Quality Assurance in Systems Development

BUTLER COX
P,E,P

PEP Paper 9, February 1989

BUTLER COX

P,E,P

Quality Assurance in Systems Development

PEP Paper 9, February 1989
by Martin Langham

Martin Langham

Martin Langham is a senior consultant with Butler Cox, specialising in the development and implementation of technically advanced computing applications. He has a particular interest in database techniques, distributed computing, and data communications. He has been involved in a wide range of consultancy assignments for both manufacturing and service companies, and was co-author of the Butler Cox Foundation Report *Managing the Evolution of Corporate Databases*.

Prior to joining Butler Cox, Martin Langham was with BIS ASL, where he was responsible for consulting in the area of distributed systems, and gained considerable experience in the insurance and banking industries. His early career was with ICL and UNISYS as a consultant supporting major customers in both the public and private sectors.

He has a BSc in physics from Bristol University and is a member of the British Computer Society. He has lectured and published widely in the trade press.

Martin Langham was assisted in the research for this report by Les Green, a consultant at Butler Cox, who specialises in the design and implementation of computer systems and who has a particular interest in the retail sector.

Published by Butler Cox & Partners Limited
Butler Cox House
12 Bloomsbury Square
London WC1A 2LL
England

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

BUTLER COX

P,E,P

Quality Assurance in Systems Development

PEP Paper 9, February 1989
by Martin Langham

Contents

Chapter 1

A new solution to quality assurance is needed

The need for companies to guarantee the quality of their goods and services has never been more important. Customers are increasingly demanding better-value products, measured not only in terms of price, but also in terms of price/performance and quality. In Europe, this need was confirmed in September 1988 by the signing of letters of intent by the presidents of 14 European multinationals to establish the European Foundation for Quality Management. Their intention is to spread the message to fellow manufacturers and service industries across Europe that quality management must be extended beyond the traditional approach of a separate quality-assurance function charged with checking that products meet specified standards. These organisations believe that, to improve competitiveness, quality management must spread to every part of the organisation and that every employee must be responsible for the quality of his or her work if the organisation is to improve competitiveness. Systems development departments are not, of course, exempt from these demands.

THE NEED FOR HIGH-QUALITY SOFTWARE IS GREATER THAN EVER BEFORE

The importance of high-quality software is growing

The importance of developing and maintaining high-quality software is growing, as systems are used much more widely throughout an organisation, and in areas that are much more critical to the business, and much more visible to the customer. Quality is therefore essential, not only in terms of the accuracy of information, but also in terms of reliability and speed of response. In addition, there will be a need to ensure that all types of applications are developed to consistent standards of quality. In the past, the only people to suffer from the poor quality of a standalone application were the users of that application. In the future, all types of applications will have to be more closely integrated with an organisation's core application systems, and they will therefore need to be developed to similar standards of quality.

Systems of very indifferent quality are still being produced

In spite of these growing pressures for high quality in application systems development, systems of very indifferent quality are still being produced. In particular, there is a great deal of evidence that time, money, and effort are being spent on systems that bear little resemblance to the product that the customer actually wants. In March 1988, a survey conducted by the UK Government estimated that British companies spend £500 million each year correcting software errors and maintaining software that has been inadequately quality-controlled at the design stage. A survey of nine US federal projects costing $6.3 million reveals a similarly unacceptable state of affairs. Half of the software was delivered but never used; a quarter was paid for but not delivered;

one-fifth required major modification before it could be used. Software that was used as delivered, or that required only minor modification, accounted for only one-twentieth of the total.

PEP members also recognise that current practices do not guarantee that high-quality applications software will be produced. Figure 1.1 lists some of the quality problems mentioned during the research. Clearly, there is a need to re-examine the concepts of quality assurance and quality control in the context of systems development.

The concepts of quality need to be re-examined

Figure 1.1 Quality problems encountered during recent systems development projects

User dissatisfaction	Many complaints about minor deviations from the specification Users have difficulty defining the deliverables Quality compromised to meet time and cost constraints
High levels of change following implementation	Unforeseen business changes Inaccurate estimates of transaction volumes Specification changed in order to meet an earlier deadline
Poor operational performance	Operational use underestimated Use of fourth-generation languages reduces operational performance
Low technical quality	Hardware limitations lead to poor user interface Technical quality taken account of too late in the development process
Other	Misunderstandings between development staff and users, particularly about timescales and costs Poor documentation Poor project management

(Source: Survey of PEP members)

TRADITIONAL CONCEPTS OF QUALITY ASSURANCE ARE INADEQUATE

The concepts generally used to achieve quality in manufacturing and the service industries are also widely used by software developers. They are based on the traditional principles of quality assurance and quality control. According to British Standard 4778, quality assurance is defined as "All activities and functions concerned with the attainment of quality", and quality control is "The operational techniques and activities that sustain the product or service quality to specified requirements; it is also the use of such techniques and activities".

Software developers use the same quality concepts as manufacturing and service industries

In the context of systems development, there are two aspects to the attainment of quality — product quality-control procedures to ensure that individual systems are developed to appropriate levels of quality, and the quality of the systems development process itself. The characteristics of these two aspects are set out in Figure 1.2.

In the past, systems quality-control procedures focused on the product by checking that a completed computer system met the

Figure 1.2 Attaining quality in systems development means that attention must be paid to the quality both of the development process and of individual projects

Development process	Individual projects
The process is designed to build high-quality software products	Quality control checks that software products meet predefined criteria
The same process applies to all projects	Quality-control checks are carried out for each project
Quality control is carried out entirely by systems staff	Quality-control procedures involve non-systems staff
Costs are incurred only in designing the process, not for each project	Costs are incurred for each project

original specification. The techniques used include system and acceptance testing, and a post-implementation review. Few, if any, quality checks were carried out at intermediate stages of the development process. This approach to systems quality assurance is concerned only with checking that the final system meets the original requirements, and not with the overall process by which the product is developed. The result is often that the system delivered meets neither the users' requirements nor their expectations, in terms of functionality, operational performance, usability, development cost, and delivery date. The defects discovered when the final system is inspected are often caused by mistakes made at the early stages of the development process — the requirements-definition stage, for example.

Traditional systems quality-control procedures are concerned only with the application being developed

To overcome the shortcomings of this approach to assuring systems quality, many systems development departments have been encouraged, by the availability of methods and tools, to concentrate on improving the effectiveness of the development process. These methods and tools make it possible to enforce a standard approach to development and make it easier to check the quality of the software at various stages of its development. In this way, the quality of the software being developed can be checked at each stage. The stages of the life cycle required to complete a project, from initiation to completion, are precisely defined, as are the deliverables to be produced at each stage. The deliverables can then be checked before development staff proceed to the next stage. Instead of the whole product being inspected once at the end of the process, it is reviewed in smaller, more easily examinable pieces, during its development. In this way, defects can be detected earlier and corrected before the software is delivered to the users.

More recently, quality assurance has been concerned with the development process

The role of many systems quality-assurance departments today is to define the development process that will be used and to carry out the quality-control checks at the end of each development stage. The development procedures, and the procedures for carrying out the checks, are usually defined in great detail and enshrined in the 'systems development standards manual'. The quality-assurance staff themselves are perceived as 'policemen', whose main role is to ensure that the procedures are followed and that those who break the rules are identified.

Many organisations have found this an inadequate approach to improving the quality of systems. Indeed, many would argue that the existence of such quality-assurance departments hinders, rather than helps, the development of new systems; all that has been achieved is the creation of an additional layer of bureaucracy concerned with enforcing standards, ensuring that rigid procedures are followed, and insisting that lengthy checklists are completed.

Quality-assurance departments sometimes hinder, rather than help, new systems development

The difficulty arises because traditional systems quality-assurance concepts are based on too narrow a definition of systems quality. The procedures described above are concerned with ensuring that the final system has the specified *functionality*; this is insufficient to ensure that the system meets the users' *real* needs. Other equally important aspects of quality, such as the quality of the user interface, the operational performance of the system, the ease with which the system can be modified to meet changing business requirements, and the quality of the documentation, are largely ignored by conventional approaches to systems quality assurance.

For the purposes of this paper, we therefore define a high-quality system as one that "conforms to users' expectations, in terms of its functionality, its operational performance, the way in which it is constructed, its ease of use, and the documentation that is supplied with it". Two features of this definition are particularly important: the emphasis on users' expectations, and the fact that quality is not limited to the software itself. It is the entire package — the software, documentation, manuals, training, and user-support — that determines the users' satisfaction and thus their perception of the quality of the software. This definition, does of course, encompass traditional concepts of systems quality — in particular, the need to produce software to budget with the minimum number of errors. However, the emphasis on the way the software is constructed is recognition of the fact that quality in software is also a matter of how easy it is to modify and extend systems to meet changing business requirements, and how well systems can meet performance criteria.

A broader definition of quality is required

AN APPROACH BASED ON A BROADER DEFINITION OF SYSTEMS QUALITY WILL PRODUCE BETTER RESULTS

Adopting a broader definition of systems quality will, of course, have wide-ranging implications both for the systems department and for the organisation as a whole. It will not be possible simply to extend the responsibilities of conventional systems quality-assurance departments to include all the aspects of our wider definition of systems quality. This would require an even larger number of 'policemen' to enforce the quality-control procedures and checks, which would be unacceptable, both in terms of cost, and in terms of the bureaucratic demands that it would place on systems development staff.

Instead, we recommend that existing quality-assurance departments be replaced by a small quality-management group whose prime role is to be the driving force for creating a 'quality culture' both within the systems department and within the departments that use its services, and to act as the quality 'champion' by

overseeing a quality-management programme. One of the group's early tasks will be to create a set of guidelines that will form the basis of all the department's efforts to improve the quality of the systems it develops. These guidelines will provide advice about when and how to use systems quality-control techniques, but they will extend far beyond these to cover the broader aspects of quality that we discussed above.

The drive for better-quality systems depends on the commitment of individual staff

It will not be the responsibility of the quality-management group to carry out quality-control checks itself nor, indeed, to ensure that the guidelines are being followed; on the contrary, the group will arrange for as much of the responsibility as possible to be devolved to project managers and their teams. The department's drive for better-quality systems must centre on making individual development staff responsible for producing quality output. For this approach to be successful, everyone in the systems department must therefore be committed to it, and take responsibility for the quality of his or her own contribution to systems development. The advantages of such an approach are obvious. It is not, however, easy to achieve in practice.

PURPOSE AND STRUCTURE OF THE PAPER

In this paper, we propose an alternative to the conventional systems quality-assurance approach, based on our broader definition of systems quality. Our proposals take the form of 'best practice' guidelines because, as yet, there is no comprehensive method that takes account of all of the aspects of systems quality implied by our definition. The guidelines should be used by systems managers who have responsibility for the development and implementation of application systems, and for ensuring their quality.

In our proposed approach, the quality-management group is the key to the success of a quality-management programme based on our extended definition of quality. In Chapter 2, we describe the shortcomings of existing systems quality-assurance concepts and show how the current narrow focus on functionality needs to be expanded to include operational performance, technical quality, and the user interface. In Chapter 3, we explain how the case for establishing a quality-management programme can be made. The four main steps are to review the existing situation, to define the objectives of a new approach, to devise a plan for achieving them, and to estimate the costs and benefits that can be expected.

The role and responsibilities of the quality-management group are described in Chapter 4. Its main tasks are to define quality guidelines and development standards, to support development staff in the quest for quality, and to establish a programme for measuring the quality of application systems. Without such a programme, it will be impossible to monitor the success of the quality-management programme.

RESEARCH SOURCES

We conducted telephone interviews, based on a structured questionnaire, with 14 organisations forming a representative sample of PEP members. (The questions asked are listed in

Figure 1.3.) In addition, we interviewed 15 employees in three PEP member organisations, about their experience of ensuring quality in software. The questions were similar to those used in the telephone survey but a much greater emphasis was placed upon the individual's experience of implementing quality programmes within his or her organisation. The overall experiences of these three organisations are set out as case histories later in this paper.

Figure 1.3 Questions asked during telephone interviews with PEP members

What do you believe to be the objectives of quality in systems development?
What do you believe to be the main methods of quality assurance?
What do you believe to be the main methods of quality control?
Do you currently measure the quality of your applications? If so, how, and who measures them?
Has quality been a problem in recently completed applications?
Have you recently attempted to improve quality in systems development:
- By forming a quality-assurance group?
- By adopting new methods, tools, or organisational structures?
- By educating development staff and users?
- By adopting quality plans?
- By adopting quality techniques?
- By adopting quality measurements?
- By improving testing procedures?
- By formalising customer relationships?
- By other means?

What benefits do you expect to achieve by improving quality:
- Improved user satisfaction?
- Lower maintenance costs?
- Lower computer costs?
- More effective applications for the organisation?
- Easier-to-use applications?
- Longer-lasting applications?
- Lower development costs?
- Other benefits?

How would you measure the expected benefits?

We also reviewed the published literature on the topic of systems quality assurance to ensure that we were fully conversant with recent developments in the theory and practice of quality programmes.

Chapter 2

Systems quality concepts need to be extended

Before examining the shortcomings of existing systems quality-assurance concepts, and suggesting ways of extending them, it is instructive to review the more general standards that already exist in the field of quality assurance, and to see how these might be applied in the context of applications software development.

GENERAL QUALITY-ASSURANCE STANDARDS CAN BE APPLIED TO SOFTWARE DEVELOPMENT

National quality-assurance standards are widely used

Various national and international organisations have defined general quality-assurance standards. In the United Kingdom, the most widely used of these is British Standard 5750 Part 1, which is identical to ISO Standard 9001. These standards are entitled "Quality Systems — Model for quality assurance in design/development, production, installation and servicing". (Note that the word 'systems' here is used in its widest sense; it does not mean information systems.) They specify the minimum requirements in 19 different areas, including:

- *Quality system*: To comply with the standards, organisations must establish, document, and maintain an effective and economical quality system to ensure that their products conform to the specified requirements. The system must include quality-management objectives, policies, organisation, and procedures.
- *Organisation*: The responsibility for functions affecting quality must be delegated to specific personnel, who must have the authority to identify and evaluate quality problems, and to initiate, recommend, and provide effective solutions.
- *Review procedures*: The quality system must be periodically and systematically reviewed to ensure its continued effectiveness.
- *Work instructions*: The organisation must develop and maintain clear and complete documented instructions that prescribe how the requirements are to be communicated to those performing the work.
- *Records*: Records must be developed and maintained to demonstrate that the required quality is being achieved and that the quality system is operating effectively.

British Standard 5750 Part 1 is based on the concept of a separate quality-management group that is responsible for developing quality procedures and guiding staff in the use of the procedures.

Other standards relate specifically to the achievement of quality in software. In the United Kingdom, the best known is probably DEF STAN 00-16, published by the Ministry of Defence, and entitled "Guide to the Achievement of Quality in Software".

Although it provides specific guidance on the methods and procedures necessary to establish confidence in the quality of software used in military applications, it is likely to be useful in other types of computer applications too.

DEF-STAN 00-16 begins by defining the precontractual arrangements (specifying the customers' requirements, the proposal that responds to those requirements, the procurement specification, software life-cycle management planning, and planning for software quality). It sets out codes of practice and software quality-assurance procedures under the following headings:

- Project management.
- Design techniques and methods.
- Programming techniques and methods.
- Software development facilities.
- Documentation.
- Configuration management.
- Design review.
- Tests and trials.
- Transfer to customer.
- Post-design services/maintenance.
- Subcontractors.

In this quality standard, the emphasis is on ensuring that systems development standards are adhered to and that the software produced meets the agreed functional specification. It is written in the form of a series of checklists to be followed, and therefore provides pragmatic advice on the steps that can be taken to assuring quality within the systems development department. However, its scope is narrower than that of BS 5750 Part 1. We believe that to perceive systems quality simply in terms of ensuring that the functional requirements are met is too restrictive. The wider view, envisaged in BS 5750, is more appropriate.

SYSTEMS QUALITY DEPENDS ON MORE THAN MEETING FUNCTIONAL REQUIREMENTS

Today, most systems quality-assurance procedures are designed to ensure that the functionality provided by applications software meets the users' requirements. However, even where the quality of the system is checked at intermediate stages of the development life cycle to ensure that the finished product does meet the functional requirements, it may still be regarded as being of poor quality by the user community. This is because the quality-assurance procedures do not take account of the users' needs in other areas — operational performance, ease of use, and the ease with which the system can be modified are obvious examples. Figure 2.1 describes how one organisation with a conventional quality-assurance function is realising that it needs to take a broader view of systems quality.

Quality-assurance procedures need to deal with more than functional requirements

Analyses of users' expectations for applications software have been carried out by Barry W Boehm and his colleagues. In their

Figure 2.1 Conventional quality assurance is not sufficient to guarantee the quality of systems

Insurance company

This organisation has a large systems department, with more than 100 development staff. The quality-assurance function resides within the systems department and is staffed by three people — a manager and two project managers. There is a well established quality-assurance culture, based on a comprehensive code of practice that covers methods, techniques, and the use of tools. The code of practice was developed four years ago and is updated annually.

At the start of a project, user expectations are documented on a project-authorisation form. This includes a quality plan, although in most cases, the plan indicates only that the code of practice will be followed. Systems development staff believe that a more detailed statement of quality objectives is desirable because it would enhance the value of post-implementation reviews.

The quality-assurance staff are invited by a user or a departmental manager to review application systems at regular intervals. An initial review can take several weeks, and the actions agreed are followed up later. Because of the number of development staff, the quality-assurance staff are very busy. However, they usually carry out reviews when a problem is detected rather than at predetermined stages in the development life cycle, even though they realise that scheduled reviews are a better means of detecting problems earlier in the development process. External consultants have also been employed, with considerable success, to review particular projects, both from a business and a technical viewpoint.

The quality-assurance manager considers that quality assurance based solely on controlling and reviewing the development process is insufficient to provide the quality required — in particular, for the business aspects of a system. The existing procedures mean that insufficient attention is paid at the beginning of a project to issues such as the feasibility of changing work practices in user departments. Many of the quality-control reviews carried out at present are concerned with technical issues — for example, program walkthroughs require up to 25 per cent of the programming effort. To progress beyond this to a wider quality-management programme, senior management must lend their support to giving quality assurance greater prominence throughout the organisation. This support is now being sought.

The quality-assurance manager told us that his aim is to make users responsible for quality in systems development projects by providing them with a code of practice and making them accountable for the business success of the projects. He believes that, when the wider quality-management programme is in place, his department will need fewer staff because the quality-assurance process will be an integral part of the whole organisation.

early work, *Characteristics of Software Quality* (published by Oxford: North Holland in 1978), they identified a large number of software characteristics that contribute to users' overall perceptions of software quality.

There are four characteristics of software quality

We believe that four of these characteristics are particularly important: functional requirements, operational performance, technical features, and ease of use. By defining and meeting quality objectives specified in terms of these characteristics, it will be possible to build application systems that the user community regards as high-quality. Although the functional requirements of a system are generally defined in great detail, the other three characteristics are often ignored in systems specifications. These characteristics are usually determined by ad hoc decisions made at the analysis and programming stages.

FUNCTIONAL REQUIREMENTS

Functional requirements define what the application system has to do

The functional requirements define what the application system has to do, down to the level of describing the data to be entered, the rules for deciding whether to accept or reject the data, and the processing to be performed once the data has been accepted. Most systems specifications contain adequate functional requirements, and it is relatively straightforward to assess the quality of a system in terms of how well it meets these.

OPERATIONAL PERFORMANCE

The operational-performance characteristics of a system define the expected performance in terms of response times (for online systems), and the elapsed time required to perform specific processing loads for batch systems. If these characteristics are defined at the outset, the quality of the final system can be assessed against them. However, the objectives should be set bearing in mind special factors that will degrade performance, such as peak processing loads or changes in workload.

Performance is defined in terms of response times and elapsed time required to perform processing loads

TECHNICAL FEATURES

The technical features of a system relate to the way the software itself is constructed. The technical quality of a system can be specified in terms of its mean time between failures, the ease with which it can be maintained and extended, how easy it is to change the basic system by parameters specified at run time, for example, and how easy it is to re-use parts of the software in other applications. Checklists should be constructed for each of these characteristics, and used to assess the technical quality of the software. Figure 2.2 shows a sample checklist for assessing how easy it will be to extend a particular application.

Technical features relate to the way software is constructed

Figure 2.2 Technical quality: a checklist for assessing how easy it will be to extend a system

This checklist can be used to assess how easy it will be to modify or extend a system's existing computational and/or data-storage limits (field sizes, record length, file sizes, and so on).

System characteristics indicating that modifications or extensions will be easy

- The system allows key parameters to be modified at run time. It should also validate the run-time entries to ensure they are within allowable boundaries.
- The documentation adequately describes what constraints of the system may be altered and how to do it.
- The system specifically tests for each code that can be input to the system, so that any code not explicitly recognised by the system is rejected.
- There are enough fields of an adequate size to allow for reasonable growth.

System characteristics indicating that modifications or extensions will be difficult

- Parameters are coded into the program logic.
- Files are sequential or index-sequential.
- Low-level protocols are used for network communications.
- Incompatibilities between system modules have been resolved by linking them via specially written programs.

(Adapted from: *"The Quest for Quality"*, published in *Datamation*, March 1, 1985)

EASE OF USE

The increasing use of PC-based software packages by the user community has raised users' expectations considerably about ease of use. Despite this, mainstream applications are still developed that users find boring, tedious, or difficult to use. A poor or inadequate user interface can mean that a system is regarded as being of poor quality even though it meets all of the functional requirements, has high operational performance, and is technically sound.

The characteristics of the user interface are a determinant of quality

The quality of a system therefore depends also on its user-interface characteristics. For example, the quality of the user

interface might be specified in general terms as one that provides clear, unambiguous messages for users, that requires the minimum number of keystrokes to be used, that provides a rapid response time, and that has simple, unambiguous error-recovery procedures. These general terms can then be defined in more detail. Clear messages for users might be defined in terms of clear command prompts, and the existence of a help facility, a tutorial mode, a terse mode, audio responses, and pointers to the most likely next activity. Specifying the user-interface characteristics in these terms will allow the quality of the user interface to be defined and assessed.

DIFFERENT APPLICATIONS HAVE DIFFERENT QUALITY CHARACTERISTICS

The full requirements for an application system can be defined in terms of the four types of system characteristics described above, and the extent to which these requirements are met provides an indication of the quality of the system. It is important to remember, however, that users' perceptions of quality are determined largely by their expectations. Different types of application are used by different types of user with different expectations. The implication is that the relative emphasis given to each of the four types of system characteristics will vary according to the type of application. For some types of application, its quality will be judged largely on the quality of the user interface; for others, it will be judged largely on the technical quality of the software.

Different types of application will have a different 'quality profile'

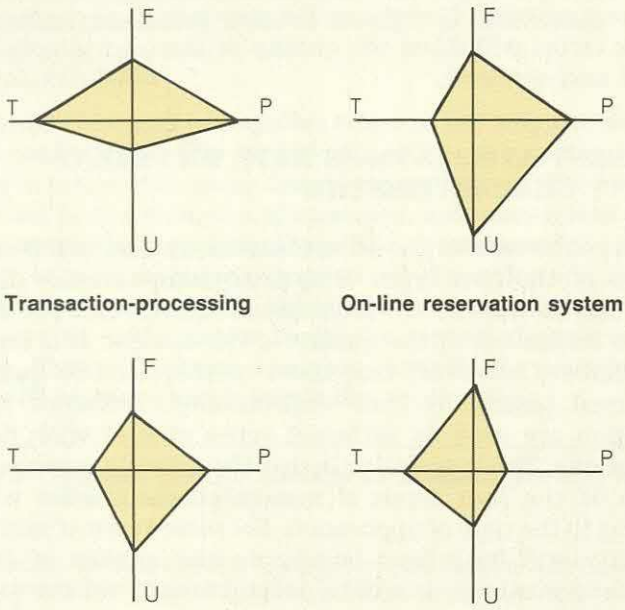
Different types of application will therefore have a different 'quality profile', which can be expressed diagrammatically, as shown in Figure 2.3, overleaf. Transaction-processing applications, for example, require a high level of technical quality, and high levels of operational performance, whereas the quality of an accounting package is determined much more by how well it meets the functional requirements and by the quality of its user interface.

The different quality profiles also imply that different emphases are required on checking the quality of the software product being produced and on assuring the quality of the development process itself. Ensuring that the software meets the functional requirements requires a heavy emphasis on quality-control checks as the software is developed. High technical quality and good operational performance are determined more by the quality of the development process. Figure 2.4, overleaf, shows the relative emphasis on product and process quality required for each of the four system characteristics.

In general, greater emphasis on product quality will increase the cost of developing an application because it will be necessary to carry out a greater number of, and more extensive, quality-control checks. Greater emphasis on process quality means that substantial initial effort is put into defining a formal development process and ensuring that it is followed. However, emphasising process quality will result in better-designed and more flexible software.

Figure 2.3 Different types of applications have different quality profiles

The characteristics of a system can be expressed in terms of the functional requirements, operational performance, technical quality, and ease of use. The quality of a system can be assessed in terms of how well the software matches these characteristics. The relative emphasis of each of the characteristics will be different for different types of application.



F = Functional requirements
P = Operational performance
T = Technical quality
U = Ease of use

Figure 2.4 Different system characteristics require different emphases on product and process quality

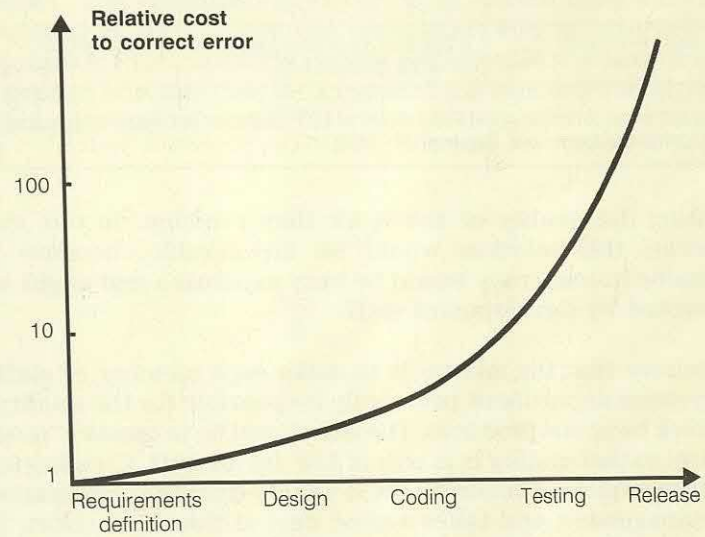
System characteristic	Relative emphasis on:	
	Product quality	Process quality
Functional requirements	★★★	★
Operational performance	★★	★★
Technical quality	★	★★★
Ease of use	★	★★★

QUALITY ASSURANCE SHOULD BE APPLIED THROUGHOUT THE DEVELOPMENT LIFE CYCLE

Most systems development departments realise that it is not sufficient to check the quality of applications software once only, at the end of the development life cycle. Errors or mistakes discovered as a system is implemented may have been caused by an error made right at the beginning of the development process, and will be very expensive to correct because much of the work

already done will have to be redone. Barry Boehm states in *Software Engineering Economics* (published in 1982 by Prentice Hall) that the cost of correcting an analysis error at the maintenance stage is 100 times more than the cost of detecting and correcting the error immediately. Other research indicates that the cost of correcting an error made early in the life cycle increases exponentially, the longer it remains undetected (see Figure 2.5).

Figure 2.5 The cost of detecting an error made at the requirements-definition stage increases exponentially as the development life cycle progresses



(Source: Computer Weekly, June 23, 1988)

Quality assurance should be applied at each stage of the development life cycle

These problems can be overcome by applying quality assurance at each stage of the development life cycle. To achieve this means that the stages of the life cycle must be clearly defined, so that quality checks can be carried out at the end of each stage. In this way, errors can be detected as they occur and can be corrected at minimum cost.

The deliverables at the end of each stage should be specified in detail and should reflect the *four* types of system characteristics described earlier in this chapter. The quality of the application system being developed can then be assured by checking that the work delivered conforms to the specification. Figure 2.6, overleaf, shows the deliverables that may be specified for various stages of the life cycle.

DEVELOPMENT STAFF SHOULD BE RESPONSIBLE FOR THE QUALITY OF THEIR WORK

The extended concepts of systems quality described earlier in this chapter imply that far more quality-control checks will have to be carried out than has typically been the case. One solution would be to increase the number of staff employed in conventional quality-assurance departments and make them responsible for ensuring that development staff are obeying the rules and for

Figure 2.6 Deliverables must be specified for each stage of the development life cycle so that quality checks can be made

Development stage	Sample deliverables
Feasibility study	Project scope; analysis of current system; project plan and justification
Logical system design	System flowchart; logic charts; illustrative outputs
User procedure design	User manuals and examples
Computer procedure design	File layouts; test requirements
Program design	Structure chart; source code; object code; test results
System evaluation	Estimated vs actual (costs, timescales, effort, benefits, and so on)

(Adapted from: "Improving the Productivity of EDP Systems Development", published in *Systems Development*, September 1988)

checking the quality of the work they produce. In our view, however, this solution would be unworkable, because the inevitable bureaucracy would be very expensive and might well be resisted by development staff.

We believe that the answer is to make each member of staff in the systems department personally responsible for the quality of the work he or she produces. The aim should be to create a 'quality culture' so that quality is 'a way of life' for all staff. Creating such a culture requires a commitment to quality from the organisation's top management and takes a good deal of time and effort, but it produces two main benefits: the quality of the products is improved, and the cost of assuring quality is minimised, because it is not necessary to employ a vast army of quality-control inspectors.

The aim should be to create a 'quality culture'

The advantages of a quality culture (albeit it in a manufacturing environment) are illustrated by the experience of Datsun at its car assembly plant in the north-east of England. Datsun has deliberately set out to create a quality culture at this factory, and produces cars that are of a high quality. It does this with a smaller proportion of quality-control staff than most other car manufacturers. Japanese companies in general have built quality cultures by ensuring that staff have the opportunity to work on different stages of the production process. Doing this ensures that staff experience all facets of the work and, eventually, gain knowledge of the complete process. This means that, when they are working at a particular stage, they understand the consequences of defects introduced at earlier stages. It also means that they are in a better position to make recommendations for improving the process, and are able to check the quality of the product at each stage in the process.

Suppose, for example, that the production process has five stages (A, B, C, D, and E). Staff working on stage B would be in a position to review the output from stage A; staff working on stage C would be able to review the output from both stage A and stage B; staff working on stage E would be able to review the outputs from

All staff are involved in checking the quality of a product at all stages

stages A, B, C, and D. The cycle is completed because the output from stage E can be reviewed by the staff who work on stage A. If this concept is taken to its logical conclusion, there is no need to employ separate quality-control inspectors because all staff are involved in checking the quality of the product at all stages. The major benefit of this approach is that the staff working on the production process no longer perceive quality to be the responsibility of a separate quality-control group.

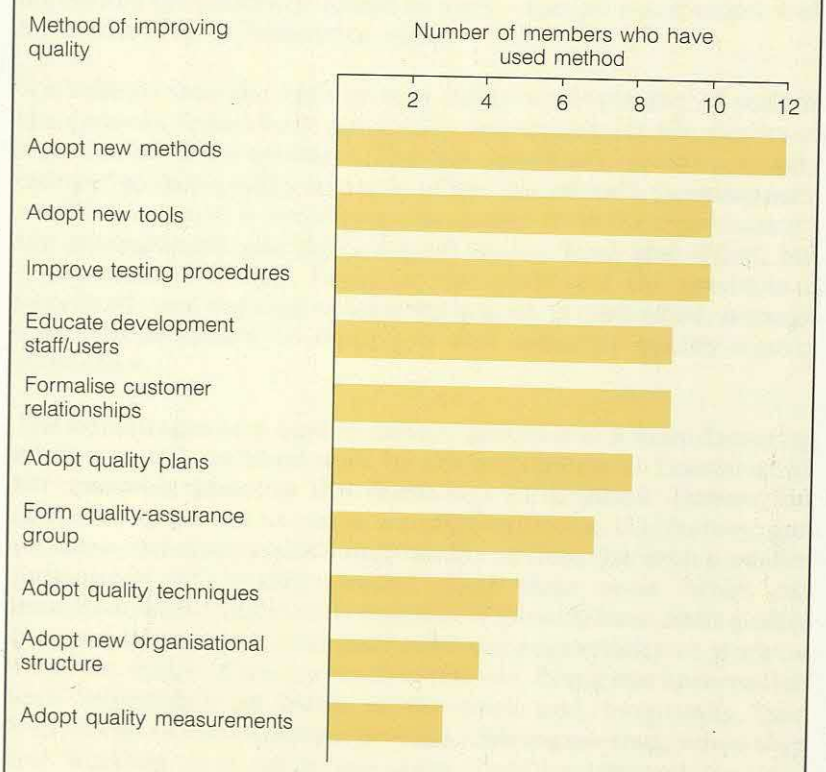
In a systems-development context, the way to apply these principles is to establish a quality-management programme, which is coordinated and administered by a quality-management group. Quality-control techniques will still be used as part of the programme. The emphasis, however, is on encouraging individual development staff to use the techniques and to take personal responsibility for producing quality software. In the next chapter, we explain how the case for a establishing a quality-management programme can be made.

Chapter 3

Making the case for a quality-management programme

Establishing a quality-management programme means changing reporting lines and the way people do their work. These changes take time to implement and to become effective, and initially, cause extra costs to be incurred. As a result, it may not be possible to achieve an immediate payback from a quality-management programme, and many organisations therefore find it difficult to make the case for introducing one. Instead, as Figure 3.1 shows, they concentrate on attaining greater software quality through the use of systems development techniques and tools, and fail to tackle the larger, and more fundamental, organisational issues.

Figure 3.1 In recent systems development, PEP members have selected various ways of improving quality



(Source: Butler Cox survey of PEP members)

Nor is it sufficient to establish a quality-management programme for the systems department alone. Unless the department's 'customers' (that is, the user community) have the same overall commitment to quality, and understand the tradeoffs that can be made between quality and costs or timescales it will be difficult, if not impossible, to develop high-quality application systems. A quality-management programme for the systems department,

Ideally, a quality-management programme should be corporate-wide

therefore, should ideally be part of a corporate-wide quality-management programme. Figure 3.2 describes the experience of an organisation where this has been achieved.

Figure 3.2 Ideally, the quality-management programme for systems should be part of a wider corporate programme for quality

Mineral exploration group

Quality is a strategic objective for all the business activities of this organisation. A steering group is responsible for quality throughout the group, and a manager is responsible for the quality-management initiative in each company. He or she is the focal point for quality, managing a network of quality-management groups, and establishing new ones wherever they are required.

We talked to the IT quality-management group that is staffed by a full-time quality manager and five staff. Finding suitable staff for quality management is difficult because they need to have a range of skills. Both personal characteristics and communications skills are important as quality-assurance staff need to influence and persuade people. They also need wide business and technical expertise in order to be credible and to be able to initiate management changes.

The IT quality-management group provides two major services to IT projects — advice on project start-up and a quality review. The quality-assurance programme is itself reviewed in the light of the feedback received from both types of service.

Project managers are not obliged to use the project start-up service, although they usually do because they are given sound advice about planning a new project. The member of the quality-management group who works with the project manager is himself an experienced project manager. The start-up service is based on a checklist of questions that is reviewed to ensure that all tasks have been identified and that all project risks have been assessed.

Quality reviews are carried out during the project by two people — a quality-assurance specialist and an independent expert. The review takes a week: six weeks after the review, the quality-management group checks that the agreed actions have been carried out.

The manager of the IT quality-assurance group told us that a major objective was to move towards a quality-management approach that would embrace all business functions — not only projects that have an identifiable goal. The aim is gradually to change the culture in the business so that management is increasingly aware of the need for quality.

The IT project managers recognise the value of quality assurance to their projects. Assistance with planning at project initiation and the ability of the quality-management group to resolve difficult problems by getting senior management involved are quoted as particular advantages.

Senior management commitment is essential

For these reasons, senior management in an organisation must be fully committed to the establishment of a quality-management programme. The case for a programme must therefore include indications of the organisational implications, the timescale of implementing the programme, and the likely costs and benefits, which may be difficult to define. We recommend a phased approach that begins with a review of the shortfalls of the current situation.

REVIEW THE EXISTING SITUATION

The first step in making the case for establishing a quality-management programme is to review the existing procedures and processes used for software development to identify those aspects of a quality programme that exist and those that do not. To perform the review effectively, staff experienced in project management, quality assurance, and software development are required. If the relevant experience is not available in-house, external consultants should be engaged. The review should cover both development issues and the wider aspects of software quality within the organisation.

Those carrying out the review should identify whether quality-assurance and quality-control techniques are used elsewhere in the company and, if so, what measurements of quality are used. They should examine the effectiveness of the planning and control mechanisms used for systems development, identify the tools and methods used and the life-cycle stages used in the development process, and assess the current departmental structure, highlighting both its strengths and its weaknesses. The types of development projects undertaken should be broadly classified in terms of size, application type, complexity, and team size, because it is these that determine the organisational structure and development process that will be appropriate.

The quality (or lack of quality) of application systems produced by the existing procedures and processes can be assessed in terms of time and cost overruns, the amount of rework and maintenance required, and the degree to which systems meet users' requirements. It is likely that the current approach is failing to meet the organisation's expectations in one or more of these areas. Specific examples should be quoted to support the case for a quality-management programme — the number of errors found per thousand lines of code, for example. Evidence such as this provides an indication of the inadequacy of the current approach, although it does not necessarily mean that a quality-management programme will solve the problem. The case will be stronger if evidence can also be supplied of where errors originally occur in the life cycle and what the cost is of allowing them to remain undetected until much later in the life cycle. A quality-management programme will allow the errors to be detected soon after they occur, thereby reducing substantially the cost of correcting them (see Figure 2.5).

The current approach is probably failing to meet some of the organisation's expectations

DEFINE THE OBJECTIVES OF THE QUALITY-MANAGEMENT PROGRAMME

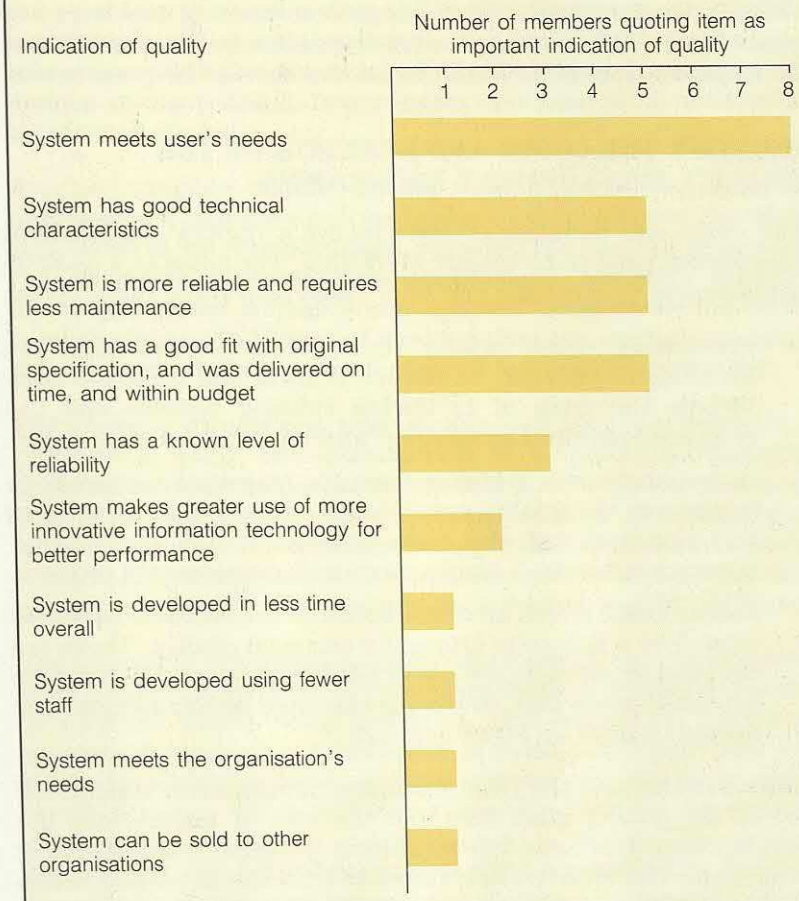
The findings of the review need to be analysed against the perceived lack of quality in application systems. The aim is to identify those areas of the current development procedures and processes that are the root cause of poor quality, so that a quality-management programme can be designed to address those areas. As Figure 3.3 shows, different organisations have different perceptions of what constitutes a quality system although overall, the need to meet users' requirements is regarded by PEP members as particularly important.

The aim is to identify the root cause of poor quality

Thus, for many organisations, the main objective of a quality-management programme will be to produce systems that are a better fit with users' requirements. For others, the objective will be to produce more reliable systems that require less maintenance. For some organisations, particularly those that produce software that will be sold as commercial products to government departments, the main objective in establishing a quality-management programme might be to demonstrate that they are complying with a national or international standard for quality assurance.

Meeting these objectives will no doubt imply certain changes and these must be clearly identified. They are likely to be mainly of an organisational nature — in particular, the need to create a

Figure 3.3 Delivering a system that meets the user's needs is considered the most important indication of quality



(Source: Butler Cox survey of PEP members)

quality-management group. (The role and responsibilities of such a group are described in detail in the next chapter.) The requirements for additional staff or funds should also be identified.

Sometimes, the changes may be required to demonstrate conformance with national or international quality-assurance standards. Even where this is not the case, the proposed changes should identify the quality-assurance standards that are to be adopted. In other organisations, the changes might be concerned with the introduction of new systems development techniques and tools. These, too, should be described when making the case for establishing a quality-management programme.

DEVISE AN IMPLEMENTATION PLAN

The implementation plan must have the full backing of development staff

The plan for implementing the quality-management programme should identify the major activities to be performed, the time they will take, and the resources required to carry them out. The programme should be applied initially to pilot projects. Once the lessons from these pilots have been learnt and the programme amended accordingly, it can be extended gradually to cover all

projects. It is essential that the proposed implementation plan has full backing and support from all the development staff, from its inception. Inevitably, the quality-management programme will require development staff to adopt new working practices and procedures. The sooner potential resistance to the changes can be reduced, the easier it will be to implement the programme.

IDENTIFY THE COSTS AND BENEFITS OF THE QUALITY-MANAGEMENT PROGRAMME

The costs of a quality-management programme are of three main types

The costs and benefits of introducing a quality-management programme need to be clearly identified. The costs of a quality-management programme can be divided into three main types:

- Prevention costs, which result from action taken to investigate, prevent, or reduce defects and failures. They include the costs of reviewing existing systems and the resources required to develop and update the programme.
- Assessment costs, which include the resources necessary to implement the quality-control procedures (such as reviews and planning) and the costs associated with the quality-management group's involvement with development projects.
- Failure costs, which are those incurred to overcome problems caused by a failure to attain the required quality. These are the costs incurred by the use of additional resources to reduce extended timescales, overcome customer dissatisfaction, and provide higher levels of support.

Other costs include the cost of training development staff in the use of the quality guidelines and the cost of maintaining the development standards. The total costs of a quality-management programme therefore include far more than the direct staff costs of the quality-management group. A paper in the January 1982 edition of *Communications of the ACM* estimates that once the requirements of a quality-management programme are imposed, the cost of a development project can double. This stems largely from the requirement to produce project documentation that achieves the desired standard of quality. However, the paper does not take account of the lower maintenance costs that should result from better-quality documentation.

Considerable benefits can be gained from a quality-management programme

There are considerable benefits, both tangible and intangible, to be gained from a quality-management programme. The tangible systems-development benefits come in two main forms: reduced life-cycle costs and reduced support costs. Once a quality-management programme is established, the resulting applications software should require substantially less maintenance effort, which, today, can represent 50 per cent or more of total life-cycle costs.

Contrary to the evidence quoted above, which suggests that a quality-management programme can increase some elements of development costs, many organisations report that the total life-cycle costs, and timescales, are reduced by establishing such a programme. British Rail, for example, reports that, two-and-a-half years after the introduction of a quality-management programme, total development costs were reduced by between 26 and 40 per cent, and development timescales by 16 per cent. The apparent

contradiction arises, we believe, because of different definitions of what is included as part of development activities. A quality-management programme usually requires more time and effort to be put into the earlier stages of the development process. The benefits are gained at the coding, testing, and maintenance stages, where time and effort are reduced substantially because the better designs created contain fewer errors that have to be resolved at these later stages.

Another tangible benefit arising from a quality-management programme is that the resulting higher-quality software can often be re-used in other applications, reducing the cost and improving the reliability of subsequent applications. Software modules developed under a quality-management programme can often form the basis of a library of modules that can easily be re-used, thereby reducing the overall costs of systems development.

For those organisations that supply software products on a commercial basis, the introduction of a quality-management programme that enables them to conform with quality-assurance standards may enable them to broaden their customer base. Increasingly, software purchasers, particularly in the government sector, are insisting that their suppliers can demonstrate that they comply with quality-assurance standards such as BS 5750 Part 1.

The intangible benefits of a quality-management programme arise from better control of the development process, leading to systems of a known quality being produced. This means that the whole process of producing applications is more predictable and less risky. In addition, better-quality systems will result in more-satisfied users, which will bring benefits both to the systems department and to the organisation as a whole. The systems department will benefit because it will be regarded by the user community as more professional and more responsive to its needs. The organisation will benefit because the user community will be more willing to use IT in general and will actively seek new ways of exploiting computer systems for the good of the business.

In summary, the costs and benefits of a quality-management programme are much wider than the direct costs associated with a quality-management group and the reductions that can be achieved in software development and maintenance costs and timescales. The intangible costs and benefits are at least as significant. The key to establishing a successful quality-management programme is to create a quality-management group. We turn now to discuss the role and responsibilities of such a group.

Chapter 4

A quality-management group is the key to success

The quality-management programme should be run by a small group of people — the quality-management group. Typically, there will be no more than two people in the group for a systems development department of up to 50 people, and three or four should be sufficient in a department of 200 or so development staff. The role of the group is to ensure that quality guidelines are established and translated into specific actions for individual projects. While a quality-management group for the systems development department can produce benefits in its own right, it will be even more effective if it is just one element of a corporate-wide quality-management programme.

The quality-management group is a small group of people

In effect, the quality-management group acts as the 'quality champion' within the systems department. Its role is not to police the work of development staff to check that their work meets the relevant standards. Rather, it is to be instrumental in creating a quality culture throughout the department and to advise project managers and their teams on producing software of increasingly high quality. The existence of the group does not, in any way, absolve development staff from their responsibilities for producing high-quality work. On the contrary, it is the responsibility of every person within a project team to ensure that his or her work is of the highest possible quality, at every stage of the development process. Figure 4.1 describes the experience of one organisation that has applied these principles successfully.

It acts as the 'quality champion'

To perform its role effectively, the quality-management group must be independent of individual development teams and projects and must report at a sufficiently senior level, so that no conflicts of interests are created. It is therefore preferable for the group to report to the systems director, rather than to the systems development manager. In this way, it will be less easy to compromise the quality of an application in order, for example, to reduce the development timescales in response to commercial pressures, and the quality-management group will have greater credibility with the user community.

It must report at a high level

The members of the quality-management group must also have the status and authority to earn the respect of development staff at all levels. The best way to achieve this is to use staff who have already established themselves as seasoned and successful systems developers. In the past, it has been difficult to assign such staff to conventional quality-assurance departments because, at best, they saw the assignment in terms of moving from being a 'poacher' to a 'gamekeeper', and at worst, saw it as the end of their active systems development careers. This difficulty can be overcome by assigning the best development staff to work in the quality-management group for 12 to 18 months, before returning to work on mainstream development activities.

Members of the group must have the respect of development staff

Figure 4.1 Development staff should be responsible for checking the quality of their own work

Retail organisation

The managers of this organisation, and in particular, the IT director, consider quality to be extremely important. The systems department has invested considerable effort in improving the quality of all aspects of development work — having recently completed an 18-month project to tailor a proprietary development method to suit its own environment. The documentation describing the method and how to use it is made available to development staff online, using a text-retrieval system.

Prior to introducing the method, this organisation used quality-control checks on the finished applications. These were carried out by separate operations-assurance and systems-assurance groups in the systems department. These groups have now been disbanded because they created a bottleneck in the development process and were expensive to run.

The aim now is to devolve the activities of quality control and assurance to project teams, so that project managers are responsible for systems quality. The development method defines, for each stage of the project, the specialised systems groups that have to be consulted to review and approve the documents produced. These groups include user representatives as well as the data administrator, the systems architect, and representatives from the information centre and operations function. The task of the systems architect is to ensure that the application being developed fits into the overall application architecture.

The documents are considered in formal reviews, each of which lasts for approximately one-and-a-half hours. Approval must be given at the review before the project can proceed to the next stage. The review process is popular with development staff because problems are identified earlier and users are able to gain a better understanding of the application as it develops. Changes initiated as a result of a quality-assurance review are handled as part of a change-control process. Feedback from the reviews can also result in the development method itself being amended. The documentation is reviewed every six months so that any changes can be incorporated.

The quality-management programme in this organisation is still evolving. One difficulty is that business demands sometimes take precedence over quality. However, by involving users in quality assurance, users' expectations are better managed, and quality and cost trade-offs are better understood by users.

However, it is essential that the staff assigned to the group are not re-assigned temporarily to work on development activities. Because of the nature of their skills, there will be a temptation to use them to resolve difficult, immediate, and urgent problems. This temptation must be resisted. It is important that the group is seen to be a critical part of the systems department and that its staff are committed and dedicated to it.

Members of the group must be dedicated to it

The activities of the quality-management group fall into three main areas — defining the guidelines for achieving quality (including the definition of systems development procedures and standards), supporting development staff in the achievement of quality, and measuring the quality that is actually achieved. The measurements will be used to review the effectiveness of the quality-management programme at regular intervals, and to make recommendations for improving it.

DEFINITION OF QUALITY GUIDELINES

The primary role of the quality-management group is to define the quality guidelines that form the framework within which all applications are developed. They provide development staff with clear rules for achieving quality in their work. They are also the basic set of quality requirements against which the quality-management group can evaluate the development of specific applications.

Quality guidelines are the framework within which all applications are developed

The quality guidelines are drawn up by the quality-management group, but they are implemented and used by the individual project managers. Project managers use the guidelines to prepare a quality plan (that is, a plan for achieving quality in the application about to be developed), which is submitted to the quality-management group for approval. In effect, the quality plan for a project forms a contract between the project team and the quality-management group.

The quality plan forms a contract between the project team and the quality-management group

The users of the system about to be developed also need to be involved in the preparation of the quality plan. The quality guidelines are translated into a quality service agreement between the project team and the users. This agreement specifies the characteristics of the system in terms of its functionality, operational performance, technical content, and user interface, using terminology that is meaningful to the users. It is then used by the development staff to carry out quality-control checks as the project progresses through the various stages of the development life cycle. Individual quality service agreements do not necessarily have to include every item described in the guidelines. Only the subset of the guidelines relevant to a particular project need be considered.

The quality service agreement forms a contract between the project team and the users

Thus, the achievement of quality is inextricably linked with the systems development process itself. For this reason, the quality-management group should also be responsible for defining systems development standards. The quality guidelines will therefore cover the following areas: systems development life cycle, development standards and tools, organisational relationships, and quality-control techniques.

SYSTEMS DEVELOPMENT LIFE CYCLE

The guidelines should indicate the stages of the development life cycle and the likely deliverables at the end of each stage. The proportions of the total effort and elapsed time to be spent at each stage should also be specified. When the guidelines are translated into a quality plan and quality service agreement, the precise deliverables (including internal documentation) and budgets for that project will be defined.

The guidelines should define the likely deliverables for each stage of the life cycle

DEVELOPMENT STANDARDS AND TOOLS

The development standards to be used within the systems department should be defined as part of the quality guidelines. The standards must cover all stages of the development life cycle and must be applicable to all types of application. When defining the development standards, it is useful to consider the requirements of formal quality-assurance standards (BS 5750 Part 1/ ISO 9001, for example) and incorporate aspects of these into the development procedures and standards.

Development standards cover all stages of the life cycle and all types of application

The standards should cover any conventions to be adhered to, such as field-naming or library-naming standards, and version numbers. They should also define the change-control procedures to be followed, at each stage of the life cycle, for error reporting and fault correction. In addition, the standards should describe the development techniques and tools to be used at each stage of the life cycle.

Many organisations have standards that are rarely used

The difficulties involved in defining usable software-development standards should not be underestimated. Many organisations have expended a lot of effort defining standards that are rarely used. There are a variety of reasons for this, but the most common are:

- Senior managers in the systems department do not support the use of the standards.
- The standards do not cover all life-cycle stages or development activities.
- The standards are impractical or over-restrictive.
- The standards require huge amounts of documentation to be produced.
- The instructions for using the standards are inadequate.
- The standards are obsolete because there is no procedure for reviewing and updating them.

The quality-management group needs to give careful thought to overcoming these difficulties. For example, the development staff who will use the standards should be involved in defining them, and clear instructions on how to use the standards should be available to all development staff, with advice about when certain standards are, and are not, applicable. The standards will need to evolve if they are to remain useful. The quality-management group should therefore ensure that the responsibilities for defining, reviewing, and updating the standards are clearly assigned.

ORGANISATIONAL RELATIONSHIPS

Organisational relationships need to be clearly defined

The quality guidelines should define the organisational relationships between the quality-management group, the development project teams, and the user community. In particular, the responsibilities for producing documentation and for reviewing and accepting the deliverables at each stage of the development life cycle should be noted.

The relationship between development staff and users will inevitably have to become more formal to take account of the requirements of the quality-management programme. Quality software cannot be achieved without the active involvement of the user community, both in defining the bespoke quality service agreement and in checking that the quality objectives are being met as the project progresses. Users must be able to specify their requirements in terms of functionality, operational performance, the level of flexibility required to accommodate future changes and enhancements, and the user interface. They must also be aware of the implications on cost and timing if the requirements are changed significantly before the development project is complete.

The guidelines should also indicate how relationships with subcontractors, who may be used to develop some or all of the software, are to be organised and managed. Procedures will need to be developed to ensure that subcontractors conform with the quality guidelines.

QUALITY-CONTROL TECHNIQUES

The quality guidelines should describe the quality-control techniques available for use by development staff at various stages of the development life cycle, and provide advice about when and how to use them. Most of the techniques are already widely used by systems development staff. The most frequently used are project reviews (sometimes known as system audits), and walk-throughs.

The quality guidelines should describe the quality-control techniques available for use

Formal reviews and audits have been used for a decade or more and often involve users and staff from outside the systems department. They have proved to be the most effective means of identifying and correcting errors. One study by Michael Fagan (reported in *Writings of the Revolution*, which was published by Yourdon Press in 1982) found that peer reviews discovered 83 per cent of the errors detected during systems development; conventional test runs discovered only 17 per cent of the errors.

Reviews can also be applied to operational systems, where they might be used to assess:

- The impact of planned hardware and software changes.
- The quality of user, operating, and maintenance documentation.
- The acceptability to user departments of system inputs and outputs.

A review is often more suitable for quality-assuring deliverables produced early in the development life cycle. Reviews can also be used to ensure that documents are accurate and complete. Action points are noted during a review and are followed up later.

A walkthrough is a structured review of a system or program conducted with the originator's peers. Its success derives from the fact that the peers can spot flaws that the originator is blind to and that management is not aware of the errors discovered during the walkthrough. Although many people initially resist the idea of subjecting their work to peer review, they can be persuaded of the advantages.

Another form of technical review is the inspection method, which was developed by Michael Fagan while he was working for IBM in the early 1970s. The method requires a team of people, each of whom is assigned a specific role. An important feature of the method is the use of checklists and statistics about the errors to be searched for. The result of an inspection is an action report, and the inspection process is not complete until satisfactory action has been taken on all the points.

SUPPORT FOR DEVELOPMENT STAFF

The responsibility for assuring the quality of a particular application system lies firmly with the development team itself. However, the quality-management group will need to provide development staff with support as new projects are initiated, particularly in helping them to construct the quality plan and the quality service agreement. The group must also be prepared to provide support on an 'as-required' basis to help development staff resolve quality-related problems that are proving to be

The quality-management group provides support as new projects are initiated

particularly difficult (ensuring that user departments assign appropriate staff to the project, for example).

The quality-management group also has a continuing responsibility for providing training and education. One of the most important aspects of this responsibility is to ensure that everyone in the systems department has a common understanding of what is meant by quality. We suggest that a dictionary of the terminology used to describe systems quality be developed. For example, *acceptance testing* might be defined in such a dictionary as: "Tests conducted by users at the end of the system test stage to ensure that an application conforms to the standards set in the quality service agreement".

The quality-management group is responsible for providing training and education

The education and training activities of the quality-management group can be divided into three areas:

- A regular series of presentations and discussions aimed at making systems development staff more generally aware of quality-assurance practices and procedures. These sessions would not necessarily be specific to systems quality assurance but could, for example, be concerned with quality assurance in the manufacturing or engineering industries.
- Training in how to use specific quality-control techniques, such as walkthroughs or system audits.
- Training in how to use specific systems development techniques and tools.

The quality-management group is in an ideal position to identify where development techniques and tools are not being used effectively, and to define and provide the training that will rectify the situation.

MEASURING QUALITY

Measuring the quality of application systems is not straightforward or easy. Fewer than one-third of the organisations contacted during the research had attempted to measure the quality of their systems, and most of these used measures based on the number of errors found or on budget variances against time and cost estimates. Only one organisation was even contemplating using quality measures based on factors other than time or cost. Other organisations assess, rather than measure, the quality of their applications, usually by system reviews and audits. PEP members are, of course, familiar with PEP assessments. These are primarily intended to measure development productivity, although they do include some aspects of quality measurement.

A measurement programme is essential if realistic quality objectives are to be set

A measurement programme is essential if the benefits of the quality-management programme are to be quantified. It is required so that realistic quality objectives can be set in the quality service agreements made between the development teams and the user community. The quality-management group should therefore establish a programme to measure various quality-related aspects of application systems.

We set out below some of the measurements that can be made and how they can be used to improve the quality of applications software. When implementing a measurement programme, it is

important that early benefits can be gained from it. Unless benefits can be demonstrated, development staff will be unwilling to provide the time and effort required to make the measurements. The measurements set out below are in the sequence in which they can provide the earliest benefits, and for this reason, we suggest that they be implemented in this sequence.

SYSTEMS OPERATION

The quality of an operational system can be measured, for example, in terms of the number of errors per 1,000 transactions, or the mean time between failures. Other measures include the number of system failures in a given period, the number of operator and user errors, classified by transaction types, processing time per 1,000 transactions or records, the number of output errors, and the number of requests for system maintenance. By keeping and analysing records of this type for a range of operational systems, it is possible to monitor overall performance and to set realistic and achievable operational-performance objectives in the quality service agreement.

The quality of an operational system can be measured in many ways

PROGRAMMING

Analysis of the PEP database shows that the number of errors increases significantly as the size of a system increases. High productivity ratings and low manpower build-up indices tend to reduce the expected number of errors. The implication is that individual programs should be kept as simple as possible. Other studies have confirmed that the number of defects in a program is directly proportional to the complexity of the program. The aim, therefore, is to reduce the complexity of programs and thus improve their quality. One measure of complexity was defined by McCabe in "A Complexity Measure", published in 1976 by IEEE in *Transactions on Software Engineering*. This measure is based on 'cyclomatic' numbers derived from the decision structure of a program. Another measure was proposed by Halstead in 1977 (in *Elements of Software Science*, published by Elsevier). Halstead's measure is based on studies of the length and volume of programs, measured by counting the occurrences of distinct operands.

Limiting the complexity of a program will reduce the number of defects it contains

SYSTEMS DESIGN

It is much harder to measure the output or the characteristics of the systems design stage than it is to make measurements at the programming stage. The problem is that there is no standard unit of output equivalent to a line of code, and it is not easy to identify that a design error has occurred until it manifests itself as a problem later in the development life cycle. The best way of improving quality at the design stage is to use structured design techniques and the CASE and other development tools that support them. These formalise the design process. By keeping records of the time and effort required at the design stage, it will be possible to improve the accuracy of future budgets and estimates.

Use of structured design techniques can improve quality at the design stage

DEVELOPMENT PROCESS

Some organisations assess the quality of the overall development process by relating measures of output (lines of error-free code) to input (staff days worked). These types of measures provide only

The quality of the development process can be assessed by measuring the effort expended at each stage of the life cycle

an approximate indication of the quality of the development process. A better indication can be obtained by collecting data about the effort expended at each stage of the development life cycle. The aim is to identify the proportions of effort at various stages of the life cycle that result in poor- or high-quality software. Analysing this data will highlight the development stages that are the root causes of poor-quality systems, and action can be taken to modify the development process accordingly. As more PEP members provide information about software errors, the PEP database will be analysed with this aim in mind.

USER SATISFACTION

Measuring user satisfaction traditionally depends on subjective judgements

No assessment of software quality can be complete without some sort of measurement of overall user satisfaction. This is probably the most difficult measurement of all to make because it requires users to make subjective judgements about the quality of application systems. These judgements in turn will be made in the light of their initial expectations about the application. In the long term, the quality service agreements will be a good way of measuring user satisfaction, but this will require users to set objective and realistic targets. In the meantime, the best way of measuring user satisfaction is to carry out surveys of the user community.

However, an assessment of user satisfaction can sometimes be made by analysing statistics relating to operational systems. For example, where users have discretion as to whether to use a system or not, the volume of usage will give an indication of user satisfaction. On the other hand, user satisfaction is likely to be inversely proportional to the number of errors reported in a system. Care should be taken when assessing user satisfaction in this way, however, because some errors will be perceived by users as being much more significant than others.

Chapter 5

Summary of guidelines for a systems quality-management programme

The guidelines described in this paper for implementing a systems quality-management programme are based on the best practice used both by the organisations contacted during the research and by Butler Cox's consultancy clients. They are summarised below:

- To establish a quality-management programme, a review of existing procedures should be conducted to produce a clear plan of action for senior management. To get the most from the programme, quality should ultimately become a central part of corporate culture; the quality-management programme should not be limited to the systems department. The programme itself is run by the quality-management group.
- The quality-management group should devise quality guidelines for the systems development department. These are a set of standards drawn up in conjunction with the development staff who will use them, and they define how quality can be guaranteed in application systems. They cover all stages of the development life cycle and both functional and nonfunctional characteristics of application systems.
- A quality plan and a quality service agreement for specific projects should be drawn up, based on the quality guidelines. The quality plan is validated by the quality-management group and ensures that appropriate quality checks are built in to the development process. The quality service agreement is, in effect, a contract between the project team and the user department, and defines the output expected at each stage of the life cycle in terms that can be understood by users.
- The quality-management group is responsible for monitoring and enhancing the quality guidelines. However, the group is not responsible for carrying out the quality-control checks at the various stages of a particular project. That responsibility lies with the development staff.
- Development staff should be encouraged to gain experience of working at all stages of the development life cycle. They will then be in a position to check the quality of each other's work.
- The quality-management group should ideally be staffed by people with a proven record of successful systems development. They should be assigned to the group for periods of about 12 to 18 months.
- The quality-management group should train and counsel development staff to ensure that quality is maintained.
- Measurements of systems quality need to be devised so that the success of the quality-management programme can be monitored. The measurements should cover the various stages of the development process (including design, coding, and operation), and customer satisfaction.

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.

P E P

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

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

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

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

PEP Papers
Four PEP papers are produced each year. They focus on specific aspects of systems development productivity and offer practical advice based on recent research and experience.

Meetings
Each quarterly PEP forum meeting and annual symposium focuses on the issues highlighted in the PEP papers, and permits deep consideration of the topics. They enable participants to exchange experience and views with managers from other subscriber organisations.

Topics in 1989

Each year, PEP will focus on four topics directly relating to improving systems development and productivity. The topics will be selected to reflect the concerns of the subscribers while maintaining a balance between management and technical issues.

The topics to be covered in 1989 are:

- Quality Assurance in Systems Development.
- Making Effective Use of Fourth-Generation Languages and Application Generators.
- Staffing the Systems Development Function.
- Trends in Systems Development among PEP Members.

Butler Cox & Partners Limited
Butler Cox House, 12 Bloomsbury Square,
London WC1A 2LL, England
☎ (01) 831 0101, Telex 8813717 BUTCOX G
Fax (01) 831 6250

Belgium and the Netherlands
Butler Cox BV
Burg Hogguerstraat 791,
1064 EB Amsterdam
☎ (020) 139955, Fax (020) 131157

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

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

United States of America
Butler Cox Inc.
150 East 58th Street, New York, NY 10155, USA
☎ (212) 891 8188

Australia and New Zealand
Mr J Cooper
Butler Cox Foundation
3rd Floor, 275 George Street, Sydney 2000, Australia
☎ (02) 236 6161, Fax (02) 236 6199

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

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

The Nordic Region
Statskonsult AB
Stora Varvsgatan 1, 21120 Malmo, Sweden
☎ (040) 1030 40, Telex 12754 SINTABS

Spain
Associated Management Consultants Spain SA
Rosalía de Castro, 84-2°D, 28035 Madrid, Spain
☎ (91) 723 0995