontie 5, 02630 Espoo, Finland
☎ (90) 358 0502 731, Fax (90) 358 05022 682

Ireland
SD Consulting
8 Clanwilliam Square, Dublin 2, Ireland
☎ (01) 764701, Fax (01) 767945

Italy
RSO SpA
Via Leopardi 1, 20123 Milano, Italy
☎ (02) 720 00 583, Fax (02) 86 45 07 20

Scandinavia
Butler Cox Foundation Scandinavia AB
Jungfrudansen 21, Box 4040, 171 04 Solna, Sweden
☎ (08) 705 83 60, 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