Research Report

BUTLER COX
FOUNDATION

THE BUTLER COX FOUNDATION

REPORT SERIES NO. 45

BUILDING QUALITY SYSTEMS

ISSUED FEBRUARY 1985

Abstract

Many organisations are concerned about the quality of their information systems but are unsure how to set about building better-quality systems. This report reviews the traditional industrial approaches to quality assurance and assesses their suitability in a systems context. It also describes specific systems quality assurance techniques.

The report concludes that achieving systems quality is not difficult. The necessary tools already exist, but a change of attitude concerning the way in which systems are developed is required to make effective use of them.

Research Method

The research for this report was carried out during the second half of 1984 and was led by *Elisabeth Somogyi,* a principal consultant with Butler Cox specialising in all aspects of information systems management.

In addition to an extensive literature search (which included reviewing the application of formal quality control and assurance techniques in an industrial context), the research team also held group discussions with Foundation members and conducted more than 60 in-depth interviews. The enquiries had three purposes:

- —To understand the general principles of quality in an industrial context.
- —To meet with representatives of professional, government and standards bodies active in the quality assurance field.
- —To meet with IT suppliers and user organisations and to identify the different approaches to systems quality being adopted.

The research set out to identify an outstanding example of a high-quality system to present as a case history in the report. The installation was visited and both developers and users were interviewed, so that the quality approach adopted could be analysed in depth.

Photoset and printed in England by Flexiprint Ltd., Worthing, Sussex

THE BUTLER COX FOUNDATION

REPORT SERIES NO. 45

BUILDING QUALITY SYSTEMS

REPORT SYNOPSIS

Users and suppliers of information systems are seriously concerned about the quality of their systems, but are making few attempts to tackle the problem from first principles. Over the next few years the quality of systems will become as important a competitive factor as it already is in other areas of business life. To achieve quality, both in the development of systems and in their performance for the user, senior managers must be committed to the pursuit of quality and must be prepared to invest the time and effort required to build an environment in which quality becomes 'a way of life'.

No discussion of quality can ignore the overriding emphasis on quality that is the keynote of Japanese industrial success in so many consumer and capital goods markets. Most of the product quality techniques pioneered by Japanese companies have been adopted by their competitors in other countries, and some of these techniques are applicable to information systems building.

But software development also needs its own special techniques of quality assurance and control. Certainly it is no longer good enough to judge the success of a system simply on the answers to the traditional three questions: does it work, was it delivered on time and was it delivered within budget? For information systems, as for other products and services, the simple measure of customer satisfaction acts as the final proof of quality.

The keys to building better-quality systems are to:

- —Recognise that quality is important.
- —Accept that quality is difficult to measure.
- —Realise that tools are the link between quality and productivity.
- —Establish clear responsibilities for quality assurance.

In discussing these points (Chapter 6), the difficulty of measuring quality is linked to the difficulty of rewarding those who achieve quality in system development. Hence attention to quality must begin at the top of the organisation, and strategies for system development must concentrate on satisfying user expectations.

Quality (and productivity) problems usually arise with large and complex systems. Thus systems should be designed for simplicity, making best use of a well-defined development method and effective development tools that focus on analysis and design.

Establishing a quality assurance function does not necessarily mean setting up a separate quality control department. Responsibility for producing quality should lie with those who produce systems, within a balanced, helpful working environment.

The principles of achieving systems quality are detailed in the first three chapters of the report. A knowledge of conventional industrial approaches to quality helps to put the subject in context, with 'fitness for purpose' as an apt general definition.

Many factors contribute to the quality of computerised systems: they include correctness, reliability, efficiency, integrity, usability, maintainability, testability, flexibility, portability, reusability and interoperability. In any given system, some factors will be more important than others, and trade-offs between quality factors may be necessary during development. Systems quality can also be defined in terms of the ease with which the system can be changed, since some modification will be inevitable.

From the human standpoint, simplicity is the most important prerequisite for good design. This implies modularity. Yourdon's structured method for analysis and design is the best-known successful, systematic application of relevant concepts for developing highly modular, structured and flexible systems.

In industry in general, four main approaches to achieving quality have evolved: quality management, quality circles, zero-defect production and 'quality as a way of life' (a concept which originated in the United States with Edward Deming but which is now regarded as a Japanese characteristic). Quality objectives usually cannot be divorced from the overall management objectives of an organisation, and industrial experience has shown that the execution of most quality assurance functions should remain the direct responsibility of the line managers concerned.

Quality in information systems involves three separate characteristics: goals (of the organisation and of system users), methods (policies, procedures, standards and guidelines), and performance. The role of quality assurance in today's systems environment is much wider than is customary in industry generally, and includes evaluating each of the three characteristics. The mechanism for ensuring quality has many parts, and everyone in the information systems department needs to be aware of all the elements that contribute to systems quality.

Three techniques are specific to quality assurance in information systems: walkthroughs, inspections and testing. Walkthroughs are organised reviews of a system conducted by the system originator's peers with the aim of detecting errors. For success, this technique demands an 'ego-less environment'. The inspection method was developed in the 1970s by Michael Fagan of IBM as a form of technical review that validates the quality and accuracy of a product, and detects the errors. By applying this method at points prior to system completion it is possible to obtain early warning of problems and to correct errors relatively inexpensively.

The inspection method does not itself provide solutions to the problems. But significant improvements in productivity and decreases in error rates in both system programs and application programs have been reported. This method is most widely applied in the programming area, but can be useful also in other phases of development. It requires a relatively high investment prior to its introduction (in particular for compiling checklists and error statistics), and is more widely used in the United States than in Europe.

An important role of the systems quality assurance function is to ensure that an adequate testing strategy has been adopted. The report (Chapter 3) reviews the various testing techniques that are available.

Chapter 4 of the report discusses the practical experiences of some 30 companies, less than half of which had established a formal systems quality function. Companies' definitions of systems quality varied considerably. In general, 'quality control' was seen as a policing activity, checking correctness of work and conformance with standards and highlighting deviations. 'Quality assurance' was regarded as being broader in scope, including methodologies and skills.

Six main types of problem had been encountered by the companies in their attempts to set up a systems quality function:

— Lack of management commitment.

— Management attitudes and understanding.

— Attitudes of development staff.

— Lack of influence of the systems quality function.

— Motivating systems quality staff.

— Difficulty of assuring quality of systems developed by end users.

Companies had difficulty in quantifying the success of their formal systems quality functions. But it appeared that those organisations with formal broad-based quality programmes were the most successful.

A Danish organisation, VP-Centralen, was extremely successful in developing a major high-quality system on time, with zero defects and within budget. This project is described in detail in the report as a case history (Chapter 5). The task was a highly demanding one — to computerise Denmark's primary method of financing private housing, a system of bonds which are issued by the financial institutions and traded on the Copenhagen Stock Exchange.

The success of this project, we believe, stems from the realisation that high quality is an integral responsibility of management, and the fact that VP-Centralen took this responsibility very seriously. The management team deliberately built and maintained a high-productivity, high-performance work environment, based on the zero-defect principle. Their intimate knowledge of the development process helped them to select excellent tools and methods to support and control the work. As planners, they opted for simplicity in both project and organisational design. They demonstrated that quality can indeed be made a way of life.

Overall, the main message to emerge from this report is that the necessary tools to achieve systems quality already exist — but to make effective use of them requires a change in attitude concerning the way in which systems are developed. Another prerequisite for success is that the commitment to the pursuit of quality must permeate from the top to become an integral part of the corporate culture. Because these lessons can apply only to new systems, the need for managements to start thinking along these lines is urgent. The longer the delay, the longer existing poor-quality systems will survive.

THE BUTLER COX FOUNDATION

REPORT SERIES NO. 45

BUILDING QUALITY SYSTEMS

CONTENTS

CHAPTER 1

THE MEANING OF SYSTEMS QUALITY

This report is concerned with all aspects of the quality of systems — both with the quality of the final system, and with quality assurance and quality control during the development process.

Most organisations rely heavily on their information systems, and so the quality of these systems will affect the successful and efficient running — and often the control — of the business. It might therefore be expected that the pursuit of quality would be a major preoccupation of information systems departments. Undoubtedly, most such departments strive to deliver the best systems they can within the constraints of time and budget. But the systematic and rigorous pursuit of quality is not usually evident.

At present, the 'success' of a system tends to be judged on three things:

—Whether it 'works'.

—Whether it was delivered on time.

—Whether it was delivered within budget (ignoring the fact that most of the real cost will not be incurred until after the system has been implemented).

Quality control is usually equated with rigorous testing, though whether a system does what it was specified to do is only one aspect of quality. Quality assurance, on the other hand, is often not even understood, let alone practised.

This lack of understanding is partly a legacy of the early days of data processing, when systems often were delivered late and did not work properly. It also stems in part from inadequate attention being given to quality because the effects of poor quality are not always immediately apparent. It is also a consequence of managers not appreciating just what can be done routinely to improve the quality of systems.

To clarify the issues associated with systems quality we first look outside the information systems area to identify the lessons that can be drawn from other areas — areas that have been concerned with delivering quality products for a generation or more. In some disciplines the concept of quality is well understood and clearly defined. We begin, therefore, by examining the concept of quality, drawing on the experiences of other industries and disciplines. Later in this chapter we relate the generally accepted definitions of quality to the specific field of information systems.

THE MEANING OF QUALITY

To achieve quality in its goods and services an organisation needs to be aware of:

—The need to make quality an important and generally recognised goal of the enterprise.

—The mechanisms by which quality is produced.

—The measures by which quality can be assessed and evaluated.

In an industrial context the word "quality"refers to the evaluation of a product (or service). To define quality, it is therefore necessary to think in terms of the use to which the product is put, and "fitness for purpose" is a commonly used definition.

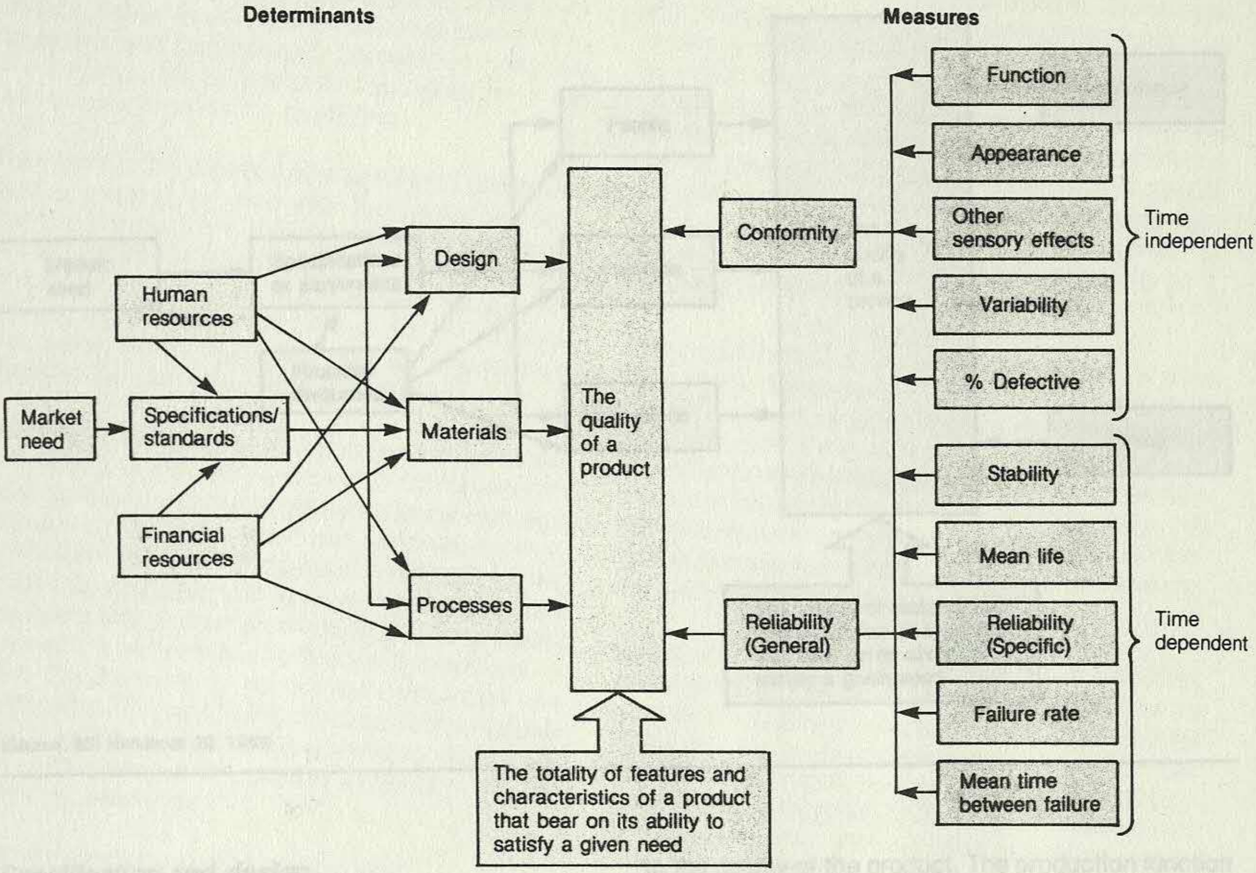
Formal industrial standards for quality describe the quality of a product or service as "the totality of features and characteristics that bear on its ability to satisfy a given need". These standards also describe the respective determinants and measures of the quality of products and of services (see Figures 1.1 and 1.2 overleaf).

The formal definitions imply that quality is so important that it might need to be 'assured', 'controlled' or 'improved'. For these reasons, it is first necessary to be able to evaluate quality. The definitions also require the features that determine the "fitness for purpose" of a product to be identified. These will include economic factors as well as characteristics such as specification and design, maintainability and availability.

Quality assurance and quality control

Users and suppliers of a product have different perceptions about its quality. The user is normally

Figure 1.1 Some of the determinants and measures of the quality of a product



(Source: BSI Handbook 22: 1983)

interested only in the quality of what he receives, whereas the producer or supplier is interested also in the reasons for poor quality and the ways to correct the shortcomings. The supplier needs to consider the elements that interact to produce quality:

—The raw materials.

—The specification and objectives of the product.

—Those who produce the goods.

—The process by which the product is made.

The supplier's need for regular high quality translates into two different types of action:

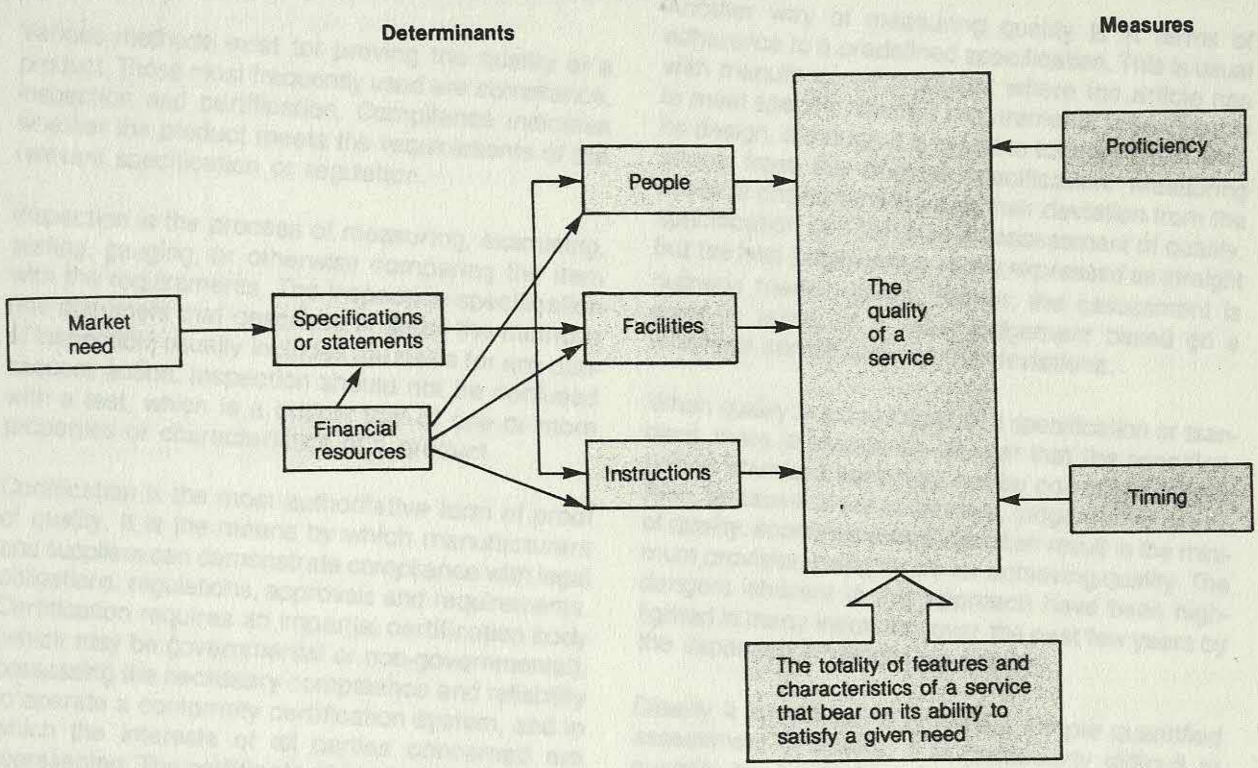
—The need to consider all elements and the complete process from inception to delivery. This approach usually is referred to as quality assurance.

—The need to consider the manufacturing process that produces the final article. Here, the supplier uses statistical sampling to measure the occurrence of faults, inserts checkpoints in the process to detect faults earlier, and inspects the product randomly. This approach usually is called quality control.

Quality assurance can therefore be defined as the activities and functions that are concerned with attaining quality. It is not concerned only with the provision of proof of quality, as implied by the word "assurance". Thus quality assurance includes the determination to achieve quality, the mechanisms for producing it and the means of assessing it.

Quality control is the aspect of quality assurance that is concerned with the practical means of ensuring product or service quality as set out in the specification. It is concerned with the operational techniques and activities that ensure the product is produced to the quality specified in the requirements. The techniques may be applied either to the system of control or to the product or service being controlled.

Figure 1.2 Some of the determinants and measures of the quality of a service



(Source: BSI Handbook 22: 1983)

Specification and design

Figures 1.1 and 1.2 indicate that specifications are an important determinant of quality. Quality can be prescribed and measured only if:

—The specification includes all characteristics of the product that are deemed to be important.

—These characteristics can be measured.

The words "specification" and "design" as used in an industrial quality context are more general than in data processing. Specification, in the general sense, is the document that describes in detail the requirements with which the product or service has to comply. Specification is of paramount importance in the achievement of quality: in many cases poor products or services are the result of inadequate, ambiguous or imprecise specifications.

The design process is concerned with defining products that fulfil the needs expressed in the specification. Defining quality in terms of fitness for purpose implies two important aspects of the design stage. First there is the intrinsic quality built into the design ("the quality of design") and, second, there is the extent to which this quality is achieved at the production stage ("the quality of conformance"). Thus the designer must specify the factors that contribute to the quality of the product. The production function then has the job of building the product according to the design specification. If the design was right, any deviation from it at the production stage can only lower the quality.

Maintainability

Many items are designed to receive attention during their life, either to compensate for the effects of wear, or to replace consumable supplies. The ease with which such work can be carried out is called "maintainability", and the operational function of the work is called "maintenance". Maintainability is therefore the ability of an item, under stated conditions of use, to be retained in, or restored to, a state in which it can perform its required functions. To carry out maintenance effectively, maintenance requirements and characteristics need to be specified.

Availability

The ability of an item to perform its required function is a combination of its reliability and maintainability. The combination of these two elements is called "availability". The use of this term in this sense is different from its use in connection with human or physical resources. In quality terms, the availability of an item does not necessarily imply that it is performing, but that it is in a fit state to perform.

THE PROOF OF QUALITY

Various methods exist for proving the quality of a product. Those most frequently used are compliance, inspection and certification. Compliance indicates whether the product meets the requirements of the relevant specification or regulation.

Inspection is the process of measuring, examining, testing, gauging, or otherwise comparing the item with the requirements. The inspection specification (the document that describes in detail the methods of inspection) usually includes the basis for any consequent action. Inspection should not be confused with a test, which is a critical trial of one or more properties or characteristics of a product.

Certification is the most authoritative form of proof of quality. It is the means by which manufacturers and suppliers can demonstrate compliance with legal obligations, regulations, approvals and requirements. Certification requires an impartial certification body (which may be governmental or non-governmental), possessing the necessary competence and reliability to operate a conformity certification system, and in which the interests of all parties concerned are represented. The certificate of conformity is a document signed by a qualified party affirming that, at the time of assessment, the product or service met the stated requirements.

THE MEASUREMENT OF QUALITY

Quality is a subjective characteristic, but a prerequisite to controlling and proving it is the ability to measure it. Measurement of quality is important, not as an end in itself, but for comparative reasons. The quality of two similar products can be compared only on the basis of values associated with the quality of similar features. And a purpose-made product can be judged only on the basis of relevant qualities or features. These qualities and features need to be measured against some base, which represents the assumed or required values of those features.

Today, it is fashionable to attach a numeric scale to opinions, thereby attempting to quantify the characteristic features that contribute to quality. Such an approach can be helpful, but it can also be misleading. High scores for irrelevant characteristics do not ensure quality.

An alternative approach is to measure quality through the potential effect its loss might create. Here, the probability of loss of quality needs to be taken into account together with the measured consequences of any such loss. This approach is taken by auditors and accountants when assessing the 'quality' of a company, its finance-related processes and the validity of its accounts.

Another way of measuring quality is in terms of adherence to a predefined specification. This is usual with manufactured products, where the article has to meet specific detailed requirements prescribed in its design, although it is usual to tolerate small deviations from the original specification. Measuring physical characteristics and their deviation from the specification can serve as an assessment of quality, but the final judgement is rarely expressed as straight numeric measurement. Rather, the assessment is given in terms of a value judgement based on a weighted combination of the deviations.

When quality is judged against a specification or standard, there is always the danger that the specification or standard itself may not be correct or tested. Also, because of the subjective, judgemental nature of quality, economic pressures often result in the minimum provision being made for achieving quality. The dangers inherent in this approach have been highlighted in many industries over the past few years by the Japanese emphasis on quality.

Clearly it is difficult to provide a simple quantified assessment of quality. It is particularly difficult to quantify the quality of one-off (or bespoke) services and goods. Here, it is usual to set up 'standards' that act as specifications against which the service or some regularly recurring features of the product can be measured. Again, the quantified assessment tends to be judgemental.

Earlier, we emphasised that quality is created through the process that produces the product. In the case of a service, quality is normally associated with the process or activity and its end results. Because of the difficulty in measuring quality and because of the strong tie between the process and its end result, service quality measurement is often focused on the process rather than the product. Using this kind of quality measure requires a regular production process and a great deal of knowledge about the process itself.

One way of overcoming the difficulty of measuring quality is to measure the quality of the product independently of its production process. The measure will still be judgemental, but the base against which it is measured is not the technical specification or standards. Rather, quality is measured against the original need for the product — that is, the purpose or objective that gave rise to it. Such "fitness for purpose" measurement checks the practical qualities of the product, irrespective of the way it was produced, to see if it fits its defined objectives. The fundamental prerequisite for this approach is that the objectives should be clearly specified.

Precise measures for quality require a unit of measurement that is related closely to the characteristics by which quality is perceived. Industries with a long history of quality control have developed their own unique measures. Even so, individual companies tend to have their own interpretations of the measures. And even in those industries that possess quality measures associated with physical characteristics, judgement seems to be an important part of proving quality.

Experience shows that there are, in addition to a unit of measurement, four other prerequisites for measuring quality successfully. These are:

—The definition of quality for the particular product.

—The knowledge of when the product is ready to be inspected for the desired quality.

—The description of the process by which the product is being built.

—The process by which quality is measured.

If any one of these prerequisites is missing, true measurement of quality is not possible.

QUALITY OF COMPUTERISED SYSTEMS

Computerised systems comprise hardware and software, but are designed to provide a service to the user. When a system is being developed, it may be regarded as a 'product'. Thus, its quality might be perceived in terms of conformance and reliability. Indeed, the software engineering approach encourages this view. However, in the hands of the user, the software and hardware, together with the way it is operated, create a 'service'. So a system that may be viewed as a 'product' by the data processing department will be viewed as a service by the user department.

To complicate matters further, the organisation's customers will perceive the computer system as a 'facility' which affects the service provided by the user department. The problem facing system developers is that they need to consider all the features of the system that will satisfy three quality-related perspectives:

—The engineering-type 'product' view.

—The user-type 'service' view.

—The customer-type 'facility' view.

All of these features need to be included in the original objectives and requirements for the system.

Sometimes it is difficult to define the requirements for commercial systems precisely and completely. In turn, this makes it difficult to build systems to meet the requirements. As a result, the quality of commercial systems is normally specified in terms of their efficiency — how well they support the company's operations, for example, and how easy it is to adapt them to changing circumstances — rather than in terms of reliability.

Thus, the definition of quality for commercial systems can be defined as the 'fit' of the system into its working environment, and can be measured by its efficiency in changing circumstances. Provided the requirements state very clearly the need for change, adherence to low-level functional requirements might be the necessary, but not always sufficient, measure for high quality in commercial systems.

Quality factors and system characteristics

The factors most often quoted as relevant for the quality of computerised systems are:

—Correctness, which is the extent to which a system satisfies its specifications and fulfills the users' objectives.

—Reliability, which is the extent to which a system can be expected to perform its intended function with required precision.

—Efficiency, or the amount of computing resources required to perform a function.

—Integrity, which is determined by the extent to which access to the system or data by unauthorised persons can be controlled.

—Usability, or the effort required to learn, operate, prepare input for, and interpret output from the system.

—Maintainability, which is measured in terms of the effort required to locate and fix an error in the operational system.

—Testability, or the effort required to test a system to ensure that it performs its intended function.

—Flexibility, which is the effort required to modify an operational system.

—Portability, which is measured by the effort required to transfer a system from one hardware configuration and/or software system environment to another.

—Reusability, or the extent to which a system can be used in other business environments or applications. This factor is related to the functionality, packaging and scope of the functions from which the system is built.

—Interoperability, which is the effort required to couple one system with another.

There are no universal measures for these factors because the relative importance of each one is subservient to the overall characteristics of the sys-

Figure 1.3 Relevant quality factors for specific system characteristics

System characteristic	*Quality factors*
Human lives are affected	Reliability Correctness Testability
Long life cycle	Maintainability Flexibility Portability
Real-time application	Efficiency Reliability Correctness
Navigational computer application	Efficiency Reliability Correctness
Processing classified information	Integrity
Interrelated systems	Interoperability

(Source: W E Perry, Effective Methods of EDP Quality Assurance)

Figure 1.4 Relationships between software design criteria and system quality factors

Design criteria	*Quality factor*				
	Correct-ness	*Relia-bility*	*Maintain-ability*	*Usability*	*Interoper-ability*
Traceability	•				
Consistency	•	•	•		
Completeness	•				
Error tolerance		•			
Accuracy		•			
Simplicity		•	•		
Conciseness			•		
Modularity			•		•
Self-descriptiveness			•		
Operability				•	
Training				•	
Communicativeness				•	
Communications commonality					•
Data commonality					•

(Source: W E Perry, Effective Methods of EDP Quality Assurance)

tem in question. Figure 1.3 shows typical system characteristics that might require one or another of the factors to be emphasised when assessing the quality of the system. Quality factors for systems also have strong links with software design criteria, as shown by Figure 1.4. (The definitions of these criteria can be found in Appendix 1.)

The importance of designing modular systems

In addition to conforming to functional objectives, a high-quality system will also be easy both to change and to adapt to changing requirements. A system designed to satisfy a fixed, static specification might initially be acceptable but, inevitably, it will need to be modified in some way or other. Systems quality can therefore also be defined in terms of how easy it is to change a system.

It is not sufficient to include as an objective "ease of modification". Such an aim is far too general to be helpful. Yet it is not possible to translate terms such as "flexibility" and "changeability" into meaningful elements of the specification, because these requirements can be met only by the process of design. The focus therefore needs to be on the design stage, not on the functional specification stage.

Designing a high-quality system requires an understanding of the problems of complexity and structure. Real understanding and real utility occur when the number of 'things' to interrelate is relatively low. From the human standpoint, simplicity is the most important prerequisite for good design.

There are three interrelated means of reducing the complexity of systems. First, the system should be partitioned into identifiable and understandable parts, where each part is defined by an inherent purpose, objective or function.

Second, the parts should be interconnected by a structure that provides order for command and control of the partitioned parts. (Note that such a structure implies a hierarchy, not a network.)

Third, the independence of the parts should be maximised by minimising their interdependence.

The first two are aids to comprehension, and the third is the key to flexibility. A system comprising simple, well-defined parts that interconnect loosely with each other is relatively simple to change. Changes are effected by decoupling from the structure parts that will remain the same, adjusting the structure and placing new, simple parts into it.

Two consequences arise from treating the parts and the structure separately:

—A bug or error in the system cannot easily corrupt the whole system because its effects will be localised.

—"Maintenance" and "enhancement" are two separate activities. Maintenance is concerned with repairing faults in existing modules. Enhancement is concerned with making changes to the structure and replacing or discarding or rearranging modules. Clearly, these enhancement activities cannot be carried out without first reanalysing the whole structure. In other words, system modifications must be preceded by analysis and design.

The quality measures associated with good systems design can therefore be defined in terms of the three characteristics of systems simplicity:

—Partitioning.

—Control structure.

—Module independence.

Glen Myers and Ed Yourdon have developed such measures for computerised systems (details can be found in their books, which are listed in the bibliography at the end of the report). The structured method developed by Yourdon for analysis and design is the best-known successful, systematic application of these concepts for developing highly modular, structured and flexible systems. The case history reported in Chapter 5 is based on the Yourdon methodology.

The impact of not specifying system quality factors

During the development stages of a system there are many opportunities to influence the quality of the final system. However, the impact of not specifying or measuring software quality factors during the development process shows up much later in the systems life-cycle, as Figure 1.5 illustrates. The effect is usually low quality which manifests itself through increased maintenance costs.

During the development stage it may be necessary to trade off quality factors against each other. For example, high integrity often can be achieved only by reducing the efficiency of the system. Some typical quality-factor trade-offs are shown in Figure 1.6 (overleaf).

These trade-offs are possible because, in general, quality factors are not independent. Figure 1.7 illustrates some of the relationships between quality factors for systems.

In order to measure the quality of a system it is necessary first to define the attributes that are critical to the system, and then to set the objectives for achieving these during the development stage. Most commercial system installations can develop ratings and base-line measures for the most commonly occurring factors that, in general terms, influence the quality of their own systems.

Beyond these general factors the most important quality factors (the functional objectives of systems), can be described only in specific terms. This also means that their measurement depends on the precise definition of the objectives set for the system. Tom Gilb's work on Design by Objectives is concerned with turning these objectives into measurable quality characteristics and ensuring that an orderly process of transformation is used to build these characteristics into each relevant part of the system during the development stage. Tom Gilb's method has previously been described in the transcript of the Foundation Conference held in Venice in May 1980.

Figure 1.5 The impact of not specifying or measuring software quality factors

Quality factors	Life-cycle phases							Expected cost saved vs cost to provide
	Development			Evaluation	Post-development			
	Requirements analysis	Design	Code and debug	System test	Operation	Revision	Transition	
Correctness	●	●	●	X	X	X		High
Reliability	●	●	●	X	X	X		High
Efficiency	●	●	●		X			Low
Integrity	●	●	●		X			Low
Usability	●	●		X		X		Medium
Maintainability		●	●			X	X	High
Testability		●	●	X		X	X	High
Flexibility		●	●			X	X	Medium
Portability		●	●				X	Medium
Reusability		●	●				X	Medium
Interoperability	●	●		X			X	Low

● = where quality factors should be measured X = where impact of poor quality is realised

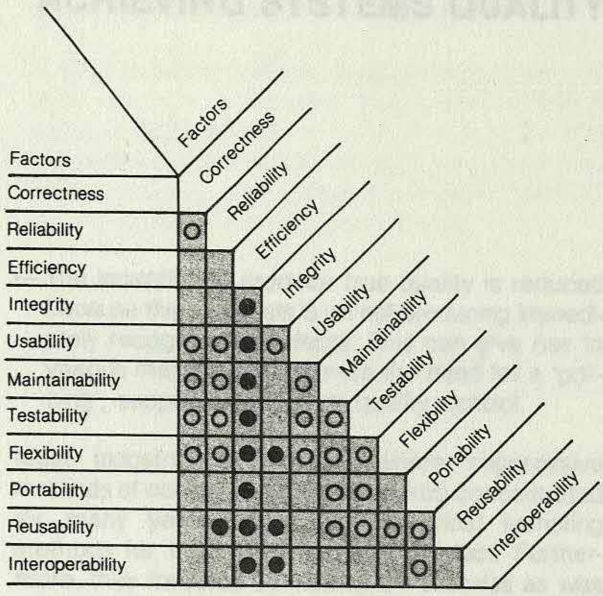
(Source: W E Perry, Effective Methods of EDP Quality Assurance)

Figure 1.6 Typical quality factor trade-offs

Trade-off	*Impact*
Integrity vs efficiency	The additional code and processing required to control access to the software or data usually increases the run time and requires additional storage
Usability vs efficiency	The additional code and processing required to ease an operator's tasks or provide more usable output usually lengthens the run time and requires additional storage
Maintainability and testability vs efficiency	Optimised code, incorporating intricate coding techniques and direct code, always provides problems for the system's maintainer. Using modularity, instrumentation, and well-documented high-level code to increase the maintainability of a system usually increases the overhead, resulting in less efficient operation
Portability vs efficiency	The use of direct code or optimised system software or utilities decreases the portability of the system
Flexibility and reusability vs efficiency	The generality required for a flexible system decreases the efficiency of the system
Interoperability vs efficiency	Again, the added overhead for conversion from standard data representations, and the use of interface routines, decreases the operating efficiency of the system
Flexibility vs integrity	Flexibility requires very general and flexible data structures. This increases the data security problem
Reusability vs integrity	As above, the generality required by reusable software provides severe protection problems
Interoperability vs integrity	Coupled systems allow more avenues of access to more and different users. The potential for accidental access of sensitive data is increased as well as the opportunities for deliberate access. Often, coupled systems share data or software that compound the security problems as well
Reusability vs reliability	The generality required by reusable software makes providing error tolerance and accuracy for all cases more difficult

(Source: W E Perry, Effective Methods of EDP Quality Assurance)

Figure 1.7 Some relationships between systems quality factors



If a high degree of quality is present in one factor, the chart illustrates the degree of quality expected for the other.

○ = High
● = Low
Blank = No relationship or application dependent

(Source: W E Perry, Effective Methods of EDP Quality Assurance)

CHAPTER 2

ACHIEVING SYSTEMS QUALITY

We now turn to the practical aspects of achieving systems quality. We begin by examining the lessons that can be learnt from the experiences of other industries that have a long history of quality assurance and quality control, and then we focus on the specific issues for information systems.

APPROACHES TO INDUSTRIAL QUALITY ASSURANCE

Formal quality control has its origins in mass production methods. Assuming there is a market for them, the goods can be sold, provided they are not defective — that is, they have no (or only a few) recognisable faults. In this situation, quality equates reasonably well with earnings: every faulty product that cannot be sold represents a loss of revenue. This means that the elusive characteristic of quality has a financial value attached to it. Quality in this environment is measurable.

Quality in the above sense is important because it is a key selling characteristic which helps to convert the product into cash. This implies that the cost of quality can also be calculated. Typically, this cost is the amount of money that needs to be invested to ensure either zero defects or that only a tolerable number of defective items end up as scrap.

Interpreting quality in this way led to the original approach to quality control used in industry, which developed almost into a separate discipline. There are two key characteristics of traditional quality control:

—Quality is measured in terms of physical characteristics that can be inspected.

—Quality is ensured by inspecting the article after it has been produced.

The effects of this approach can be summarised as follows:

—Control of quality is retrospective, where the emergence of a fault is the signal to initiate corrective action. But, as the damage has already occurred, the action is directed to ensuring that it does not continue to occur.

—The incentive to produce true quality is reduced because the emphasis is on not producing immediately recognisable defects. This can give rise to various malpractices. Hence the need for a 'policing', suppressive type of quality control.

Most industrial economies (where suppressive methods of control had been the norm) concentrated for many years mainly on statistical sampling methods for inspecting the final product. Furthermore, they invested as little in the process as was possible. More recently, industry has recognised the inadequacy of this approach to quality control and has devised four main approaches to achieving quality: quality management, quality circles, zero-defect production and 'quality as a way of life'. These approaches are explained below.

Quality management

The need to manage quality came about as a result of the recognition that those companies and countries doing best are the ones offering the best value for money — that is, providing competitive quality at a competitive price. Often price is negotiable but quality is not, and the reputation for competitive quality is recognised more and more as a company's greatest asset. It has also been recognised that dramatic improvements in quality need not increase costs unduly.

The management of quality is the sum of the following elements:

—Knowing the customers' needs.

—Designing to meet those needs.

—Constructing faultless products.

—Purchasing reliable components and sub-assemblies.

—Meeting certified performance and safety requirements.

—Producing clear instruction manuals.

—Ensuring that the products are suitably packaged.

—Ensuring that the products are delivered on time.

—Providing an efficient back-up service.

—Taking account of feedback from field experiences.

These elements add up to "fitness for purpose" and value for money. The aim is to create "ownership satisfaction", which brings customers back to buy a company's products and services again and again.

The four golden rules of managing quality can be summarised as follows:

—Goods must be designed to meet customers' needs and to make manufacture and maintenance easy.

—Goods must be made exactly and consistently to the specified design.

—Product marketing must ensure that advertising is accurate, that deliveries are on time, that servicing is efficient and that effective market research is carried out and fed back into continual design improvements.

—Above all, there must be a total and organised commitment to quality. This can spring only from the very top — from the chairman, managing director or chief executive. In fact, quality should be a way of life throughout the organisation.

Quality circles

Quality circles are problem-solving forums for those who are involved in the day-to-day operation of the business. They are based not on cash reward but on satisfaction and recognition of achievement. A quality circle is a group of people who meet voluntarily and regularly to identify and solve their own work-related problems — and then implement their solutions with management approval.

In practice, small groups of employees, usually from the same workplace and under the same supervisor, volunteer to meet for problem-spotting and problem-solving sessions. The participants examine problems that occur in their work area and that affect their own jobs. The group itself applies the solutions if they have the authority to do so. Otherwise, management is presented with the recommendations, and the decision to implement the recommendations then rests with management.

Quality circles concentrate on tasks and problems, and their mode of operation is depicted in Figure 2.1. Quality circles recognise the need both to increase the importance of the role of the foreman and to draw upon the knowledge and expertise available on the shopfloor.

Quality circles were first introduced in Japan in the 1960s. The literal translation of the Japanese term for quality circles is "the gathering of the wisdom of the people". In the mid-1970s, industry in the United States and Europe began to show practical interest in the technique. So far, however, quality circles seem to work best in Japan, where they are a natural extension of corporate and trade union structures, and of employee and management attitudes.

Experience has shown that quality circles do not work when the idea is imposed from above, although management's attitude is very important for their successful operation. Beyond improving quality and reducing costs, quality circles also improve communications between workers and management.

Zero-defect production

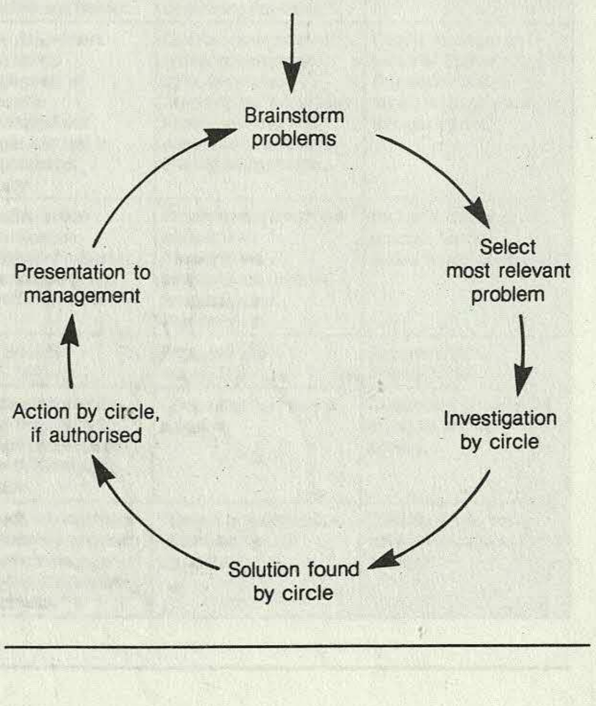
The concept of zero-defect production was formulated by Philip Crosby, one of the major advocates of quality in manufacturing. Crosby is chief executive of PCA Inc., a Florida-based quality management consultancy. Prior to founding PCA he was corporate vice president for the ITT corporation where for 14 years he was responsible worldwide for quality assurance.

His belief is that quality is the all-important catalyst that makes the difference between success and failure. In fact, he maintains that quality is free — it is "un-quality" that costs money. The main task of the production process is to make quality certain — in other words to do things right the first time.

Crosby's principles are simple:

—Quality is the absence of faults and defects (quality in conformance to standards and specifications).

Figure 2.1 Mode of operation of quality circles



—Quality is built in by the people who do the work. Therefore quality is ultimately the responsibility of those who manage the operations.

—Quality problems are non-conformance problems and these have to be prevented. But prevention is a difficult idea to put over and manage successfully.

Interestingly, Crosby implies that management's lack of understanding contributes to a large extent to a lack of quality. He advocates a quality organisation. He sees an organisation's approach to quality management evolving through five stages, as depicted by the quality management maturity grid shown in Figure 2.2.

Quality as a way of life

Many Western economies have lost markets to Japan in recent years. The spectacular success of Japan in the seemingly impenetrable European and North American markets was the result of the consistent high quality and reliability of Japanese products. The remarkable achievement of Japan in producing high quality for an acceptable price is due largely to a philosophy of quality control that is different in many respects from Western industrial practices. What is less well known is that the quality concept used by the Japanese originated in the United States with Edward Deming.

The heart of Deming's approach to the production of quality is the belief that quality is manufactured. In other words, quality is built into a product through a process. But defects are also built into a product as the result of a faulty process. Mass inspection to spot faulty products is expensive, and scrapping the defective articles also costs money. This approach to achieving quality does not improve the process or the system that is the root cause of the defect.

Deming argued that, instead, industry should spend money on ensuring that the system and the process are right so that items are manufactured correctly in the first place. To improve quality, the production process needs to be subjected to statistical control, and this implies that the process or system needs to be regular. Thus, disturbances and haphazard interference with the process need to be eliminated.

Deming believes that quality failures are not always caused by defective production processes; often, they are caused by an inadequate system. The responsibilities for poor quality therefore lie firmly with

Figure 2.2 Crosby's quality management maturity grid

Measurement Category	*Stage 1: Uncertainty*	*Stage 2: Awakening*	*Stage 3: Enlightenment*	*Stage 4: Wisdom*	*Stage 5: Certainty*
Management understanding and attitude	No comprehension of quality as a management tool. Tend to blame quality department for "quality problems".	Recognising that quality management may be of value but not willing to provide money or time to make it all happen.	While going through quality improvement program learn more about quality management; becoming supportive and helpful.	Participating. Understand absolutes of quality management. Recognise their personal role in continuing emphasis.	Consider quality management an essential part of company system.
Quality organisation status	Quality is hidden in manufacturing or engineering departments. Inspection probably not part of organisation. Emphasis on appraisal and sorting.	A stronger quality leader is appointed but main emphasis is still on appraisal and moving the product. Still part of manufacturing or other.	Quality department reports to top management, all appraisal is incorporated and manager has role in management of company.	Quality manager is an officer of company; effective status reporting and preventive action. Involved with consumer affairs and special assignments.	Quality manager on board of directors. Prevention is main concern. Quality is a thought leader.
Problem handling	Problems are fought as they occur; no resolution; inadequate definition; lots of yelling and accusations.	Teams are set up to attack major problems. Long-range solutions are not solicited.	Corrective action communication established. Problems are faced openly and resolved in an orderly way.	Problems are identified early in their development. All functions are open to suggestion and improvement.	Except in the most unusual cases, problems are prevented.
Cost of quality as % of sales	Reported: unknown Actual: 20%	Reported: 3% Actual: 18%	Reported: 8% Actual: 12%	Reported: 6.5% Actual: 8%	Reported: 2.5% Actual: 2.5%
Quality improvement actions	No organised activities. No understanding of such activities.	Trying obvious "motivational" short-range efforts.	Implementation of the 14-step program with thorough understanding and establishment of each step.	Continuing the 14-step program.	Quality improvement is a normal and continued activity.
Summation of company quality posture	"We don't know why we have problems with quality."	"Is it absolutely necessary to always have problems with quality?"	"Through management commitment and quality improvement we are identifying and resolving our problems."	"Defect prevention is a routine part of our operation."	"We know why we do not have problems with quality."

(Source: Philip Crosby, Quality is Free) management. He makes the point that it is little use exhorting workers to improve quality because most things that contribute to quality — having the right tools, adequate training, good materials, workable production processes, a reasonable working atmosphere — are largely outside their control.

Deming asserts that quality should be a way of life for the whole organisation. He has summarised his philosophy in 14 action points for managers (see Figure 2.3).

Figure 2.3 Deming's fourteen action points for producing quality

1. Create constancy of purpose toward improvement of product and service, with a plan to become competitive and to stay in business. Decide whom top management is responsible to.
2. Adopt the new philosophy. We are in a new economic age. We can no longer live with commonly accepted levels of delays, mistakes, defective materials, and defective workmanship.
3. Cease dependence on mass inspection. Require, instead, statistical evidence that quality is built in, to eliminate need for inspection on a mass basis. Purchasing managers have a new job, and must learn it.
4. End the practice of awarding business on the basis of price tag. Instead, depend on meaningful measures of quality, along with price. Eliminate suppliers that cannot qualify with statistical evidence of quality.
5. Find problems. It is management's job to work continually on the system (design, incoming materials, composition of material, maintenance, improvement of machine, training, supervision, retraining).
6. Institute modern methods of training on the job.
7. Institute modern methods of supervision of production workers. The responsibility of foremen must be changed from sheer numbers to quality. Improvement of quality will automatically improve productivity. Management must prepare to take immediate action on reports from foremen concerning barriers such as inherited defects, machines not maintained, poor tools, fuzzy operational definitions.
8. Drive out fear, so that everyone may work effectively for the company.
9. Break down barriers between departments. People in research, design, sales, and production must work as a team, to foresee problems of production that may be encountered with various materials and specifications.
10. Eliminate numerical goals, posters, and slogans for the work force, asking for new levels of productivity without providing methods.
11. Eliminate work standards that prescribe numerical quotas.
12. Remove barriers that stand between the hourly worker and his right to pride of workmanship.
13. Institute a vigorous programme of education and retraining.
14. Create a structure in top management that will push every day on the above thirteen points.

(Source: W E Deming, Quality Productivity and Competitive Position)

ORGANISATIONAL ASPECTS OF QUALITY

The success of any undertaking directed at fulfilling a specified need depends on the collective and individual success of its several activities and functions. Thus, all activities and all functions in any undertaking, and all personnel engaged in them, contribute to quality and are inherently involved in quality assurance. The attainment of high quality starts with recognising this fact through initiatives that come from the top of the organisation. Conversely, low quality is often associated with management's non-recognition of this fact, or management's abdication of its responsibility for quality.

Quality objectives usually cannot be divorced from the overall management objectives of an organisation. There are some well-documented case histories of long-established organisations going out of business, primarily because of failures in quality — not in the production or manufacturing area, but due to a combination of shortcomings in planning, design and development (optimistic forecasting and tendering based on insufficient development engineering and proving of a design, for example).

Placing quality assurance in the organisational structure

Most organisations are structured in one or more of the following ways:

- Groupings of personnel according to occupational level (normal line management).
- Groupings of personnel according to subject of specialisation.
- Groupings of personnel according to project (each new product).
- Groupings of personnel according to the phase of each project.

These existing groupings are likely to be a governing factor in deciding whether to establish a specialised quality department. Where a department has been established, its title can cause difficulties. For example, where the term "quality control department" has been adopted, an undue emphasis on the word "control" can lead to interdepartmental difficulties if other departments suspect there is a move to usurp their functions. It is therefore rarely sufficient merely to change the name of an established inspection department to "quality control" (or some similar title) and then assume that the quality objectives of an organisation will be attained.

Many recent investigations in industry suggest that an integrated approach that involves the inspection or quality control departments with the manufacturing process is the best organisational model for quality assurance. An added benefit is that the approach can lead to increased operator responsibility, with the inspection or quality control department being mainly concerned with monitoring, measuring and guidance.

The effect of education and training

The scope and intensity of quality-related education and training will help to determine the type of quality assurance organisation that is established, in both the national and corporate sense.

Where training has been extended beyond specialists to managers at all levels, to supervisors and even to non-supervisory staff, the coordination of quality assurance activities will usually be carried out by the line supervisors and managers in the regular chain of command. In this situation, the role of any quality specialist is mainly in reviewing, auditing and consulting, and usually there is no need either to employ large numbers of quality specialists or to establish large, centralised quality assurance or quality control departments.

On the other hand, where intensive training in quality control methods has been restricted mainly to specialists (such as quality control inspectors), a strong, centralised quality assurance or quality control department is more likely to be formed. Although its staff will be well trained in their speciality, the department will sometimes experience difficulties in its relationships with other departments, because of the disparity in training.

THE QUALITY MANAGEMENT ROLE

Industrial experience has shown that the execution of most quality assurance functions should remain the direct responsibility of the individual line managers concerned. Rather than attempt to usurp these responsibilities, the quality assurance manager should advise and assist on all quality-related tasks and should monitor and coordinate them throughout the organisation. He will probably be directly responsible for the following activities:

- Developing the quality assurance programme and quality manual.
- Controlling incoming and defective materials.
- Operating the quality control system.
- Analysing defects and failures.
- Measuring the cost of achieving quality.

In addition, the manager of a quality department might be required to train staff throughout the company in matters relating to quality assurance, or to coordinate such training.

In this context, he is involved in developing the relevant elements of quality management, and acts as a reference point for them. These elements are:

- The quality manual, which sets out the general quality policies, procedures, and practices of the organisation.
- The quality system, which (in a quality context) defines the organisation structure, responsibilities, activities, resources and events that together provide organised procedures and implementation methods to ensure that the organisation can meet its quality requirements.
- The quality programme, which comprises a documented set of activities, resources and events serving to implement an organisation's quality system.
- The quality plan, which is derived from the quality programme (extended if necessary) and which sets out the specific quality practices, resources and activities relevant to a particular contract or project.

THE QUALITY ASSURANCE FUNCTION FOR SYSTEMS

We now turn our attention to the way in which the general lessons of quality assurance can be applied in the specific area of information systems.

In many countries the national standards body has developed a formal approach to quality issues. The standards usually relate to a wide range of issues and typically are very general in nature. Nevertheless, some of them can be useful in the quality assurance of commercial systems development projects. In particular, the standards can provide either a checklist against which to compare current practices, or a 'shopping list' that can be used when setting up a quality assurance function.

Many of the formal approaches have their origins in the military field, where the pioneering work done by the United States Department of Defense is often used as a model. For example, the United Kingdom Ministry of Defence published a standard in February 1984 which relates exclusively to systems development. (Appendix 2 contains a description of this standard.)

Industrial experience shows that those organisations that achieve a high level of quality in their products and services first establish an acceptable level of quality, and then build a mechanism to assure that this level is maintained. Control of quality includes more than simply evaluating the end product. It begins by examining the raw materials and continues throughout the entire manufacturing cycle. The implication is that data processing organisations must first assume the responsibility for determining an acceptable level of quality for their systems, and then must establish the mechanism to ensure that the level is maintained.

CHARACTERISTICS OF THE SYSTEMS QUALITY ASSURANCE FUNCTION

Quality in systems is associated with three separate characteristics: goals, methods and performance. Goals can be stated in terms of achieving the objectives of both system users and the total organisation. The goals of the organisation are primary; the goals (or requirements) of system users are secondary. If the user requirements conflict with the goals of the organisation, the systems quality assurance function must point this out. The goals of any one user also should be in harmony with the goals of other users.

Standardised methods are needed to perform the data processing function. These methods manifest themselves as policies, procedures, standards and guidelines.

Performance is concerned with optimising the use of computer hardware and software when implementing applications. This requires relevant business analysis, proper systems design, the use of appropriate programming and systems techniques, and the best use of the available hardware and software.

Goals (users' needs) can be better satisfied if appropriate quality standards exist. But methods should not interfere with goals, nor should methods reduce performance. And goals should not override performance. It is only through the use of appropriate standards, however, that the proper balance between goals, methods, and performance can be achieved. If the organisation fails to set quality standards, technical staff will set their own. Far too often, management relies on technical staff setting standards, but provides them with little guidance on overall objectives. As a consequence, management later reprimands the technical staff for poor performance.

In this context, the aims of the systems quality assurance function are:

— To ensure compliance with the approved methods of building applications.

— To ensure a reasonable level of performance.

— To ensure that the goals of the organisation, and especially those of the system user, are satisfied.

These aims can best be accomplished by establishing a quality-oriented information systems environment.

Most information systems departments have only just begun to address the quality issues associated with computerised applications. In many organisations the first stage of systems quality assurance will be to define what is meant by systems quality in their particular environment. Also, establishing a systems quality assurance function might be regarded as an evolutionary step along the path of moving data processing from an art to a science. But a successful systems quality assurance function requires that standards be established against which quality can be measured. A key characteristic is that the function should advise information systems personnel, but should derive its authority from the top management of the organisation.

ROLE OF THE SYSTEMS QUALITY ASSURANCE FUNCTION

The role of quality assurance in today's systems environment is essentially to assist system developers successfully to face many differing challenges. The role is seen to be much wider than that normally associated with quality control.

In addition to providing guidance, there are three broad roles the systems quality assurance function can undertake:

— Applying the quality controls.

— Building an environment in which higher-quality systems can be developed.

— Reviewing applications and systems.

A checklist of the tasks associated with the above roles is given in Figure 2.4.

Applying controls

Quality assurance controls should apply both when developing new systems and when enhancing old ones. Irrespective of the way in which quality assurance is implemented, the goals, methods and performance of each system must be evaluated.

Evaluating system goals

For each system, the quality assurance function must confirm that:

— The system meets the objectives and needs of the user department(s) and other users.

— The system is consistent with the needs of other users, so that a system built for one group of users does not infringe on the rights of other systems users.

— The goals are consistent with the objectives of the organisation. In all cases, the organisation's objectives should have a higher priority than the goal of an individual user.

Figure 2.4 Systems quality assurance tasks

Providing guidance about: Methods, tools and standards Training needs Project performance, review and control Technology Audit requirements Organisation of technical staff Recurring errors and how to avoid them
Applying controls: Project initiation, planning and monitoring Interpreting management control and audit requirements Ensuring that policies and standards are adhered to
Building the environment: Develop quality objectives Develop standards and guidelines Select life-cycle model and methods for development Select development and maintenance tools
Reviewing systems: Initiating reviews Evaluating methods Assessing technical criteria Assessing projects (cost performance, organisation) Identifying quality, technical and cost problems

(Source: W E Perry, Effective Methods of EDP Quality Assurance)

—The goals of the system match the objectives of the information systems department. Any conflict should be resolved before the system is implemented.

—The goals of the system are consistent with industry and government requirements.

—The system complies with the intent of management (that is, controls are adequate) and the system is auditable.

Evaluating system methods
When evaluating system methods, the systems quality function examines each project to determine that it is following the organisation's and the information system department's policies, procedures, standards, and guidelines to ensure that:

—The policies conform with the broad-based course of action selected by the organisation.

—The procedures conform with the particular methods outlined by the organisation to accomplish its objectives.

—The standards and rules being used conform with those set up by the organisation for the measure of quantity or quality of work.

—The guidelines conform with the department's recommended methods for performing work. (When a standard methodology is used, the local guidelines are often referred to as "codes of practice".)

Evaluating system performance
The performance-related aspects of the systems quality assurance function consist of evaluating each system to determine that its design is:

—Economical, by checking that the system is to be operated in a way that incurs minimal costs.

—Effective, by confirming that the system will accomplish the results desired with minimal effort.

—Efficient, by checking that the system is designed to optimise the use of people and machines.

These three aspects of system performance are not synonymous, and each requires a different evaluation of the system or project. The economic evaluation is concerned with value for money. Effectiveness needs to consider the criteria of time and ease of performing the task. Efficiency considers such things as the choice of technology and its application, whether all or parts of the system need to be computerised, or whether the need can be met better by a manual procedure.

Establishing the right environment

No matter how the systems quality assurance function is implemented, the emphasis should be on establishing an environment in which quality can flourish, rather than on reviewing individual applications. Reviewing a single application can only improve the quality of that application; creating an environment that encourages quality can improve the quality of all applications. Such an approach also places the quality assurance function in a position where its contribution is more visible. Establishing an environment that promotes quality is, in effect, no more than drawing together all those functions that support and improve the quality of systems.

Reviewing applications and systems

There are three aspects of project reviews: deciding what to review; deciding when to do it; and determining the purpose of the review.

Ideally, the quality assurance function should review all systems in depth but in practice this is rarely possible. A method is therefore required for determining priorities and allocating the time of the review personnel. The most relevant criteria are:

—The type of application.

—The complexity of the application.

—The technology being utilised to implement the application.

—The people involved in specifying and implementing the system.

Figure 2.5 lists the factors for each criterion that should be considered when selecting an application or a development project for review.

Formal reviews should be carried out at the major decision points of system development:

—Before the decision is made to implement a system solution (at the end of the feasibility study).

—Before any major expenditure is committed.

—After the technical development has been completed.

—After the trial period of live operations has been completed.

—When the project is showing signs of being off-course.

Systems should also be reviewed at the early stages of development because this is the best time to influence the quality of systems. Once the design of a system has been determined, it is costly to make significant changes because most development costs occur after programming has commenced. Furthermore, management tends to become involved with a system only when the cost becomes significant and the implementation date approaches. Thus, paradoxically, management's influence and concern often come at the point in the systems development life cycle where it is least able to influence the design. A quality assurance focus early in the development cycle can solve this problem.

For major systems that require large capital expenditure, reviews also need to be planned throughout the operational phase of the life cycle. Large systems often have an operational life span of ten years (or more), and this period usually involves modifications, adjustments and difficulties with operating an ageing system. Regular, pre-planned reviews, carried out with specific review criteria in mind, should indicate the need for redevelopment and major adjustments. (The life cycle of systems based on pre-planned reviews was discussed in Foundation Report No. 25 — System Development Methods.)

A systems review is concerned with the total performance of a project during its development stage, and the performance of the system during its operational phase. Before carrying out a systems review, it is important to specify its precise purpose. This might be to test the conformance of the system and the adequacy of the controls, validate the standards and methods being used, and so forth.

Many systems will need to be reviewed as a matter of course by the organisation's auditors. (Sometimes the auditing firm may be asked to carry out a systems review — or audit — independently of the regular financial audit cycle.) Systems are rarely designed with ease of auditing in mind. Special requirements for audit control often are not recognised until the first audit is carried out. Moreover, auditors find it difficult to suggest appropriate standards that are specific to computerised, as opposed to manual, procedures.

Figure 2.5 Factors to consider when selecting a project for review

Criterion	*Factors*
Type of application	Financial application Statutory requirement Exposure to loss Fundamental administrative or control process
Complexity of application	Application history (problems) Large effort required to develop application Short time span allowed to complete the project Large number of expected users Cross-technology application
Technology	New hardware, software, techniques or tools New technology project
People	High level of user involvement Low level of internal audit involvement Limited skill level of project team

(Source: W E Perry, Effective Methods of EDP Quality Assurance)

From the experiences of the organisations we talked with during the research for this report it is clear that system audits can be made easier by:

—Involving auditors in the early design of a system. In this way the auditors can influence the design as far as general financial or management control requirements are concerned. And they can indicate special requirements that need to be included as technical facilities for simplifying audit work.

—Designing databases with easy data-extraction facilities, so that the auditors can have easy and independent access to the data.

SUMMARY

Quality cannot be built into a product, or measured, until it is defined. But quality cannot be universally defined; it must be defined for the item in question as a list of stated attributes and characteristics. This means that quality in a data processing environment must be defined by the organisation, and one organisation's definition of quality may vary significantly from another's. The main problem facing data processing personnel is that systems technology is not well understood by the organisation in general.

If an application 'works', few people (if any) in the business know whether the system developers did an outstanding job or just a satisfactory job. There is a clear need for a means of evaluating the excellence of system development work, and a clear definition of systems quality would help significantly in this area.

We have also established that the mechanism for ensuring quality has many parts. The totality of the mechanism is called quality assurance, which is different from formal quality control. Quality assurance does not necessarily require a separate formal organisation, but it does need to be initiated from the top of the organisation. Everyone in the information systems department needs to be aware of all the elements that contribute to systems quality, and quality assurance requires a focal point for the activities by which quality is produced.

CHAPTER 3

QUALITY ASSURANCE TECHNIQUES FOR INFORMATION SYSTEMS

Having described in general terms the characteristics, objectives, role and tasks of the systems quality assurance function, we now describe the specific techniques that are available.

Most systems quality assurance techniques overlap considerably with conventional management, project management and development tasks. Three techniques, however, are specific to systems quality assurance:

—Walkthroughs.

—Inspections.

—Testing.

WALKTHROUGHS

A walkthrough is a structured review of a system or program conducted by a group of the originator's peers. The walkthrough technique derives from a common human characteristic: the originator of any creative work is often blind to any flaws there might be in it. Yet the originator's peers, who are competent to understand the product but do not have the emotional involvement with it, can spot the flaws. Many people initially resist the idea of subjecting their work to scrutiny by their peers. Nevertheless, experience shows that people can be persuaded to take a positive approach to walkthroughs and overcome the initial difficulties.

The aim of system walkthroughs is to ensure error-free (that is, zero-defect) development. For walkthroughs to be successful they must be organised carefully in a constructive environment where open criticism is possible.

The walkthrough technique was developed following some ten years of work by psychologists at IBM's various programming workshops. The results were documented by Gerry Weinberg in his book "The Psychology of Computer Programming". Since the original work was carried out, it has been realised that:

—Programming is not the only activity where walkthroughs are relevant for zero-defect production. In fact, almost all creative activity (such as decision making, analysis, design, and research) can benefit from the technique.

—The technique works only if what is termed an "ego-less environment" is maintained.

An ego-less environment aims to separate the product of an activity from the producer. As long as the producer regards the results of his activity as an extension of his own ego, he will be hard put to find his own errors. More importantly, he might be unable to receive any assessment of the product constructively because he would regard any criticism as a personal attack. In extreme cases he will be reluctant to release the product for any inspection for fear of criticism. Such an approach has two adverse effects:

—Faulty items might be produced without the errors being detected early enough for effective correction.

—By receiving no feedback or by not being able to receive constructive feedback on the errors, producers are unlikely to improve their methods of production.

The ego-less environment requires trust to be built between those who regularly review each other's work. This trust is built by encouraging the producer to select his own reviewers.

Building trust also means that the documents resulting from the walkthrough are *not* deposited with the quality assurance function or project librarian or the personnel department. Using the results of walkthroughs for direct or implied reviews of personal performance is strongly discouraged by users of the technique.

Organising a walkthrough

A walkthrough is conducted as a semi-formal meeting between the producer (the reviewee) and his peers (the reviewers) for the purpose of finding errors in the producer's work. The meeting is organised by the producer of the work and is attended by those whom he invites. The reviewers must perform the tasks specified for them by the reviewee. There are only three regular tasks:

—Being a critic of the work.

—Being a scribe for the proceedings.

—Acting as a moderator when arguments are heated.

The reviewee has to circulate in advance all relevant documents related to the work being reviewed, and must also follow up the points suggested as a result of the walkthrough.

Practical guidelines for walkthroughs are:

—Each participant should be a peer of the person producing the product.

—Each participant should receive the product and any supporting materials before the walkthrough, and should be given reasonable time to review them.

—The reviewee should select the participants and invite them to play the roles of scribe, moderator, quality assurance champion, etc. The participants should not regard their contribution as a favour to the reviewee but as a positive aid that helps him to avoid later problems with the product.

—The walkthrough is not the place for displays of ego and assertive behaviour: comments should be made as questions about the product, not as statements about the producer.

—The scribe is responsible for noting the errors found and the recommendations made. He is also responsible for distributing the appropriate copies to the walkthrough participants.

—The moderator is responsible for opening and closing the walkthrough on time, maintaining a sense of urgency, resolving conflicts, and stopping any irrelevant discussion.

—Action points should state concerns that need attention, rather than give solutions: ideas, solutions, irreconcilable differences of opinion, etc. should appear as "other notes".

Guidance on conducting walkthroughs and providing time in the project plans for walkthroughs appear to be the major prerequisites for using the technique effectively.

Walkthroughs have produced some spectacular results in the system development area. The case history presented in Chapter 5 describes a project where the only formal means of quality control was the walkthrough. This was sufficient to achieve zero-defect development in a large and complex system.

THE INSPECTION METHOD

The inspection method was developed in the early 1970s by Michael Fagan at IBM's laboratory in Kingston, New York. Fagan outlined the method in his article "Design and code inspections to reduce errors in program development" published in the IBM Systems Journal. Fagan is an engineer and he developed the inspection method by applying engineering quality control rules to software, in particular to the designing and coding phases of systems development.

The inspection method is a form of technical review that validates the quality and accuracy of a product, and detects the errors. In other words, it is used to maintain quality control. By applying it at points prior to system completion it is possible to obtain early warning of problems and to correct errors relatively inexpensively. The method may also be used to ensure the completeness of a stage by checking that the product meets the criteria specified for the end of that stage.

The inspection method requires a team of people, each of whom is assigned a specific role in the inspection. An important feature of the method is the use of checklists and statistics about the errors to be searched for during the inspection process. These lists and statistics are compiled by carefully analysing both the experiences with similar pieces of work and the results of previous inspections. The result of an inspection meeting is an action report, and the inspection process is completed when satisfactory action has been taken on all the points.

The inspection team

The inspection team should have a minimum of two members and a maximum of seven. The usual number is four. Each member of the team must be qualified to make a positive contribution. Some of the team should ideally be from outside the project whose product is being inspected.

The following participants are essential to the success of the inspection method:

—The inspection moderator. The role of the moderator is to lead the inspection process, and the person chosen must have received appropriate training.

—The author of the product being inspected.

—The user of the product being inspected. (In some circumstances the author also plays the role of the user.)

Other roles may be assigned if considered desirable.

The inspection process

There are six steps in conducting an inspection, each of which has specific objectives and involves specific personnel:

—Planning: the moderator sets up the inspection schedule and assembles the team.

—Overview: relevant project and back-up documentation is distributed to the team members, together with a high-level summary of the material to be inspected.

—Preparation: the team members read and review the documentation with the aid of checklists, and they list any questions.

—Inspection meeting: the material is analysed, and all the errors are noted. Checklists are used to ensure the completeness of inspection. A problem-definition report is produced, summarising all the errors detected.

—Rework: the errors are corrected and the corrections are implemented.

—Follow-up: the inspection team verifies that all errors have been corrected, using the problem-definition report as a checklist.

The inspection method has several special features, including the following:

—The moderator, not the author of the material being inspected, leads the inspection. Training courses are available to train people to be effective inspection moderators. Most advocates of the technique believe that this training is essential.

—Checklists also are an essential ingredient of the inspection method. They ensure that attention is focused on detecting problems and/or errors. Checklists are used during preparation and during the inspection meeting. When compiling the checklists, Fagan recommends that errors should be classified by type, and ranked according to the frequency with which they occur. In this way it is possible to highlight the most serious types of errors to look for during inspections. The analysis of inspection results should be used to refine the statistics.

Fagan expressed the rationale for using checklists in these terms: "Numerous experiences have shown that people have to be taught or prompted to find errors effectively. So it is prudent to condition them to seek the high-occurrence, high-cost error types, and to describe the clues which usually betray the presence of each error type".

Preparation for an inspection meeting should take between half an hour and half a day per team member. The inspection meeting itself should last no more than two hours. If more time is required several meetings should be scheduled.

It is important to hold as many inspections as are required to produce a correct and good-quality product. Where more than one inspection is necessary, more than one inspection team may be involved. IBM's experience suggests that higher-quality systems are achieved if re-inspections are carried out by a new team.

Finally, it is important also to remember that the aim of the inspection process is to detect problems and/or errors. It does not provide solutions to the problems.

Experiences with the inspection method

The inspection method was first used at IBM during the programming phase of a development project for a new operating system. Subsequently, IBM extended its use to the programming phase of application systems, to maintenance programming and to other phases of development prior to programming.

The most widely quoted statistics on the success of the inspection method are those produced by Fagan. His stated improvements are based on the difference between the results produced by inspections (and by walkthroughs) and those produced by traditional methods for similar development projects. For system programs, the increase in productivity of the coding operation is stated as 23 per cent. The reduction in discovered errors in seven months following unit testing is stated as 38 per cent.

For application programs, the improvements quoted by Fagan are:

—Twenty-five per cent saving in programmer resources.

—Between 38 and 82 per cent reduction in the number of errors found per 1,000 statements before unit test.

—Between eight and 18 per cent reduction in the number of errors found per 1,000 statements before acceptance test.

—Zero errors at acceptance test and in the first six months' use.

The inspection method is most widely applied in the programming area (for which it was specifically developed). Nevertheless, the method can be adapted for use within other development phases, in particular to assess the completeness, accuracy and quality of the work being produced. Experience has shown that the concept of inspections may be applied to any functionally complete part of a system. The method may be adapted for use at end-of-phase checkpoints or at predefined checkpoints within a phase.

Gerry Weinberg's "Ethnotechnical Review Handbook" provides guidelines and checklists to be used with technical reviews such as the inspection method. He considers the inspection method to be applicable at all phases of development, including:

—Functional specification.

—Design.

—Coding.

—Documentation.

—Test plan.

—Tools and packages selection.

—Operations and maintenance.

The inspection method requires a relatively high investment prior to its introduction, in particular for compiling checklists and error statistics. For this reason, the method is more likely to be used for software systems where the coding is complex, rather than for application projects. Once implemented, however, the inspection method takes less time to apply to application programs than to system programs.

Many believe that the inspection method is also particularly relevant to smaller projects and maintenance activities. With smaller projects, the relative importance of lower-level reviews is increased, and the inspection method is an appropriate way of conducting such reviews. The method is relevant to maintenance activities because modified systems are likely to contain a high number of errors.

The inspection method is more widely used in the United States than in Europe. However, its use in the United Kingdom is sufficiently wide for several moderator training courses to be run — in particular IBM's "Software Inspections: Moderator Training", which is run three times a year.

TESTING TECHNIQUES

Clearly, thorough testing is an important aspect of producing a high-quality system. In practice, system testing usually takes one of two distinct forms:

—Demonstration-type testing, which shows that the system works in a known environment using selected inputs.

—Exposition-type testing, which sets out to expose hidden errors, by comparing system outputs with values determined by an independent process.

Demonstration-type tests assume that the system will work, and the test data will be chosen accordingly. Exposition-type tests set out to prove that the system does not work and because of this they are likely to be more rigorous and comprehensive than demonstration tests. In a systems environment, exposition testing is the nearest equivalent to conventional industrial quality control.

If there is pressure to reduce costs, the bias will be towards demonstration testing. But this type of testing can lead to lower-quality systems because it does not try to uncover hidden errors. On the other hand, exhaustive exposition tests will be very expensive to conduct and, beyond a certain level, such tests may not provide any substantial extra benefits.

System testing can also take either a 'black-box' or 'transparent-box' approach. A black-box approach assumes no knowledge about the internal working of the system, whereas the transparent-box approach considers the overall design of the system.

The traditional approach has been to test systems comprehensively using the black-box approach. There is an inherent weakness in this approach because it only proves that the system works (or does not work) under specific conditions — it does not provide assistance in identifying in what way the system is wrong or where the error is. Increasingly, systems staff are realising that a transparent-box approach is more efficient, especially where precision, time and cost are important factors.

Testing levels

Technical tests ensure that there are no defects in the individual parts of a system. Tests at this level may need to be repeated several times if there are several sub-component levels. Examples of this type of test are unit, module, program and sub-system tests. At the technical exposition-testing level, much detailed work has been done by hardware manufacturers and military-oriented government agencies. As a result, manual and automatic methods for testing program code have been developed, with the emphasis being on correctness and verification tests. We question whether these efforts alone will lead to high-quality systems being produced. There is ample evidence to show that technically correct and verifiable programs do not necessarily produce usable systems.

The usability of a system depends as much on whether its behaviour is predictable. This applies particularly to interactive online systems, where users will judge the quality of the system in this way. People can learn to live with a system that behaves in a predictable way, even though it may not be completely accurate. The implication is that system tests will in future need not only to check for technical correctness, but must also concentrate on the behavioural, human-interaction aspects of systems to demonstrate that the system as a whole meets the users' expectations. The term "system test" more accurately describes this second level of testing but, in practice, most system tests are usually technical tests.

The impact of the development approach on testing

The development approach influences the way sub-assemblies of modules and programs can be tested. With a top-down approach, the uppermost (and usually the most important) control modules can be tested first. Lower-level modules can then be included in the overall test plan after they have been unit-tested. The advantages of a top-down testing approach are that:

—Testing can proceed in parallel with development activities, thereby reducing considerably the overall elapsed time.

—The top-level modules can act as a test-harness for the rest of the system, thereby eliminating the need for expensive test drivers.

—The top-level, critical, modules will be tested more thoroughly than lower-level modules.

The case history in Chapter 5 clearly illustrates these advantages.

The design approach adopted will also affect the way in which system testing is carried out, particularly at the behavioural, human-interaction level. People relate the behaviour of a system to the function it performs, and testing and identifying errors will cost more if the system has been constructed from multi-purpose units, because:

—An additional level of testing is required to verify that the functional requirements have been met. This level of testing is not required if the system was designed using modules that correspond with a specific function (or sub-function), because a module test will also test the functional correctness of the module. In addition, the higher-level function (made up of sub-functions) will also be tested when lower-level tests are performed.

—Investigation of an error will first identify the likely functions associated with the failure. If multi-purpose modules have been used, all parts of the system associated with the function will be suspect, and tracing the error will require more time and effort.

The additional levels of testing necessary also means that maintenance and system enhancements will require more effort. The clear message is that the quality of systems (and their costs) is influenced strongly by the development approach and design method adopted.

Designing test cases

System tests can have one of three objectives:

—To test that the system as built matches the system as specified.

—To discover if the system as built behaves in ways not included in the specification because the developer has included additional features (deliberately or accidentally).

—To demonstrate that selected inputs produce the required outputs.

It is usually impractical to design test cases that cover all possibilities, and the aim has always been to reduce the number of cases to the minimum required to test the system as comprehensively as possible, so that the span of the tested behaviour very nearly covers the span of the specified behaviour. Several methods have been suggested for achieving this aim, and some of them are reviewed briefly below.

Logic coverage

This method is designed to test the key logic components of a system by testing the paths between decision points. Test cases can be designed to test every decision-to-decision path at least once. Alternatively, the test cases may only test the paths between the most significant decision points. Another variation is to test the paths into and out of modules.

Equivalence partitioning

For each specified input condition there is a range of cases (or classes) that are valid and invalid. These classes need to be identified, and a test case designed to test each one at least once.

Boundary value analysis

This type of testing stretches the system to its limits by designing cases to test the boundary of the specified behaviour.

Cause and effect graphing

The basis of this method is that inputs can be regarded as causes and output as effects. Using a simple decision table, the relationships between causes and effects can be tabulated. An analysis of the table will identify the number and nature of test cases required.

Error guessing

Error guessing is used mainly to measure the effectiveness of a testing tool, a test method or a test case. The test is first performed as intended. A known error is then deliberately introduced into the system and the test is repeated. If the results are identical, the test is not effective. Error guessing is also useful when developing new test cases.

Testing tools

Testing tools, in the form of test data generators and testing harnesses, have been used for many years. More recently, sophisticated automated tools have become available with the aim of highlighting anomalies in the software. These tools can take two forms: static tools and dynamic tools.

Static tools

A static tool analyses the software itself and highlights code that logically cannot be executed or data items that will never be used. (According to recent studies, many operational systems contain a significant percentage of code that will never be executed.) At present, static analysers are expensive, but the Fortran-based Toolpack represents a step towards the provision of inexpensive testing tools.

Toolpack is a set of Fortran testing tools created by Leon Osterweil of the University of Colorado. One part of the Toolpack environment is a static analyser that detects a wide class of program errors. Toolpack is being developed by six universities and research institutes in the United States, together with the British Algorithm Group. It should be available as a commercial product during 1985.

Dynamic tools

Dynamic testing tools are used as the program under test is being executed. Their primary purpose is to measure the effectiveness of test data. Many such tools work by inserting 'software probes' that report on how much of the program has been executed. The effectiveness of the test data is measured by the number of program statements or segments executed.

One problem with dynamic testing tools is that statement coverage is not necessarily an indication of the thoroughness of a test. Research has shown that errors may still be present even if every statement in a program has been exercised by the test data.

New developments

A new development in the field of testing tools is symbolic execution, which analyses the logic of a program by executing it not with data but algebraically. Instead of numerical results, the tool produces an algebraic representation of the program output.

Two of the leading researchers in symbolic execution, Lori Clarke and Deborah Richardson at the University of Massachusetts, are attempting to use the technique to build automatic test data generators.

Another new development is an attempt to automate the error-guessing technique mentioned earlier. Timothy Budd at the University of Arizona is pioneering this work.

Dynamic testing tools are also being developed in the United Kingdom by Mike Hemmel at Liverpool Data Research Associates. His linear code sequence and jump (LCSAJ) technique also tests the thoroughness of the test and the adequacy of the test data.

Summary

Without clear objectives and careful planning, testing can easily become haphazard and misdirected, and can result in wasted effort and resources. An important role for the system quality assurance function is to ensure that an adequate testing strategy has been adopted. Another is to ensure that system developers are aware of, and have received training in, the various testing techniques.

CHAPTER 4

PRACTICAL EXPERIENCE OF SYSTEMS QUALITY ASSURANCE

During our research we discussed systems quality assurance with representatives of about 30 companies. In this chapter we present an overview of their experiences, which are drawn from both the software development functions of suppliers and the information systems functions of user organisations.

In broad terms, a significant minority of organisations are trying to apply traditional quality assurance techniques to information systems work. A few are applying general quality programmes, normally following the introduction of company-wide quality programmes initiated by top management. This latter group comprises mainly information technology suppliers (both hardware and software) — usually companies who have either suffered from, or felt increasingly threatened by, Japanese competition.

RECOGNITION OF THE NEED FOR SYSTEMS QUALITY

Most of those we spoke with recognised that some aspects of the work their organisations undertake could be considered to be some form of quality assurance or quality control. However, less than half of the companies had established a formal systems quality function. Furthermore, very few such functions have been in existence for more than three years (less than ten per cent in our sample). Even those information technology suppliers with a reputation for high quality may have established formal functions only within the last three to four years.

Our research confirmed that there are considerable variations in the definition of systems quality. Many of those we talked with used terminology borrowed from the manufacturing environment, using the fitness-for-purpose or meeting-requirements definitions of quality. Others described quality in terms of quality factors, such as reliability and maintainability. A few equated the word quality with concepts (such as "a way of life") that appear to originate from the works of quality specialists such as Deming and Crosby.

Although there were some minor differences of interpretation, there was more general agreement on the meaning of quality assurance and quality control in a systems context. Quality control was seen as a policing activity, checking correctness of work and conformance to standards and highlighting deviations. Quality assurance was regarded as being broader in scope, giving attention to aspects such as methodologies and skills. Some regarded quality assurance as ensuring that quality control is carried out.

ORIGINS OF FORMAL SYSTEMS QUALITY FUNCTIONS

We identified five origins for the formal systems quality functions that companies had established:

1. Company-wide programmes
Some companies had established a formal systems quality function as a consequence of company-wide quality programmes. These companies were all information technology suppliers.

2. New methodologies
Many systems development methodologies, such as Arthur Andersen's Method/1, include quality assurance and/or quality control activities. Implementing these methodologies often includes establishing a systems quality function to carry out the relevant activities.

3. Attention to standards
Sometimes, a formal systems quality function has been established following the creation of a formal standards unit. This move is often seen as a natural progression to ensure that the standards are followed. On the other hand, establishing a formal systems quality function can identify the need for a standards unit to be set up.

4. Commercial reasons
Companies supplying software or systems development services, particularly to government defence departments, have found that a prerequisite is to establish a formal systems quality function.

5. Acceptance testing
Several companies have set up a systems quality

function that is quite separate from the systems development function. The main aim of such a function is to test systems prior to operational use. This may be by undertaking complete testing in order to establish that the system meets the specified requirements, or by focusing on the operational acceptability of systems, particularly with respect to operational performance.

ORGANISATIONAL POSITION OF SYSTEMS QUALITY FUNCTIONS

Most formal systems quality functions report either to the head of systems development or to the head of the technical services or support function (about 40 per cent in each case). The remaining 20 per cent had a variety of reporting relationships, such as direct to the head of the information systems function or to a staff function, such as planning or strategic programmes.

Quite often the systems quality function was associated with the standards function, and was usually positioned two or three levels below the head of the information systems function. Some companies believed that organisationally separating the systems quality and systems development functions was crucial to the success of the quality function, despite the fact that political difficulties were likely to ensue.

QUALITY FOCUS

Organisations varied in the particular focus they adopted for their systems quality activities, irrespective of whether a formal function had been established. We identified six different approaches. Some organisations were using a combination of two or three of these approaches.

1. Line management focus

The major characteristic of this approach to quality assurance is that it clearly places the responsibility for quality on the line managers of the various operating units. Our research showed that this approach was being used in the information systems departments of some large organisations where the department had been established as a separate operating unit or company. In these cases the whole group or organisation was committed to quality, and the information systems company's quality approach was in line with the group's philosophy and practices for quality.

The quality focus is achieved by appointing a quality manager at a very senior level, usually reporting to the managing director. The quality manager's responsibility is to see that line managers adopt the appropriate practices and mechanisms for ensuring quality in their own areas, and his role is mainly one of informing, guiding and monitoring. He provides advice on company policy and practices and on available methodologies and training facilities. He also acts as a high-level consultant for quality issues.

2. Planning focus

Where the systems quality resources were limited and the emphasis was on quality assurance rather than on quality control, some organisations had chosen to pay close attention to the project management aspects of systems projects. This attention may be limited to the initiation stage of projects or may be directed to the start of every stage of development. The intention was to ensure that the way the project was set up (manning levels and the skills deployed, for example) matched the nature of the project and gave it a good chance of fulfilling its mission.

Some organisations' standards required project managers to prepare periodic quality plans. These plans identified the measures that would be taken to produce and ensure quality, and were a principal focus of attention for the systems quality function.

3. Process focus

We identified two types of process focus, depending on whether the emphasis is on quality assurance or quality control.

Most organisations that have adopted a process focus emphasise quality control. Their approach is a policing one, ensuring adherence to methodologies and standards. The intention is to ensure the completeness of a system rather than to examine its content. Frequently, the policing is undertaken in formal reviews, typically by two-man teams who spend two to three days examining documents. Pre-review briefing meetings and post-review meetings (about four to six weeks later) are also held.

A minority of organisations with a process focus emphasise quality assurance. The focus here is on trying to understand, measure and improve the development process itself.

4. People focus

A few organisations have included responsibility for some aspects of staff training in their quality assurance or quality control functions. The emphasis usually was on improving quality by making good any weaknesses in skills or knowledge.

Those who were operating a broad quality programme tended to focus on providing managers with training both in quality concepts and in techniques that would help them assess their own and their staff's quality performance. Other organisations provided all staff with training about quality concepts, usually supported by a poster campaign. Hewlett-Packard, for example, makes a point of emphasising to its software development personnel that they must assume that the products of their labours will go straight to the customer without further checking. The company provides its staff with what it terms "customer-satisfaction" training.

5. Product focus

Some organisations placed their quality emphasis on "fitness for purpose". We identified two types of focus, depending mainly on the resources available.

Those with limited resources focus on examining what is being proposed following an initial feasibility study. The prime aim of this approach is to make sure that the proposal makes good business sense, and that the development risks, if any, are acceptable.

Those with greater resources pay most attention to testing the system prior to operational implementation, to ensure it meets the specified requirements.

Interestingly, we found no organisation with a product focus putting its formal systems quality effort into the intermediate stages of the development process.

6. Tools and techniques focus

Some systems quality functions concentrated on trying to find or develop tools and techniques that would help to build quality systems. Included in this group were companies using techniques such as walkthroughs and inspections, either on a selective or mandatory basis.

One organisation was using quality circles on an optional basis in its management services function. Quality circles were used as a platform for staff to identify and consider ways in which quality could be improved, and only matters directly concerned with quality could be discussed. This was done to avoid a recurring problem with quality circles where specific quality issues are neglected as wider issues become the focus of the work. The danger is that the emphasis of quality circles can change to providing staff with opportunities for involvement rather than with opportunities to focus on quality matters.

Yet another organisation was placing particular emphasis on developing automated tools for testing, which is generally regarded as being one of the least well-supported aspects of systems work.

FACTORS AFFECTING QUALITY

We asked the representatives of the organisations involved in the research to identify the factors that they believed affected the quality of the systems that were developed. The responses to our question were very varied, but the following list indicates the types of factors that were cited:

—Availability of well-trained development staff and quality assurance (or quality control) staff.

—Existence of good project control procedures, formal reporting lines and well-structured projects.

—Use of techniques such as walkthroughs and regular project-team meetings.

—Use of good basic techniques for data analysis and structured systems analysis.

—Use of effective change-management procedures.

—Availability of good tools for supporting testing activities.

—Availability of programmers' workbenches and automated development aids.

—Availability of adequate typing and word processing capacity.

STAFFING THE SYSTEMS QUALITY FUNCTION

The number of staff employed in systems quality assurance or quality control functions was generally quite low — typically one or two people, representing about two per cent of the total development resources. We believe that those with a broad quality focus, rather than a quality assurance or quality control focus, were getting the best results, mostly from one person working with managers (providing them with awareness training, for example).

One organisation, however, has a systems quality function with a staff level equivalent to about 25 per cent of the staff in the development area.

Recruiting suitable staff for the systems quality function was generally regarded as a problem. The ideal was someone who had development experience, was well respected by his peers, and was diplomatic. Typically, such staff were seconded to the systems quality function from the development area for a period of one or two years. Some organisations used outside consultants and others worked on the basis of temporary short-term (two to three weeks) assignments where people were assigned to perform a particular role in one specific quality assurance or quality control activity.

Those organisations engaged in a broad quality programme tended not to be as concerned about the need for previous development experience. What seemed to be more important was to have a good understanding of quality concepts and principles, and to be able to apply them.

COSTS OF THE SYSTEMS QUALITY FUNCTION

The costs attributed to the systems quality function varied considerably across the organisations that participated in the research. Much of this variation can be explained by the varying awareness of what all the systems quality costs might be. Even so, there was a general consensus that the cost of the function was usually about two per cent of the total information systems budget, though we were told of cases where it was as low as one per cent and as high as eight per cent. The major exceptions were those organisations where the quality focus was on acceptance testing, where costs were as high as 25 per cent of the total information systems budget.

When indirect quality assurance or quality control efforts (that is, those carried out by line personnel) were included, estimated costs were put typically at between 30 and 40 per cent of development costs.

MEASURING SYSTEMS QUALITY

Although some research (particularly that by Tom Gilb, the independent specialist) has attempted to define possible measures for assessing levels of quality in systems (often referred to as quality metrics), we did not encounter any company that was using them. Nevertheless, the few organisations that were undertaking broad quality improvement programmes had devised their own practical and simple measures. For example, IBM's internal systems development function uses the following measures:

— Number of faults per thousand lines of code, although this was soon to be replaced by number of faults per function point.

— Number of faults per system-user per month.

— Number of user complaints (of all types).

— Number of computer service hours lost.

— Number of network hours lost.

Hewlett-Packard, on the other hand, takes a broad view about metrics. Any measure that indicates a poor or decreasing performance is an indicator of some failure to produce quality. For example, Hewlett-Packard tracks actual and anticipated completion dates against originally planned dates on what it calls "bug-charts" (see Figure 4.1).

QUALITY FACTORS

We also asked about the quality factors that were judged to be important. The following factors were identified:

— The project meets its planned timescales within budgeted costs.

— The system meets the performance requirements for which it was designed.

— The conduct of the project and its documentation conform to the laid-down standards.

— The resources and skills deployed on the project are sufficient and appropriate.

— The project is properly organised.

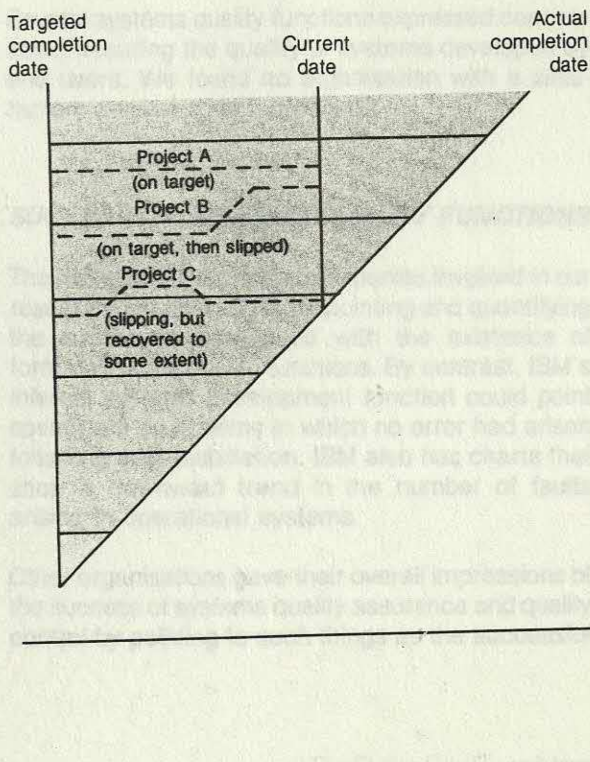
There were other factors that most companies believed important, but few, if any, of the companies were able to judge these objectively. The most frequently mentioned factors were functionality (the degree to which the system provides the functions that are required), maintainability, reliability (the traditional system quality factor), operability (the degree to which the system is easy to operate and use), and acceptability (how much users like the system and can understand it).

Although this list of factors seems reasonable, we found that in most cases systems quality assurance or quality control functions did not focus on any particular factors. Part of the problem appears to be the lack of suitable metrics.

PROBLEMS ENCOUNTERED

Six issues were frequently mentioned when we asked about the problems that had been encountered in setting up a systems quality function.

Figure 4.1 Hewlett-Packard's bug chart



1. Lack of management commitment

The most frequently voiced problem concerned the lack of management commitment, in particular in those organisations that had not introduced their systems quality function as a part of a corporate programme. In this situation, the systems quality function is vulnerable to budget cutbacks, and its existence has to be constantly justified. Quality functions appear to be particularly at risk in information systems departments with a strong focus on keeping to deadlines. It is also recognised that it is difficult to recruit the right staff in such circumstances. Those who are successfully addressing this problem are paying particular attention to educating their managers.

On the other hand, the general level of management commitment is not a problem in those organisations that have introduced quality programmes from the top. Formal quality-awareness training for managers is usually undertaken at the beginning of the programme, and is built into standard management training courses. Managers may also be trained to use techniques that help them identify the costs of correcting poor-quality work.

2. Management attitudes and understanding

Management attitudes to, and understanding of, quality issues can be an inhibiting factor, even in those organisations whose senior management is committed to a company-wide quality programme. A few examples of the problems illustrate some of the concerns.

Quality programmes attempt to draw attention to areas in which people may be failing in order to initiate corrective action. People usually are not happy to talk about these areas, particularly if the failings influence the next performance appraisal (and related salary increase).

Substantial effort is usually required to identify the costs associated with quality failures. Managers may be reluctant to expend this effort, particularly if they are sensitive about the failures.

One organisation found it particularly difficult to motivate its managers to think in terms of preventing quality failures. This was similar to the problem experienced by others of getting managers to identify and understand the costs associated with quality failures.

3. Attitudes of development staff

The problem of getting managers to be open about quality failures is also evident in development staff.

A few organisations encourage what is termed an "ego-less environment". In such environments, people welcome the views of their peers and others, so as to improve the quality of the work they produce. Problems are openly discussed so that possible training needs and/or changes in procedures can be identified.

4. Lack of influence

Many systems quality functions would like to have more influence, in order to prevent development projects from proceeding when quality failures have been identified. This issue is of particular concern to organisations operating a quality control process in which reviews or inspections are undertaken. Recommended corrective action is not always carried out because, typically, deadlines are regarded as more important.

5. Motivating systems quality staff

Many organisations find it difficult to motivate their systems quality staff, particularly those whose main role is to check the work of others by using checklists. This difficulty can be overcome to some extent by making it clear that staff are assigned to the systems quality function for relatively short periods. However, such assignments are not perceived by the staff as opportunities for personal advancement. Indeed, staff may feel they run the risk of missing out on opportunities that might arise in their normal place of work.

Some organisations recognise that the lack of a career path for systems quality staff is a problem, but one interviewee regarded a period in systems quality assurance or quality control as an important aspect in general staff development or training.

6. End-user computing

Several systems quality functions expressed concern about assuring the quality of systems developed by end users. We found no organisation with a satisfactory answer to this problem.

SUCCESS OF SYSTEMS QUALITY FUNCTIONS

The representatives of the companies involved in our research had difficulty in pinpointing and quantifying the successes associated with the existence of formal systems quality functions. By contrast, IBM's internal systems development function could point specifically to systems in which no error had arisen following their installation. IBM also has charts that show a downward trend in the number of faults arising in operational systems.

Other organisations gave their overall impressions of the success of systems quality assurance and quality control by pointing to such things as the successful introduction of new methodologies and standards, and obtaining new business as a result of having a formal systems quality function.

All companies possessing a formal systems quality function believe they can justify its existence in qualitative terms, and can identify signs of its success. However, those organisations with formal broad-based quality programmes seem to be achieving more success, and seem better placed to create an environment in which that success is likely to be sustained.

CHAPTER 5

BUILDING QUALITY SYSTEMS: A CASE HISTORY

This case history describes the development and early operational phases of a major system that computerised Denmark's primary method of financing private housing. The project is noteworthy because the quality of the system was a key consideration from the outset. The case history illustrates how a high-quality system can be developed on time, with zero defects and within budget.

The system is operated by VP-Centralen on behalf of 240 financial institutions (commercial and savings banks and independent brokers). It went live in April 1983 when, within the space of a few days, the 46 million paper bonds, with a total value of $80 billion, traded on the Copenhagen Stock Exchange were eliminated and replaced by an automated electronic system.

During the research for this report we met the project director, Dines Hansen, then vice president of VP-Centralen, now deputy general manager of BRF, one of Denmark's biggest mortgage credit institutes, and asked him to describe the mechanisms and methods used for building a quality system. (His book "Up and Running" describes the project in detail.) We also met both with key members of the original development team and with current operators and users of the system to find out:

- How they originally perceived quality.
- How they ensured that quality would be built into and delivered with the system.
- How they now see quality manifesting itself through the system, more than a year after it went into full operation.

BACKGROUND TO THE PROBLEMS

The Danish finance sector can be regarded as the equivalent of a huge company consisting of 240 divisions (the financial institutions), governed by a board (formed from government and finance sector representatives). The data processing department of the 'company' is VP-Centralen, but all the 'divisions' have their own local computing facilities. Some of the smaller divisions have joined forces and opened local data centres. They all make their own decisions, using various, and sometimes incompatible hardware, and they jealously guard their operations and data. Their systems communicate with each other through networks of different kinds.

Private housing in Denmark is financed through a bond system. Bonds are issued by the various financial institutions and are held by these institutions on behalf of the owners and traded on the Copenhagen Stock Exchange. The problem of bond handling became a major issue during the early 1970s as the number of bonds in circulation increased rapidly. Between 1968 and 1982 the monetary value of bonds increased from $9 billion to $80 billion. By the late 1970s some 45 million bonds were registered in Denmark, representing in value about 1.5 times the gross national product of the country. On an average day about 16,000 individual sales and purchases of bonds were recorded, equivalent to three to four transactions per bond each year.

The totally manual system of issuing and handling bonds was cumbersome. Registration and changes of ownership meant that many pieces of paper had to be circulated. In addition, the coupon sheets showing the interest payable twice a year to the bearer had to be cut physically. The costs of printing and distributing the coupons had risen to $2 per bond, equivalent to an annual handling charge of $8 million. An estimated three per cent of all those employed in the finance sector were engaged in bond handling.

After five years of debate between the 'divisions' and the 'board', the decision was made in 1980 to replace the old manual system. There was really no choice, because the manual system was so costly and cumbersome that it would soon have caused the 'company' to collapse.

At the start of the project, there were only four people in the data processing department, and one of these was an administrator. Needless to say, all the divisions had unique requirements and their existing systems would require differing and slightly incompatible interfaces with the new system.

Development of the new system would therefore benefit by recruiting or seconding divisional staff. But the divisions were reluctant to release their personnel. They each had their own data processing salary structure, making it difficult to assign staff to the central data processing department.

Another problem to be faced was that the changeover from a completely manual to a completely automated operation would require all the divisions to change long-established working practices.

Once the decision was made to proceed with the automated system, it had to be installed to a very tight timescale. The system could go live only during the Easter break. The target date was Easter 1983, which allowed only two years and three months to develop, test and implement the design. If this date was missed, the system would be delayed by a year.

To complicate the issue further, there was no way of returning to the old system once the new one was installed. Should the new system fail, it would bankrupt the 'company'.

There were also stringent financial constraints on the project. No direct, corporate funds were available to finance the new system. All development and operational costs were to be met from 'divisional' budgets.

THE REQUIREMENTS

The financial institutions of Denmark realised that radically new procedures were needed for issuing, trading and storing the bonds. The solution was clearly an approach that would reduce, or preferably eliminate, paper handling.

The system for bond handling had to satisfy the following requirements. Approximately 240 organisations would be connected to the system for bond registration and trading. About 1.2 million investors' accounts had to be registered and the system would need to handle about 22,000 trading transactions a day. The system would have to make five to six million dividend payments annually. Interest was to be calculated 30 days before due date on a fixed percentage of the nominal bond value, and transferred automatically on the due date to the investor's bank account. Dividends and the capital repayable on maturity would need to be handled in the same way. Bondholders needed to receive a statement annually, and ad hoc statement enquiries also would have to be accommodated. Cash transfers between the various issuing and holding institutions and banks would go through an existing electronic funds transfer system. Direct electronic Stock Exchange dealing, however, was not a requirement of the system.

The system would have to accommodate several access methods: large investors and major bond dealers would need to access the system via their own terminals, and must be able to initiate inter-account transactions. Other users would enter the system either through their own terminals or through some 30 existing regional data centres, run by groups of savings banks in different parts of the country. Altogether, about 15,000 terminals in the various financial networks would require access to the system.

There were two additional and very important requirements:

- The system had to be fair — that is, it should not compete with the financial institutions and it should allow free competition amongst the participating financial companies.
- The system should not disrupt the normal dealing between the financial institutions and their customers (the investors and bond owners).

HISTORY OF THE PROJECT

In 1975 the Ministry of Housing formed a committee of representatives of the finance sector to develop a comprehensive solution to the bond-dealing problem. The committee published in 1979 "Yellow Report No. 793", in which it analysed the legal, technical and economic aspects of establishing a centralised computer system to replace physical securities. The proposal was to set up a centre for trading electronic bonds, and to include in the centre's work all registration, interest and dividend payment activities related to the handling of bonds. In 1978, a small team (the secretariat) consisting of four members was established to plan the project.

Between 1978 and 1980 the secretariat developed preliminary definitions of requirements and a project plan for the estimated 250 man-years of development work. This strategic planning phase also included some detailed planning of the development environment. In particular, the project organisation and the framework for the very complex and entirely new development approach were defined. A software engineering approach was selected. This clearly defined the systems life cycle and specified the analysis, design and implementation techniques to be used. The software configuration for supporting with automated tools the development and future operations of the system were therefore established well before the hardware configuration was selected. Those involved with the project believe that this was a key factor in its subsequent success.

In July 1980 the Danish Parliament passed the Act that allowed the establishment of VP-Centralen, a non-profit-making organisation for handling the existing paper securities by electronic means. VP-Centralen is governed by the Mortgage Credit Council, the Copenhagen Stock Exchange, the Danish Banks Association, the Association of Danish Savings Banks, the Insurers' Society and the Danish Central Bank, through a central board whose chairman is appointed by the Minister for Trade and Industry.

A new data processing centre was built near Copenhagen and opened in May 1981. Its staff grew from the original secretariat of four in 1980 to 24 by January 1981. Among the first positions established were those of the data administrator, the database administrator, the four key project leaders and the necessary clerical personnel. The total staff at the centre was originally planned to be 107. By January 1982 there were 85 employees, and this had risen to 100 by January 1983. By late 1984, the number of staff had stabilised at 105.

The system development effort was scheduled for the period March 1981 to March 1983, with live operations to commence in April 1983. The deadline set for the users (the 240 different organisations) to agree how to modify their own systems to interface with the centre was July 1981. VP-Centralen steadfastly refused to consider solutions suggested by the user community. Instead, it demanded that users state their requirements in the form of a problem to be solved. VP-Centralen's staff believe strongly that it is their own responsibility to develop the best solution for a given problem. Nevertheless, the user organisations put in about 500 man-years of effort to make their own systems compatible with VP-Centralen's system.

In September 1981 a prototype of the complete VP-Centralen system was presented to the users, showing all the major system functions and ensuring that all user requirements were identified. Two more prototypes were presented in December 1981 and in March 1982. At the same time, selected users began testing VP-Centralen's terminal system. The final version of the user manual was produced in July 1982 on the basis of these tests, well before the full system was completed. The manual described all legal, clerical and technical aspects of the system from the users' perspective.

From July 1982 onwards VP-Centralen conducted an intensive public information campaign using newspapers, television advertising and specially printed booklets to inform investors and the public at large of the project, its aims, schedules, plans, and the likely effect of the development on securities, trading and investment.

The operational environment was ready in September 1982, and interface and selective application testing began in October 1982. Overall system testing, involving 29 data centres and 500 terminals, started in November 1982 and was completed in six weeks. (This timescale was possible because the technical system at VP-Centralen was already operational and had been tested for six months prior to external system testing.)

The decision to 'go live' was taken by the board after it evaluated the results of three different kinds of systems tests that were performed during November and December 1982. Surprisingly few (and only minor) errors were detected during the take-on phase and no major disaster halted the build-up of the system.

The system was 'inspected' closely from the start of the systems test and a rigorous problem-recording system was instituted. During the mandatory tests up to January 1983, 681 problems were recorded, and 50 of these remained unsolved, although none of them was critical to the operation of the system. These were resolved during January 1983. VP-Centralen specified strict acceptance criteria for individual data centres' systems before they were allowed to connect up for full operations.

The gradual take-on of the workload began at the end of January 1983 with the conversion to the system of two small mortgage credit institutes. These two institutes handle about 60,000 accounts and about 5 per cent of the bonds in circulation.

In April 1983 the Copenhagen Stock Exchange suspended operations for a few days so that the remnants of the old manual system could be cleared away before switching over to the completely automatic new system. During this period there were several sudden changes in the Danish political and economic environment. In addition, the Danish Central Bank announced that foreign investors would be allowed to purchase government bonds. As a result, there was a substantial increase in activity on the Stock Exchange during the first day on which the new system was used for trading bonds. The system had to handle up to three times the maximum load that it was designed for. (A similar situation caused the Stockholm Stock Exchange to shut down for ten days in May 1983.)

During the first six weeks of live operations it was realised that a major business policy had originally been incorrectly stated. The system had been built to match this faulty policy statement causing, in the event, major congestion at a certain part of the system. This had a cumulative effect, which halted the system completely. VP-Centralen's mandate did not allow it to change the system of its own accord. The financial institutions and brokers had to reverse their earlier decision before technical modification of the system was possible. The decision of the finance sector to change the policy was taken quickly and was unanimous — partly because of the clear explanation provided by VP-Centralen for the cause of the problem. The system changes (which by any standard would be classified as major) were implemented in three days.

At the end of 1984, the users were delighted with the system. The annual cost of running the centre was about $10 million, which compared very favourably with the costs of the old manual system. Staffing levels, operational efficiency, running costs and the level of maintenance were all exactly as planned in 1980. An extension of the system for handling shares and other securities was being considered, and a contract has been signed to export the system to Norway.

THE QUALITY APPROACH ADOPTED

From the beginning of the project, the planning team realised that the quality of the services provided would depend on the organisation and management control within the centre. The team also realised that VP-Centralen's overall performance depended on keeping software costs down. This meant that people's productivity had to be increased and the need for maintenance had to be decreased. Thus, the overall objectives were to maximise development productivity and minimise the maintenance workload. The quality of the system was therefore related to the level of maintenance required after development: high quality meant no maintenance, and this emphasised the need for zero-defect development.

As a consequence, the approach to quality adopted by VP-Centralen had the following characteristics:

- The planning team carefully analysed the most common defects in systems and identified the ways in which system development projects usually go wrong.
- The organisation of VP-Centralen was designed to support the development and operation of the system.
- The system development and operations processes were deliberately designed to help the creation of a quality system and the provision of a quality service.

The main reason for the length of the initial planning phase was to ensure that developers and operators would have the optimal arrangement, the most appropriate tools and the best management support for carrying out their work. The centre's management saw that its prime responsibility was to create and maintain such an environment. Management believed that, without this environment, there was little chance of success.

Six guiding principles were laid down to ensure that a quality system was built, and to prevent quality deteriorating in the future. They were to:

- Resist the temptation of "re-inventing the wheel".
- Resist the temptation of using short cuts in the system and quick fixes in the maintenance process.
- Eliminate repetitive manual processes that were likely to introduce human error.
- Simplify the processes within the system, and opt for simple and reasoned development, maintenance and operations methods.
- Resist the temptation of allowing people to specialise and allowing specialist skills to be segregated.
- Choose a reasoned organisational structure that harmonised all the interactive elements of work: people, automated tools and methods.

VP-Centralen recognised that there are four key potential causes of low systems quality:

- Large and complex system projects.
- Complex system functions, and complex data handling and access procedures.
- A multitude of highly specialised technical development skills segregated into separate units.
- Systems as implemented deviating from the specified user requirements.

VP-Centralen's approach to combating these potential problems was therefore to simplify the project, reduce the system's complexity, decrease the amount of specialisation and insist on user requirements being satisfied. The ways in which each of these aims was achieved are explained below.

Simplifying the project

Many system development problems occur with large and complex projects, and VP-Centralen's system, by its very nature, promised to be large and complex. Despite this, the planning team deliberately set out to make the development project as simple as possible by breaking it down into smaller, simpler, controllable units. VP-Centralen also needed to design and institute a mechanism by which these simpler parts could be coordinated and controlled. The organisational challenges posed by adopting such an approach were met by paying great attention to planning.

Reducing the system's complexity

VP-Centralen believed that a modular systems design is a prerequisite for maintainability. Furthermore, systems complexity can be reduced by understanding the structure of the interactions between the parts (or modules). The modules themselves have two important quality parameters:

—Coupling, or the level of dependence of one module on other modules. This parameter should be as low as possible in order to provide a high level of independence. In data processing terms this means that only the minimum amount of data should pass between modules.

—Cohesion, or the internal strength of the module. This parameter should be as high as possible to provide a specific purpose for the module. In data processing terms this means that the module should perform a single, well-defined function.

Designing modules which possess these characteristics reduces the potential impact of any system changes, because any change will be limited either to a single module or to a well-defined subset of the modules. The change process therefore becomes predictable and can be defined. Changes can be implemented without generating unforeseen side effects.

At VP-Centralen, the design method was chosen to aid modular design. In addition, every part of the developing system was evaluated for modularity, coupling and cohesion, as these were recognised as being the key design characteristics associated with good quality.

The team also realised that the use of maintainability as the prime measure of quality meant that the data handling and data access procedures had to be as simple as possible. Using traditional methods, many system errors can be introduced because:

—The data handling and data access instructions are miscoded.

—The data relationships, codes, sequences, etc. are misunderstood.

The first problem can be solved by using a database management system that allows the required data to be accessed by a single command. This feature has the added benefit that, compared with conventional file handling techniques, it reduces the number of lines of code typically by between 50 and 60 per cent.

The key to solving the second problem is to use a data dictionary. In addition to providing a central definitive description of data structures and inter-relationships, a data dictionary is also the key to controlling the analysis activities. The data dictionary may be used to record the definitive descriptions of processes and dataflows, together with their components and inter-relationships. VP-Centralen decided to use its data dictionary for this wider purpose.

The design method chosen and the need to simplify data handling not only dictated the choice of methodology and tools, but also had important implications for productivity, maintainability and organisational structure. As a consequence, the data management function became a central position in VP-Centralen's organisational structure.

Decreasing the level of specialisation

Using individuals with specialised skills in a project means that additional management time has to be expended on coordinating people, skills and activities. As a result, individual assignments often are not supervised properly. Yet many of the highly specialised areas in data processing result from using a development approach that sets out to provide bespoke solutions to a single, static problem which is not expected to change.

To avoid these problems, VP-Centralen adopted a different approach, encouraging overlapping skills and making little distinction between analysts, designers, programmers and other specialists. Development work was organised in small multi-disciplinary teams. For future enhancements and maintenance, the developers were organised into four large teams, each of which included all data processing development disciplines. Short-term specific skill requirements were satisfied by the temporary use of external consultants.

Ensuring users' requirements are met

The most common system development problems are caused by deviating from the users' requirements. Often, the assumption is that only the users know what they need, so the work focuses on the users' perception of a data processing solution, rather than on providing an accurate description of the work process that, potentially, needs to be computerised. In other professions (such as architecture), however, models are used to provide specific reference points between the professional and the client. This concept can be applied to systems development projects in two obvious ways:

—First, by building models of the processes and information flows that the user has or needs.

—Second, by implementing these models precisely in the technical environment.

The importance of building models was taken very seriously at VP-Centralen. The analysis phase was devoted solely to building models of the system at different levels of detail. Development teams, users and management were all involved in checking and re-checking the accuracy of the models.

The methods and tools chosen supported the building and documentation of the models, and ensured that they were then implemented in an orderly step-by-step way which permitted no deviation.

THE ORGANISATIONAL APPROACH ADOPTED

The planning team adopted a highly analytical approach to organising VP-Centralen so that it would produce the highest-quality work. The internal organisation of VP-Centralen was set up to match the demand for control in the context of the four key areas detailed above. The emphasis on data management and multi-disciplinary teams meant that the three key organisational positions, reporting direct to the director of system development, were:

—The data administrator.

—The database administrator.

—The development project leaders.

Development staff were assigned as required to the various development projects, but reported on general matters to the director of operations (see Figures 5.1 and 5.2). Interaction with the various user organisations was coordinated by the director of user liaison and was organised through various committees (see Figure 5.3).

A separate maintenance section was not established. The original estimates for staff were based on the belief that it was possible to reverse the normal 20:80 per cent relationship between the development and maintenance workload. The argument was that a well-planned, well-structured, well-developed system, supported by extensive automatic aids, would not require traditional maintenance activities. The planners believed it was possible to create a zero-defect development environment. New system requirements and adjustments to the system brought about by external changes need genuine development effort,

Figure 5.1 VP-Centralen's organisation chart

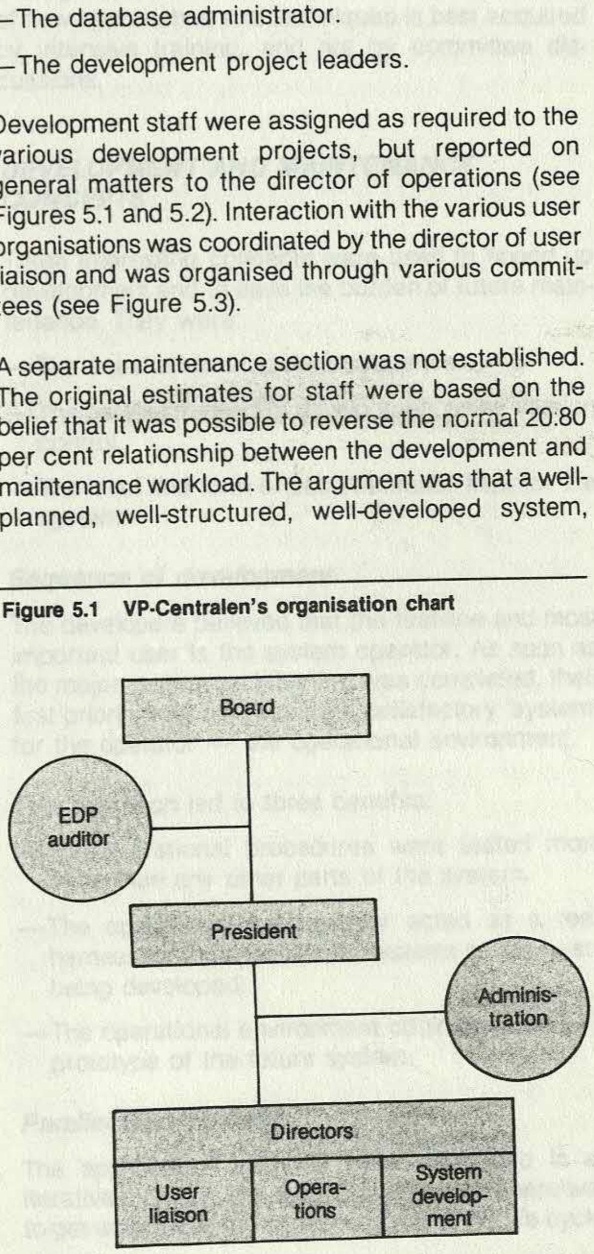
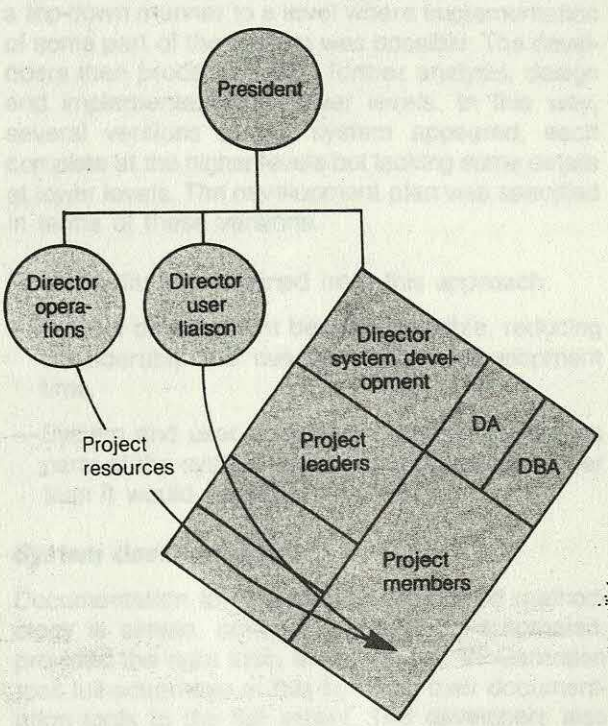
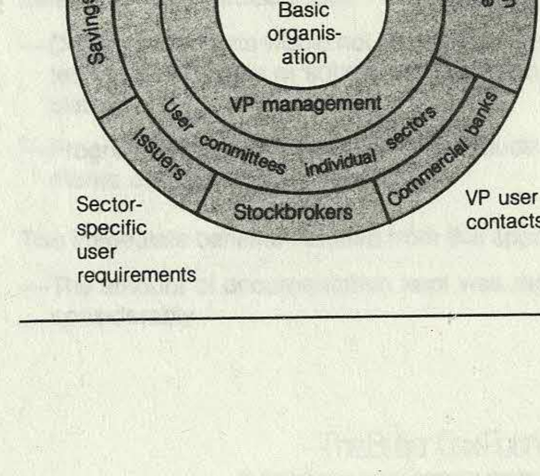


Figure 5.2 VP-Centralen's matrix organisation



DA = Data administrator
DBA = Database administrator

Figure 5.3 Organisation for interactions between the users and VP-Centralen



and these tasks should be handled by those who have an intimate knowledge of the system, not by a separate maintenance section.

For similar reasons, a separate quality assurance section or position was not set up. The planners believed that quality was built into the system by the deliberate efforts of the developers, and not by an external inspectorate. Instead of spending funds on quality assurance personnel, special methods and tools were acquired and training was provided to help the developers. An unusual feature of the training was that all levels of management were the first to be trained in the techniques that were to be employed. The rationale was that, without a common understanding of how the project should be carried out, management would not be in control. Understanding of new approaches and techniques is best acquired by intensive training, and not by committee discussions.

DEVELOPMENT AND MAINTENANCE CONCEPTS

Three interesting concepts were used to speed up development and to ease the burden of future maintenance. They were:

—The unusual development sequence.

—The parallel method of development using version control.

—The level and form of documentation kept for the system.

Sequence of development

The developers believed that the first-line and most important user is the system operator. As soon as the major system architecture was completed, their first priority was to establish a satisfactory 'system' for the operator — the operational environment.

This approach led to three benefits:

—The operational procedures were tested more often than any other parts of the system.

—The operational environment acted as a test-harness for the applications systems as they were being developed.

—The operational environment could be used as a prototype of the future system.

Parallel development

The applications systems were developed in an iterative manner. The main line of thought here was to get away from the linear concept of the life-cycle.

This was achieved by using two conventional concepts in new ways. First, the developers believed that analysis and design are techniques to be used throughout the whole of the life-cycle model, and not just in the analysis and design phases of the development process. Second, they did not insist on completing all the analysis before starting the design phase, and they did not require all of the design to be completed before starting the implementation phase. The analysis and design were carried out in a top-down manner to a level where implementation of some part of the system was possible. The developers then proceeded with further analysis, design and implementation at lower levels. In this way, several versions of the system appeared, each complete at the higher levels but lacking some details at lower levels. The development plan was specified in terms of these versions.

Two benefits were gained from this approach:

—Parallel development became possible, reducing considerably the overall elapsed development time.

—System and user acceptance testing on various parts of the system was carried out much earlier than it would normally have been.

System documentation

Documentation for the chosen structured methodology is simple, concise and can be automated, provided the right tools are available. VP-Centralen took full advantage of this by using their documentation tools to the full extent. The developers also believed that:

—The structured design phase should be concerned with merely re-packaging the results of the structured analysis phase (the specification) and therefore should not introduce any new ideas.

—Programs should be coded exactly as they were designed.

As a consequence, it is not necessary to document detailed design considerations, and it is not necessary to include comments in programs. Any such documents and comments are redundant because they duplicate information available elsewhere. VP-Centralen therefore decided that:

—Design documents would not be retained after the testing of a module or subsystem had been completed.

—Programs would not be allowed to include comments and explanatory notes.

Two immediate benefits resulted from this approach:

—The amount of documentation kept was reduced considerably.

—Because no other documents were available, any modifications to the system had to go through the analysis, design and coding phases.

In summary, the approach to development and maintenance used well-known concepts and techniques, but they were used in a new way. It is worth noting that on the only occasion when someone bypassed the strict procedures laid down for modifications, a solitary bug was introduced into one of the programs. The result was that on-line operations had to be suspended for ten hours.

THE DEVELOPMENT METHODS AND TOOLS USED

During the strategic preparations for VP-Centralen, the original team members debated the possibility of using structured development methods. Their main concerns were in two areas:

—Very few of the development team (who had yet to be recruited) were likely to be familiar with these methods.

—Without tools to support the method, the method chosen would not be efficient enough to meet the challenge posed by developing a high-quality system in the required time.

From the start, the team decided to select interactive tools that would support the chosen development method. Furthermore, the team believed that the management of the centre must be responsible for selecting the method and tools, and so it set about evaluating the products available on the market. Some specialists were used in this process, such as a consultant who evaluated available database systems, but the assignment was performed within a framework of firm objectives set by the management team.

The strategic planning team chose the Yourdon method, and adopted the life-cycle model suggested by Yourdon as the standard for the system. Every member of the centre's staff has subsequently been trained in the method. Yourdon consultants were asked to provide the initial training and then to return regularly to inspect VP-Centralen's progress.

The choice of this methodology led to four major benefits:

—The high-level analysis and design performed during the survey stage enabled the developers to prototype the system. This allowed them to try out various hardware and software environments before making their final choice.

—The dataflow diagrams, minispecs and structure charts, which are an integral part of the methodology, were all held in electronic form, thereby eliminating manual, paper documentation of the system.

—The elapsed time for development was greatly reduced by subdividing the system into small, controllable units. This was made possible by top-down functional decomposition and by using the life-cycle model recursively on smaller parts of the system.

—The small, well-documented, mainly unifunctional, highly cohesive, loosely coupled modules of the system were easy to modify.

In addition, the early analysis of data flows and data stores resulted in a completely normalised data structure, which helped greatly in the selection of the eventual database management system.

The analysis-of-requirements stage took ten months and provided the criteria for selecting the software and hardware tools required for the development and operations stages. The problem that faced the developers was the need to assemble a compatible toolkit right at the start. For the development stage this meant:

—A highly automated library for all system documentation.

—An on-line program development environment, including a text editing facility.

In addition, the future operational environment was envisaged as being totally automated, and required integration with the rest of the tools. This needed:

—An efficient database management system.

—An automated operational control and operations management system.

The eventual system needed a complex communications subsystem to interconnect with a variety of machines and communications networks. These were:

—CICS and IMS, using the Intersystem Communication Feature.

—CICS standards (with standard terminal control).

—3270 protocols.

—JES II standards for the rest of the eventual users.

All of these tools had to be supported by appropriate hardware to allow the terminal network to operate with SNA-type standards for communicating with the 240 user organisations. Readers with experience of implementing SNA-based communications systems will appreciate that the VP-Centralen plans were very ambitious.

The final choice for the database management system was between Intel's System 2000 and Software AG's Adabas. IBM's IMS was also considered, but was estimated to require between three and four times as much coding as the other systems. System 2000 was chosen together with iDIS, the System 2000 database information system that allows microcomputers to manipulate small sections of the database away from the mainframe. PL/1 was chosen as the development language because it was judged to be the best conventional language for a structured approach.

PL/1 was complemented by IBM's TSO/SPF for program development, ADR's Librarian for document filing and IBM's DCF for text editing. For controlling the automatic operational environment, Universal Computing's UCC-7/11 was chosen. MSP's Datamanager was used to provide the data dictionary facility and to integrate the tools.

Hardware was chosen by carrying out full-scale simulations of the critical operations of the future system, using the full set of the chosen software kit. Co-operation between Intel technicians and IBM's Munich laboratory made it possible to run the first prototype of the system on one of the first 4300 processors shipped by IBM.

Two hardware proposals were considered — from IBM and from National Advanced Systems (NAS). Initially, three machines were installed (two 4341s and an NAS 6050), and these were used for system development work and system implementation. Because of the sudden and unforeseen increase in the number of transactions, the configuration was later upgraded to an IBM 3083E and a plug-compatible NAS 8040. The communications processor selected was an IBM 4341. In addition, some Intel hardware was installed, including a 3805 solid-state 'disc' that speeds up the processing of on-line transactions.

All of the hardware has been installed in the basement of VP-Centralen's computer centre. Operations are controlled remotely through terminals installed in the control room, and the flow of transaction traffic is monitored by the various managers through their own terminals. Each desk at the centre has a workstation terminal that supports operations, analysis, design, programming and administrative activities. The people at VP-Centralen have, in effect, implemented the paperless office in their own environment.

THE QUALITY ASSURANCE METHOD USED

The key method for assuring quality became the review, and the walkthrough technique was used to ensure that a zero-defect system was implemented. All development activities were continually reviewed and monitored for quality. Work was reviewed at five different levels, as described below.

Development project reviews

The purpose of reviews at this level was to detect faults. During the analysis phase this meant checking the accuracy of the system models. At the design stage, the key characteristics of modularity were scrutinised and the design was checked to ensure no deviation from the original model. During the implementation stage, reviews concentrated on ensuring that details had been accurately transcribed and that testing and documentation were complete. In addition, no code was accepted for operations unless someone other than the author could read and understand it.

Data administration reviews

Through the data dictionary, the data administrator maintained an independent consistency check on all parts of the system. He attended project walkthroughs to maintain continuous liaison with the developers and to control the overall logical use of data.

Database administration reviews

The physical data stores were regularly reviewed for consistency with the specifications for using the data.

Project management reviews

Development projects were subjected to periodic reviews where the actual results were compared with the various plans.

Management reviews

The directors of system development and user liaison regularly reviewed the implementation of user requirements to check that the stated business policies were being implemented through the system. (An early discrepancy between the system and the user manual was discovered by a review at this level.)

Introducing the walkthrough technique

There were two prerequisites for introducing walkthroughs and reviews:

- The sole purpose of work-review meetings was to concentrate on the quality of what was produced, and not on the personality of the producer.
- Management needed to understand the approach fully. VP-Centralen believed that staff could not be expected to use the walkthrough technique if the managers did not understand the underlying psychological factors. The best way for management to acquire such understanding was by practical experience. Walkthroughs were therefore first used by the management team.

It took VP-Centralen's staff nine months to accept the walkthrough technique. The most common problems encountered during the initial period of using walkthroughs were:

— No involvement by the participants because they did not feel responsible for someone else's work.

— No formal evaluation of the quality of the material reviewed and no follow-up actions specified.

— Too much material presented for review at a single walkthrough.

— Lack of discipline during the walkthrough meetings.

Others who have been involved in the introduction of walkthroughs confirm that these are the most common early reactions to an ego-less development environment and walkthrough techniques.

SYSTEM SECURITY AND AUDIT

Denmark does not have a tradition of auditing data processing systems, but the financial sector needed to satisfy itself that the system and VP-Centralen's processes were sound. As well as conforming to stringent financial and security requirements, the system had to be right the first time. Danish law and the country's financial practices had to be changed for the new operations, and there was no possibility of returning to the old system after converting the paper bonds to an electronic medium.

Despite these stringent requirements, VP-Centralen did not set up a separate security function. The argument was that, by establishing such a function, the responsibility for developing a secure and high-quality system would be removed from the developers. Management believed that such a move would take away the incentive to carry out the right kind of development work. The test of time has proved that management made the right decision.

The strict financial control requirements were built into the system by specifying them as a separate set of requirements. In addition to building in these control requirements, the developers took account of the need to make the system available for inspections and audits of the centre's operations. Although the specific rules for auditing an on-line, interactive transaction-processing system had, in the main, to be invented as the project progressed, the facilities were planned for from the beginning. For example, iDIS the small relational database information system which co-exists with System 2000, was acquired for use by the auditors. With iDIS, auditors can load a selected subset of the system files onto a microcomputer for interrogation and manipulation. In addition to making extracted data easily available, separate functions for audit and control were also built into the system.

Audit and control requirements were developed by a special group formed jointly from the auditors of the user institutions. Both the developers and the auditors recall their work together as trouble-free and highly enjoyable. The system team developed a high regard for the auditors, because they were the only committee that stated problems and expected the system team to come up with solutions. The auditors believe that they are provided with far more from VP-Centralen's system than they are from the various systems in their own companies.

SUMMARY

This case history documents the successful development and implementation of a system in a highly demanding and challenging situation. We believe that the main reason for this success was the belief that high quality is an integral responsibility of management, and that VP-Centralen's management team took this responsibility very seriously. The management team deliberately built and maintained a high-productivity, high-performance work environment, based on the zero-defect principle.

In addition, their intimate knowledge of the development process helped them to select excellent tools and methods for supporting and controlling the work of the centre. As planners, they opted for simplicity, both for project and organisational design. The outcome has been to minimise the effort required at the centre: development has taken a minimum amount of time and the relatively simple organisation employs a minimum number of people. The organisational complexity is low because authorities and responsibilities are placed at the lowest possible level and are not divorced from actually carrying out the work.

Through its decisions and actions, the management team made quality a way of life at VP-Centralen. And this, we believe, is the key to its success.

CHAPTER 6

THE KEYS TO BUILDING BETTER-QUALITY SYSTEMS

The principal finding from our research is that the quality of systems is important, but insufficient attention is given to quality issues, mainly because systems quality is difficult to measure. A formal quality assurance department is not necessary for the production of quality systems, but top-level support, a good overall knowledge of the development process and good project management and development tools are essential. We also found that quality and development productivity are closely linked through the use of system development tools. As a consequence, it is not, or should not be, difficult to start producing better-quality systems. In this final chapter of the report we highlight the implications of our findings for Foundation members.

RECOGNISE THAT QUALITY IS IMPORTANT

Foundation members, software and hardware suppliers, government agencies and official bodies all told us that they were deeply concerned about the quality of their systems. Evidence of this concern emerged also in the results of a recent survey of top information systems professionals in 54 companies in a variety of United States industries. The survey was conducted jointly by the Society for Information Management and the MIS Research Center, School of Management, University of Minnesota. Improved system quality was ranked as the fourth most important issue, ahead of other issues such as data management, telecommunications, decision support, office automation, graphics and artificial intelligence. (According to the survey, the three most important issues were improved information systems planning; end-user computing; and integrating data processing, office automation and telecommunications.)

In other industries there is also great concern about the quality of goods and services. Quality management, quality circles, and zero-defect development and manufacturing represent ideas and practices that have been developed as aids to competitiveness, including differentiation through quality rather than price. We have found that companies in the IT supply industry are adopting these approaches and practices, and that commercial information systems departments are also adopting them as a result of company-wide quality programmes.

ACCEPT THAT QUALITY IS DIFFICULT TO MEASURE

The difficulty of measuring quality is not unique to the systems area. Other industries face the same problem. But, if there is no metric, it is hard to encourage quality production because it will be hard to reward the achievement of quality. This difficulty manifests itself in the systems area in a particular way. Systems personnel usually are employed because of their technical competence, and they are promoted because of technical and fiscal successes. The latter means that personal advancement is linked strongly with meeting deadlines and staying within budgets, and both of these are reasonably easy to measure. There is no equivalent reward or recognition for achieving quality in the process of system development.

Industry in general also recognises the importance of deadlines and budgets. But in strong competitive situations, where all other factors are nearly equal, the quality of goods and services becomes the issue that determines success or failure. The simple measure of customer satisfaction acts as the final proof of quality. Equally, the real measure of systems quality lies in the user being satisfied with the system. Of course it is not easy to monitor user satisfaction systematically, nor is it possible to evaluate it, like a budget. The key to satisfying user expectations is to pay greater attention to the users' case by concentrating more on analysis and implementation, by not deviating from specifications for technical convenience, and so forth. Such an approach assumes technical and fiscal competence as a minimum prerequisite.

A recent survey conducted by two Canadian universities set out to identify the criteria that determine a quality system. The survey found that recognising the success factors from the users' perspective, and subsequently evaluating the tangible benefits of the system based on these factors, were the most important criteria. The results of another survey of 32 business applications were presented at the 1984 conference of the Administrative Science Association of Canada. These indicated that projects where the analysis phase took longer required less time for implementation, and resulted in greater user satisfaction. Moreover, they met the specified budgets and deadlines.

We draw two very different but important conclusions from these findings. First, the difficulty of measuring and rewarding quality achievements means that attention to quality can be seriously established only if the expectation and encouragement come from the top of the organisation. The information systems director whose only concern is meeting deadlines and staying within budgets is as unlikely to achieve high quality and lasting success as the company executive whose main focus is on profit and balance-sheet figures.

Second, a desire for a quality focus in the systems area implies there are preferred strategies for system development. In turn, this implies that, to be successful, the strategies have to be managed and controlled.

REALISE THAT TOOLS ARE THE LINK BETWEEN QUALITY AND PRODUCTIVITY

For several years, the strategy adopted in systems departments has been largely determined by the rising cost of systems. Hence the productivity of system development staff has became an important issue. But the word "productivity" is misleading. It is not sufficient solely to carry out the same activities faster than before, or to motivate better the systems development staff. The problems are more likely to be caused by unresolved complexity in the project and in the system design.

Large projects, and systems designed with complexity rather than simplicity in mind, inevitably cause problems. A system that was wrongly conceived is likely to contain errors and will need higher levels of maintenance, and so its quality is unlikely to be high. Moreover, the re-work required to correct the faults reduces the productivity of development staff.

Reducing the complexity of systems will therefore increase both the productivity of development staff and the quality of systems, because parts of the work will not need to be repeated several times. The key to increasing both productivity and quality is straightforward — it is to get the system right first time. There is a simple set of procedures that will help to achieve this aim:

—Create small, manageable organisational units for development activities and provide these units with clear responsibilities for producing quality.

—Use a simple but well-defined and well-understood development method.

—Select a set of interactive development tools, preferring those that reduce or eliminate internal documentation.

—Ensure that the developers themselves carry out regular reviews of the work.

Selecting an appropriate method and the tools to support it should create an environment that encourages development staff to spend more time on the intellectual activities of analysis and design and less time on the repetitive activities of coding, documenting, searching for references, and so forth. In some circumstances it will be possible to eliminate altogether some of the latter stages.

We believe that producing high-quality output from systems work requires less innovation than most people believe. Moreover, the development process can actually be speeded up. Most organisations now use a well-defined procedure for developing their systems. They have also realised that small development teams are better than large ones. It is relatively easy to introduce regular walkthroughs and reviews in small groups — it is only the large, bureaucratic, hierarchical organisations that resist the introduction of such frank and open practices.

The tools selected to support the chosen method are a key element of producing high-quality systems. The really useful tools are those that help with the routine tasks of system development. These tools do not apply only to individual phases of development. Instead, they help to reduce the burden caused by the large number of system parts that need to be recorded and interlinked. They also help to produce the text and diagrams that any large project generates. These tasks are common to all phases of the development process, irrespective of the life-cycle model adopted.

The case history in Chapter 5 showed how the sensible packaging and specific use of conventional tools can help to produce quality systems. We conclude that the tools required to build quality systems already exist. What is lacking in many organisations is the will and the intent to use these tools to the best purpose.

We believe, therefore, that the prerequisites for producing quality systems are already present in most organisations, and that they are more-or-less the same as the tools for increasing development productivity. We must emphasise, however, that this approach to building quality systems applies only to new systems. Existing systems will continue to experience the problems inherited from their original development approach, and will continue to do so until they are redeveloped.

ESTABLISH A QUALITY ASSURANCE FUNCTION

Our research showed that establishing a formal quality control department does not necessarily ensure the production of high-quality systems. Conversely, we found that high-quality systems can be produced where there is no formal quality control section. The keys to success are the attention given to the development process and its control, and the existence of an environment in which quality production is possible.

These aims can be achieved by:

—Giving responsibility for producing quality to those who produce systems, and providing them with appropriate incentives.

—Building a balanced and helpful working environment by providing the relevant tools for the job (and the training to use them), encouraging open reviews of system problems through regular walkthroughs, and ensuring that development staff have a thorough understanding of the system development process.

Such an environment can best be created and maintained by establishing a quality assurance function (as opposed to a quality control department). The precise form of the quality assurance function will depend on the type of organisation and its systems. Financially oriented organisations such as banks and insurance companies will probably need to establish a separate department whose role is to inspect and police the quality aspects of systems development.

Other organisations will choose to entrust the quality function to one person whose role will be to ensure that the quality culture of the organisation is reinforced and promoted within the information systems department. This approach requires that every line manager in the information systems department has very clearly defined quality objectives.

Those organisations that are just beginning to focus on systems quality issues may first need to set up a separate department whose role is to establish standards and procedures for development and quality assurance. Once the necessary discipline has been introduced, and training provided, the emphasis may well evolve to quality issues being primarily the concern of line managers.

Whatever form it takes, the quality assurance function has three roles:

—To establish a focus for quality issues by developing quality objectives.

—To provide the prerequisites for quality work: standards; codes of practice; selection of relevant tools and methods; provision of training, checklists for walkthroughs, statistics on usual errors and guidelines on how to avoid them.

—To assist in the production of quality by helping those in line positions to adopt practices that conform to the quality objectives; by giving practical help to developers in understanding the development process and in using standards, codes of practice and tools; and by taking part in reviewing the work produced.

The actual work of any quality assurance function is likely to focus more on one or another of these roles at different times, and organisations will need to decide for themselves the approach that best suits their current stage of development.

CONCLUSION

The main message of this report is deceptively simple. Achieving quality in the systems area is not difficult. The necessary tools already exist — but to make effective use of them requires a change of attitude concerning the way in which systems are developed. Given that change in attitude, there is no reason why better (that is, quality) systems cannot be produced.

A prerequisite for success, however, is that the organisation as a whole must be committed to the pursuit of quality in everything it does. That commitment must originate from the top of the organisation, and it must be an integral part of the corporate culture.

Nevertheless, the lessons learnt from this report can be applied only to new developments. The poor quality inherited from past systems will remain until they are eventually scrapped, and then replaced using the methods and tools we have outlined.

APPENDIX 1

DEFINITIONS OF SOFTWARE DESIGN CRITERIA

Traceability: The attribute that provides a thread from the requirements to the implementation with respect to the specific development and operational environment.

Completeness: The attribute that provides full implementation of the functions required.

Consistency: The attribute that provides uniform design and implementation.

Accuracy: The attribute that provides the required precision in procedures, calculations and outputs.

Error tolerance: The attribute that provides continuity of operation under non-nominal conditions.

Simplicity: The attribute that provides implementation of functions in the most understandable manner. (Usually avoidance of practices that increase complexity — i.e. multi-functionality, complex interfaces, etc.)

Modularity: The attribute that provides a structure of highly independent modules.

Self-descriptiveness: The attribute that provides explanation of the implementation or operation of a function.

Operability: The attribute that determines the ease of operations.

Training: The attribute that provides transition from current operation or initial familiarisation.

Communicativeness: The attribute that provides useful inputs and outputs that can be assimilated.

Communications commonality: The attribute that helps the use of standard protocols and interface routines.

Data commonality: The attribute that aids the use of standard data representations.

Conciseness: The attribute that provides for clarity with a minimum amount of information.

APPENDIX 2

DEF STAN 00-16: GUIDE TO THE ACHIEVEMENT OF QUALITY IN SOFTWARE

Because the United Kingdom Ministry of Defence is a large user of bought-in software, it published a standard in February 1984 that relates exclusively to systems development (Def Stan 00-16 — Guide to the Achievement of Quality in Software). This guide brought together the work that had been done on quality by the Ministry and several other national groups (including the British Standards Institute), to show how quality can be obtained during systems development.

Specifically, it provides guidance on the methods and procedures necessary to establish confidence in the quality of software for computer systems used in operational defence applications. Defence applications of this type have a very great need for complete accuracy — a requirement that does not necessarily apply to other application development projects. Nevertheless, the basic principles of the guide are of general application, although great emphasis is placed on the 'contract' between the purchaser and the developer of the project. This emphasis implies a level of requirements specification and planning not usually undertaken in the majority of in-house system developments.

QUALITY CONCEPTS ADOPTED

The guide follows closely the common approach used for other quality-related national standards. The concepts of quality and quality assurance have a common definition in the United Kingdom national standards. Quality is defined as the "totality of features and characteristics of a product or service that bear on its ability to satisfy a need". Thus, the all-embracing idea of "fitness for purpose" is being used, rather than other concepts such as "degree of excellence", where products are ranked on a relative basis, or "quality level", which measures product quality quantitatively. The implication is that the needs of all phases of a project (development, design and implementation) must be met. Also, the initial specification should be sufficient to ensure that the end-product will be 'fit'.

"Quality assurance" is defined as "all activities and functions concerned with the attainment of quality".

Thus, the scope of quality assurance is extensive, affecting all aspects of the life cycle of systems development projects (including the post-implementation phase), and also incorporating the determination and assessment of quality.

The standards also universally subscribe to the "way-of-life" concept of quality assurance (described in Chapter 2 of this report). In essence, this means that if "quality assurance" is used as the title of a department, this department's authority and activities should not supplant or reduce the responsibilities of other departments to contribute to quality, and that quality is an inherent part of the whole process.

The various standards also suggest a common approach to the task of quality-assuring systems. This approach involves four aspects.

Specifying a standard. At the start of a project, a detailed specification of requirements should be generated and agreed by all participants, particularly the end user. This can then be used as a 'standard'.

Planning the project. Detailed plans showing how the project will be managed and how quality will be assured should be developed.

Developing codes of practice. The codes of practice describe generally the activities and functions that need to be performed in the development process and how they should be performed.

Specifying quality assurance procedures. These procedures control and monitor the way in which activities are performed.

These four aspects interact to form an iterative process (depicted in Figure A.1), which ensures both the development of quality systems and the establishment and maintenance of quality standards. During the planning process, the requirements specification will be used to help determine which activities and functions from the code of practice should be used. Any issues or problems raised while developing systems can be fed back into the codes of practice to ensure the continuing attainment of quality.

We now discuss Def Stan 00-16 in terms of the four aspects of the common approach.

SPECIFYING REQUIREMENTS

Because the Ministry of Defence typically purchases software from external suppliers, it perceives the attainment of quality though requirements specification differently from most commercial organisations. It recommends the following procedure.

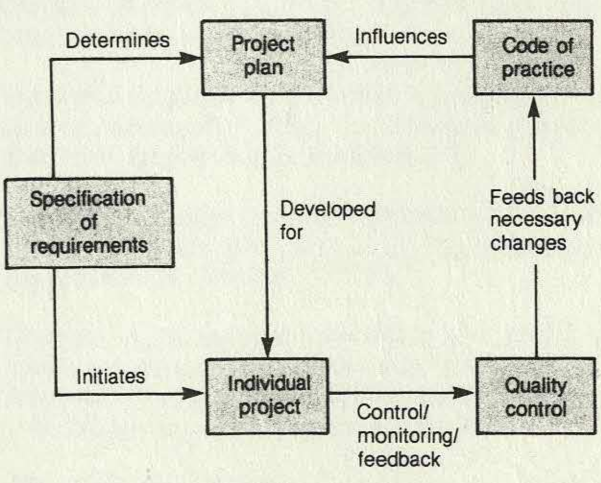
- —The user (or customer) prepares a detailed statement of requirements that specifies, for example, the function of the system, the performance required, any environmental or design constraints, etc.
- —The supplier responds point-by-point to this statement of requirements, detailing both how it would meet the requirements and how it would plan and manage the project and assure project quality.
- —If the supplier's response is accepted, a mutually agreed 'procurement specification' is produced specifying how the software will be supplied and what the criteria for acceptance will be.

Obviously, this procedure offers considerable value to a company buying in development effort, but it would also be useful, in a modified form, when developing in-house systems. It emphasises the agreement between developer and user on what exactly is to be achieved, by when and how.

PLANNING

Two types of plans are recommended by the Ministry of Defence:

Figure A.1 British standards — the quality assurance approach



- —A software life-cycle management plan which defines the total system and identifies major events (or milestones) in the project's life.
- —A software quality plan which specifies quality targets or milestones, such as design reviews or audits.

The two plans should be separate and independent, but both should be developed with the co-operation of users, project management staff and quality management staff. The concept of a quality plan is particularly useful to commercial organisations.

CODES OF PRACTICE AND QUALITY ASSURANCE PROCEDURES

Codes of practice and quality assurance procedures should be used in the software development process. Both relate to every stage of software development, from the initiation of a project to the installation of a system and its use by the end user.

Codes of practice

Codes of practice are the documented standards that explain what procedures, methods, methodologies, etc. have to be applied at every stage in the development of a system. There are five aspects of codes of practice implicit in Def Stan 00-16:

- —The codes of practice are a documented set of 'best practices' that are expected to be achieved in each project.
- —Their development should be undertaken separately from project development, with input from external sources as well as from the best in-house practice.
- —They establish the basic 'rules and regulations' for system development, but they should be used flexibly in different projects. The same emphasis is not necessary for a short project as would be required for very large projects.
- —Feedback from their use in projects should be incorporated into the codes of practice so that any difficulties in using them are removed, and any insights achieved are built in.
- —Codes of practice should evolve continuously as improved methods and methodologies are incorporated.

The clearest example of a code of practice is the one suggested by the Ministry of Defence for the programming activity. It suggests that a programmers' manual should be produced in which programming techniques and methods are defined and documented. The manual should describe both approved and prohibited programming practices, explaining the reasons behind these decisions. A section on programming languages should include, or refer to, a full definition of both the language and the compiler and their use. Other sections should include information about programming conventions, debugging aids, etc.

Quality assurance procedures

Quality assurance procedures are required at various stages of a project, depending on its size and importance. The procedures consist of a mix of direct action and involvement, and the auditing of project activities. Auditing should be based on the codes of practice. Again, the clearest example of quality assurance procedures lies in the programming area, where the Ministry of Defence recommends that:

- —The quality assurance function should ensure programming techniques and methods are clearly defined.
- —During the project, checks should be made to ensure programming standards are being complied with.
- —An acceptable level of standardisation should be achieved across a project.

Often these quality assurance procedures are referred to as "quality control".

Bonissone, P.P. QUAISY/DAISY: An on-line qualitative assurance information system/diagnosis and system. Proceedings of the International Conference on Cybernetics and Society, Atlanta, Georgia, 26-28 October 1981, p324-330. New York, IEEE, 1981.

British Institute of Management. What's New in Quality Management. Transcript of Senior Management Conference, 3 June 1980.

British Standards Institution. Quality Assurance. BSI Handbook Number 22. Milton Keynes: British Standards Institution, 1983.

Collard, R. How to go one better than Japan. Management Today, January 1982.

Crosby, P.B. Quality is Free. New York: The New American Library Inc., 1980.

Deming, W.E. Quality, Productivity and Competitive Position. Cambridge, Mass: MIT Center for Advanced Engineering Study, 1982.

Dickson, G.W. et al. Key information systems issues for the 1980s. MIS Quarterly, September 1984.

Dunn, R. and Ullman, R. Quality Assurance for Computer Software. New York: McGraw-Hill, 1982.

Electronic Engineering Association. The Guide to the Quality Assurance of Software. London: Electronic Engineering Assocation, 1978.

Electronic Engineering Association. Establishing a Quality Assurance Function for Software. London: Electronic Engineering Association, 1981.

Fagan, M.E. Design and code inspections to reduce errors in program development. IBM Systems Journal, Volume 15, Number 3, 1976.

Forrester, M.H. Automatic test equipment software quality assurance. Proceedings of the Conference on Automatic Testing and Test Instrumentation, Brighton, 7-10 December 1981, Volume 4 p179-188.

Gilb, T. Software Metrics. Englewood, New Jersey: Winthorp Publishing Company, 1976.

Gilb, T. Computerware Technoscopes. Course Notes, 1981.

Hansen, H. D. Up and Running. New York: Yourdon Press, 1984.

Ince, D. A case of bugs, mutants and software tests. Computing, 5 July 1984.

Jones, T.C. Measuring programming quality and productivity. IBM Systems Journal, Volume 17, Number 1, 1978.

Langevin, R.G. Quality Control in Service Industries. New York: AMACOM, 1977.

Lockyer, K. and Oakland, J. The quest for quality. Management Today, July 1981.

Mair, W.C. et al. Computer Control and Audit. Wellesley, Mass: QED Information Sciences Inc., 1978.

McCall, J.A. et al. Software Quality Measurement Manual. Springfield, VA: US Department of Commerce, National Technical Information Service, 1980.

McKeen, J.D. Successful development strategies for business application systems. MIS Quarterly, September 1983.

Meek, B. Failures can bring pleasure. Computer Weekly, Number 906, 12 April 1984.

Ministry of Defence. A Guide to the Achievement of Quality in Software. Defence standard 00-16/issue 1. Glasgow: Ministry of Defence, February 1984.

Morland, J. Quality Circles. London: The Industrial Society, 1984.

Myers, G.J. Reliable Software through Composite Design. New York: Petrocelli, Charter, 1975.

Myers, G.J. Composite/Structured Design. New York: Van Nostrand Reinhold, 1978.

O'Neill, H. The quest for quality. Management Today, July 1981.

Perry, W.E. Effective Methods of EDP Quality Assurance. Wellesley, Mass: QED Information Sciences Inc., 1981.

Pyper, W.R. Historic files for software quality assurance. Proceedings of the Computer Performance Evaluation Users Group (CPEUG) 17th Meeting (N135-SP-500-83), San Antonio, Texas, 16-19 November 1981, p101-106.

Rivard, S. and Huff, S.L. User developed applications: Evaluation of success from the DP department perspective. MIS Quarterly, March 1984.

Ruby, R.J. Software Quality Assurance. Seminar notes, State of the Art Conference, London, June 1984.

Shaw, J.C. The Quality/Productivity Connection in Service Sector Management. London: Van Nostrand Reinhold, 1978.

Streeter, D.N. Productivity of computer-dependent workers. IBM Systems Journal, Volume 14, Number 3, 1975.

Weinberg, G.M. The Psychology of Computer Programming. New York: Van Nostrand Reinhold, 1971.

Weinberg, G.M. and Friedman, D.P. Ethnotechnical Review Handbook. Ethnotec Inc., 1979.

Yourdon, E. Structured Walkthroughs. New York: Yourdon Press, 1978.

Yourdon, E. and Constantine, L.L. Structured Design. New York: Yourdon Press, 1978.