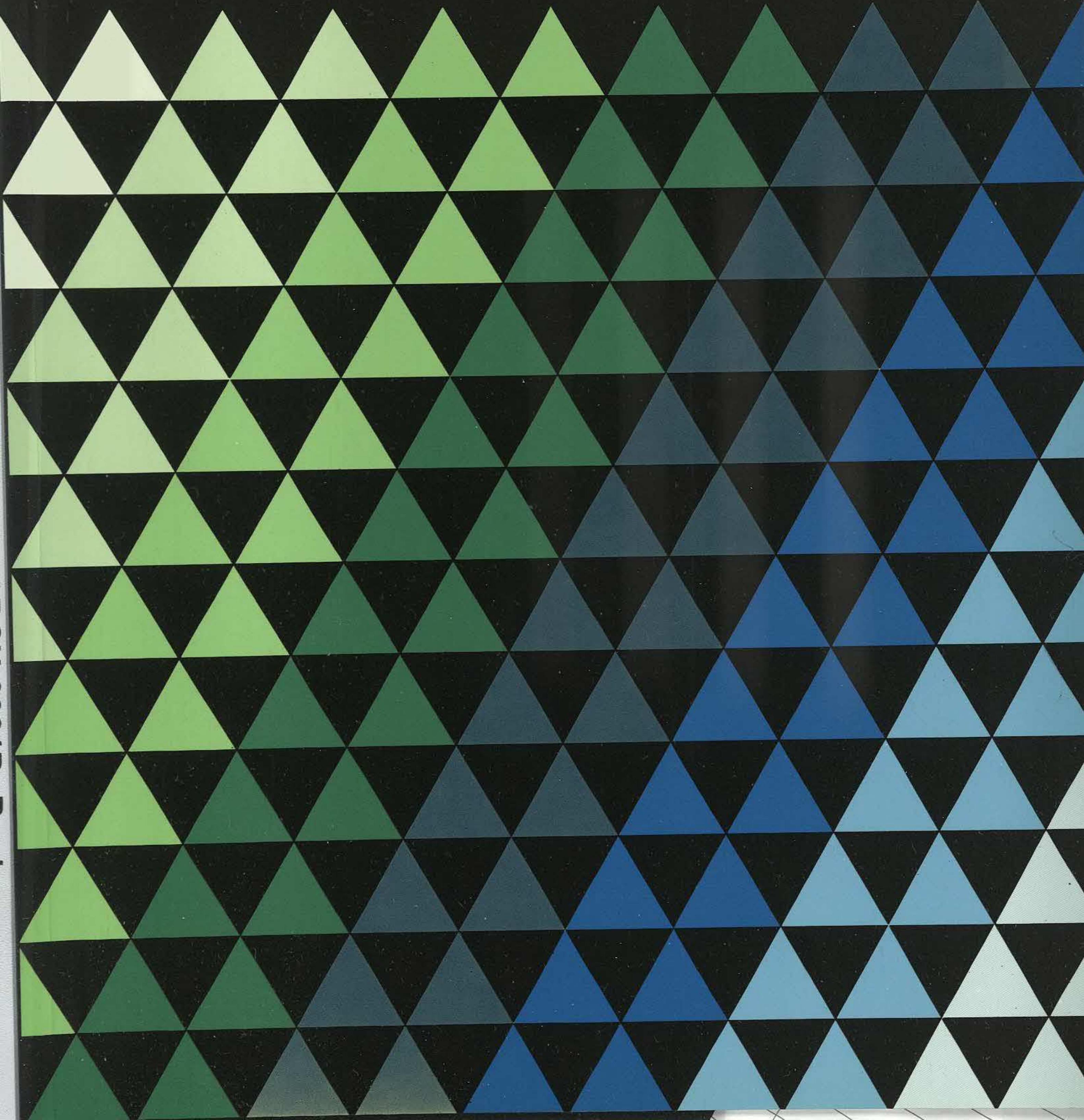
Software Strategy

BUTLER COX FOUNDATION

Research Report 69, May 1989



BUTLER COX
FOUNDATION

Software Strategy

Research Report 69, May 1989

Butler Cox & Partners Limited

LONDON
AMSTERDAM MUNICH NEW YORK PARIS

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

Copyright © Butler Cox & Partners Limited 1989

All rights reserved. No part of this publication may be reproduced by any method without the prior consent of Butler Cox.

Availability of reports
Members of the Butler Cox Foundation receive three copies of each report upon publication; additional copies and copies of earlier reports may be purchased by members from Butler Cox.

Photoset and printed in Great Britain by Flexiprint Ltd., Lancing, Sussex.

BUTLER COX FOUNDATION

Software Strategy

Research Report 69, May 1989

Contents

A Management Summary of this report has been published separately and distributed to all Foundation members. Additional copies of the Management Summary are available from Butler Cox.

BUTLER COX FOUNDATION
© Butler Cox & Partners Limited 1989

Report synopsis

Investing in new software and managing existing software are important management concerns. This report highlights the benefits of a software strategy and identifies the three main elements — the software infrastructure, software standards, and the software-procurement policy. An appropriate software infrastructure reduces the need for bespoke development because it recognises that the boundary between system and application software is changing. Many of the traditional applications functions are now available within 'infrastructure' software, which can be used by the user community to construct more of its own applications. In time, the main applications-software role of the systems department will be to manage the software infrastructure.

Chapter 1

The need for a strategy

Originally, the word 'strategy' was used to describe the management and deployment of armies and other resources so as to win a war. In business today, however, the word has a variety of meanings, ranging from an approach to solving a problem, to a detailed plan for achieving certain objectives. In this report on software strategy, we use the word 'strategy' in a way that is closer to its original meaning. By 'software strategy', we mean the use and management of software by an organisation to achieve its desired goals in a changing and competitive business environment.

The purpose of this report is to describe the elements of a software strategy, the benefits that can be gained by defining and implementing the strategy, and the factors that have to be taken into account as decisions are made about the strategy. Formulating a software strategy includes making decisions about which software functions are needed by the organisation to help achieve its goals, how the organisation should migrate from its existing software base, what standards are needed, and how the functions should be procured. It does not, however, include detailed plans for installing new items of system software nor for developing new applications software.

We have not included application-development priorities in our definition of a software strategy because we believe that, in the long term, application-development skills will be spread throughout the organisation. Decisions about which applications to develop or enhance next will therefore be taken by individual business managers, not by the systems department. Ultimately, most applications will be developed (or, more likely, constructed) by the business departments themselves. The main role of the systems department will be to develop and maintain the 'software infrastructure' (this term is explained later in the report) that allows these developments to occur.

Nevertheless, an agreed set of application priorities will be a necessary input to developing a software strategy. The priorities should be set as a result of a strategic systems planning process. An approach to strategic systems planning was described in Foundation Report 49, *Developing and Implementing a Systems Strategy*, which was published in October 1985. In that report, we stressed the importance of relating systems strategy closely to business strategy, and described several tools that could be used to do this. (We know that, in the intervening period, several Foundation members have used the tools with considerable success.) In particular, competitive-impact analysis and critical success factors can help to determine which applications will provide the greatest business benefits. The decision-conferencing technique, which was also described in Report 49, can be used to arrive at a consensus about application priorities.

The importance of software strategy

Other Foundation Reports have covered specific software issues, but this is the first time we have researched software strategy in its entirety. (The research team, and the scope of the research, are described in Figure 1.1, overleaf.) Without doubt, Foundation members regard software strategy as an important topic. The response to the annual poll of members showed clearly that this topic was by far the most important of those suggested for 1989. We believe that there are three main reasons for this.

First, business managers are demanding a much faster response to their needs either for new

© Butler Cox & Partners Limited 1989

Figure 1.1 Research team and scope of the research

This report was written under the direction of Andrew Milner, Director of Butler Cox's Productivity Enhancement Programme. The conclusions and recommendations of the report are derived from an international research programme carried out during the second half of 1988. The research was led by Jenny Clayton and Michael Lloyd, consultants based in Butler Cox's London office and specialising in systems strategy. They were assisted by Onno Schroder (Amsterdam), Lothar Schmidt (Munich), Michel Lederman (Paris), and John Cooper (Sydney).

In order to understand recent developments in the software industry fully, we conducted an extensive review of the published literature. We also conducted detailed interviews with software suppliers and with standards-setting organisations, including X/Open and the Open Software Foundation (OSF). The views of 170 Foundation members were gathered through the questionnaire sent out at the beginning of the study. In addition, the experiences of members were sought in telephone and face-to-face interviews in Australia, France, Italy, Scandinavia, the Netherlands, and the United Kingdom. Three focus groups were held in Germany, the Netherlands, and the United Kingdom, attended by a total of 18 organisations. In total, research was carried out in eight countries.

software applications or for enhancements to existing applications. In addition to providing administrative support, computer applications are now essential for the day-to-day operation of most organisations, and have a direct effect on their business goals. They can also be the key to achieving a competitive advantage. The growing importance of applications software means that changes in business strategy now have a far greater impact on the software needed to support the business. As a result, systems departments are expected to be able to react much more quickly to new requirements. A two-year lead time for a new application, for example, is now regarded as unacceptable by most business managers.

Second, expenditure on software (especially on commercially available software products) continues to account for a large part of the overall IT budget. Typically, the market for software products and services is growing at more than 20 per cent, whereas expenditure on in-house IT staff (a large proportion of whom are involved in developing, installing, and supporting software) is growing at less than 10 per cent. We believe that the different growth rates are due partly to the fact that organisations are developing their own software infrastructures, a concept that is discussed in more detail in Chapter 2.

The third reason for the growing importance of software strategy is that interworking between software applications is seen as a major problem that prevents organisations from making best use of their software investments.

The benefits of a software strategy

A software strategy provides benefits in many areas, the most important of which are:

— *Business demands will be responded to faster* because a strategy helps to minimise the variety of software being used, which means that the systems department can concentrate its software skills onto a smaller number of areas. In turn, this means that development staff can work on a greater variety of business applications. Hence, projects are less likely to be delayed because staff with the required skills are working on other projects.

— *Expenditure on software will be reduced* by minimising the need for different software products that perform basically the same function, by minimising the costs of replacing (or renewing) applications, and by allowing bulk discounts to be obtained wherever possible. One Foundation member told us that his organisation had saved \$5 million by standardising the software-infrastructure products used throughout all its offices worldwide. Buying the software licences centrally enabled substantial discounts to be obtained.

— *Training costs may also be reduced*, both for systems staff and for the business users of the software. Training costs usually represent a greater investment than the cost of the software itself. In particular, a lack of standardisation can result in very high training costs when staff are moved from one development environment to another. Training costs in the user community can also be minimised if all applications use the same user-interface conventions. In this way, staff moving from one department to another will not have to undergo extensive retraining before they can use the applications in their new department.

© Butler Cox & Partners Limited 1989

– *Interworking between applications is facilitated* by a software strategy because the strategy will ensure that the software products selected can be interlinked. Often, the full extent of the need for an application to interwork with other applications is not apparent when the application is designed. Using software products that conform with the strategy ensures that, when future requirements for interlinking arise, an application will not require major modifications and that the need to create a bespoke interface between two applications is minimised. One organisation we spoke to during the research told us that the lack of a software strategy in the past meant that it was now faced with the problem of supporting three mainframe environments. Integrating the applications will be impossible until the software infrastructure has been rationalised. Interfaces to external systems may also present a problem. (The problems of managing multiple hardware environments will be addressed in Foundation Report 72, *Managing Multivendor Systems*.)

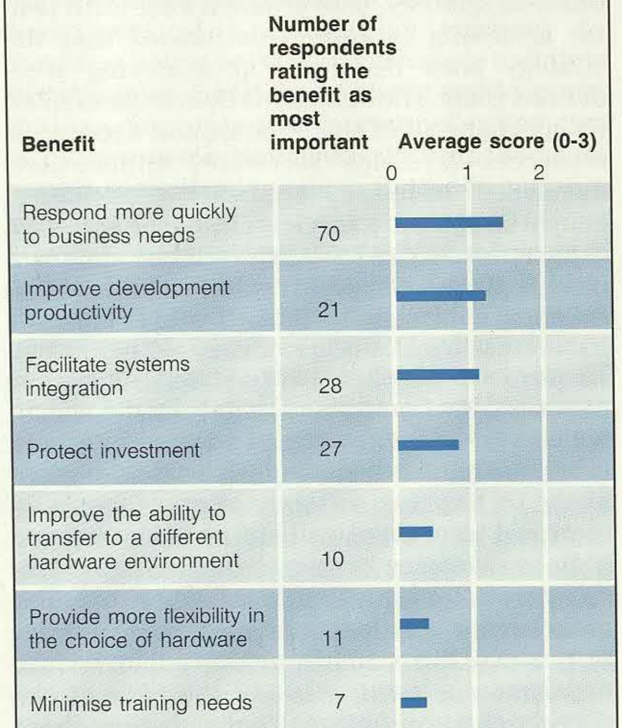
– *A more flexible choice of hardware will be possible* with a defined software strategy. Few large organisations have a single-vendor policy for computer hardware. Even those that use one vendor's equipment for their mainstream computing find that they need other suppliers for scientific, technical, or manufacturing applications. Others, such as Aachener und Münchener (a German insurance company), have a business policy to reduce their reliance on a single hardware vendor. To run the same software on different suppliers' hardware (or even on different-sized hardware from the same supplier) requires the systems department to think ahead and choose the right applications packages or system-software products.

– *Staff commitment will improve* if a clearly defined software strategy is in place. If systems staff believe that no attempt is being made to improve the development environment or use up-to-date techniques, they may feel that their career objectives would be better served in a more forward-looking organisation.

It may be argued that in a rapidly changing business and technical environment, it is better *not* to have a software strategy. Inevitably, a software strategy will be based on predictions about future business and technical developments, some of which will turn out to be incorrect. This means that some of the 'strategic' investments will be a waste of money. Perhaps it would be less expensive in the long term just to respond to specific software needs as they arise?

We do not believe this to be the case. Indeed, the need to respond rapidly to new business requirements is the main reason for developing a software strategy. This view was supported by the results of our survey of Foundation members carried out at the beginning of the research (see Figure 1.2). Without a software strategy that includes a flexible software infrastructure, there will inevitably be delays

Figure 1.2 The ability to respond more quickly to business needs is seen as the most important reason for having a software strategy



Benefit	Number of respondents rating the benefit as most important
Respond more quickly to business needs	70
Improve development productivity	21
Facilitate systems integration	28
Protect investment	27
Improve the ability to transfer to a different hardware environment	10
Provide more flexibility in the choice of hardware	11
Minimise training needs	7

Respondents were asked to rate the importance of each benefit on a scale of 0 to 3

(Source: Survey of 170 Foundation members)

© Butler Cox & Partners Limited 1989

in implementing applications to meet new business requirements.

The consequences of an inappropriate software infrastructure are illustrated by the experience of an insurance company that was unable to obtain new business because it did not have the appropriate videotex software in place. This company wanted to obtain a place on the 'panel' of insurance companies with whom a particular building society (building societies are UK savings and home-loans institutions) placed its business. On approaching the building society, the insurance company was told that it could be put on the panel within two months, provided that it could send and receive data via a videotex-based value-added network service. Unfortunately, the lead time required to connect to this service was six months, and the insurance company believes that it lost a significant amount of business in the interim.

The need for business, rather than technical, goals

The aim of a software strategy, as for any other business strategy, is to create a long-term plan for achieving success. This implies that the strategy must be aimed at achieving well-defined goals. The strategy is then implemented by managing all of the organisation's software resources (systems software, applications developed in-house, application packages, bought-in software services, in-house software support, and so on) to achieve those aims in a changing and competitive business environment.

The terms in which software-strategy goals are expressed are determined largely by the way in which a systems department perceives its basic mission. An inward-looking department is more likely to express software-strategy goals in technical terms, and will aim to achieve those goals by managing its own resources (hardware, systems staff, existing software, and so on). The main barrier that has to be overcome in order to achieve the goals is perceived to be the department's users, who, in the eyes of the department, are continually complaining about the service they receive and are often trying to find ways of breaking the department's monopoly-supply position. Hardware vendors are seen as another significant barrier, because of their efforts to lock their customers into their own range of products. Software vendors are perceived as yet another barrier; their products never seem to be able to interface to other systems that have already been developed. Finally, top management is seen by the systems department as a barrier to achieving the technically based software-strategy goals, because senior managers are perceived as not providing the commitment to the systems department to allow it to do the professional job that it would like to do. It is no surprise, therefore, that if software-strategy goals are expressed in technical terms, the link between business strategy and software strategy is at best tenuous, if it exists at all.

It was clear from our research, however, that many systems departments defined their software goals purely in technical terms. Typical goals of this type are to convert all major software applications from IMS to DB2, to use fourth-generation languages for all new development work, and to phase out the use of VM. While these types of goal may well represent sensible technical decisions, there are serious dangers in using them as a basis for a software strategy.

The problems that can result from basing a software strategy purely on technical goals are illustrated by the experience of an insurance company, which is summarised in Figure 1.3. By failing to develop a close relationship with its users, the systems department of this organisation was unaware of the business goals, and thus directed all its attention to achieving its own technical goal of providing a sound software base. Although this may well have been needed, the user community could not relate the large expenditure on software to any direct benefits in terms of its own business strategy. The result was that the systems department's budget was reduced substantially, and its only possible goal was then to concentrate on surviving.

Another example is provided by the UK division of a European bank. Before the crash in the world's financial markets in 1987, the systems department had embarked on ambitious plans for developing completely new systems to support the bank's trading operations. A very large, complex, and expensive software infrastructure was being developed, and attention

© Butler Cox & Partners Limited 1989

Figure 1.3 **Defining software strategy in terms of technical goals, rather than business goals, can lead to serious problems for the systems department**

An insurance company

This insurance company (which wishes to remain anonymous) employs about 7,000 people, 350 of whom work in the information systems department. All of the computing resources are supplied by IBM, and the software strategy is technology-driven, being determined largely by IBM's current products and future product developments. Most development work is done in-house, using IBM products for the software infrastructure. The systems department is not represented at main-board level, and communication between the department and top management is poor. There is no mechanism for systems staff to feed ideas upwards and, although there is some downward communication of business strategy, many of the most important business decisions are not relayed to the systems department.

As a consequence, the systems department is not well regarded by the user community, which feels that its needs are not understood and not being properly met. Not surprisingly, user departments are beginning to acquire their own computing resources, and are increasingly reluctant to pay for the systems department's services. As a result, the department's budget has been severely cut, and its staff has been reduced by about 25 per cent. These reductions mean that the systems department is now able to operate only in 'firefighting' mode, and its ability to support users properly is declining still further. The department is caught in a vicious circle from which it can see no means of escape.

was focused mainly on acquiring and using state-of-the-art development tools, rather than on ensuring that the new systems provided business benefits. After the crash, the large development costs being incurred every month became the focus of senior-management attention in the drive to cut costs. In the ensuing cutbacks, the systems department was forced to abandon its technical objectives and to concentrate instead on delivering real business benefits as early as possible. An important point, which we discuss in more detail in Chapter 4, is that expenditure on software should be in line with the *business* benefits that will be achieved. For this reason, many financial-services organisations have had to rethink their software strategies as a consequence of the low trading volumes after the crash.

The situation is very different if the software-strategy goals are expressed in business terms. In this case, the systems department is outward-looking, perceiving its main mission to be that of helping the organisation achieve its business goals. In turn, the organisation perceives IT as just one of the business resources that needs to be managed in order to achieve those goals. The barriers to achieving the goals are determined by the business and commercial environment in which the organisation operates — its competitors, the political and economic environment, legal and regulatory constraints, and so on. Thus, IT is perceived as making a real contribution to achieving business goals, and the software strategy is an integral part of the business strategy. Users are no longer perceived as a barrier to achieving the software-strategy goals, but as one of the resources to be used in achieving them. In this way, the increasing ability of users to take responsibility for providing their own computing needs is explicitly recognised. An added advantage is that business managers will no longer think of the systems function as an expensive overhead.

In general, software-strategy goals should therefore be expressed in terms of helping other business functions achieve the organisation's *business* goals. For example, online-ordering and stock-availability systems can help the sales department of a distribution company to provide a better service to customers and thus increase sales. Similarly, in a large multinational group, a corporate-wide personnel system can help the organisation make the best use of its human resources.

In some organisations, the use of software is essential to achieving the business goals. A good example is Lloyd's of London, which provides services to the members of the Lloyd's insurance market in London. The systems department realised that, along with other financial markets, Lloyd's could not survive in its traditional form. Huge benefits could be gained by creating an electronic market, and if Lloyd's did not move in this direction, its members would do so themselves, in a less controlled way. By understanding the business environment, the systems department was able to convince top management at Lloyd's that investment was needed for a network and networked applications to serve the market. The Lloyd's Market Network is now in operation and the corporation's current business strategy is dominated by the opportunities to be gained from this network and the applications

© Butler Cox & Partners Limited 1989

supported on it. It is clear that, in future, software will be a critical business resource for Lloyd's.

Another example, also from the financial-services industry, illustrates how the right software strategy can help achieve business goals. The Woolwich Building Society, one of the United Kingdom's largest savings and home-loans institutions, decided to enter the insurance market. When this was made known to the systems department, it investigated the use of IT in the insurance industry and found that extensive use was made of videotex. As a result, the systems department decided that videotex software should be incorporated into the society's software infrastructure. This software was available for use 12 months before the society launched its insurance services, and is now used for several applications.

The elements of a software strategy

A complete software strategy has three main elements:

— *The software infrastructure*. This is the base of software on which new applications are built. It includes system software such as a database management system, as well as core applications. In Chapter 2, we discuss the components of software infrastructure, and describe how to implement and manage them.

— *Software standards*. Software standards need to be specified by the software strategy. Decisions need to be made about whether open or proprietary standards should be adopted, and which internal standards should be specified. We discuss standards (including developments in Unix-based standards and IBM's Systems Application Architecture) in Chapter 3.

— *Software procurement policies*. The software strategy needs to specify the policies for procuring new software, whether it be system software, application packages, or bespoke software developed in-house. We discuss the most important policies in Chapter 4.

© Butler Cox & Partners Limited 1989

Chapter 2

Defining and implementing the software infrastructure

A software infrastructure comprises a coherent set of software tools and building blocks that is used as the basis for developing specific applications. An infrastructure should include as many as possible of the common functions that will be needed to build new applications, so that bespoke development is reduced to a minimum. Some of the functions will be low-level, general-purpose facilities, such as those provided by the operating system, or machine utilities supplied by the hardware vendor (in other words, all the software that used to be known as system software). Other functions will be core applications that are specific to the business, but that are used by many software applications within the business. In general terms, core applications either store or modify data held in corporate databases. In other words, the data generated or modified by core applications will be accessed by other applications. An application that accesses corporate data, but that creates data only for its own use, will usually be a non-core application and is therefore not part of a software infrastructure.

We are concerned, in this chapter, with the developments in software products that are making it possible to create a software infrastructure, and to use it as the basis for a software strategy. There may, in fact, be a need for more than one such infrastructure, but it is important to limit the number if the greatest benefits are to be realised. Organisations will need to devise a plan for the gradual implementation of the five main components of the infrastructure approach because it is a complex task, with significant implications for the systems department, in particular.

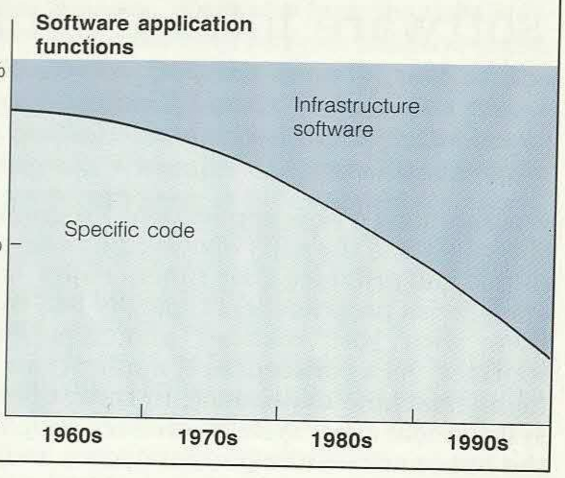
New developments are making the infrastructure approach possible

In the early days of commercial computing, it was necessary to start from first principles when programming a new application. Programming languages and the early operating systems were simple and provided little functionality, so that application programmers typically had to code their own sort routines and input/output routines. As a consequence, applications were highly machine-dependent. In time, operating systems and other system software supplied by the hardware vendor provided more and more prepackaged functions that could be used by application programmers. Languages became more sophisticated, so that a single line of code generated a complex function that would have required many assembler-language instructions. The use of standard languages and other system-software interfaces meant that, in theory, it became possible to transfer an application written for one hardware environment to another machine.

Today, by choosing appropriate system building tools, most of the standard programming functions can be generated automatically. Application developers can therefore concentrate on providing application-specific functions and on building links between different applications and systems and between different parts of systems. Figure 2.1, overleaf, illustrates how more of the functions that formerly had to be coded specifically for an individual application are now available within system-software products. During the 1970s, for example, database management systems and communications software were developed, and provided functions that, hitherto, had to be written by application programmers. During the 1980s, products such as fourth-generation languages, expert-system shells, and CASE tools have continued this trend and have extended the scope of the software infrastructure. We expect to see the scope extended further during the 1990s, by developments in integrated CASE tools, data dictionaries, and dialogue-management products.

© Butler Cox & Partners Limited 1989

Figure 2.1 An increasing proportion of the software functions required by an application is provided by the software infrastructure rather than by specific code



Developments in infrastructure software are not confined to application-independent system software, however. Some of the major suppliers of system-software products (Oracle, for example) have developed application skeletons (or templates) that can be used as a basis for developing a specific application. Other software suppliers, such as Cullinet and Cincom, who have hitherto provided database management systems and development tools, now also provide application packages that are built on top of their existing system-software products. These packages are purchased for the application functionality they provide, rather than for the quality of the system software on which they are based. Others (Digital, for example) are encouraging value-added resellers to build and market application packages on top of their system-software products.

As a consequence, traditional system-software suppliers are busy repositioning themselves in the marketplace so that their customers will also perceive them as suppliers of application software. This is an important trend because it shows that the traditional distinction between system and application software is breaking down, and that system-software vendors are encroaching on the domain of specific applications software suppliers.

The result is that it is increasingly difficult to delineate the boundary between system software and application packages. In the past, packages were inseparable from most of the software infrastructure on which they were based. For example, a stock-control package would need database (or at least file-management) facilities to store records of items, stock levels, and so on. However, as often as not, these facilities would not be based on a standard database management system, and the data would not be easily accessible by other systems; it was therefore difficult for the package to interwork with other systems or to generate additional reports. Purchasers of such a package had to accept it as it was, or perhaps pay the package vendor to develop minor enhancements. Any significant tailoring of the package was uneconomic. As a consequence, packages came to be regarded as cheap solutions used either by small companies that could not afford to develop their own bespoke systems, or by larger organisations for unimportant applications.

Today, however, the use of advanced development tools in conjunction with an existing database management system has led to the development of 'soft' packages than can easily be tailored to create a bespoke application. (Soft packages are described in more detail in Chapter 4.) This means that it is also increasingly difficult to classify suppliers as either system-software suppliers or application-package suppliers. Traditional package suppliers have become aware that advanced development tools are a threat to their market, and have responded by using proprietary development tools and database management systems to develop their packages. The resulting soft packages provide in-built tools for screen formatting, report generation, and other functions. In some cases, package suppliers have even developed their own system-software tools for application-package development.

The importance of a common and stable software infrastructure for all of an organisation's applications has also been recognised by the many package suppliers who are now building new products that can be used in conjunction with the most popular system-software products. For example, the French software house, SOPRA, markets a leading European human-resources package, called PACHA, which is available for a wide variety of

© Butler Cox & Partners Limited 1989

hardware environments. SOPRA has now recognised that making its product available for a range of different hardware is not sufficient. The company has decided to launch a new IBM version of the package that uses DB2, and to follow this with a version that can use Oracle's database management system. In one of our focus groups, several of the participants mentioned the availability of application packages as a reason for choosing a particular system-software product.

Figure 2.2 illustrates how the boundary between applications and system software is being breached. System-software suppliers are extending their product ranges to provide application templates, or even complete application packages, and package suppliers are extending their products to create general-purpose soft packages that can be tailored to meet an organisation's specific requirements. The result is that organisations now have the chance to build a coherent software infrastructure that can support both in-house development and bought-in application packages.

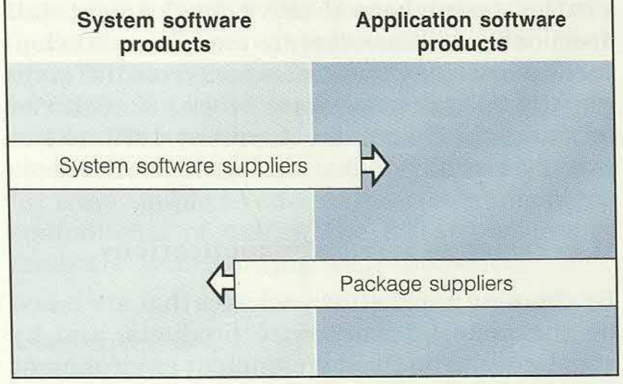
The infrastructure approach provides significant benefits

The infrastructure approach to software strategy has three main benefits: it provides the flexibility to respond quickly to changing business requirements; it helps reduce the variety of software in use and thus reduces costs; it makes it easier to integrate applications.

It provides the flexibility to respond quickly to changing business requirements

In a rapidly changing business environment, it may be difficult, or impossible, to forecast new application requirements even as far as six months ahead. However, it is often in precisely this type of environment that a rapid response to new business needs is vital for survival. The systems department in a major Australian oil company found that the decision to standardise on IBM's CSP and DB2 products for new systems has led to improvements in its ability to deliver on time and within budget. Systems integration has also become easier. The systems department can now respond more quickly to changing business requirements and, in particular, facilitate a move to a centralised marketing approach.

Figure 2.2 The line between system software and application software products is being bridged



This could not have been achieved 10 years ago when there was no software strategy.

By making sure that as many as possible of the functions that are likely to be needed are incorporated into the software infrastructure, the development time for new applications can be minimised, even when the likely requirements are not well understood at the time the infrastructure is defined. Figure 2.3, overleaf, describes an extreme case of an organisation that found that the only way to cope with the uncertainty about the application requirements was to adopt the infrastructure approach to software strategy.

Reduced variety leads to reduced costs

By standardising on a specific software infrastructure, it is possible to reduce support costs, training costs, and in some cases, the costs of the software itself. Training costs, in particular, should not be underestimated. One focus group participant had recently conducted a review of his organisation's total expenditure on an analysis tool. This amounted to around $135,000 for software licences and $270,000 for hardware, but more than $850,000 for training. (Foundation Report 67, *Computer-Aided Software Engineering*, showed how training and support costs for CASE tools often exceed the cost of purchasing the tool itself.)

By reducing the variety of software in use throughout an organisation, it may also be possible to widen the choice of development teams able to undertake new projects. A major oil company uses a standardised software infrastructure from Software AG in all of its offices throughout the world. The infrastructure includes a common development environment (based on products from Software AG). This enables the company to move development staff from one site to another, to carry out development work using team members from different countries, and to allocate work to countries where the software development staff are less heavily loaded.

It is easier to integrate applications

By choosing application packages that are based on the same infrastructure products, and by standardising on the development environment for new software, it will be easier to exchange data between applications and to build links between systems. One Foundation member told us that his organisation was in the process of rationalising its software infrastructure in order to achieve these benefits. This organisation operates in a business sector where recent deregulation has led to huge growth in new demands for applications from the business managers. To meet this demand, the systems department had been forced into buying packages that did not fit into its chosen infrastructure. Moreover, a recent merger had further complicated the situation. With the prospect of further mergers in the near future, the systems department is anxious to have a consistent and comprehensive software infrastructure in place before it has to cope with the need to integrate the different systems.

Figure 2.3 A flexible infrastructure is essential when information systems requirements are unclear

The Securities and Investments Board

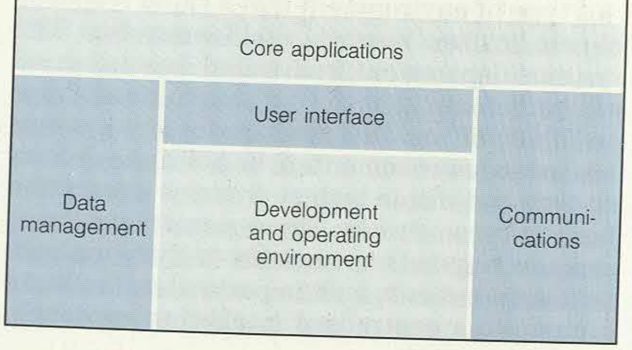
The Securities and Investments Board (SIB) was set up as a result of the UK Financial Services Act, which became law in 1986, and is designed to protect investors. The SIB monitors and regulates the activities of all the types of financial-services organisations recognised by the act. At the time the SIB was set up, it was not clearly understood how it should monitor the activities of the financial-services industry. Obviously, information systems had to play a central role, but it was by no means clear what the requirements for such systems were, or what the scale of the software development effort was likely to be. Given the very short timescales for developing the required systems, the SIB had no choice but to use a bureau. Because of the nature of the regulations, there were no packages that could meet the requirements, so it was necessary to develop bespoke applications. In order to deal with the uncertain and changing requirements, the SIB chose a software infrastructure that would enable it to respond to users' requests quickly and effectively. To minimise the risks, the infrastructure was based on a tried and tested IBM hardware and software environment.

The infrastructure included a relational database, a fourth-generation language, and networking facilities for connection to the bureau. A core set of enquiry and reporting functions was developed initially, with no attempt being made to develop a comprehensive system. Facilities were gradually added as users needed them. The need for these additions had to be fully justified because all development was done by the bureau. After two years of operation, the SIB is now reviewing its software strategy and considering ways in which it can respond even better to the volatile business environment. The software infrastructure is unlikely to change, however.

A software infrastructure has five main components

In essence, a software infrastructure consists of a list of software functions and a description of how they are to be provided. The majority of these functions will be purchased as packages, often from the hardware vendor, although some of them may be acquired from other software vendors. The remaining infrastructure functions have to be written in-house, especially industry-dependent software (core applications) and gateway, conversion, or interfacing software, to enable different parts of the infrastructure to be interlinked. The components of the software infrastructure will vary according to the type of business and the role that IT plays in the business. Figure 2.4 shows the five main software components that make up the infrastructure. Each is discussed in more detail below.

Figure 2.4 The software infrastructure has five main components



© Butler Cox & Partners Limited 1989

Development and operating environment

The development and operating environment includes the operating system, machine utilities, language support, CASE tools, performance tools, and any other functions needed to develop, implement, and run software efficiently in the chosen hardware environment. Choosing the right products for this part of the infrastructure is possibly the most difficult decision the systems director has to make on software strategy. As we discussed in Foundation Report 67, *Computer-Aided Software Engineering*, there is, as yet, no integrated set of development tools that adequately covers the whole of the development life cycle. Indeed, those tools with the greatest life-cycle coverage may not be the most cost-effective because they may be expensive to implement and they may necessitate extensive training. Report 67 lists the criteria to use for selecting the most appropriate development tools.

Data management

The main element of the data management component of the software infrastructure is a database management system. In Foundation Report 64, *Managing the Evolution of Corporate Databases*, we showed that most Foundation members are implementing, or plan to implement, relational database management systems. The development environment should therefore include tools that use SQL (the *de facto* standard for accessing relational databases) to create and manipulate the database, and a data dictionary (or data repository) to describe the information held in the database. These tools and products should, of course, be chosen so they can interwork with the database management system. One of the common problems mentioned by those participants in our focus groups who had chosen IBM's DB2 as their relational database management system was that IBM does not yet have a comprehensive data-dictionary product that works with DB2.

Communications

The communications component of the infrastructure should include software functions for both internal and external communications. Software support for local and wide-area networking is not difficult to achieve. The problems arise mainly in the area of conflicting standards and in interworking between different infrastructures. One organisation in the chemicals industry, for example, uses an IBM-based infrastructure (including SNA communications) for some sites, and a DEC-based infrastructure (including DECnet communications) for other sites. Within either infrastructure, it would not be difficult to choose a coherent set of communications products for applications such as electronic mail and file transfer. However, it is far from simple to choose the communications components of one of the infrastructures to facilitate interworking with the other.

User interface

Very few products are currently available to use within the user-interface component of the software infrastructure. As we discuss in Chapter 3, there is a need for common user-interface standards and products to support those standards. IBM's Presentation Manager will go some way towards establishing *de facto* standards for the user-interface component in the personal-computer environment, but products are needed to support cooperative processing (where two or more processors need to interwork) and dialogue management. At the moment, the main choices to be made are concerned with which in-house standards to set rather than which products to buy. However, it is worthwhile creating re-usable modules that implement an organisation's chosen standards for the user interface. These are routines developed in-house (usually in the course of developing a specific application) that are designed in such a way that they can be re-used in other applications.

Core applications

The final component of the software infrastructure is the organisation's set of core applications. These are applications that are essential to the day-to-day operation of the business. Different business sectors will have different core applications (some examples are listed in Figure 2.5, overleaf). Moreover, an application that is a core application for one business may not necessarily be a core application in another. Similarly, a non-core application for one business may be a core application for another. Although non-core applications may be essential to run the business efficiently and to remain

© Butler Cox & Partners Limited 1989

Figure 2.5 Different businesses have different core applications

Business	Example of core application
Tour operator	Holiday reservation system
Distribution company	Order processing/stock control
Insurance company	Policy database
Retail bank	ATM systems
Manufacturing company	Process control
Retailer	Point-of-sale systems
Public utility	Customer billing

within the law, they do not normally affect the day-to-day operations of other departments, and the software itself does not form a building block for other departments' applications.

Nevertheless, non-core applications should, wherever possible, comply with the other components of the software infrastructure, particularly the user-interface component. However, a non-core application that does not conform with the infrastructure but that provides a good business solution is preferable to one that conforms with the infrastructure but that is inferior in business terms. The infrastructure should not be modified to accommodate a non-core application, however.

Managing core applications as part of the software infrastructure ensures that they are designed so that they can provide a flexible basis for developing non-core applications. In particular, the database used by a core application, such as order processing, will often form the basis for other non-core applications. In making the distinction between core and non-core applications, we are not implying that the former are more important than the latter. In fact, competitive advantage gained as a result of using IT is more often derived from non-core applications. The purpose of the distinction is to emphasise the need to manage core applications as part of a coherent software infrastructure.

A common problem now faces the many large financial-services organisations that have not, in the past, managed their customer-account applications as core applications. Each financial-services 'product' (such as current-account banking, life insurance, and savings accounts) typically had its own set of applications software based on a sales database accessed by account numbers or policy numbers. From the product managers' perspective, this was the most convenient and efficient system design. However, now that large financial organisations have diversified into a wide range of products, they would like to be able to exploit the opportunities for cross-marketing. The problem is that there is no efficient and easy way of finding out which products have been supplied to an individual customer because all the databases are organised around products. Worse still, many of the applications are based on different underlying software infrastructures. These organisations now have a difficult strategic choice to make: either they accept the business limitations imposed by their present software base, or they embark on a massive redevelopment programme.

The number of software infrastructures should be limited

Most organisations will need two, or even three, infrastructures to ensure that the best technical solution can be chosen for different types of application. There are several situations in which this need may arise:

— Different hardware environments, and hence different software infrastructures, may be necessary for applications with special requirements. Thus, specialised scientific computing (such as finite-element analysis) needs high-speed processing capacity but a very simple and efficient software infrastructure; high-performance, free-text retrieval applications may require parallel-processing capabilities and very-high-speed disc access and data retrieval.

— The hardware and software suppliers used for an organisation's mainstream commercial applications may not be able to provide the most appropriate hardware environment and software infrastructure to support certain types of business

© Butler Cox & Partners Limited 1989

applications. Office system and point-of-sale applications are typical examples.

– The ideal package for a particular application may not be available for the mainstream hardware or software infrastructure. Several Foundation members told us that they had purchased a personnel-management package that runs on McDonnell Douglas equipment for this reason.

– Some organisations may have several software infrastructures for historic reasons. It is not usually possible to transfer all systems to a new infrastructure at once; for a time, it is therefore necessary to support applications using both the new and old infrastructures.

Wherever possible, however, organisations should avoid having more than one software infrastructure. Multiple infrastructures mean that integration of applications implemented in different infrastructures is difficult, support costs are increased, and systems staff and users have to be trained in how to develop and use applications in each of the environments. We recommend that an additional software infrastructure should be implemented only where at least one of the following situations exists:

— The additional infrastructure is completely separate, with no requirement to link systems to the main infrastructure.

— The additional infrastructure, and the applications within it, can be managed by a third party (a computer bureau, for example).

— The infrastructure supports just one application and can easily be maintained by the users or by the software supplier.

— The problems of interconnecting applications supported by different infrastructures can be minimised by using simple file-transfer and conversion facilities. This could occur, for example, with a point-of-sale system where an in-store processor provides all the detailed information required by the store managers, and transmits files of consolidated data to the head office for processing on a central mainframe.

Implementing a new software infrastructure requires a migration plan

Few Foundation members are able to start with a blank sheet when designing and implementing a new software infrastructure. Their existing applications, some of which may be 10 or even 20 years old, are based on a range of infrastructure products, each of which may have been installed over the years to meet a specific need. Mergers and acquisitions may have complicated the problem by adding further suppliers and software infrastructures. In the long term, the main options are either to scrap old applications or convert them to run on the new infrastructure. We describe here the best way of migrating to the new infrastructure.

Add infrastructure components as new applications are developed

The first step in migrating to a new software infrastructure is to define and agree on the infrastructure that the organisation wants to end up with. Doing this will limit the migration difficulties because new systems can be procured to conform with the desired infrastructure. It is unlikely to be cost-effective to implement all of the infrastructure at once. This would require a large capital expenditure on items that would not be used fully for some time, until new applications have been built or old ones converted. It would also impose an unacceptable load on the systems department.

The most cost-effective way of migrating to the new infrastructure is to take a series of small steps based on the infrastructure needs of new applications as they occur within the development plan. The application priorities should be derived from established business needs and the corresponding business benefits that the applications will provide. For example, a major life assurance company told us that it is redeveloping its 15-year-old policy and customer applications to use a relational database management system because of the marketing department's business requirement for information to support the company's cross-selling initiatives. This project provides an excellent opportunity to begin to migrate to a more modern software infrastructure that will provide business benefits for most departments within the

© Butler Cox & Partners Limited 1989

company. The systems department realises that several other steps need to be taken before the new infrastructure is fully in place. These include implementing improved document-handling and other office systems functions. However, these functions will not be added to the software infrastructure until they can be properly justified in business terms, and until systems resources are available to implement them.

In summary, the new software infrastructure should not be implemented all at once. Infrastructure components should be acquired to meet the needs of new applications as they are developed. The development priorities should be set according to the size of the benefit that each application will provide to the business.

Consider the possibility of bringing forward the systems replacement point

As the components of the infrastructure are implemented, it may well be possible to bring forward the point at which it is cost-effective to replace existing applications with new systems that conform with the new infrastructure. The portfolio of existing applications should therefore be reviewed at regular intervals to see if the infrastructure functions already available make it cost-effective to rebuild existing applications earlier than was originally planned. The case for doing this will be even stronger if the rebuilt applications require further components to be added to the new infrastructure. (The type of calculation required to determine whether an application can be rebuilt earlier than planned is the same as that set out in Figure 3.3 of Foundation Report 67, *Computer-Aided Software Engineering*; in that report, we showed how the use of CASE tools can bring forward the date when it is cost-effective to redevelop existing systems.)

Foundation Report 64, *Managing the Evolution of Corporate Databases*, gives several examples of tools that can help in migrating to the database components of the new infrastructure. These include fourth-generation languages, application generators, and database transparency software. It is therefore important to keep abreast of developments in software products that may significantly reduce redevelopment costs or provide significantly greater business benefits. Figure 2.6 gives some examples of developments that are likely to occur in the next five to ten years. Once a new infrastructure product that could be of benefit has been identified, it may be necessary to redefine part of the infrastructure to incorporate the product and to re-assess whether ageing applications should be replaced, making use of the new product.

Sometimes, however, components of the new infrastructure can be used to extend the life of old applications, but in a way that will make their conversion to the new infrastructure easier at a later date. The Swiss Bank Corporation, for example, has used the Telon application generator in this way. Existing applications were written in Cobol and use the IDMS database management system. This is out of line with the bank's new software infrastructure, which uses the DB2 relational database management system and Telon as the development language. Although the existing applications will eventually need to be converted, the bank has neither the resources nor the business justification for embarking on the conversion exercise now, because there are other more urgent requirements. However, some changes and enhancements to the existing systems are essential, and these are being developed as new functions written in Telon. Since Telon can generate different versions of Cobol programs that can be used with either IDMS or DB2, conversion of these enhancements will be relatively simple.

Figure 2.6 Developments in software infrastructure products may bring forward the systems replacement point

New industry-specific soft packages.
Full I-CASE products.
Distributed databases.
Graphics interfaces and windowing systems.
New distributed architecture (leading to a reduced requirement for mainframe systems).
Voice input.

© Butler Cox & Partners Limited 1989

Formalise the maintain-or-replace decision

The migration to a new software infrastructure will not be complete until all the existing applications have been replaced by applications that conform with the new infrastructure. It is therefore essential to formalise the procedure for deciding whether existing applications should continue to be maintained, or whether they should be rewritten using (and perhaps adding to) the new infrastructure. The purpose is to identify the best time to redevelop an application. This means comparing the costs of continuing to maintain and run an application with the cost of rewriting it for the new infrastructure. The costs of maintaining both the old and new infrastructures also need to be considered.

It is well known that many organisations spend about 65 per cent of all their analysis and programming effort on software maintenance. The purpose of maintenance is to protect the organisation's investment in systems by prolonging their useful life and by improving the benefits they bring to the business. There are three categories of maintenance:

- *Corrective maintenance,* which is concerned with resolving errors.
- *Adaptive maintenance,* which involves enhancing and modifying systems in line with the needs of the business.
- *Perfective maintenance,* which consists of changes to the application structure and coding to improve performance and maintainability, and to reduce the likelihood of errors.

As software ages, the workload in all three categories of maintenance tends to increase. The business environment changes, creating a need for software enhancements, which gives rise to adaptive maintenance. These enhancements introduce more software errors, requiring corrective maintenance. Eventually, the software reaches a point where it has been changed so often that the original structure is lost, and making further changes increases the probability of introducing more errors. At this point, perfective maintenance is needed if the software is to be upgraded to a state where further maintenance can be carried out.

It is easy to continue maintaining old systems without appreciating that costs are escalating and that it would be less expensive to redevelop the applications. The best time to replace an application can be chosen by keeping track of the effort spent on maintaining each application and subjecting every system to a regular formal review. It is helpful to keep detailed records of the effort spent on each category of maintenance for each module of the system. As well as helping to identify when the system as a whole should be replaced, such records can also indicate the need for perfective maintenance.

The formal review of systems should be carried out at least annually to assess:

- The extent to which the application currently meets users' requirements.
- The risk and impact of a system failure.
- The effort required to maintain the system adequately.
- New requirements, or growth in the processing load, that may force the application to be redeveloped.

In the United Kingdom, the Central Computer and Telecommunications Agency (CCTA), which helps central government departments to make the best use of information technology, has developed a 'system-maintenance profile' that helps to identify the point at which a system should be replaced. There are nine criteria in the system-maintenance profile and a total of 16 measures (between one and three measures for each criterion), as shown in Figure 2.7, overleaf. Each measure provides a score. The scores are totalled for each system, and systems scoring 100 or more are candidates for renewal.

Some organisations restrict maintenance effort to a predefined percentage (typically 30 to 50 per cent) of the systems development department's budget. Such a policy can be useful because it ensures that an adequate proportion of development resources is assigned to new applications. However, this type of control is less effective than a formal evaluation of the type advocated by the CCTA because it focuses on the problems of the systems development department and largely ignores the needs of the business. In particular, it may encourage the systems department to carry out more development work (which systems staff tend to

© Butler Cox & Partners Limited 1989

prefer), when the real need is to provide a stable and reliable base of existing systems.

Formalising maintain-or-replace decisions is not only essential for ensuring that the migration to a new software infrastructure proceeds as smoothly as possible. Once the new infrastructure has been implemented fully, it is still necessary to review applications at regular intervals to determine whether the software-strategy goals will be better met by continuing to maintain them, or by rewriting them.

Figure 2.7 There are nine criteria in the CCTA's system-maintenance profile

Category	Criteria	Measures
Adequacy to user	Desirable changes	—(Man-days per annum on desirable changes/thousand lines of code) + 1.
		—Degree to which desirable changes are being blocked (1 = not at all, 5 = completely).
	Changes backlog	—(Estimated man-days to clear backlog of changes/thousand lines of code) + 1.
		—Degree to which system is failing to meet requirements (1 = fully, 5 marginally).
Risk to business	Staffing	—Degree to which staffing is a problem (1 = none, 5 = major).
		—Quality of documentation (1 = excellent, 5 = non-existent).
	Change control	—Change-control procedures (1 = good, 5 = non-existent).
		—Testing procedures (1 = good, 5 = non-existent).
	Errors	—(Errors per annum/thousand lines of code) + 1.
	Impact of errors	—Rating of effect of errors on the business (1 = nil, 5 = significant).
	State of code	—System age (1 = 1 to 7 years, 2 = 8 to 14 years, 3 = >14).
		—Structure (1 = good, 5 = bad).
		—Program size (thousand lines of code/number of programs).
Support effort	Staffing	—(Maintenance effort per annum/thousand lines of code) + 1.
	Mandatory changes	—(Annual effort on mandatory changes/thousand lines of code) + 1.
		—Reduction in mandatory changes if system redesigned (1 = nil, 5 = substantial).

Each measure provides a score. Systems scoring a total of 100 or more are candidates for renewal.

(Source: Managing Software Maintenance, CCTA, October 1987)

The main applications-software role of the systems department will be to manage the software infrastructure

The infrastructure approach to software strategy will change the roles and responsibilities of the systems department with respect to applications software. In particular, systems staff will be less concerned with developing and implementing applications, and more concerned with defining and managing the software infrastructure. These changes are consistent with the results of other recent work carried out by the Butler Cox Foundation. For example, we pointed out in the first Directors' Briefing Paper (*Managing Information Systems in a Decentralised Business*, published in March 1989) that there is an established trend towards devolving responsibility for the information systems function to business units.

The impact of this trend on the role of the systems department can already be seen in the way software suppliers are now selling their products. For example, Management Science America (MSA) Inc, the world's largest supplier of applications for mainframe computers, has moved the emphasis of its selling effort from the systems department to user departments. In general, MSA now puts about three times as much sales effort into convincing users that a package can meet their business requirements as it does discussing technical issues with systems staff. MSA's experience is that the role of the systems department is typically to develop a 'long list' of potential suppliers, and to give a final technical endorsement after the users have made their choice. This is a logical division of responsibilities because users are in a far better position than systems staff to decide whether an application package provides the required facilities and whether it will fit in with existing working methods.

The developments being made in software-infrastructure products will, in the future, also

© Butler Cox & Partners Limited 1989

have an impact on the systems department's role in creating bespoke applications. Because most of the standard functions will be provided from within the infrastructure, the main skill required to develop an application will be business analysis, not the ability to generate program code. By using the building blocks provided by the infrastructure, users will therefore be able to construct their own applications.

The need for technical skills will not, however, disappear. In several previous Foundation studies (Report 64, *The Evolution of Corporate Databases*, Report 65, *Network Management*, and Report 67, *Computer-Aided Software Engineering*), we discussed specific components of the software infrastructure in detail. They have all emphasised that managing and choosing a comprehensive infrastructure are difficult tasks that require highly specialised technical skills. These tasks cannot be left to non-systems staff to perform. Thus, the need for user organisations to employ programmers is not about to disappear. We believe that, even in the long term, when most applications development may well be carried out by user departments, programmers will still be needed to develop interfaces and conversion software that will enable different parts of the infrastructure to interwork. The skills needed in future will principally be those of systems integration, analogous to the skills found among the current generation of systems programmers.

To ensure that the software infrastructure is fully defined and properly managed, the systems department will need to be responsible for:

- Defining and managing the procedures for implementing the infrastructure and for converting existing applications to conform to it. In particular, there is a need to ensure that new core applications conform to all aspects of the software infrastructure, and that software-infrastructure products provide adequate capacity, performance, reliability, and availability. This applies to all the components of the infrastructure, whether they be networking software, development tools, or applications, which, in turn, may either be bought-in or constructed in-house.
- Defining rules and guidelines for using the software infrastructure to build new applications or enhance existing ones. Standards must be defined for carrying out feasibility studies, requirements analyses, system designs, system constructions, and systems testing, and for producing documentation. These standards should specify both the best practices and the methods to be used, to ensure that each stage of the development cycle is completed satisfactorily before progressing to the next.
- Charging users for systems services in a way that ensures that the best use is made of the software infrastructure. There is a danger that flexible and easily used infrastructure facilities could be used in situations where a simpler, possibly manual, approach would be more cost-effective. The systems department should therefore operate a chargeback mechanism that encourages the user community to use the corporate software infrastructure in the most effective way. (Foundation Report 66, *Marketing the Systems Department*, provides detailed advice about chargeback schemes.)
- Resolving difficulties that might arise from allowing user departments to make their own software-procurement decisions. In particular, there is a need to ensure that systems built by different departments can, where necessary, interwork, and that development effort is not duplicated. This responsibility can be discharged by providing high-level consultancy advice on the best technical solution to meet functional, user-interface, performance, reliability, and maintainability requirements, and by providing a help-desk service to handle ad hoc requests for assistance about using the infrastructure.
- Planning the future evolution of the software infrastructure. This means evaluating new technologies to assess how they could support the business better than the components of the existing software infrastructure.

However, the most important decision that the systems department has to make when defining a software infrastructure is to decide on the set

© Butler Cox & Partners Limited 1989

of software standards to which the infrastructure products must conform. In the next chapter, we discuss developments in open and proprietary software standards, and identify the main options for the standards element of the software strategy.

© Butler Cox & Partners Limited 1989

Chapter 3

Forming a policy for software standards

In the previous chapter, we discussed the need for a software infrastructure that can provide the building blocks for new applications. The software products that comprise the infrastructure will, of course, need to conform to an agreed set of standards, and a policy on standards is a necessary component of a software strategy. In general, a separate set of standards will be required for each software infrastructure, although, as we emphasised in Chapter 2, the number of infrastructures should be limited.

In this chapter, we identify the benefits that a well-thought-out set of software standards will bring to an organisation. There are several choices available, the main ones being open standards defined by standards-making bodies such as the International Standards Organisation (ISO), open standards based on the Unix operating system, and IBM's *de facto* proprietary standards. Our discussion of proprietary standards is limited to IBM's because standards developed by other hardware or software vendors have not been widely adopted, except in a few specialised application areas. Indeed, many vendors have abandoned their proprietary standards in favour of open standards. Moreover, the importance of IBM standards is well recognised in the marketplace — some vendors are even migrating their own standards to make them compatible with IBM's *de facto* standards.

Although developments are continuing to take place in proprietary standards, recent developments in open standards mean that the latter are becoming an increasingly viable choice, particularly for intelligent workstations and minicomputers. In time, open standards also have the potential to become a genuine alternative for mainstream (mainframe) computing. However, it will not be possible just to standardise on open standards or a proprietary architecture like IBM's SAA. These standards and architectures are defined in such wide terms that it will be necessary for an organisation to choose a subset (or 'profile') on which to base its software standards. In the final section of the chapter, we provide advice on how to choose the appropriate subset.

The benefits sought from setting standards should be defined

By specifying the standards that its software must comply with, an organisation is essentially setting out to simplify the technical environment in which its application systems are developed and operated. Software standards may also be used as a means of breaking free from a dominant hardware supplier. Many organisations have based their mainstream hardware environments on the architectures and standards of a single supplier and have allowed this supplier's development plans to set the pattern for their own computing strategy, even when they have been able to buy compatible products from other suppliers. This approach has, in some cases, worked well. More often than not, though, the reasons for originally choosing the now dominant supplier have been eroded by time and by the advances made by other, more innovative, suppliers. As a result, many organisations find themselves in a situation where the single supplier is, in effect, a monopoly supplier and is able to charge high prices and impose strict contract conditions.

Different conditions have developed in the personal-computer and engineering-workstation markets, however. In both cases, *de facto* standards (based on MS-DOS and Unix respectively) have emerged which, although not strictly non-proprietary, have been sufficiently open, and sufficiently stable, to allow intense

© Butler Cox & Partners Limited 1989

competition to develop in the supply of hardware and software. New companies, some of which have grown quickly to the stage where their annual turnover exceeds one billion dollars, have emerged from this competitive market, while established suppliers have been subject to unprecedented, and very desirable, competition. As a consequence, users of personal computers and specialised workstations have benefited from increased innovation, greater choice, and lower prices.

As we explain more fully later in this chapter, developments in non-proprietary standards (particularly those based on Unix) have reached the stage where the standards can be used as the basis for competitive general-purpose minicomputers. Within the next few years, there will also be general-purpose mainframes based on the same standards. When this happens, user organisations will, for the first time, have the option of adopting non-proprietary standards for the whole range of their computing requirements, and will be able to enjoy the benefits of a fully competitive marketplace.

The advantages of such a policy are considerable, and have led many public-sector (and some private) organisations in Europe and North America to specify that products must comply with non-proprietary standards. There are also disadvantages, of which the largest are the cost and timescale required to transfer a large portfolio of existing applications to the software infrastructure implied by the standards. Any decision about whether to change from proprietary to non-proprietary standards must be based on a wide range of business and technical considerations, including hardware issues, and we shall not attempt to resolve it in this report. (We intend to address this question more fully in a Foundation Technology Briefing to be published later in 1989.)

It is essential, however to identify the benefits sought from software standards before deciding which standards to set, because each type of benefit requires different kinds of standard, and different degrees of compliance with the standards. In general, software standards make it easier to plan and manage the technical environment, and provide specific benefits in the following five areas:

— *Software interworking*. The standards may be required for exchanging data between applications, for implementing distributed systems, or for establishing cooperative-processing systems. Interworking between software systems requires standards both for the format and interpretation of information, and for transferring the information between the systems, often via a communications network. For software interworking to take place, the communicating systems must comply precisely with the standards.

— *Software portability*. Many organisations would like to be able to run the same application systems on different ranges of hardware without having to rewrite the application to suit the new hardware environment. To be able to achieve this requires standards for the interfaces between applications and infrastructure-software components such as operating systems, database management systems, communications packages, and perhaps, a user-interface package. Again, software portability will be achieved only if the applications comply precisely with the standards.

— *Development staff flexibility*. User organisations often want the flexibility to assign individual development staff according to business priorities, not according to the specialist language or machine skills that a person has. Although the implementation of a common software infrastructure will help, it is not sufficient. To have complete freedom in assigning staff to development teams, computer sites, and development projects, it is necessary to have standards for development methods and tools, programming languages, data-naming conventions, data access, communications, and user-interaction methods. In this case, precise compliance with the standards is not necessary because, to some extent, development staff can adapt to local variations.

— *User-training requirements*. By adopting a standard style of user interface for all systems, it is possible to reduce the training needed by users and to improve the efficiency with which they use the systems.

© Butler Cox & Partners Limited 1989

This is more likely to be the case if the chosen interface style is a modern, user-friendly one, such as the window-icon-mouse-pulldown-menu (WIMP) style, pioneered by Xerox.

– *Communications between organisations*. In Foundation Report 59, we showed that electronic data interchange (EDI) is becoming increasingly important for many organisations. Communications between organisations require standards for the format and interpretation of business data, and sometimes of graphics. Standards for the EDI communications network are necessary in some cases, although protocol- and format-conversion services may sometimes be used instead. It is not essential to use the same standards for the organisation's in-house systems, but there may be advantages if this is done. There is, unfortunately, no comprehensive, international standard for EDI. In some countries, however, and in industries such as retail, banking, insurance, and chemicals, interim *de facto* standards have been developed or set by early EDI systems.

The relative importance of these benefits will vary between organisations, and even between different parts of the same organisation. For most Foundation members, the ability to deploy systems development staff according to business priorities, and the reduced need to retrain users when they begin to use a new application will be more important than software portability. For some organisations, software interworking is increasingly important, although for others, communications with other organisations may be more important than any of the other benefits.

While the benefits from setting software standards are considerable, there are disadvantages as well. In particular, an organisation will not be able to consider any software product that does not conform with its standards, even though that product may in all other respects match its requirements. The more rigid the standards are, the smaller will be the list of products that meet the standards. Moreover, products complying with the standards may have fewer functions or be less easy to use than non-standard ones. Another disadvantage is that products complying with all the functions of an all-embracing standard are likely to be more complex, more expensive, and less efficient than those that have been optimised for a narrower range of functions.

Progress in public, open standards remains slow

The need for open standards that facilitate interconnection between systems based on different hardware was recognised in the early 1970s. This led to the formulation by the ISO of the seven-layer 'open systems interconnection reference model', known as the OSI model. Although this model is defined in an international standard, it is a framework into which more specific public international standards for interconnection should fit, rather than a specification with which products must comply. (Suppliers' claims to have products that comply with OSI should therefore always be treated with caution.) Thus, the X.25 packet-switching standard fits into the lowest three layers of the OSI model, the Ethernet standard into the lowest two layers, and the FTAM (file transfer, access, and manipulation) standard into the uppermost layer. To achieve full interworking between systems, it is necessary to use appropriate standards for each layer of the model.

The scope of the OSI model is being broadened

The initial set of standards (or protocols) conforming with the OSI model were concerned with the basic functions required to provide communications paths between applications, and between terminals and a remote operating system. However, the OSI model did not originally address either network management or cooperative processing (although it did envisage an FTAM protocol, which enables files to be accessed and manipulated over a communications network). Under pressure from smaller vendors, from users, and from governments, the scope of OSI (and of public standards generally) has been significantly widened to include the concept of systems interworking, in which applications running on different machines interact with each other.

This has led to new standards initiatives in the areas of data access and operating systems. For

© Butler Cox & Partners Limited 1989

example, the SQL standard for database access (which was discussed in Foundation Report 64, *Managing the Evolution of Corporate Databases*) is now well established as an international standard, and standards for data dictionaries (the Information Resource Dictionary System, or IRDS), are being developed, albeit slowly. The newly defined Posix standard, which specifies a standard programming interface to Unix operating systems, is the first venture of the international standards-making organisations into the standardisation of operating systems.

A disadvantage of the wider scope of public standards is that, because so many interested parties have contributed to the standards-making process, the final version often includes so many options that it is no longer really a single standard. This is precisely what has happened with many of the OSI standards. As a result, users and suppliers wishing to implement communications software conforming to OSI standards have been forced to choose subsets of the standards within each layer of the model to form 'protocol stacks' or 'functional profiles' that are specific to that organisation or to the type of application that the communications facilities are to support. Thus, the US and UK governments have each developed, for procurement purposes, functional profiles called the Government Open Systems Interconnection Profile, or Gosip. The US and UK Gosips are, inevitably, different, but there are moves to develop a common Gosip. Internationally, Government buying power is immense and will continue to be a strong influence on suppliers' adoption of standards.

Public standards usually lag behind product developments

The main international standards-making body is the ISO, but there are, as shown in Figure 3.1, many other organisations with an interest in standards representing national interests, computer manufacturers, the PTTs, and user groups. With so many interested parties involved in standards-making, it seems inevitable that progress will be slow. Figure 3.2 (on page 24) outlines the procedure for gaining approval for a new ISO standard. Although this process can be completed quickly if everything goes smoothly, the opportunities for delay are considerable, and it usually takes four to five years. It is not surprising, therefore, that suppliers have been hesitant to design products that conform to emerging international standards. Long before standards have become established, suppliers have marketed products that embody proprietary standards.

Figure 3.3 (on page 24) illustrates the typical relationship between the development of standards and the development of technology. It shows that, usually, standards are fully established only once a technology is in widespread use. During the early stages of development, standards are non-existent, or just emerging. Because the development of standards lags behind the development of the technology, suppliers are either reluctant to accept an emerging standard, or they insist on providing their own 'added value' features not included in the standard. SQL is a typical example of this. In theory, if database-access code is written in SQL, it should be possible to replace the underlying relational database management system without having to change the application software. However, virtually all relational database management systems have non-standard features that system developers are reluctant to forego. In practice, therefore, SQL applications are not fully portable.

Frustration with the usual timescale for defining an international standard, and pressure from users (particularly governments), has led to the development of standards by industry bodies, and their subsequent transfer, in largely complete form, into the formal standards-making process. In several cases (the ANSI standard for the Fibre Distributed Data Interface, or FDDI, for instance), this has resulted in standards that are ahead of the technology that they relate to. Sometimes, even the OSI procedures can be speeded up. In the case of the EDIFACT standard for electronic document interchange, the draft proposal was approved by the ISO in just 12 weeks.

Once the formal standards-making process is complete, there is, however, no guarantee that suppliers will implement the standards in compatible ways. Thus, X.25 implementations by European PTTs are often incompatible, because they comply with different subsets of the same overall standard. There is, in fact, no guarantee that the standards will be implemented at all. At present, for instance, hardly

© Butler Cox & Partners Limited 1989

any products comply with the OSI Virtual Terminal Protocol (VTP). The functions that VTP provides are appropriate only for dumb terminals, for which good *de facto* standards already exist. Moreover, as we explained in Foundation Report 63, *The Future of the Personal Workstation*, dumb terminals are being superseded by intelligent workstations.

Figure 3.1 **Standards-making organisations may be international or national, or they may specialise in an industry or a function**

International standards organisations

ISO — *International Standards Organisation*
Part of the United Nations, based in Geneva.

CCITT — *International Telegraph and Telephone Consultative Committee*
Geneva-based arm of the International Telecommunications Union.

CEN — *European Committee for Standardization*
Work is dictated by the European Commission and driven by the perceived need to harmonise IT standards by 1992.

National standards organisations

AFNOR — *Association Française de Normalisation*
French national standards association.

ANSI — *American National Standards Institution*
A non-profit US organisation founded by manufacturers. Does not produce standards itself — documents are fed to ANSI from over 250 organisations.

BSI — *British Standards Institution*
British national standards body.

DIN — *Deutsches Institut für Normung*
West German national standards organisation.

Professional and industry bodies

ECMA — *European Computer Manufacturers' Association*
A vendor consortium that develops specifications that are often incorporated into the final OSI standards.

EIA — *Electronic Industries Association*

EWOS — *European Workshop for Open Systems*
Established in 1988 to develop profiles based on the OSI model. Aims to coordinate European activity and liaise with groups in the United States and Japan. The output will be documents in a form that is directly usable for rapid standardisation both by CEN/CENELEC and ISO.

IEC — *International Electrotechnical Commission*
US organisation that produces world standards for electrical and electronic engineering.

IEE — *Institution of Electrical Engineers*
UK arm of IEEE.

IEEE — *Institute of Electrical and Electronic Engineers*
US learned institute. An independent professional body that creates public standards that are subsequently adopted by such bodies as ANSI and ISO.

NBS — *National Bureau of Standards*
Recently renamed the National Institute of Standards and Technology (NIST). Works on behalf of the US government to establish specifications that have to be satisfied by manufacturers bidding for government computer contracts.

Bodies that promote functional standards

COS — *Corporation for Open Systems*
Non-profit organisation, based in Washington DC. Aims to promote inter-operable, multivendor products and services operating under agreed-to OSI, ISDN, and related international standards.

OSF — *Open Software Foundation*
A US non-profit, industry-supported organisation founded in 1988, which is developing products around Posix and IBM's version of Unix, AIX.

OSITOP — *Open Systems Interconnection Technical and Office Protocol*
A European user association with the objective of including users' requirements in the standardisation process by promoting OSI-based standards and international standardised profiles. Has over 120 members, both users and vendors.

SPAG — *Standards Promotion and Applications Group*
European equivalent to COS, based in Brussels. Formed from major IT companies in the EC to advise on implementing policy decisions on standards.

Unix International
Comprises vendors supporting System V, Release 4, the latest version of AT&T's Unix operating system. Aims to work closely with X/Open, and will conform with the Posix standard.

X/Open
A non-profit, Europe-based, independent consortium of international computer-systems vendors who are promoting the development of an open, multivendor Common Applications Environment based on *de facto* and international standards. Founded in 1984. Liaising closely with IEEE to keep the Common Applications Environment in line with Posix and other IEEE work aimed at providing a complete operating environment.

© Butler Cox & Partners Limited 1989

Figure 3.2 Developing a new ISO standard is a long procedure

The following procedure is followed, assuming that each step is successfully completed. The whole process may take upwards of two years.

1. Proposal is drafted and agreed by ISO member (for example BSI, DIN).
2. Proposal is submitted to ISO.
3. ISO technical committee votes on draft proposal.
4. Draft proposal is passed to appropriate ISO subcommittee as work item.
5. Subcommittee produces working draft of ISO standards.
6. Postal ballot is held on working draft.
7. Draft International Standard (DIS) is formulated.
8. Vote is held on DIS.
9. DIS is elevated to full international standard.

Unix-based open standards are becoming viable

Unlike proprietary operating systems such as IBM's MVS and DEC's VMS, Unix was not designed by a hardware vendor. It was, in fact, developed by two researchers at Bell Laboratories. For nearly 20 years, Unix advocates have claimed that it is the vehicle for providing machine-independent, and vendor-independent, applications. In theory, it should be possible to take a Unix application written for one environment and run it in any other Unix environment. In practice, that has not happened because there have been (and still are) several Unix 'standards'.

Originally, major hardware vendors did not take Unix very seriously. Today, however, most major suppliers offer Unix and Unix-based applications on all types of hardware, from workstations to mainframes. In addition, software houses are increasingly making their products available for a Unix environment. In Germany, for example, the ISIS catalogue (which lists all the software products in that country) contains 1,300 Unix applications.

Figure 3.3 Standards usually lag behind technology, as the following examples show

Technology	No standards (proprietary solutions only)	Competing proprietary products (standards emerging)	Established standards (commodity products)
Data management	Hypertext	IDMS IMS Total Natural	Cobol data division SQL
User interface		X-Windows Common user access Open Look	
Business programming languages	Expert systems	Focus Nomad Mapper Linc dBASE II	Cobol
Realtime language		Coral RTL/2	ADA
Operating system interface	Proprietary operating systems		Posix CP/M MS-DOS

© Butler Cox & Partners Limited 1989

Unix is now well established

Unix systems are now well established in the networked-workstation environment and are increasingly being used for 'niche' applications such as front-office support in financial-services companies, computer-aided design and manufacturing, and process control. Because of the highly competitive nature of the Unix market, the price-performance of Unix systems is generally better than that of systems based on proprietary architectures. In some cases, this advantage is a factor of 10 or more and is likely to be sustained, and even increased, as Unix benefits from the impact of new computing technologies such as reduced-instruction-set computers (RISC) and parallel processing. Within a few years, it will be technically feasible to use Unix for the whole of a major organisation's computing.

There is no doubt, therefore, that Unix and other open standards are here to stay. Suppliers such as Apollo, Datapoint, and Norsk Data have lost business because of the closed natures of their proprietary standards and have been obliged to move towards open standards.

Supplier groups have been formed to promote Unix

As a consequence, most of the major suppliers now see Unix as a good basis for software-infrastructure standards. Three organisations — X/Open, OSF, and Unix International — have therefore been formed by various groups of suppliers to develop and promote open software-infrastructure standards based on Unix. Two of these groups — OSF and Unix International — are working hard to make their particular version of Unix the *de facto* standard. Regretfully, it is too early to advise Foundation members as to which group, if either, will succeed in this aim.

The membership of the three groups is compared in Figure 3.4, overleaf, which shows that several of the leading hardware vendors belong to two of them. IBM and DEC, for example, belong to OSF and X/Open, and AT&T belongs to X/Open and Unix International. Membership of X/Open is restricted to hardware vendors, whereas membership of OSF is open to any organisation with an interest in software. Unix International comprises vendors who support AT&T's latest version of the Unix operating system, System V, Release 4.

We believe that the formation of these organisations is a significant step forward in the development of open software standards. The fact that IBM and DEC are members of both X/Open and OSF, and are both making positive contributions to the development of open standards, indicates their acceptance of the market demand for software-infrastructure standards. In addition, the activities of these three open standards organisations have contributed to the increasing viability of Unix in the marketplace.

X/Open

X/Open is an independent non-profit consortium of hardware vendors whose principal activity is the development of a comprehensive set of standards for an open, multivendor, software infrastructure known as the Common Applications Environment (CAE). CAE is a consciously created alternative to the IBM software environments. It is intended to boost sales of the members' equipment by attracting software developers and reassuring users that they will not be locked in to a single supplier.

X/Open operates in the field of emerging standards, selecting those *de facto* and international standards that are thought to be the most practical and technically acceptable for the CAE. The consortium aims to publish standards covering operating systems, programming languages, data management, the user interface, transaction processing, and communications. The standards are intended to be internationally acceptable. X/Open has already published a 'portability guide' and recently announced a 'branding programme' that will indicate which software products have passed the compliance tests for the X/Open interfaces. Standards are agreed in the conventional way, with all members evaluating proposals and arriving at a consensus.

Open Software Foundation

The Open Software Foundation (OSF) is a direct response to AT&T's dominance of Unix standards. The founders, who are major suppliers of Unix systems, seek to wrest control of the standards from AT&T and place it in the hands of an independent body. OSF describes itself as a software house whose aim is to construct and license an Open Applications

© Butler Cox & Partners Limited 1989

Figure 3.4 Groups of suppliers have established three organisations to develop and promote Unix-based open software infrastructure standards

X/Open[1]

AT&T	IBM	Philips
Bull	ICL	Siemens
DEC	NCR	Sun Microsystems
Ericsson	Nixdorf	Unisys
Fujitsu	Nokia Data	
Hewlett-Packard	Olivetti	

OSF[2]

Apollo	Hewlett-Packard	Software AG
Boeing Computer Services	Hitachi	Sony
Bull	IBM	Sumitomo Electric Industries
Canon	MIT's Information Systems	Texas Instruments
Carnegie Mellon University	Nixdorf	University of Guelph
CETIA	Norsk Data	University of Maryland, Department of Computer Science
CSK Corporation	Philips	Wang
Data General	Project Athena at MIT	Xerox Corporation
DEC	Quantum	
Dell Corporate Services Corporation	Siemens	

Unix International[3]

Addamax	ICL	Prisma
Alcatel-SMH	Informix	Pyramid
Amdahl	Intel	Ricoh
Arix	Interactive Systems	Sony
AT&T	Lachman Associates	Stellar Computer
Computer Consoles	Locus	Stratus
Concurrent	Micro Focus	Sun Microsystems
Control Data	MODCOMP	Texas Instruments
Convergent	Motorola	Tolerant
Data General	NCR	Toshiba
Dupont Fibre Division	NEC	UniSoft
Ericsson	Oki Electric Industry	Unisys
FP Computing	Olivetti	Wang
Fujitsu	Omron	Xerox
Fuji-Xerox	Oracle	88Open
Gould	Phoenix Technologies	
HCL	Prime	

1 All the members as of January 1989

2 Not the total membership; there are now over 90 members

3 All the members as of March 1989

Environment. Eventually, this will comprise an operating system, user-interface software, a database management system, communications software, and software-engineering tools. All of the members of OSF have committed themselves to provide the Open Applications Environment for their Unix products.

OSF believes that the most effective standards are those embodied in existing software products, and is therefore basing its own developments on existing products. The OSF operating system will be based on a future version of AIX, IBM's version of Unix, and will comply with the Posix standard. The user-interface software (known as OSF/Motif) will comply with the X-Windows protocol. OSF invites submissions of software products from its members or, in theory, from any other organisation. Products are evaluated technically by an OSF team and the final choice is made by the Board. This procedure is different from that used by X/Open

© Butler Cox & Partners Limited 1989

and reflects the different nature of the two organisations; OSF is a software house and X/Open is a standards-promotion body.

Unix International
Unix International, which was announced towards the end of 1988, is AT&T's response to OSF. The aim is to align other major Unix suppliers with AT&T's new version of the Unix standard, System V, Release 4, which consolidates the three most important commercial variants of Unix (Berkeley, System V, and Xenix). The group will not write any software itself, but the members will be provided with early (and simultaneous) access to Unix source code. Unix International will work closely with X/Open.

Unix will not supersede mainstream operating systems in the near future

Despite the growing success of Unix in the marketplace, and despite the activities of X/Open, OSF, and Unix International, many Foundation members are sceptical about Unix, and question whether it will ever supersede established mainstream operating systems like IBM's MVS and DEC's VMS. In our view, Unix-based operating systems will evolve quite quickly to the stage where they could be used for mainstream corporate computing, but it is unrealistic to expect major users to switch to Unix in the foreseeable future. Their existing applications portfolios make it difficult for them to migrate away from their existing software infrastructures. Both users and suppliers have immense investments of hardware, software, and skills in the established proprietary systems, and the cost, time, and effort involved in migrating away from these systems are so high that many organisations will not be prepared to contemplate such a move. Furthermore, the range of application packages that is compatible with proprietary systems and suitable for use by large organisations is greater than for Unix.

The major suppliers of proprietary operating systems are, of course, anxious to prevent their customers from migrating to open standards, and go to great lengths to emphasise the superiority of their proprietary products compared with those based on open standards, even when they offer both. Thus, IBM and DEC have restated their commitments to MVS and VMS respectively, and they are particularly concerned to position Unix as an unsuitable base for large-scale commercial applications.

There is a fair degree of truth in this view. Unix was not designed as a comprehensive operating system and it has several major limitations, especially in the security, integrity, and performance it provides. Until recently, for example, there were no in-built file-locking facilities, a fundamental requirement of any mainstream operating system. Its resilience and recovery features are still weak, as are job scheduling and access control. As commercial interest grows, suppliers will address these deficiencies, and some progress has already been made. However, some of the deficiencies, notably those concerned with system security, are rooted in the fundamental design of Unix, and will be very difficult to remedy. It will therefore be some time before Unix will reach the stage of being a genuine alternative to mainstream operating systems.

Unix will not, therefore, supersede the mainstream proprietary operating systems for large-scale systems in the short to medium term. It will, however, be increasingly marketed and seen as a valid alternative to them. There may also be further moves to converge Unix and proprietary environments. DEC, for instance, has announced Posix support for VMS, allowing Unix applications to run alongside VMS applications.

An organisation that chooses to base its mainstream software infrastructure on Unix will have to look carefully at the limitations of today's Unix systems in deciding on the migration plan, and will need to invest a higher than usual level of effort in tracking standards developments. It may even have to play a part in advancing those standards. In return, it will obtain a very wide choice of supplier, and excellent price-performance from its hardware and software products.

IBM's *de facto* standards will remain important

While Unix is set to become an important force in certain well-defined areas, proprietary software standards will remain important in the area of computing that is of most significance

© Butler Cox & Partners Limited 1989

to the majority of Foundation members — large-scale corporate data processing systems based on mainframes. In most countries where there are significant numbers of Foundation members, the dominant mainframe supplier is IBM. Even in France, where Bull is the market leader, it is not possible to ignore IBM and the importance of its *de facto* standards to the IT industry as a whole.

In the past, IBM has not hesitated to introduce new hardware architectures and its associated software standards, as it has enhanced its hardware ranges and introduced new ones for specific purposes. Even in the communications area, where SNA was intended to provide a universal solution, there has been a proliferation of incompatible 'logical units' within the SNA architecture, and IBM has also adopted some public communications standards, such as X.25.

IBM is now making a determined effort to rationalise its standards. Progress has already been made in the communications area, with the announcement of the Advanced Program to Program Communications (APPC) protocol. This is intended to be a universal solution to SNA communications interworking, defining a single, standard unit. IBM has now begun to provide products that implement the APPC protocol. A second initiative has resulted in the transfer of established products from one systems environment to others. For instance, following the release of CICS for OS/2, IBM indicated that CICS might also be offered for the AS/400 and AIX environments in the future.

Systems Applications Architecture is a significant initiative

The most important standards initiative from IBM, however, is Systems Applications Architecture (SAA). In the two years since IBM announced the SAA concept, there has been intense speculation about IBM's motivations. Some see SAA as an attempt by IBM to pull together its three main product lines — the System/370 and successor mainframes, the mid-range AS/400s, and the PS/2 range of personal computers. Others see SAA as a competitive response to DEC, aimed at achieving the applications portability already available to VMS users. Yet others see it as a competitive response to Apple, providing IBM users with the type of user-friendly graphical interface already available to Macintosh users. More sceptical observers believe that SAA is little more than a smokescreen, designed to divert attention away from IBM's real product-development intentions. These sceptics question IBM's real commitment to SAA and claim that their view is based on informed comment from within IBM.

In our view, there is no doubt that IBM is fully committed to SAA. It should be seen as nothing less than IBM's grand design for the future of commercial information systems. The original initiative came from IBM's Corporate Management Committee (CMC), which is chaired by John Akers, IBM's chief executive. At the beginning of 1986, the CMC set up a special task force to draw up a plan for bringing the company's different operating environments closer together. The result was SAA.

Systems Applications Architecture has four main elements

SAA has four main elements — common programming interface, common user access, common communications support, and common applications. The SAA architecture is illustrated in Figure 3.5. Three of these elements are based firmly on existing IBM products and standards. Common communications support provides a standards framework for a subset of IBM's existing communications products; much of the common programming interface is defined in terms of existing products or IBM (or international) standards; the common applications concept extends an idea first applied to IBM's office systems products. Only the common user access element is genuinely new, although that part of the common programming interface which relates to it is also new.

It is important to realise that SAA is not a set of product specifications; it is a collection of selected software interfaces, conventions, and protocols that will provide the framework for developing consistent applications that can operate in the three main IBM computing environments — System/370 and its successors (TSO/E running under MVS/XA, and CMS running under VM), AS/400 (running under the OS/400 operating system), and PS/2 (running under the OS/2 Extended Edition operating system).

© Butler Cox & Partners Limited 1989

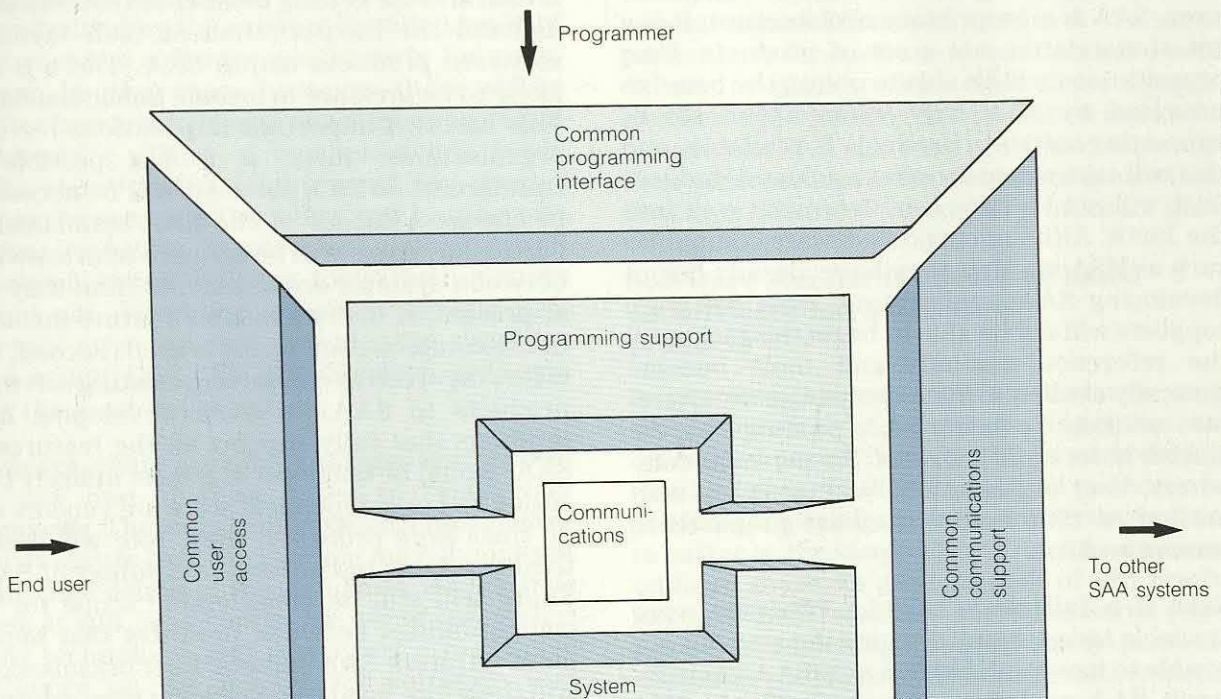
Provided that they use no features outside the interface definitions, applications written to comply with the common programming interface will be able to run in any SAA environment. As a consequence, applications development staff will also be more easily able to transfer between computing environments, as the business demands. Applications that use common communications support will be able to interwork with one another, even if they are running in different hardware environments.

Users will more easily be able to learn how to use a new application that implements the common user access standards, making it easier to move staff from one department to another. In short, SAA provides all the benefits of software standards identified at the start of this chapter, except those associated with inter-organisational communications (which is largely outside the scope of an initiative from a single vendor).

Systems Applications Architecture has implications for user organisations

The original IBM personal computer rapidly became the *de facto* standard for business

Figure 3.5 SAA is not a set of products; it is a collection of conventions, interfaces, and protocols



(Source: IBM)

© Butler Cox & Partners Limited 1989

microcomputers. In time, the PS/2 running SAA applications will do the same, and this will mean that applications written for one vendor's personal computer will also be able to run on equipment from other vendors. Thus, in the personal-computer marketplace, SAA looks set to provide genuine applications portability. It is not yet clear whether SAA can also do this in the minicomputer and mainframe market-places. As we explained earlier in this chapter, open standards, particularly Unix-based standards, are strong contenders to achieve this for minicomputers, and possibly for mainframes as well.

Nevertheless, SAA is a significant development, not just for IBM, but for the whole of the IT industry, and it cannot be ignored by either users or suppliers. Like SNA 15 years ago, however, SAA is a proprietary architecture; it is a set of standards, not a set of products. User organisations will be able to obtain the benefits promised by SAA only when a full set of supporting software products is available, and this will take several years to achieve. Indeed, SAA will not be fully complete until well into the 1990s. Although major software companies such as MSA say that they have already begun developing SAA applications, most software suppliers will not be able to begin this task until the reference manuals and tools become generally available. Such companies are therefore unlikely to bring SAA products to the market before 1991, except for personal computers. User organisations will probably wait until after 1991 before they are prepared to commit to SAA.

Even if a full range of SAA products were available today, many organisations would not be able to move quickly to a new SAA environment. SAA excludes several significant IBM hardware ranges and standards that are in widespread use today, and because of these investments, user organisations will need to migrate to SAA over an extended period of time. Thus, many organisations will continue to be dependent on SAA 'orphans' for many years, so that SAA will not achieve, for IBM users, the degree of consistency and simplicity that is already enjoyed by, for instance, DEC VMS users.

In the short term, the main benefits of SAA for user organisations will be derived from the common user access standards. Until now, user-interface standards have been a significant gap in the standards field. IBM's Presentation Manager for the OS/2 operating system supports SAA's common user access. The standards and conventions of this element of SAA are designed to make applications look and feel the same, irrespective of the hardware on which they are running. This can be achieved only if applications conform to the same user-interface standards and conventions, or if front-end workstation programs are written to provide common user access interfaces. This latter approach can be used for applications running in non-SAA environments such as CICS, IMS, and AIX.

In general, SAA standards are already very broad, and are getting broader as IBM customers demand the incorporation of their favourite software products within SAA. There is also likely to be pressure to include public standards. This has two important implications for user organisations. First, it is not possible to standardise on SAA *per se*; it will be necessary to choose a subset of the SAA standards. In particular, users will be no more able to switch between systems environments than they are at present, if every available feature included in the common user access is used. Second, the effort involved in converting existing software products to SAA, or even developing new products that fully support all the features of SAA, would be immense. It is most unlikely that either IBM or independent software vendors will do this. More probably, they, too, will design their products to conform to a subset of SAA. There will still be considerable scope for incompatibilities between products that implement different SAA subsets. User organisations will therefore need to investigate very carefully exactly what is meant by a software vendor who claims SAA compatibility, and whether the product fits in with the organisation's own software standards.

Organisations will need to choose a subset of an appropriate family of standards

An organisation's software standards need to cover all the components of the software infrastructure — communications, database,

© Butler Cox & Partners Limited 1989

programming, user interface, and so on. Unfortunately, it is not possible to choose standards for each of the components independently because application systems will make use of each of the components. It is therefore necessary to choose a family of standards. The earlier sections of this chapter have shown that there are a bewildering number and variety of standards families, which have evolved, and are continuing to evolve, in several different ways. Some, such as SNA (and more latterly, SAA), are proprietary, being developed and promoted by hardware vendors. Others, such as Posix, are open standards that have evolved from successful products. Yet others are created by standards-making bodies.

In choosing which family of standards to adopt, organisations will usually find that their choice is constrained by the existing installed bases of hardware and software, particularly for mainframes. In other cases, however, there will be greater freedom to choose the most appropriate architecture. Thus, a business-needs study might show that a requirement for greater responsiveness and flexibility can best be met by implementing a relational database and advanced system building tools. Such a study would not determine which of several proprietary environments provided by suppliers such as Bull, DEC, IBM, ICL, Siemens, or Unisys was the most appropriate.

However, there are disadvantages with choosing any one of these types of families of standards. The main problems with proprietary standards are that the vendors may choose not to disclose details of the standards, may change them at will, may charge for use of the underlying technology, and may exert a restrictive influence on the market for software that conforms to the standards. These problems may persist even after control of the standard has passed to an independent standards body, if the vendor has a dominant position in the relevant part of the market.

The major drawback of standards developed by standards-making bodies is the slow pace of development, both of the standards themselves and of software products that conform to them. The software-supply industry is still hesitant to commit to open-standards products, even though the standards may be more balanced and more comprehensive than proprietary standards. The all-embracing nature of these standards (and of proprietary architectures like SAA) means that user organisations must select a subset of the facilities relevant to their own needs. Unrestricted use of all the facilities would largely negate the benefits of software standards, because the large number of options and the flexibility provided by the standards would make it impossible to ensure that applications could interwork, that user interfaces were consistent, and that development staff could move freely between different hardware environments. Moreover, product support is available only for certain combinations of the standards defined by the overall family.

The need to select a subset of the facilities is already evident in the OSI functional profiles defined by the UK and US governments. Commercial organisations will need to do the same, particularly for open standards that have been defined by starting with a clean sheet of paper. Neither will choosing an all-embracing architecture like SAA solve the problem completely. User organisations will still need to create their own equivalents of functional profiles for SAA.

Software standards should be based on software-infrastructure products

In practice, however, open standards, even those created by standards-making bodies, are often developed from concepts embodied in existing products. Standards based on successful products, especially *de facto* standards, tend not to suffer to the same extent from the problems outlined above. (A good example of a successful *de facto* 'open' standard is the Postscript page-description language used by many personal-computer packages to enable the data from the package to be passed to a laser printer.)

We recommend, therefore, that once a family of standards has been chosen, Foundation members base their software standards on the products in their software infrastructure. This would mean, for example, standardising on a particular database management system, rather than standardising on SQL and then allowing a free choice of database management system.

For most purposes, the most effective types of standards are those that are encapsulated in the products that make up the infrastructure.

© Butler Cox & Partners Limited 1989

Unlike standards that are defined by means of formal specifications, product-based standards are clearly defined by the way the product behaves. In addition, it is much easier to test whether new software conforms to a product-based standard: if it cannot interwork with the main product that defines the standard, it does not meet the standard.

In deciding which products to include in the software infrastructure, and hence the product-based standards that will be adopted, the needs of the business should be the paramount concern. Thus, an oil company has made a particular distribution package a central element of its software infrastructure, and this has required it to adopt IBM technical standards for a wide range of data processing and office systems within the company. This particular company happens to have chosen to base its software infrastructure on a product that requires a proprietary architecture. While we believe that such infrastructure products should form the basis for selecting software standards, there is increasing scope for these products to be based on open, rather than on proprietary standards.

User organisations need to specify their own rules for using the standards

Choosing a subset of an appropriate family of standards is not sufficient, however. It will still be necessary to provide detailed house rules describing precisely how the chosen products and facilities will be used. Allowing development staff free reign in the way they use a programming language, or construct the user interface for an application, is a recipe for disaster. For example, detailed rules and guidelines are required for the programming style that will be used, and for the conventions to be used for naming data items and procedures. Most organisations, of course, already have standards manuals that cover these aspects of software standards.

Similar, though less familiar, problems arise in setting user-interface standards. User-interface packages, such as DECWindows, Microsoft's OS/2 Presentation Manager, and the Macintosh Toolbox, allow development staff considerable freedom in designing both screen layouts and the dialogue between the application and the user. Although suppliers of such packages produce useful style guides, they are unlikely to be sufficiently specific for a particular organisation. User organisations will therefore need to create their own in-house style guides that define the user interface in specific terms.

Although a standards policy is a necessary and essential component of a software strategy, it is not sufficient by itself. The strategy must also specify the policy for acquiring new software. We address the issues associated with the software-procurement policy in the next chapter.

© Butler Cox & Partners Limited 1989

Chapter 4

Procuring new software

In this chapter, we discuss the third element of a software strategy — the policy for procuring new software. We have deliberately used the word 'procuring' to indicate that there are several options for obtaining software. Usually, the lower-level components of the software infrastructure, such as language compilers, database management systems, and communications software will be software packages purchased from hardware or software suppliers. Some smaller components, typically 'bridgeware' (special software used to link applications) or conversion software, may need to be written specially, but these components should form only a very small proportion of the total infrastructure. However, there are a greater number of options for procuring applications software.

For most organisations, the choice is usually between using a package or developing a bespoke application. We provide guidelines on how to decide whether a packaged solution or bespoke development is the best option. In a surprisingly high number of cases, packages will be preferable to bespoke development if business benefits, rather than users' requirements, are the main criterion. The chapter concludes by showing how certain types of software-procurement decisions can be decentralised, and how an appropriate software infrastructure can allow user departments to procure their own applications by constructing them themselves.

There are four main options for procuring applications software

The four main options for procuring applications software are:

— To use an application package.

— To develop bespoke software in-house.

— To employ a software house or a bureau to develop the application.

— To set up an industry consortium to develop applications software that will be used by all members of the consortium.

Often, the decision about which option to choose is made automatically, based on the policy of the organisation, rather than by formally evaluating each option. For example, some organisations will always develop their own applications, whereas others have a policy of first looking to see if there is a package that meets the requirements.

Schering, the worldwide pharmaceutical company with its headquarters in Germany, has adopted an interesting variation of the application package option. Having failed to find a suitable package for a particular application requirement, Schering identified a US software house that it felt would be well placed to develop such a package and sell it. The company persuaded the software house to fund some of the development costs of a package that met Schering's own requirements. So far, several dozen copies of the package have been sold. This experience has led Schering to recognise the value of the company's knowledge to software houses, and it has decided to use this approach wherever it is practical. It points out, however, that the risks are higher than for buying established packages, and that a very close working relationship must be maintained with the software house throughout the development of the package.

The consortium option for developing application software has been used successfully by public-sector organisations that have similar (maybe identical) requirements. For them, the

© Butler Cox & Partners Limited 1989

attraction of this option is the ability to share development costs between the members of the consortium. In the private sector, however, where organisations in the same business sector are likely to be competitors, consortium developments have not been so successful. For example, an organisation called London Clear was established to provide an electronic-clearing service for the London money markets. Unfortunately, when the costs of developing the specialised applications software began to escalate during the early system-specification stages, the consortium members withdrew their funding, and London Clear collapsed. Strong leadership, tight project management, and full commitment from the partners are needed to make a consortium project successful.

For most organisations, however, the main choice for procuring new application software is either to use a package or to develop a bespoke application.

Benefits, not requirements, should be the basis for software procurement

In deciding whether to use a package or to develop a bespoke application, it is important to make a clear distinction between the essential and non-essential application requirements. Unless this is done, the bespoke-development option will invariably be preferred, and opportunities for using packaged software will be missed. Most organisations start a requirements-definition exercise by carrying out a detailed survey to find out what users would ideally like. Much less attention is devoted to determining the *business benefits* that the organisation expects to gain from the new application. This approach almost inevitably leads to the conclusion that there are no packages that meet all of the users' requirements. As a consequence, the systems department, under pressure from the users, falls into the trap of trying to develop a comprehensive bespoke system.

More often than not, however, there will be an application package that meets most (if not all) of the *essential* requirements and provides most of the business benefits. Moreover, the additional business benefits (in terms of increased profitability, reduced costs, and so on) provided by a bespoke system are unlikely to justify the additional costs and time required to develop the system. Figure 4.1 describes the experience of one organisation that abandoned a major bespoke development project in favour of a package because it realised that the development costs could not be justified in business terms. We believe that paying greater attention to the business benefits obtained from computer applications would mean that, in many cases, application packages would be seen to be a better investment than bespoke in-house developments.

Although most organisations will analyse carefully the costs and benefits of purchasing new plant, opening new retail outlets, or launching new products, very few will conduct the same analysis for software investments. Figure 4.2 shows how the Net Present Value (NPV) technique can be used to analyse and compare the return on investment from two options for implementing a new application. The idea behind NPV is that future benefit values (increased profits, for example) are worth less than the same benefit obtained today. The difference is measured by considering the 'rate

Figure 4.1 Application packages can meet most essential business requirements

A fund-management organisation

This fund-management organisation used to obtain all its IT services (apart from minor PC-based systems and direct input from financial-information services) from a bureau. The applications were mainly batch-based systems. The organisation realised that in view of the changes that were likely to occur as a result of the deregulation of London's financial markets in October 1987, its survival would be in jeopardy if it did not have in-house online systems to enable fund managers to react more quickly to changes in the market, and to meet new regulatory reporting requirements. A software house was commissioned to develop new bespoke systems from scratch.

After a year of development work, it became apparent that progress was much too slow and that the final cost would be exceptionally high. With the deregulation deadline approaching, the only other choice was to see if there was a package that could meet the requirements. There was considerable resistance to this approach from one principal user who wanted an all-embracing bespoke system that could handle every type of financial instrument automatically. However, the finance director realised that the cost of such a system was out of line with the fees that the organisation could expect to earn over the next five years. A package capable of meeting more than 80 per cent of the requirements was identified, and the recommendation to buy this package was accepted.

© Butler Cox & Partners Limited 1989

of return' that an organisation might expect to achieve by investing the money in a different way. For example, $1 million deposited in the bank at an annual interest rate of 10 per cent will grow to $1.1 million after one year. Thus, at a rate of return of 10 per cent, $1 million obtained in *one* year's time would be worth $1 million divided by 1.1, or $909,091, today. One million dollars obtained in *two* years' time would be worth still less today. Thus, an all-embracing comprehensive bespoke development that takes a long time to implement (and to start producing benefits) may have a lower NPV than a package-based solution that can be implemented immediately, even though the package does not meet all the requirements.

The example NPV calculation in Figure 4.2 shows that a package is by far the better investment, even though the benefit value it

Figure 4.2 The Net Present Value technique can be used to compare two options for implementing a new application

An organisation has a choice of buying a package which, after some tailoring, will meet most, but not all, of its requirements, or developing a bespoke system from scratch. The package will cost $125,000 to purchase and a further $175,000 to tailor. It will take one year to implement, and at present values, will then produce a net benefit of $400,000 a year. The bespoke system will cost $1.5 million, spread equally over three years. Once it is implemented, it will produce a net benefit of $600,000 a year.

Assuming a return on investment of 25 per cent, the package approach produces a cumulative benefit after two years, whereas the bespoke system does not produce a cumulative benefit until the eleventh year (see the table below). By the eleventh year, the package approach will have produced a cumulative benefit of more than £1.1 million, even though its annual benefit is one-third less than that of the bespoke system.

		Package			Bespoke development		
Year	Present value factor at 25%	Benefit or (cost)$	Net present value of benefit or (cost)$	Cumulative benefit or (cost)$	Benefit or (cost)$	Net present value of benefit or (cost)$	Cumulative benefit or (cost)$
1	1.000	(300,000)	(300,000)	(300,000)	(500,000)	(500,000)	(500,000)
2	0.800	400,000	320,000	20,000	(500,000)	(400,000)	(900,000)
3	0.640	400,000	256,000	276,000	(500,000)	(320,000)	(1,220,000)
4	0.512	400,000	204,000	480,000	600,000	307,200	(912,800)
5	0.410	400,000	164,000	644,800	600,000	246,000	(666,800)
6	0.328	400,000	131,200	776,000	600,000	196,800	(470,000)
7	0.262	400,000	104,800	880,000	600,000	157,200	(312,800)
8	0.210	400,000	84,000	964,000	600,000	126,000	(186,800)
9	0.168	400,000	67,200	1,032,000	600,000	100,800	(86,000)
10	0.134	400,000	53,600	1,085,000	600,000	80,400	(5,600)
11	0.107	400,000	42,800	1,128,400	600,000	64,200	58,600

Net present values are calculated by multiplying the expected benefit or cost by the present value factor. The factor for year $n+1 = 1 \div (1+r)^n$, where r is the expected rate of return on investment. In this example, r is assumed to be 0.25.

Note that the above example is highly simplified because no account is taken of the fact that costs and payments are likely to be spread throughout the year, rather than accounted for once, at the end of the year. Nor is any variation in maintenance costs after implementation taken into account, and the example is based on the assumption that subsequent benefits are net of these costs.

© Butler Cox & Partners Limited 1989

provides each year is only two-thirds of the benefit value provided by a bespoke system. The reason for this is that, in the example, the package is implemented two years earlier and costs substantially less to implement. Thus, instead of using a comprehensive study of requirements as the basis for deciding which option to pursue for procuring the application software, we believe that it makes more sense to concentrate on the business benefits that will result from the application. The following procedure should be adopted:

— Identify, and as far as possible, quantify all the benefits expected from the application.

— Identify the basic functions that it is absolutely essential for the application to perform.

— Determine which benefits are achieved by these essential functions.

— Determine the additional costs of increasing the functionality to achieve the remaining benefits.

— Use the NPV technique to analyse the rate of return expected from these additional investments.

This procedure concentrates users' attention on the full implications of insisting that the system includes special or expensive requirements — in other words, the type of requirements that may rule out the use of packages. Some of the benefits may not, of course, be quantifiable, and it may be worth going ahead with a bespoke application even if the NPV calculations show a net cost. However, by explicitly listing the unquantifiable benefits and calculating the *cost* of achieving them, it is much easier for user managers to make informed decisions. A similar approach to that shown in Figure 4.2 can be used to compare different methods of implementation.

Packages will usually be a better investment than bespoke development

When the NPV technique is used to compare a package with bespoke development, the package will nearly always prove to be a better investment, provided, of course, that it meets the essential requirements. This is not just because it is less expensive, but also because it can be implemented more quickly so that the benefits can be achieved earlier.

Two arguments have traditionally been used against the use of packages. The first is their lack of flexibility, which has made systems departments reluctant to advocate the use of packages. If new requirements emerge after a package has been implemented, and the package is unable to handle them, the systems department can be accused of a lack of foresight. The second argument is that it is not possible to achieve a competitive advantage by using a package because the same capabilities will be available to all users of the package.

The advent of 'soft' packages has weakened both of these arguments. A soft package is one that can be tailored to meet an organisation's specific application requirements. Such packages may include report-generation and screen-formatting facilities, and a fourth-generation language. Furthermore, a soft package is usually based on a well known database management system, which means that the database and associated system software on which the package is based can be used to extend the scope of the package or to develop interfaces to other software. Figure 4.3 describes a soft package produced by SAP, a German software house.

Several Foundation members told us that, because of the increasing availability of soft packages, they now intend to use packages much more and they believe that products such as SAP will have a major impact on the development of the package market. The description in Figure 4.3 shows that the SAP package provides considerable flexibility to meet new requirements — more, in fact, than would usually be achieved with a bespoke application. Each organisation implementing SAP selects those parts of the package that suit its own requirements, so that there is considerable scope for using a product like SAP to achieve a competitive advantage.

The one drawback of soft packages is, of course, the cost of implementing them. For an extremely flexible package that provides a wide variety of tailoring options, the implementation cost may be several times more than the purchase price of the package. Nevertheless, the

© Butler Cox & Partners Limited 1989

total cost will still be substantially less than the cost of developing an equivalent bespoke application.

Some software suppliers have recognised the need for tools that will help user organisations to tailor soft packages to their specific requirements. For example, Cullinet and SAP both provide a personal-computer-based implementation workbench to assist in this task. This type of product provides facilities for storing project plans and monitoring progress against the plans, for producing online documentation (using graphics and text) that shows a top-down representation of the package functions selected, for training users, based on a question-and-answer format, and for converting existing data so that it can be used with the package.

Once a package has been tailored and implemented, customisation workbenches can

Figure 4.3 System R/2 is a highly integrated, yet extremely flexible, 'soft' package

SAP is a software house based in Germany, and with several offices in other countries. It was formed in 1972, now employs more than 900 staff, and has a turnover of about DM240 million ($135 million). Its main product is a highly integrated, yet extremely flexible, application package, called System R/2. This product contains many of the features that we expect to see in a 'soft' package.

System R/2 was designed to enable user organisations to select and combine the applications and functions they need at several different levels. SAP considers this facility to be unique to System R/2. The functions provided cover the following main application areas:

— Sales.
— Production control.
— Materials requirements planning.
— Planned maintenance.
— Personnel.
— Project planning and accounting.
— Cost accounting.
— Financial accounting.
— Assets register management.

These application areas can be used individually or they can be integrated. They are grouped around a core system that contains operating system interfaces, modules for accessing the data communications facilities, database and table-handling modules, a fourth-generation language, and other routines common to all applications. The core system is supplied regardless of which combination of application areas is implemented.

Within each application area, user organisations can choose which individual functions to use. Within the materials-planning application, for example, users can choose to use only a very limited range of functions to create a basic inventory-control system, or they can use all the functions necessary to create a complete materials-planning system that is integrated with a production-control system (including bill-of-materials processing, purchasing, and so on). In addition, many functions provide options — different ways of calculating costs, for example. Some of the options can be defined by the user. Other functions permit 'what-if' simulations.

Additional tailoring facilities are provided by allowing many of the application details to be defined as the package is implemented. This is achieved through:

— Extensive use of tables and table-driven processing (for defining depreciation rules, planned maintenance, the execution sequence in which programs will be run, and so forth).
— Use of parameters to tailor individual functions.
— An adaptable, user-dependent menu system.
— Modifiable screen fields (controlled through the data dictionary).
— A menu-driven query system.
— User definition of calculation routines.
— User-definable report layouts to fit in with preprinted stationery, for example.
— Use of a database and a data dictionary, allowing users to specify the format of data items.

The system has also been structured so that it can be extended easily by user organisations, and linked to other applications. This is achieved by using the following facilities:

— A database that can be extended by user-defined fields.
— A user-accessible data dictionary that can store descriptions of user-defined data.
— Input and output interfaces to other application systems.
— Interfaces to computer-aided design and factory data-collection systems.
— Source code that is available to user organisations.
— A fourth-generation language (ABAP).
— The ability for user organisations to add their own code, either using Cobol or ABAP.
— The ability to access user-defined DB2 tables, as well as System R/2 internal data.
— The ability to add user-defined reports to the standard reports.

With this amount of flexibility, implementing the package is not a trivial task; it requires the user organisation to carry out a significant amount of analysis to define the functions needed, and the way in which they should work. The implementors also need to have a good understanding of the capabilities of the package. SAP therefore provides consulting and training services to help its customers to make the best use of the package.

© Butler Cox & Partners Limited 1989

be used by the package users to make changes to the application. Cullinet also provides this type of product, as does Hewlett-Packard, whose Customizer can be used in this way. Such workbenches allow package users to switch smoothly from operational to development mode so that simple changes can be made to the way in which the package operates.

SAP's package is an example of the type of facilities that will increasingly be provided by soft packages. Other packages provide less flexibility, however, and this may mean that a package that can meet an organisation's essential requirements may not fit in with its current working methods. When evaluating a package, Foundation members should therefore consider the possibility of adapting working practices to fit in with the package, rather than vice versa. Doing this is likely to meet resistance from users, but will be much easier to achieve if user departments have to pay directly for their use of IT facilities. One organisation that recently introduced a recharging system found that user departments were suddenly very keen to reduce costs and to cooperate in adapting their working practices to fit in with the most cost-efficient application-software solution.

In selecting a package, either for core or non-core applications, care should be taken to ensure that the product conforms with the chosen software infrastructure. Compatibility with other infrastructure components is just as important as the ability to provide the basis for further developments. During the research, we met a German manufacturing company that had decided to install a wide-ranging production-control package covering a large part of its business operations. The company had to write a large number of bridging programs to interface the package to its existing applications because the package did not interface fully with the relational database management system already installed as part of the software infrastructure. A similar situation had arisen with other packages, with the result that this company now has four different and incompatible infrastructures — a central mainframe environment, a distributed systems environment, and separate infrastructures for warehouse systems and production control. Although it is clear that more than one infrastructure may be needed, this company is finding it difficult and expensive to manage four.

A package that is used for a core application may itself become part of the software infrastructure. When Rumbelows (a UK electrical retail chain) decided to replace its 16-year-old applications that had become unmaintainable, it first looked to see if there were any packages available that could meet its needs. As a result, a credit-management package and a package for automating Rumbelows' back-office procedures now form the basis for software infrastructures for the two halves of its business (credit and retail). The credit-management package has enabled the credit function (now known as Trinity House Finance) to provide similar services both for other parts of Rumbelows' parent group and for third parties.

Responsibilities should be allocated for software procurement

In a large organisation, it is unusual for all software to be developed or procured centrally, and an important prerequisite to developing a software strategy is to decide on the most appropriate organisational level for making various types of decisions about software. For example, many organisations need to consider variations in requirements between different parts of their organisations, and variations in the availability of products (and the quality of support for them) in different countries. Business departments also now expect to be able to buy software for their own personal computers. The same type of trend is evident for departmental minicomputers and even for mainframe software.

In addition, the management style of many organisations encourages the devolution to business units of the responsibility for the systems function and this means that some decisions about software procurement will be made on a decentralised basis. It is therefore generally impractical for a single software-procurement policy to be imposed by a central systems function on the rest of the business. A critical issue in developing a software strategy, therefore, is to decide which software-procurement decisions should be made centrally, which should be left to local discretion, and what restrictions, if any, should be placed upon the local systems function.

BUTLER COX FOUNDATION
© Butler Cox & Partners Limited 1989

Centralising all procurement decisions can cause difficulties

If the business itself is decentralised, resisting the pressure to decentralise the systems function can present major problems. In our research, we interviewed systems staff from both the parent company and a national company of a major European manufacturing group. The business strategy for the group as a whole was to become much more decentralised, giving more autonomy both to national companies and to individual business units. However, the central systems department still maintained complete control over all the major applications software used throughout the group.

The central systems staff believed that major benefits were gained by making all decisions about software at the corporate level. The systems staff from the national company were not satisfied with this arrangement, however. Their users, with their newly obtained autonomous responsibility, were complaining bitterly about the inadequacy of the standard corporate software to meet their business needs, and the slow response of the central department to requests for changes. At the same time, the corporate systems staff were complaining about the number of requests for changes and the lack of support for standard applications. "We are just the 'piggy in the middle' and there is not much we can do about it", commented an interviewee from the national systems function.

Sometimes, it is preferable to decentralise software decisions

In contrast, a Dutch company, Wavin, has adopted a novel approach to decentralising software decisions. The corporate systems manager wanted to achieve a degree of standardisation for certain types of software, but without alienating the national systems managers. He achieved this by delegating corporate responsibility for each of these types of software to an individual national systems manager. Thus, one manager was given the responsibility for investigating electronic mail products and for defining corporate standards for electronic mail.

In deciding which software decisions should be made centrally, and which should be devolved, it is crucial to take into account the nature of the organisation's style. A spectrum of possibilities exists, from full control of the operating companies by head office at one end, to full decentralisation at the other. The organisation of the systems department will itself take account of the organisation's management style, ranging from fully centralised at one end of the spectrum, to devolved, with only a vestigial central function, at the other.

Figure 4.4 shows which types of software decisions should be centralised and which should be decentralised in systems departments organised in each of four ways: centralised, coordinating, guiding, and devolved. For simplicity, the figure identifies four types of software decision: those concerned with standards and policies, those concerned with the software infrastructure (including core applications), those concerned with software procurement, and those concerned with non-core applications. (The concept of core and non-core applications was explained in Chapter 2.)

Where the systems department is centralised, all software decisions are taken centrally. Where it is devolved, all software decisions, except those to do with standards and policies,

Figure 4.4 Decisions on software should be centralised or decentralised, depending on organisational style

Organisational style	Standards and policies	Software infrastructure (including core applications)	Software procurement	Non-core applications
Centralised	C	C	C	C
Coordinating	C	C	C	D
Guiding	C	C	D	D
Devolved	C	D	D	D

C = centralised
D = decentralised

© Butler Cox & Partners Limited 1989

should be taken at the operating-company level. Between these two extremes, decision-making should be divided between the centre and the operating-company level, as shown by Figure 4.4.

Where the responsibility for software procurement is devolved to the local level, there may still be a need for some central control of the way in which the responsibility is discharged. This might take the form of a checklist of items to be considered when a product is evaluated, or the level of financial return required to justify a departure from corporate guidelines for software standards.

User departments should be encouraged to construct more applications themselves

In the long term, user departments will therefore be responsible for constructing the majority of their non-core applications. Increasingly, younger staff in organisations will have received some training in computing during their formal education or as part of their technical training. As a consequence, user departments will become more self-sufficient in their use of IT and in the construction of applications. In the short to medium term, however, the systems department will continue to be responsible for constructing many of the non-core applications. In particular, high-volume, online transaction-processing systems require specialist skills, and in many areas, system-software products have not yet reached the stage where users can (or want to) use them.

Nevertheless, systems departments cannot ignore the trend towards user departments wanting, and being able, to take on more responsibility for constructing their own applications. If they do ignore this trend, systems departments will be perceived as constraining business development in the interests of protecting their own roles. Software products, whether application packages or lower-level infrastructure components, should therefore be chosen with this trend in mind. This means that the chosen products should provide facilities such as the ability to generate customised reports, to add new transaction types, and to define data and tables that allow the product to be customised. Furthermore, the user interface to these facilities should be simple and well-documented, and should insulate the user from the technical features and eccentricities of the operating system and other lower-level infrastructure components. If necessary, the infrastructure should be extended to include skeleton functions and application templates that business staff can use as the basis for constructing applications. Figure 4.5 lists some examples of applications that are particularly suited to user development, given a suitable software infrastructure.

The systems department can help to promote the construction of applications by users by seconding or transferring systems staff to user departments. In the long term, however, organisations should recognise that experience in systems analysis and applications construction should form part of graduate-training and management-development programmes for business staff. These are skills that will, in the future, be needed in most user departments, particularly when the software infrastructure has developed to the point where users (who best understand their own needs) no longer need professional systems staff to construct the majority of their non-core applications.

The shift towards the construction of applications by users should be encouraged to develop in a gradual and controlled way. If systems departments resist this trend, users will be encouraged to 'go it alone' without adequate professional advice, standards, and controls. The systems department should therefore provide a consultancy service to help users

Figure 4.5 User departments can implement a variety of applications

Sales and marketing systems (based on the customer/product database).
Management information (extracted from corporate databases).
Personnel management.
Project control.
Communication with the sales force.
Electronic data interchange

BUTLER COX FOUNDATION
© Butler Cox & Partners Limited 1989

choose appropriate packages and construct their applications. It should also set standards and guidelines for the user community, and provide a quality-assurance function whose role is to assist users in conforming with the standards.

Many systems staff have a natural tendency to believe that, because of their lack of professional systems training, users are not capable of designing, constructing, and documenting software to the professional standards demanded by the systems department. This view is not borne out in practice by many of the engineering and actuarial departments that have been developing their own software for many years. When problems do occur, they are more often caused by lack of standards and guidelines than by lack of ability. The need for specialists to set corporate standards is not a new concept; indeed, in many organisations, professional staff in one department set standards that must be followed by other departments. For example, in most organisations, there are standards for handling cash, keeping adequate financial records, talking to the press, and so on. A professional accountant or lawyer may well be needed to set these standards, but professional training in the particular discipline is not needed to follow them.

In conclusion, systems departments must accept as a fact of life that business staff will increasingly have the skills, and the access to the tools that they need to construct more of their own applications. Systems departments should therefore start planning for and encouraging user involvement in software selection and construction. The first step is to ensure that the organisation has an appropriate software infrastructure that is backed up by professional standards for using it to construct new applications. This will ensure that the user community adopts good practices and does not repeat the mistakes made by the systems community during the past 20 or 30 years.

Report conclusion

In this report, we have shown that software strategy should be directed at achieving business goals, not at achieving technology-related goals specific to the systems department. As computer applications become more firmly embedded in the day-to-day operations of the business, it becomes more and more important to be able to adapt the software to reflect changes in the business environment and in business strategy. A software infrastructure is needed to enable systems to be adapted more quickly in response to new business needs.

Developments in software products have enabled more and more software functions to be included within a software infrastructure that forms the basis for constructing new applications. This, in turn, has reinforced the trend to devolving the responsibility for constructing applications to user departments, so that, in the future, the main applications-software role of the systems function will be to define and support the software infrastructure.

The standards will ensure that different applications use the same style of user interface, so that business users can move from one department to another and immediately feel comfortable with the applications used by the new department. At a different level, the standards will allow development staff to move freely between different development environments, providing greater flexibility in the use of this scarce resource. The need for standardisation arises particularly in those organisations that use different computer architectures, and in those that need to communicate with the outside world. Progress has been slow in developing open standards (although Unix-based developments are now showing great promise), and IBM's SAA will not provide all the answers. We have provided guidance on how Foundation members can identify and select the appropriate subset of those standards that are likely to succeed in the IT industry in general, or in their own industry.

The trend towards soft packages has meant that in many areas where, in the past, bespoke software would have been developed, a package is now the most cost-effective option. Formal evaluation of options for new developments should be carried out. Similarly, a formal

© Butler Cox & Partners Limited 1989

method is needed to determine the point at which ageing systems should be replaced.

We have stressed the need to focus software decisions on business needs and benefits. This has been said many times before, but few systems departments have embraced this idea fully, tending instead to concentrate on the technical issues. Unless this situation changes, user departments will increasingly take the initiative by developing their own systems without regard for the information needs of the organisation as a whole. By developing a stable and consistent software infrastructure, the systems department can retain control over those software assets that rightfully belong to the organisation as a whole, while allowing users the freedom they need to construct and manage their own applications.

BUTLER COX FOUNDATION
© Butler Cox & Partners Limited 1989

BUTLER COX FOUNDATION

Butler Cox
Butler Cox is an independent management consultancy and research organisation, specialising in the application of information technology within commerce, government, and industry. The company offers a wide range of services both to suppliers and users of this technology. The Butler Cox Foundation is a service operated by Butler Cox on behalf of subscribing members.

Objectives of the Foundation
The Butler Cox Foundation sets out to study on behalf of subscribing members the opportunities and possible threats arising from developments in the field of information systems.

The Foundation not only provides access to an extensive and coherent programme of continuous research, it also provides an opportunity for widespread exchange of experience and views between its members.

Membership of the Foundation
The majority of organisations participating in the Butler Cox Foundation are large organisations seeking to exploit to the full the most recent developments in information systems technology. An important minority of the membership is formed by suppliers of the technology. The membership is international, with participants from Australia, Belgium, France, Germany, Italy, the Netherlands, Sweden, Switzerland, the United Kingdom, and elsewhere.

The Foundation research programme
The research programme is planned jointly by Butler Cox and by the member organisations. Half of the research topics are selected by Butler Cox and half by preferences expressed by the membership. Each year a shortlist of topics is circulated for consideration by the members. Member organisations rank the topics according to their own requirements and as a result of this process, members' preferences are determined.

Before each research project starts there is a further opportunity for members to influence the direction of the research. A detailed description of the project defining its scope and the issues to be addressed is sent to all members for comment.

The report series
The Foundation publishes six reports each year. The reports are intended to be read primarily by senior and middle managers who are concerned with the planning of information systems. They are, however, written in a style that makes them suitable to be read both by line managers and functional managers. The reports concentrate on defining key management issues and on offering advice and guidance on how and when to address those issues.

Selected reports

8 Project Management
20 The Interface Between People and Equipment
24 Investment in Systems
25 System Development Methods
27 Developments in Videotex
28 User Experience with Data Networks
29 Implementing Office Systems
30 End-User Computing
31 A Director's Guide to Information Technology
32 Data Management
33 Managing Operational Computer Services
34 Strategic Systems Planning
35 Multifunction Equipment
36 Cost-effective Systems Development and Maintenance
37 Expert Systems
38 Selecting Local Network Facilities
39 Trends in Information Technology
40 Presenting Information to Managers
41 Managing the Human Aspects of Change
42 Value Added Network Services
43 Managing the Microcomputer in Business
44 Office Systems: Applications and Organisational Impact
45 Building Quality Systems
46 Network Architectures for Interconnecting Systems
47 The Effective Use of System Building Tools
48 Measuring the Performance of the Information Systems Function
49 Developing and Implementing a Systems Strategy
50 Unlocking the Corporate Data Resource
51 Threats to Computer Systems
52 Organising the Systems Department
53 Using Information Technology to Improve Decision Making
54 Integrated Networks
55 Planning the Corporate Data Centre
56 The Impact of Information Technology on Corporate Organisation Structure
57 Using System Development Methods
58 Senior Management IT Education
59 Electronic Data Interchange
60 Expert Systems in Business
61 Competitive-Edge Applications: Myths and Reality
62 Communications Infrastructure for Buildings
63 The Future of the Personal Workstation
64 Managing the Evolution of Corporate Databases
65 Network Management
66 Marketing the Systems Department
67 Computer-Aided Software Engineering (CASE)
68 Mobile Communications

Forthcoming reports

Electronic Document Management
Staffing the Systems Function
Managing Multivendor Systems
Future Information Technologies

Availability of reports
Members of the Butler Cox Foundation receive three copies of each report upon publication; additional copies and copies of earlier reports may be purchased by members from Butler Cox.

BUTLER COX FOUNDATION

© Butler Cox & Partners Limited 1989

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

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

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

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

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

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

Finland
TT-Innovation Oy
Meritullinkatu 33, SF-00170 Helsinki, Finland
☎ (90) 135 1533, Fax (90) 135 1091

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

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

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

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