Application Packages

BUTLER COX
P.E.P

PEP Paper 15, September 1990

BUTLER COX
P.E.P

Application Packages

PEP Paper 15, September 1990
by Paul Green

Paul Green

Paul Green is a consultant in Butler Cox's London office, where he specialises in systems strategy and management. Since joining Butler Cox, he has been involved in a wide variety of projects, including the formulation of an information systems strategy for a large oil company, the definition of a 'vision' for the future of office systems for a major corporation considering relocation of its head office, and the development of systems development standards for a public-sector client. He has also carried out several PEP assessments.

Prior to joining Butler Cox, Paul Green spent three years with Peat Marwick McLintock, where he was involved in a wide range of consulting assignments in the systems area. During this time, he gained significant experience of projects that involved application packages, selecting and implementing packages for clients in the transport, financial services, and manufacturing sectors. His early career was with Thorn EMI Datasolve's Consultancy Division, and with Amersham International.

He has a BSc and an MA, both in economics, and is an associate member of the Institute of Management Consultants.

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

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

BUTLER COX
P,E,P

Application Packages

PEP Paper 15, September 1990
by Paul Green

Contents

Chapter 1

Application packages are playing an increasingly important role in systems delivery

There are four generations of application packages

Application packages have been widely available for the last 25 years, and during this time, they have developed significantly. They have moved away from being systems with limited flexibility that could be used only to meet a very specific application requirement, to very flexible, 'soft' systems, which address wider business needs by offering fully integrated business solutions. This development has resulted in four identifiable generations of application packages, the characteristics of which are shown in Figure 1.1, overleaf. Each generation of package still exists; as the more modern parameterised and flexible products have come onto the market, basic packages and modular packages have not disappeared. This means that application packages can now offer organisations sophisticated and relatively inexpensive 'off-the-shelf' solutions.

Both the number of application packages and the number of suppliers of packages have continued to grow. Many thousands of application packages are now available, from simple products designed for a single user to run on a microcomputer, to sophisticated applications that can support thousands of users and process millions of transactions daily on the most powerful computers. Application packages range in cost from a few tens of pounds to over £1 million for modules of some sophisticated, large-scale products.

While there are several large and well known suppliers of application packages, such as Computer Associates, there are still opportunities for new suppliers to grow rapidly if they have a good product, and new suppliers continue to appear. QSP, a supplier of accounting packages, for example, has become a major force in its chosen marketplace over the last five years.

In the last 10 years, in particular, the use of application packages has grown dramatically. In 1980, only about one-third of organisations used application packages; today, almost all do. More significantly, however, organisations are using packages for a much wider range of applications. In the past, many organisations used only one or two application packages. Currently, while packages still represent less than 25 per cent of applications in two-thirds of organisations, they represent between 25 and 50 per cent of applications in a quarter of organisations. The proportion of organisations' systems portfolios represented by application packages is shown in Figure 1.2, on page 3.

The use of application packages continues to grow

Our research shows that their use will continue to increase. We spoke to several systems development managers of large organisations who had decided, as a matter of policy, to make greater use of application packages. Typically, they believe that, within three to five years, about 80 per cent of their information systems will be based on application packages. About 90 per cent

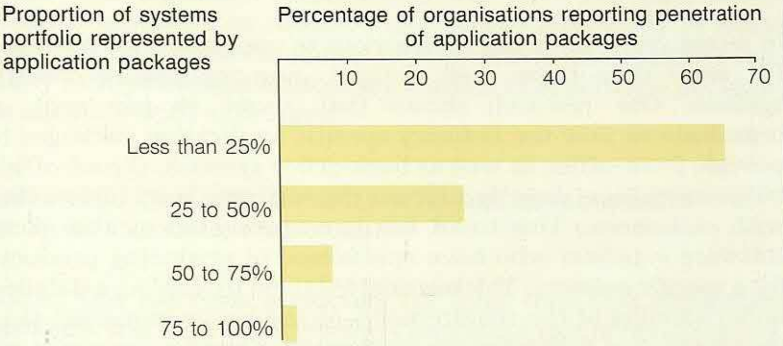
Figure 1.1 There are four generations of application packages

Generation	Package classification	Package characteristics
First	Basic package	Batch-based product. Inflexible, with few, if any, parameters. Scope for modification limited. Designed to process a specific business application. Limited reporting facilities.
Second	Modular package	Mainly batch-based product, with simple, online user interface. Flexibility is improved, with more parameters and discrete modules that can add functionality in predefined areas. Scope for modification is still limited but is enhanced, as changes can be made to modules. Predominantly a standalone application, but batch input and output can transfer information to and from other systems. Reporting facilities are still limited, but more options are available as modules provide more data.
Third	Parameterised package	Basic processing is still provided by a batch system, but a sophisticated online user interface is available. Flexibility is considerably improved, with many parameters enabling different areas of the package to be used in several ways. Modification is now easier through the setting of parameters and the use of package-specific tools. Integration with other applications is enhanced. Reporting facilities are more sophisticated, with content and format being controlled by parameters.
Fourth	Flexible package	All processing can be online. Batch processing is optional. Modern development tools are used to produce very flexible or 'soft' packages, with many parameters. Modification is easy, using parameters, package-specific tools, and standard modern development tools. The package is now part of a fully integrated software range that aims to provide a complete business solution. Sophisticated and flexible reporting facilities are available. Reports can be structured by package-specific tools and/or modern development tools.

of PEP members believe that their own use of packages will increase, and 25 per cent think that this increase will be quite dramatic.

In some quarters, however, there are still serious concerns about the use of application packages. Significant numbers of systems managers believe that no application package can meet their

Figure 1.2 Application packages play a significant role in organisations' portfolios of information systems



(Source: Survey of PEP members)

organisation's specific requirements, and there are other concerns relating to the perceived risk of a loss of control over systems, the inability to gain a competitive advantage if everyone has access to the same packages, and perceived difficulties with technical architectures and with systems integration.

Application packages should always be considered

As we show in this paper, none of these concerns is, in fact, a valid reason for dismissing application packages. We recommend that application packages should always be considered when a request for a new or enhanced computerised information system is received. The benefits of using application packages in preference to bespoke developments can be clearly demonstrated in terms of cost savings, time savings, skill savings, and guaranteed quality. Standard methods for selecting appropriate packages for particular applications and a standard approach to implementation will ensure that these benefits are realised.

For the purposes of this paper, we define an application package as *a commercially available set of one or more computer programs that is designed to create a complete business application.* Examples of application packages would therefore include an accounts payable or personnel package. Some so-called 'packages' fall outside our definition because they do not, by themselves, form a complete application. Thus, spreadsheet programs, word processing packages, fourth-generation languages, and computer-aided design systems are not application packages.

APPLICATION PACKAGES ARE BECOMING MORE SOPHISTICATED

Our research shows that PEP members are not only making more use of application packages, but are using them to provide a wider range of systems. Traditionally, application packages were used for back-office administrative tasks — that is, to provide information systems to support the departments that provide an internal service to the business, rather than an external service to the organisation's customers. A prime example of this is the widespread use of accounting packages, which have been successful because they provide a thorough and rigorous solution to processing standard types of information in a clearly defined way.

Packages are increasingly being used for front-office systems

In recent years, there has been a move to use application packages for more than these well defined, non-organisation-specific systems. Our research shows that about 40 per cent of organisations now use industry-specific application packages to provide front-office as well as back-office systems. (Front-office business systems directly support the business in its interaction with customers.) This trend has been promoted by the many software suppliers who have specialised in producing products for a specific industry. This has enabled them to build up a detailed understanding of the requirements of the business market that they serve, and facilitated the production of application packages that can deliver real benefits to the front-office operations in many organisations. Examples include application packages developed for the banking and insurance industries and for the health-care sector.

APPLICATION PACKAGES WILL BECOME EVEN MORE ATTRACTIVE IN THE FUTURE

Both business pressures (to make time and cost savings in a period of skill shortages), and technical developments will encourage systems development managers to make greater use of application packages.

BUSINESS PRESSURES WILL ENCOURAGE GREATER USE OF PACKAGES

Business pressures have continued to intensify, both in the private and public sectors. This has led to increased time pressures on systems development departments to introduce new systems at a time when many development managers already face large development backlogs. To overcome these problems, both development managers and users are increasingly turning towards packages, with the expectation that time and resource savings will be made.

Improved price/performance of hardware makes packages more attractive

The cost of undertaking bespoke system development has continued to rise sharply as skill shortages have led to salary inflation for development staff. At the same time, the price/performance of computer hardware has continued to improve. One of the traditional reservations that development managers have had about using application packages is the impact on machine resources of using inefficient code supplied as part of a package. The improved price/performance of hardware has meant that, even if application package code is inefficient, its effect is less marked.

Governments are also beginning explicitly to encourage the use of application packages. For example, in order to maintain the competitiveness of its industry, the Japanese government is encouraging both the development and use of application packages. The use of application packages in Japan is low by international standards, and in common with the rest of the

Packages are seen as a way of alleviating staff shortages

developed world, Japan faces a severe shortage of skilled systems staff. The government believes that making greater use of application packages will help to alleviate this shortage, and will mean that Japanese corporations will have the information systems necessary to continue to compete effectively in world markets. Our research among PEP members also suggests that many European organisations are turning to application packages to alleviate staff shortages.

ADVANCED TOOLS WILL FACILITATE THE DEVELOPMENT OF MORE FLEXIBLE PACKAGES

Butler Cox has been predicting for some time that application packages will form an increasingly important part of organisations' software infrastructures. (The software infrastructure is the portfolio of basic software that is used to support an organisation's applications.) We believe that the use of application packages will become more attractive because suppliers of packages are increasingly using advanced software development tools, such as CASE, fourth-generation languages, and so on, to develop their products.

At first sight, this may seem to be counter-intuitive because tools such as these will increase the effectiveness of in-house developments, and may therefore be seen as a threat to application packages. However, suppliers are beginning to use these tools to develop new types of packages that are more flexible than their predecessors. The possibility of using these flexible packages, which can be easily modified to meet requirements closely, will mean that this type of application package will be increasingly important in the future. An example is the Chameleon accounting system, supplied by Tetra. This product has been designed to offer a sound basic accounting system that can be easily modified to meet an organisation's specific needs. Such products offer what amounts to a bespoke application, at a similar price to an 'off-the-shelf' solution.

Flexible packages can be easily modified with advanced development tools

In the future, we believe that 'off-the-shelf' software components will play a greater role in applications development. Our recent research for the Butler Cox Foundation suggests that within the next five years, information systems will begin to be built in a new way, based on object-oriented methods. New systems will be built by combining different 'objects' from the software infrastructure, from in-house development, and from software objects that are purchased from the type of supplier who today provides application packages.

MANAGEMENT CONCERNS ABOUT APPLICATION PACKAGES CAN OFTEN BE ALLEVIATED

During our research, many organisations mentioned concerns they had about application packages that could militate against their use, even if there were clear, measurable benefits in using them. These issues can be grouped into five categories — concerns that no application package will meet the organisation's particular requirements, concerns that suppliers may not remain committed to supporting their products, a lack of opportunity to realise competitive advantage, technical architecture difficulties, and systems integration difficulties. Systems management can take actions to reduce all of these concerns.

APPLICATION PACKAGES CAN MEET A VARIETY OF REQUIREMENTS

Several systems development managers to whom we spoke during our research were concerned that, for at least some applications, there were no packages available that could meet their organisation's specific requirements. One organisation told us that although several packages had been evaluated, none could meet its particular needs, because its business was so different in nature from that of other organisations.

Most business areas have fundamental requirements that can be satisfied by a package

We believe that such arguments are no longer generally valid. We show in Chapter 2 that using an application package that does not fully meet all the requirements can still provide more benefits to an organisation than developing a bespoke application that meets all the requirements. Systems managers should seriously question whether their organisation's requirements really are different from those of others. In most business areas that lend themselves to computerisation, the fundamental requirements for new computer systems will be similar. The differences will usually be in the working procedures that have developed within the organisation over time; these may well be unique to particular organisations. We know of some that have successfully changed their traditional working procedures so that they can use packaged software.

THE RISK OF BECOMING VULNERABLE TO SUPPLIERS CAN BE MINIMISED

The risk is least for those who use IBM or Digital hardware

Several organisations we spoke to during our research were concerned that using application packages exposed them to unnecessary risk because they would no longer have control over the development of their computer applications. This risk occurs because suppliers of application packages introduce new releases of their software at regular intervals to maintain the competitiveness of their products. Problems can occur for users of application packages if new versions of the software do not meet their evolving requirements. Alternatively, the supplier may take a strategic decision to specialise in product development for a particular hardware and software infrastructure. If the user of the application package does not have the appropriate infrastructure, he may either have to change his policy or be faced with using outmoded software, which suppliers may refuse to support. Our research suggests that the risk is greatest for organisations that do not have one of the more widely used technical architectures — for example, one based on standard products from a leading supplier such as IBM or Digital.

To reduce this risk, organisations should include rigorous criteria in the package-selection process. Thus, if the continued development and support of the application package for a particular technical architecture is a criterion, it should be specified, investigated during selection, and if possible, included in the contract agreed with the supplier. At the very least, this will ensure that the user has a written commitment from the supplier to continue development and support of the product. If possible, organisations should insist that penalty clauses, for non-performance by the supplier, are written into their contracts.

At a time when many small suppliers of packages are entering the market with new products, there is a risk that they may not all survive, or that some may be taken over. If the supplier goes out of business, or is bought out, the future of the product is uncertain. Concerns are related to issues such as the future development and support of the product and the ownership of the software.

Package users should ensure that they can obtain the package source code if the supplier goes out of business

Arrangements should always be made for the source code of the package to be held in escrow so that it is available to the organisations using the package in the event of the supplier ceasing to trade. For complex or business-critical applications, the costs of holding source code in escrow are minimal compared with the implications of having to redevelop systems from scratch.

APPLICATION PACKAGES CAN BE USED TO GAIN A COMPETITIVE ADVANTAGE

Many organisations believe that using application packages prevents them from realising a competitive advantage from their information systems, because the same capabilities are available to all users of the package. This issue should be considered carefully when deciding whether or not to use an application package to provide a new information system. If the system is to support front-office operations, and must be significantly different from competitors' systems to provide a competitive edge, it may be appropriate to use bespoke development.

Soft packages can be used to provide a competitive advantage

However, with the growing availability of 'soft' or very flexible packages, competitive advantage may be derived from the ways in which organisations use application packages, and from the modifications that they make to them. One organisation that has used an application package in this way is a leading international airline, which has modified an accounting package to give it online access to up-to-date financial information for all its worldwide operations. This has enabled the company to improve its management of resources and finances significantly.

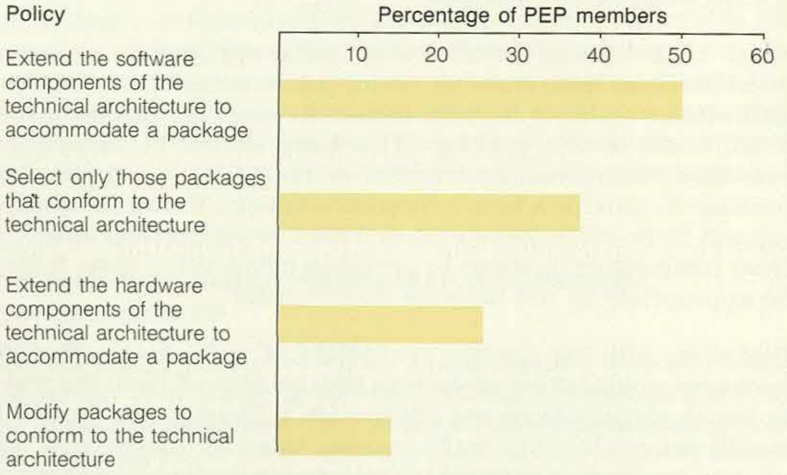
The rapid introduction of new front-office systems, based on application packages, can create an opportunity for achieving a competitive advantage. However, before an organisation can gain a competitive advantage in this way, it must usually have both an underlying technical and business infrastructure to enable it to take advantage of the application package facilities. A large financial services company, for example, realised that it could establish a new line of business, and needed to use a package to do so quickly, before competitors also became aware of the business opportunities. This organisation used an application package to establish a competitive advantage, although it could not have done so without its existing telecommunications infrastructure and branch network, which enabled the new business to be marketed effectively to customers.

TECHNICAL ARCHITECTURE DIFFICULTIES CAN BE OVERCOME

Most organisations take their technical architecture into consideration when selecting application packages. However, their policies vary widely, as indicated in Figure 1.3. The four policies shown in Figure 1.3 are not mutually exclusive. Some organisations combine policies. For example, they may consider extending both the hardware and the software components of the architecture. About 36 per cent of organisations consider only application packages that conform to the technical architecture, while about 14 per cent modify packages to conform to their architecture. About 50 per cent of organisations extend the software components of the architecture to accommodate packages; about 25 per cent of organisations extend the hardware components.

About half of organisations are prepared to extend their software infrastructure to accommodate packages

Figure 1.3 PEP members have different policies for ensuring that application packages conform with the technical architecture



(Source: Survey of PEP members)

Some organisations we spoke to during our research had selected their hardware and software infrastructure because of the availability of packages to run in their chosen environment. Thus, the choice of packages can play a significant part in shaping technical architectures.

Management will often be faced with a business decision that entails striking a balance between the business needs that can be satisfied by an application package, and the need to keep the technical architecture as simple as possible. Butler Cox has long been a champion of the concept of a coordinated organisation-wide technical architecture because of all the benefits that this can bring. Nevertheless, we do not believe that organisations should restrict themselves to selecting only application packages that conform to the architecture, or modifying packages to conform with the architecture.

By considering only conforming packages, organisations may unnecessarily limit their choice, and may fail to realise the full potential business benefits. Moreover, modifying application packages to conform to the technical architecture can be very costly, can lead to problems with the support of the package, and may prevent the easy use of future releases of the software. We do not recommend that organisations extend the hardware components of their architecture to accommodate a package as this often leads to compatibility problems and prevents information being used across systems as required. However, if suitable software interfaces exist to overcome this problem, extensions of the hardware components of the architecture may be appropriate. Extending the software components of the architecture usually causes fewer interface problems and is usually more acceptable.

Extending the hardware components of the technical architecture to accommodate a package often leads to problems

Any extension to the technical architecture to accommodate an application package must be considered carefully, so as to minimise complexity and cost, particularly in terms of developing staff skills in more than one architecture. When extending the technical architecture, PEP members must seek to minimise the problems of interfacing different systems, by ensuring that workable software interfaces are available.

DIFFICULTIES OF INTERFACING PACKAGES WITH OTHER SYSTEMS CAN BE SOLVED

Several organisations we spoke to reported complex problems when interfacing their application packages with other systems. Organisations should consider the requirements for interfaces carefully when deciding whether to use an application package, and when selecting a specific product. Short-term interface requirements must be clearly defined, and potential long-term requirements must be made explicit. PEP members should then seek commitment from suppliers that both immediate and longer-term requirements can be met. Commitment to provide the working interfaces required when the system is implemented should be included in the final contract signed with the supplier. Clauses should be included to penalise suppliers if specified interfaces are not available or do not work.

A commitment to provide the required interfaces should be written into the contract

PURPOSE AND STRUCTURE OF THE PAPER

Our research has revealed that application packages will play an increasingly important role within PEP members' applications portfolios. The purpose of the paper is to explain the benefits of using packages, and the best way of selecting and implementing them. We believe that this paper provides, for the first time, quantifiable evidence of the productivity benefits of using application packages. We show PEP members how they can achieve the best results from using application packages.

In Chapter 2, we describe the benefits of using application packages, particularly in terms of reduced costs and timescales for delivering new applications. We report the theoretical productivity improvements of using application packages, compared with undertaking bespoke system development. We also explain how the total business case for using an application package can be assessed.

In Chapter 3, we show how the selection of an application package can be successfully undertaken. We recommend that organisations use a formal method for selecting application packages, and describe the main components and characteristics of such a method. We also review commercially available methods for selecting application packages.

In Chapter 4, we show that it may be possible to modify a package so that it provides a better fit with business requirements. However, many PEP members have experienced problems modifying packages. We explain how these problems can be overcome and how different types of modification are best undertaken.

In Chapter 5, we explain the importance of application package implementation. Implementation must be successfully undertaken if the benefits are to be achieved. We describe the main differences between implementing an application package and a bespoke system development. In order to achieve the benefits of using application packages, we recommend that organisations use a method for implementing packages that is specific to packages and that takes advantage of their unique features.

RESEARCH SOURCES

At the beginning of the research programme for this paper, we circulated a detailed questionnaire to all PEP members. The aim of this questionnaire was to identify the main issues surrounding the selection, implementation, and use of application packages. There was a response rate of over 70 per cent to the questionnaire, and we have included a detailed analysis of the results as an appendix to this paper. We selected several organisations for more detailed investigation, either through personal or telephone interviews. Our aim has been to understand fully the benefits and problems associated with using application packages, and the selection and implementation procedures that are used.

We supplemented our research by talking to other organisations that either had extensive experience in the selection, implementation, and use of application packages, or that had taken a strategic position with regard to the use of application packages. We also reviewed other recent research and articles on the subject, and sought specialist opinion, where appropriate. In addition to this, we have drawn on the considerable experience of Butler Cox consultants, who have been involved in many application-package-related projects.

Chapter 2

The benefits of application packages are quantifiable and demonstrable

The benefits of using application packages are so great that organisations should *always* consider their use. When faced with a request for a new or enhanced computerised information system, systems development managers have traditionally responded by assessing the possibility of building the required systems using bespoke development before considering the use of application packages. This should no longer be the case. It should be assumed that any new system will be based on an application package, unless there are clear reasons that this cannot be so. This approach is justified by the scale of benefits that can be achieved from using application packages, whether such benefits are measured in terms of improved process productivity, time and cost savings, the overall benefits to the business, or better-quality systems.

New systems should be based on packages unless there are clear reasons for not doing so

PROCESS PRODUCTIVITY IS MUCH HIGHER FOR APPLICATION PACKAGE PROJECTS

In our survey, we asked PEP members to estimate the time and cost savings they had gained by using packages, rather than developing a bespoke system. We sought the same information in the interviews we held with PEP members. Using this information as a basis, we have estimated the benefits of using application packages in terms of the measure of process productivity, the Productivity Index (PI), and the measure of Manpower Buildup (MBI), which will be familiar to PEP members.

The average PI for all projects in the PEP database is 15, while the average MBI is 3, indicating a medium pace of manpower buildup. The savings from using application packages reported by PEP members suggest that the average PI for application package projects is about 17. This PI of 17 is the theoretical PI for application package projects, and is based on the assumption that the same project processes apply to the selection and implementation of an application package as those that apply to a bespoke system development. An increase of two points on the PI scale represents significant savings in terms of both cost and time.

Packages seem to improve the Productivity Index by two points

During our research, we were also able to obtain reliable data relating to 29 application package projects in 10 different organisations. We used the PADS software to calculate the theoretical PIs and MBIs for these projects. The average PI of these projects was 22. Not all application package projects had high PIs, however; three had PIs below the average, one of which had a PI of only 9. The average theoretical MBI of these projects is 3, the same as the overall PEP average, indicating that application package projects also tend to have a medium rate of manpower buildup.

The reason for the discrepancy between PEP members' reported productivity gains from using application packages and the data from the 29 application package projects is that many costly and time-consuming activities, such as selecting and buying a package, are not included in the information used to calculate the theoretical PIs for these projects.

Thus, while the use of application packages appears to increase PEP members' PIs by about two points, the potential improvements are even greater. Although it is unrealistic for most PEP members to expect the use of application packages to result in a seven-point increase in PIs (to an average of 22), greater improvements could be obtained by minimising both the cost and time required to select a package.

Larger productivity improvements can be obtained by minimising the cost and time required to select a package

Figure 2.1 shows the scale of benefits that can be expected. We have tabulated the elapsed time in months, the effort in man-months, and the likely costs for the main-build stage of projects with PIs of 15 (the PEP average), 17 (the average achieved by PEP members using application packages), and 22 (the theoretical average achievable using application packages). In each case, we have assumed a medium rate of manpower buildup — that is, an MBI of 3 — and a typical size of project — about 40,000 lines of code. We have assumed a cost of £4,000 per man-month of effort, this being about the average for PEP members. The table shows that a two-point increase in PI (from 15 to 17), resulting from using an application package rather than undertaking bespoke development, would reduce the delivery time for a typical new system by about one-and-a-half months. The

Figure 2.1 Using application packages can result in substantial savings of time, effort, and cost

The table shows information for a 'typical' PEP project with a size of 40,000 lines of code and an MBI of 3. Data is shown for PIs of 15 (the overall PEP average), 17 (the PI represented by PEP members' reported time and cost savings resulting from the use of application packages), and 22 (the average theoretical PI for the 29 application package projects). The costs of the projects have been calculated by assuming a cost of £4,000 per man-month of effort (this is close to the PEP average). The benefits of using application packages can be seen by comparing the costs, effort, and timescales of a project with a PI of 17 with those for a project with a PI of 15. Using an application package for a typically sized project would save about £100,000, and the project would be completed one-and-a-half months earlier and would use about half the effort.

When we compare the PI of 22, representing the theoretical average PI for the 29 application package projects, with the PEP average PI of 15, we can see that there are large potential savings in time and effort that could be made.

PI		Time taken to complete the project (months)	Effort required (man-months)	Cost (£)
PEP average	15	9.5	58.0	230,000
PI represented by PEP members' reported time and cost savings	17	8.0	31.0	125,000
Average theoretical PI for application package projects	22	4.5	6.0	24,000

development cost for this system would be reduced to nearly half that of developing a bespoke system. If a PI of 22 could be achieved, applications would be delivered in about half the time and at about one-tenth of the cost.

Obtaining measures of the lines of code and function points represented by a package will enable the productivity gains to be calculated

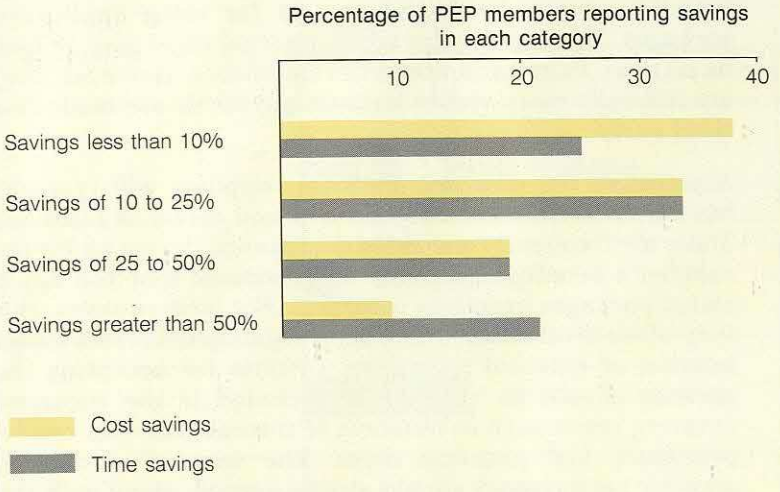
PEP members who want to estimate accurately the productivity gains that can be achieved by using an application package should try to obtain a measure of both the lines of code and function points that they will receive by buying a package. These measures should relate to the parts of the package that will be used. If only half the functions of a package will be used, estimates of the function points and lines of code necessary to provide these functions should be made. Using this data, it will be possible to estimate the PI for package-based development and to compare this with the likely PI for developing a bespoke system.

USE OF APPLICATION PACKAGES RESULTS IN TIME AND COST SAVINGS

PEP members believe that the most important benefit is time savings

Our calculation of the theoretical PIs achievable by using application packages suggests that, compared with bespoke development, reductions in effort, and hence, cost, are considerably greater than reductions in time. According to the results of our survey, however, PEP members believe that the most important reason for using application packages is time savings; cost saving is ranked third. As Figure 2.2 illustrates, significantly more PEP members reported savings of over 50 per cent in time than in cost. Small benefits (savings of under 10 per cent) are more frequently achieved for cost savings than for time savings.

Figure 2.2 PEP members report greater time savings from using application packages than cost savings



(Source: Survey of PEP members)

This apparently conflicting evidence may be explained by the fact that the costs used to calculate the theoretical PIs are based on the man-months of effort required for the main-build stage of a project. These exclude many of the cost items that PEP members will have taken into account when assessing the savings made by using an application package. These items will include the cost of the selection exercise, the cost of the package, training costs, and so on.

Package selection and acquisition involves some potentially high costs

During our research, we were unable to find any organisation that had kept complete records of all the additional costs associated with application package projects. Each of the cost items is, however, potentially high:

— *Package selection.* Considerable amounts of time and resources can be used just to select an application package. For large, complex applications, the selection process often involves a team of between three and six members of staff, working for an elapsed period of about six months.

— *Cost of the package.* Once a product is chosen, it can be expensive to acquire. The price of widely available application packages ranges from a few tens of pounds to over £1 million per module for the most powerful application packages.

— *Documentation.* Some package suppliers charge significant sums for documentation, and if many copies are required, the cost of acquiring a full set of documentation can represent up to 25 per cent of the cost of the packaged software.

— *Support and training.* Typically, organisations that use application packages have support agreements with the suppliers, and supplier personnel will undertake at least some staff training. The costs of both training and support can be high, however. The cost of initial training is often in the region of 10 per cent of the cost of the package. The costs of support will be incurred as long as an organisation uses a package. Typical annual costs are in the region of 10 to 15 per cent of the initial package cost. Of course, an organisation will incur support costs for both in-house bespoke system developments and for using application packages. Support costs for application packages may, in fact, be no more than for in-house developments. However, they are normally more visible because payments are made to a third party.

Criteria for accepting the package should be included in the contract

— *Negotiating the contract.* Package suppliers will typically have their own standard contracts and terms of business. These are frequently one-sided documents, drawn up for the supplier's benefit, and many organisations that use application packages negotiate contracts. For large projects, this very often involves seeking expert legal opinion from either internal or external specialists. Criteria for accepting the package should be agreed and included in the contract, covering issues such as numbers of transactions that can be processed, and response times. The ways of measuring supplier performance should also be agreed, along with the penalties for not meeting the agreed performance levels. The negotiations can take several months, and require agreement and commitment from all parties involved.

— *Modifying the package.* Modification costs can be greater than the initial cost of the package and are frequently difficult to estimate. This issue is addressed in more detail in Chapter 4.

THE BUSINESS BENEFITS OF USING APPLICATION PACKAGES CAN BE CALCULATED

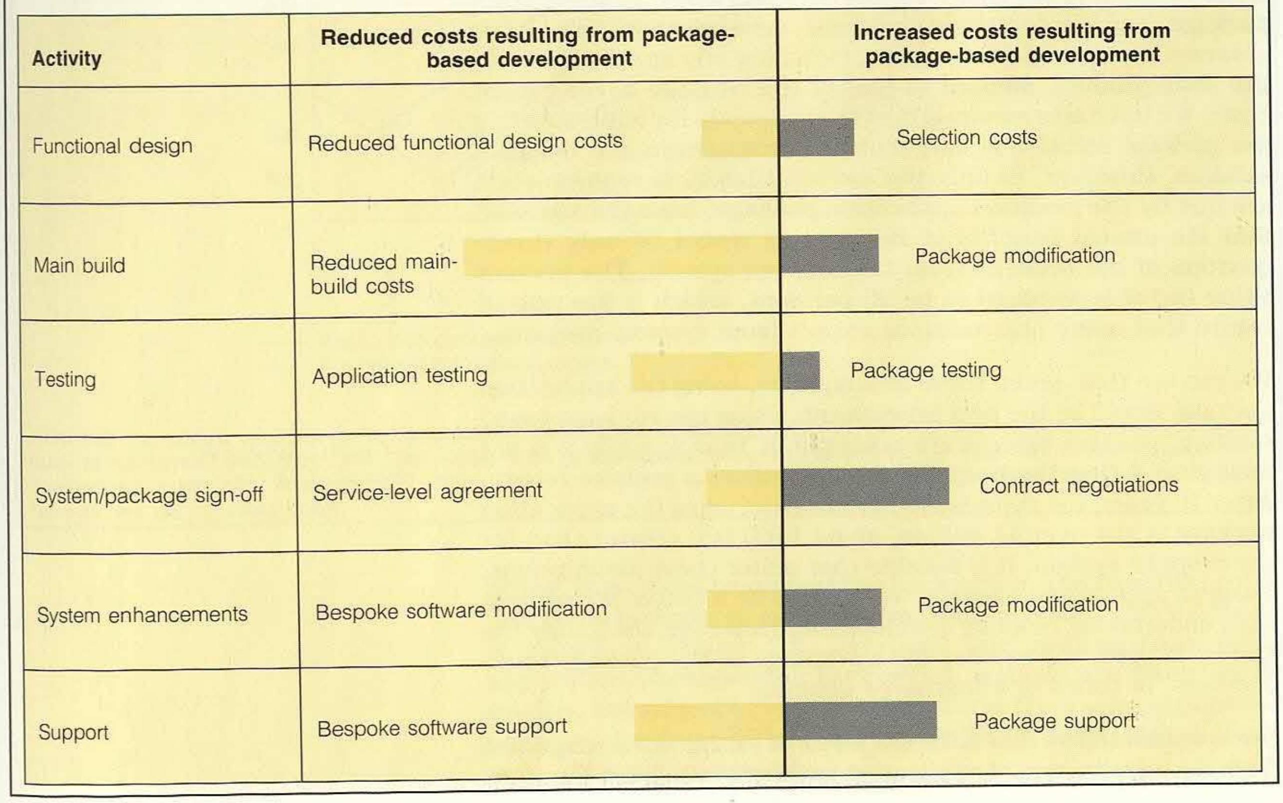
In order to assess the potential benefits of using an application package rather than undertaking bespoke system development, an organisation should understand the cost structure of both types of development, but more importantly, quantify the business benefits to be derived from the new system. To develop a reliable business case, the likely timescales for delivery for both the bespoke and package options need to be considered.

The differences between packages and bespoke development have implications for cost estimating

Typically, most systems development departments have a poor record of estimating systems delivery costs for bespoke development, even though they have more experience at doing this than estimating package-based systems delivery costs. Systems development managers must take into account the fundamental differences between the two types of development when producing estimates. These differences and their implications for cost estimating are illustrated in Figure 2.3. While some activities, such as contract negotiation, will increase the relative cost of using application packages, the overall impact of the differences should be that an application package project is delivered more quickly and at lower cost than a bespoke development. This is because using an application package effectively changes the nature of the main-build stage. Instead of developing functionality from scratch, the facilities available with the package are used to establish the required functionality.

Figure 2.3 The different activities of bespoke and application package development have implications for cost estimating

The net effect of the reductions and increases in costs is that the overall cost of using a package will be considerably lower.



Activity	Reduced costs resulting from package-based development	Increased costs resulting from package-based development
Functional design	Reduced functional design costs	Selection costs
Main build	Reduced main-build costs	Package modification
Testing	Application testing	Package testing
System/package sign-off	Service-level agreement	Contract negotiations
System enhancements	Bespoke software modification	Package modification
Support	Bespoke software support	Package support

One argument often put forward for not using packages is that a package will not provide all of the benefits that would be available from a bespoke development. It is better, so the argument runs, to wait longer for a bespoke system that can provide greater benefit. However, using the net present value (NPV) technique to compare the returns on investment of delivering a new system will usually lead to the conclusion that using an application package is a better investment than bespoke development. Even though the year-on-year benefits of a package solution are lower than those of a bespoke solution, the package represents a better investment because the benefits are obtained earlier.

NPV calculations are based on the fact that future benefit values are worth less than the same benefit obtained today. The difference is measured by considering the 'rate 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 pounds obtained in two years' time would be worth still less today. Thus, a bespoke development that takes a long time to develop and implement may have a lower NPV than a package-based solution that can be implemented within much shorter timescales, even if the application package does not meet all the detailed user requirements, and hence, achieves lower year-on-year benefits.

NPV calculations often show that a package is a better investment than bespoke development

An example of a simplified NPV comparison of undertaking a project using either an application package or bespoke development is shown in Figure 2.4. In this figure, a hypothetical organisation is faced with a choice between using an application package and undertaking bespoke development. We have assumed that total package costs, including any modifications to the basic product, amount to half of the bespoke development costs. We have also assumed that the timescale for implementing the package solution is half that of implementing the bespoke solution. However, as only the essential business requirements are met by the modified application package, we have assumed that the annual benefits of this system would be only three-quarters of the benefits from the bespoke system. The present value factor is assumed to be 20 per cent, which is the rate of return that many organisations expect from systems projects.

We can see that, given these assumptions, using the application package would be the best investment. Using the package-based solution, positive returns are achieved in Year 2, while it is not until Year 4 that the bespoke system realises a positive return. After 10 years, the cumulative benefit from using the application package is just over £1 million, about £100,000 greater than for the bespoke system. It is notable that under these assumptions, using an application package will always be a better investment than undertaking bespoke development. However, the longer the system is used, the smaller the advantage of the package route becomes, in terms of cumulative benefits.

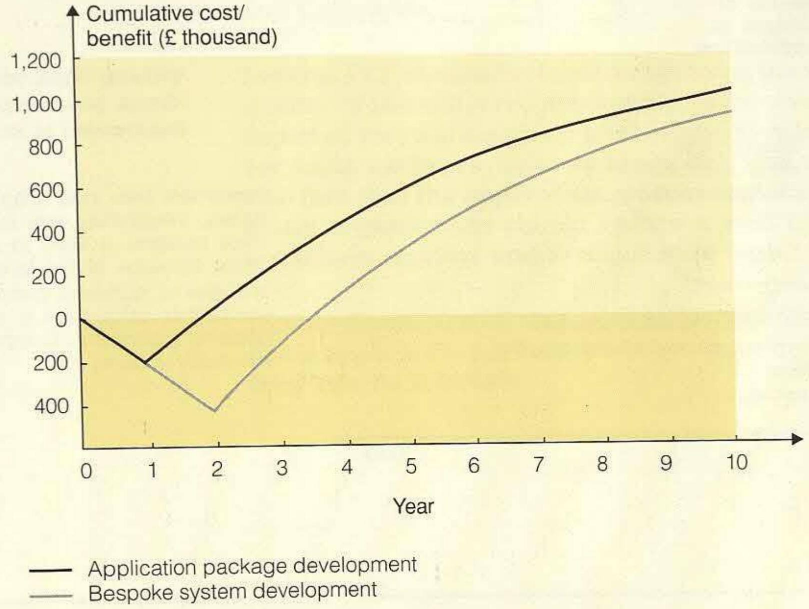
Our research shows that different levels of return can be expected from the introduction of application packages to support the front

Figure 2.4 **The net present value technique can be used to compare the business benefits of application package and bespoke system development**

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. Total package costs, including purchase and tailoring, are £200,000, and the new system will be implemented within a year. At present values, it will produce benefits of £300,000 per year. The bespoke system will take two years to develop and will cost £400,000 (£200,000 in each year). Once implemented, it will produce benefits of £400,000 per year, at present values. The present-value factor is assumed to be 20 per cent.

		Application Package Development			**Bespoke System Development**		
Year	**PV factor (20%)**	**Benefit or (cost) (£)**	**NPV of benefit or (cost) (£)**	**Cumulative benefit or (cost) (£)**	**Benefit or (cost) (£)**	**NPV of benefit or (cost) (£)**	**Cumulative benefit or (cost) (£)**
1	1.00	(200,000)	(200,000)	(200,000)	(200,000)	(200,000)	(200,000)
2	0.83	300,000	249,000	49,000	(200,000)	(166,000)	(366,000)
3	0.69	300,000	207,000	256,000	400,000	276,000	(90,000)
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•
10	0.19	300,000	57,000	1,003,000	400,000	76,000	906,000

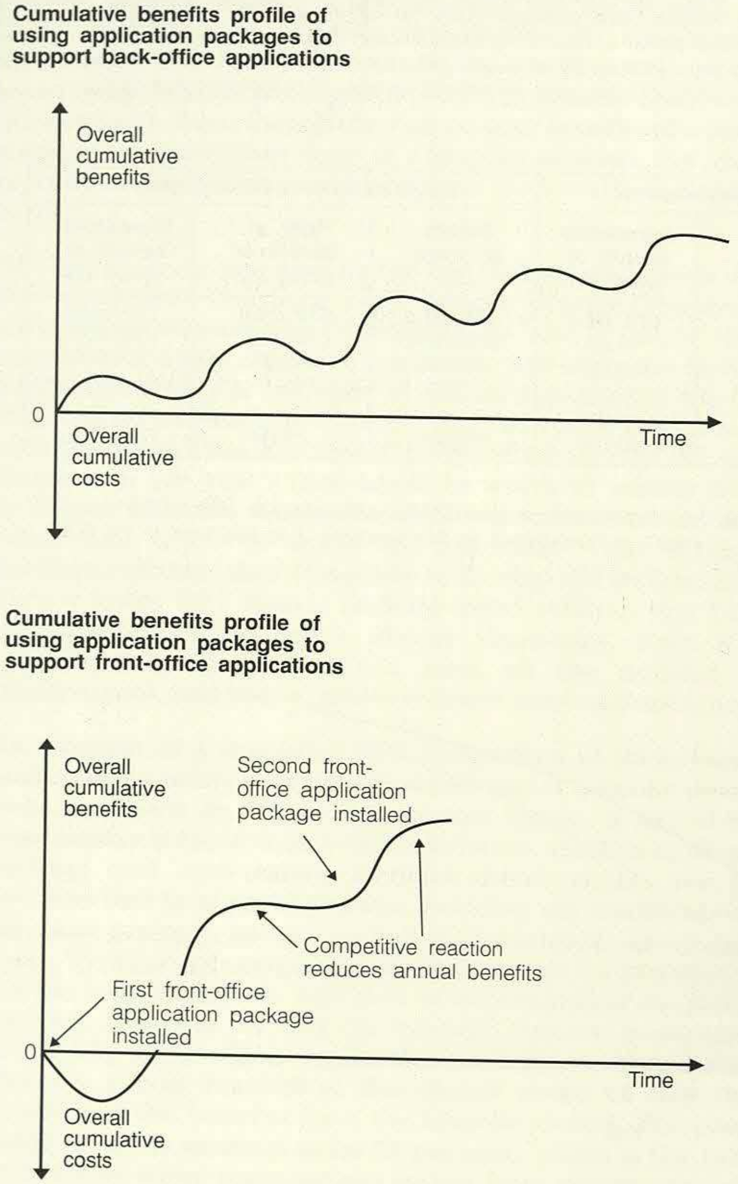
Net present values are calculated by multiplying the expected benefit or cost by the present-value factor. The factor for year n+1 = $1 \div (1+r)^n$, where r is the expected rate of return on investment. In this example, r is assumed to be 0.20, representing a 20 per cent return.



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.

office, from those to support the back office. The benefits profile that could typically be expected for front- and back-office systems is shown in Figure 2.5, overleaf. The returns from using application packages for back-office systems are likely to be smaller, but subject to less risk; the returns from using application packages for front-office systems are likely to be high, but are more difficult to quantify.

Figure 2.5 Different levels of return can be expected from investment in application packages to provide front- and back-office systems


Cumulative benefits profile of using application packages to support back-office applications

Cumulative benefits profile of using application packages to support front-office applications

Investment in application packages for the back office is worthwhile but the returns are not spectacular, because the objective of back-office systems is typically to reduce the cost of administrative processing. The 'dips' in the benefits profile are caused by the time taken to gain maximum benefit once a new (or enhanced) system has been introduced.

Investment here can bring significant benefits. Initially, introducing new (or enhanced) front-office systems is likely to result in overall costs, because of the level of investment and changes to business patterns. However, as a competitive advantage is achieved, the benefits become substantial. Competitors' reactions will eventually reduce this.

APPLICATION PACKAGES ARE OFTEN USED FOR STRATEGIC REASONS

Many organisations have used broad-based business considerations to justify the use of application packages, rather than specific cost or benefit comparisons. For example, several organisations have used packages to support a new business venture. Typically, they have used packages to provide a new service ahead of competitors, thus enabling them to become established market leaders. Very often, the decision to use an application package is made not on the basis of either the costs or the benefits involved, but on the basis of business judgement; business pressures mean that the only way of delivering a system in the required timescale is to use a package.

Packages are sometimes used to meet tight timescales

Several organisations have used application packages to introduce new computer applications to meet regulatory changes within tight timescales — for example, the financial systems of organisations that have been transferred from the public to the private sector. Again, in these cases, the decision to use application packages has usually been a broad-based business decision, and has not been based on a clear understanding of the costs and benefits involved. Some organisations have made a strategic decision that all information systems will be based on application packages, in order to exploit the perceived benefits of packages. In these organisations, a package will normally be used without an assessment of its worth in a specific case.

Other organisations have made a strategic decision not to use application packages, for a variety of reasons. They may have had poor experience with packages in the past, they may feel that their requirements are so specific that suitable packages will not be available, or they may have a systems department that believes that developing bespoke applications increases its own standing within the organisation. Such a decision will preclude the use of packages even where it can be justified in terms of reduced costs and timescales.

For some, the high quality of package-based applications is important

Several PEP members claimed that using packages improved the quality of the delivered application. While overall, PEP members reported that a guaranteed level of quality was not a major reason for using packages, in some cases, this was an important issue. The fact that the application source code of a package is used by many organisations should ensure a well tested product, with consistent quality, and few software bugs.

Having demonstrated that substantial benefits can be gained from using application packages, we now consider the ways in which they can be selected.

Chapter 3

Established methods are available for selecting application packages

PEP members employ a wide range of methods for selecting application packages, from completely informal approaches to highly formal methods. Those who do not use a formal method typically select application packages in an ad hoc manner, plan projects on a 'one-off' basis, or do not plan this type of project at all. Such an informal approach can work, but it leaves the organisation open to unnecessary risk.

Clearly, many organisations realise that the evaluation and selection of application packages is already a very important area of work — during the last two years, three-quarters of PEP members have assessed a wide range of application packages, covering several business areas. The evaluation and selection of application packages will become more important over the next few years as the use of packages increases. Almost half of PEP members already use a formal method for the selection of application packages, although several are revising their standards in this area. Many of those who currently do not, are considering introducing one. Proprietary methods for package selection are also beginning to come onto the market as an alternative to methods devised by organisations for their own purposes.

The package-selection method must be appropriate to the organisation and the type of package

Figures 3.1 and 3.2 (on page 22) describe two different approaches to selecting application packages, one informal (used by a utility company), and one formal (used by Lombard North Central). Each organisation considers its approach to have been successful. These examples demonstrate the importance of using an approach that is appropriate for the organisation and for the type of package. It was appropriate for the utility company to take an informal approach because it did not need the new packages to fit closely with any specific requirements, and it needed the new systems quickly. Lombard North Central needed to take a formal approach because it was selecting a package to provide a competitive-advantage, front-office system. It had to be certain of its requirements and had to select the best package so that competitors could not beat it at its own game by using a better product.

Formal and informal approaches both have a role to play

In general terms, it is usually appropriate to use a detailed, formal method for selecting a large, mainframe-based package, which will be expensive to acquire and of strategic importance to an organisation's business. Such an approach, however, would not be appropriate for a small, microcomputer-based package, which is inexpensive and not strategically important. These types of smaller application packages can make a valuable business contribution, but not if the cost of selecting them outweighs the benefits of using them. Other considerations are also important in deciding which approach to use for package selection. Business requirements may demand that a system is operational in a very

Figure 3.1 A utility company has adopted an informal approach to the selection of application packages

When faced with legislation that changed the organisation's status from a public utility to a privatised company, this organisation took a strategic decision to invest in application packages. The organisation's financial systems were not suitable for use within the new commercial environment. This meant that the financial applications, which had been developed in-house during the previous 10 years, all had to be replaced within a year. The organisation decided that the only way to do this was to use application packages.

The organisation adopted an informal approach to selecting application packages. It was successful in that the new financial systems were implemented within the required timescales. The approach involved users and systems staff working closely as teams at all stages. It can be summarised as follows:

Stage 1: Define requirements. Requirements are defined quickly, and at a high level. The appropriate application area and the scale of the business is defined, but no detailed specification of requirements is produced at this stage.

Stage 2: Identify packages with the required functionality. A literature search is undertaken to identify possible suppliers. These are contacted, and company representatives attend demonstrations of the functionality of the products. A high-level specification for the new system is developed and refined by taking into account the features of the products. The specification is progressively developed until there is a clear match between the requirements and the software facilities of a package. This is the preferred choice.

Stage 3: Investigate product and preferred supplier. A list of users of the preferred product is compiled. These users are contacted and the product's and supplier's performance are discussed. An analysis of the preferred supplier is undertaken, covering the turnover of the company, the number of staff employed, the future development path for the package, and the installed user base for the product being investigated. If the results of the supplier analysis are positive, a decision to contract with the preferred supplier is taken.

Stage 4: Negotiate contract. Contracts are not negotiated to any predefined standards. The best 'deal' is sought in each case.

The organisation has used this informal approach to select application packages within very short timescales. This has been a major success for the company, which believes that if bespoke development had been undertaken, it would have taken a year to complete the specifications before development could begin. Two problems have emerged, however. First, difficulties have been encountered when the company has tried to develop the necessary interfaces between various packages. Second, the most important package had only recently been developed to run in the organisation's hardware environment, and as a consequence, there have been some 'teething problems'.

short timescale, and in such a case, a lengthy, detailed selection exercise would be inappropriate.

INFORMAL APPROACHES EXPOSE THE ORGANISATION TO UNNECESSARY RISK

An informal approach to selecting application packages may simply require users to identify two or three potential suppliers, attend demonstrations of their products, and buy the package that they like most. Choosing the most appropriate package with such an approach is likely to be more by luck than judgement. An organisation that uses this type of approach is exposed to considerable risk because many important factors are not taken into account when the decision is made to acquire a package. Using an informal approach can leave an organisation vulnerable to persuasive sales skills and impressive demonstrations. In particular, the level of expertise available from the supplier after a sale has been made may not match the level implied at an impressive sales presentation.

Figure 3.2 A formal package-selection approach has been adopted for a new business venture at Lombard North Central

Lombard North Central, a large financial services company, has recently selected an application package to support a new business venture. The selection was led by the business manager responsible for establishing the new venture. Swift entry into the new business area was essential to gain a competitive advantage, and computer support was required to achieve this. However, discussions with the systems department revealed that an in-house solution could not be provided in time. As a consequence, the business manager used a formal method to select an application package. The main stages of work were:

Stage 1: Define requirements. Three types of requirements were defined — business, operational, and technical. Business requirements were rigorously defined using data flow diagrams.

Stage 2: Produce invitation to tender. An invitation to tender was produced, consisting of:

- The requirements for the new system.
- Administrative details for the tender process.
- Format for the response.
- Additional information required — for example, supplier status.

The invitation to tender specified that suppliers should define how well their basic system met the requirements, and what it would cost to modify the system to meet the requirements completely.

Stage 3: Define selection criteria. Selection criteria were defined in three categories — functional, commercial, and technical. Criteria were weighted at group and individual levels. Functional requirements accounted for 70 per cent of the weighting overall.

Stage 4: Create a shortlist of suppliers. A long list of suppliers was created from a literature search and from knowledge of available systems. This was reduced to a shortlist of four suppliers, by considering:

- The range of functionality supported for the package.
- The ability of the package to process the required volumes of work.
- The stability of the supplier.

Stage 5: Issue invitation to tender. Shortlisted suppliers were informed of the intention to invite them to tender to supply an application package. The invitation to tender was issued and a presentation was made to each supplier to explain the requirements and the tendering process.

Stage 6: Evaluate responses and make recommendation. When responses were received from the suppliers, the packages were scored on a scale of one to four according to how well they met the selection criteria. The weighted scores were used to analyse the approach proposed by each supplier and to identify the suppliers' strengths and weaknesses. One package was clearly the preferred solution.

This selection process was successful, for several reasons. The scope of the project was clearly defined, and the project was managed by a steering committee. Regular meetings were held and weekly progress reports were issued. The analysis of requirements was thorough, and suppliers were evaluated using a comprehensive set of criteria.

Interestingly, the preferred package does not conform with the company's technical architecture, which means that the business department may have to provide operational and technical support for the new system.

Figure 3.3 lists the typical stages in an informal approach. Although such an approach typically takes less time and effort, it exposes the organisation to more risk, because user requirements are not clearly defined, technical and operational requirements are not taken into consideration, the supplier's performance is not considered in detail, and the features of the packages available are not studied in detail. With an informal approach, the most significant influence on the choice of package

Figure 3.3 An informal approach to package selection might typically comprise seven stages

1 A loosely defined working party is set up to organise the selection process. Project structure is very informal.

2 A small number of fundamental requirements is agreed.

3 Suppliers are researched and demonstrations are arranged for a few products.

4 On the basis of the demonstrations, a preferred product is chosen.

5 Any further issues are discussed informally with the suppliers.

6 A minimum assessment of the preferred supplier may be undertaken.

7 Some consideration may be given to contractual implications.

With an informal approach, organisations can be over-influenced by impressive sales presentations

is the skill with which suppliers present their products and support capabilities during demonstrations. The approach to package selection adopted by those PEP members who do not use a specific method tends to be closer to an informal approach than to a formal method.

A FORMAL METHOD IS USUALLY A BETTER ALTERNATIVE

Many package suppliers prefer organisations to adopt a formal method for selecting a package because they realise that this increases the likelihood of their selling their product to an organisation for which it is appropriate. If an organisation uses an informal approach, and makes a poor choice of package, the chosen supplier will have little to gain in the longer term.

Some package suppliers encourage the use of consultants at the selection stage

Several leading package suppliers encourage organisations with little experience in selecting application packages to engage consultants to assist them. These package suppliers have typically instigated consultant-liaison programmes to update consultants about their products and plans.

A formal approach that PEP members might adopt if they have no established guidelines for selecting application packages is discussed below. It is based on 'best practice' derived from the research that we have undertaken. Members who already have a standard approach to the selection of packages should compare it with our 'best practice' guidelines and amend their approach if necessary. Organisations that do not have an established approach to selecting packages, but that do have one for bespoke system development, should draw on it to complement their approach to package selection — established techniques for requirements definition, for example, could usefully be incorporated into a formal approach, and the same project-management framework might also be appropriate.

A FORMAL APPROACH HAS SIGNIFICANT ADVANTAGES

The additional time and effort required by a formal approach is worthwhile

A formal approach typically takes more time and effort than an informal approach, but it also has indisputable advantages:

— It reduces risk. A formal approach will ensure that all appropriate issues are considered during the selection process, and this should ensure that the best product is selected.

— For those organisations involved in selecting many application packages, it should reduce the effort and time required, because staff will follow a standard approach, which will include standard criteria for evaluating suppliers and products, contractual features, and so on.

— It should enable a clear business case to be made for any planned investment, with accurate estimates of benefits and costs.

— It will ensure that all parties (users, systems staff, and suppliers) clearly understand what their roles are in the selection project, and how the project will be progressed.

- By clearly defining requirements and assessment criteria, it limits the subjective element in the selection process, and thereby removes possible partiality.
- It provides a record of the selection exercise, outlining what was done and why the package was chosen. This information is often useful after implementation, especially if enhancements are planned.

THE CHOSEN APPROACH MUST BE FLEXIBLE

Any formal approach to the selection of application packages must, however, still be flexible. PEP members will want to select different types of application packages and will do so in the face of different business constraints. To ensure that the approach used is appropriate to the occasion, the approach must have appropriate paths that can be followed depending on the type of package being assessed. For example, a project to select a package for a small, non-business-critical application should not be undertaken in the same manner as a project to select a large system of major importance. Thus, the main accounting system of a major multinational corporation should be selected with more care and rigour than a purchase-order system of a small subsidiary. Some of the methods already used by PEP members have this type of flexibility. The majority, however, give guidance for one type of application, which is to be followed for all projects.

The approach to selecting a package for a small application should be different from that used for a major application

A standard approach should also have the flexibility to keep the organisation's options open until the best solution is evident. The options to reconsider other packages or bespoke development should remain open as the shortlist is narrowed to a preferred supplier. At any stage during a selection exercise, it may be appropriate to change direction. This should be accommodated within the approach.

In Chapter 2, we showed that significant savings and improvements in process productivity can be made when the costs and timescales involved in package selection are minimised. The approach to package selection should therefore be economical in terms of effort, and especially time. It should allow the appropriate balance to be struck between reducing the timescale (and therefore the thoroughness of the selection process) and increasing the risk of choosing an inappropriate package.

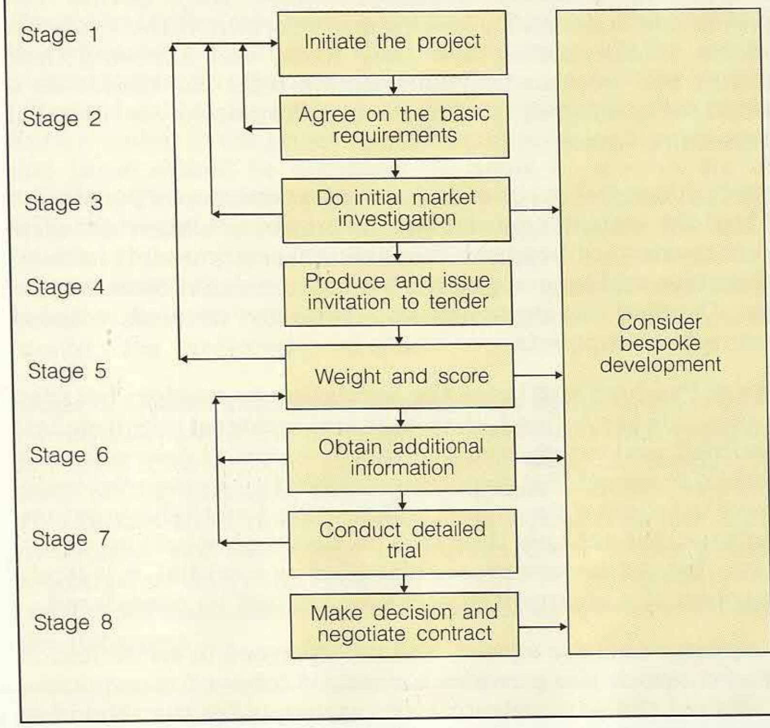
AN EIGHT-STAGE APPROACH IS RECOMMENDED

The approach we recommend is shown schematically in Figure 3.4 and has eight stages:

Stage 1: Initiate the project. In many respects, this is the most critical stage in the selection process. It is important that before the project gets under way, clear terms of reference and goals are established, and everyone involved has a clear view about what is to be achieved. The selection approach to be followed during the rest of the project is decided upon, taking into account the size, complexity, importance, likely cost, and required implementation timescales of the application. The project team is set up, and team members' roles and responsibilities are defined. At this stage, the management structure for the project is

The selection approach should take account of the application's size, complexity, importance, cost, and implementation timescale

Figure 3.4 The recommended approach to selecting an application package consists of eight stages


Stage 1
Initiate the project
Stage 2
Agree on the basic requirements
Stage 3
Do initial market investigation
Stage 4
Produce and issue invitation to tender
Stage 5
Weight and score
Stage 6
Obtain additional information
Stage 7
Conduct detailed trial
Stage 8
Make decision and negotiate contract
Consider bespoke development

also established. Progress is monitored and controlled via regular reporting and meetings.

Stage 2: Agree on the basic requirements. The fundamental requirements of the users, and the technical requirements, are defined and agreed. Fundamental requirements should be kept to a minimum, but should include the main criteria that any application package must meet if it is to satisfy the project objectives. These requirements are likely to include the main processing requirements of the business area being addressed, and the need to comply with any constraints imposed by the technical architecture.

Stage 3: Do initial market investigation. A product search is undertaken to identify packages that are likely to meet the fundamental requirements. There are many sources of information, such as software guides, magazines, and the experience of colleagues. Publications specialising in application packages are now available, and these usually contain a candid review and comprehensive information about suppliers, package costs and options, and required hardware and operating system software. An example of a package-specific publication is *Software Guide for Accountants*.

Initial demonstrations of up to six packages should be arranged

Possible suppliers are contacted, and initial demonstrations are arranged of up to six packages that appear to be capable of meeting the fundamental requirements. In our experience, having demonstrations of between three and six packages should give a fair view of the types of products available. If more packages are investigated, the additional effort may be excessive in terms of the additional information gathered. It is important that those involved in selecting a package attend demonstrations early on in the selection process, so that they have an understanding of the types of products available before they define the requirements in detail. By looking at a selection of the available products, understanding how they work, and assessing their strengths and weaknesses, the selection team can decide on a shortlist of preferred packages and formulate its detailed requirements better.

There is a danger that supplier demonstrations can be persuasive and lead the team to consider issues that are not important. The project teams must keep the fundamental requirements in mind at this stage and ignore suppliers whose products cannot meet them. Detailed requirements can then be decided without pressure from suppliers.

Packages that do not meet the fundamental requirements should be ignored

Stage 4: Produce and issue the invitation to tender. Detailed requirements are defined. Both user and technical requirements are refined, and categorised as either *essential* or *desirable*. It is important to ensure that requirements classified as essential really do have to be met if the system is to provide a workable solution. If features that are only desirable, in that they would be useful to have, but not necessary, are classified as essential, it is likely that potentially appropriate packages will not be considered.

The requirements are agreed, and incorporated in an invitation to tender, which also provides a standard format for responses, and sets out the administrative arrangements for the tendering procedure. The invitation to tender is then issued to the preferred suppliers identified in Stage 3. The typical contents list for an invitation to tender is set out in Figure 3.5.

Figure 3.5 Most invitations to tender have similar contents

Tender administration
Request to tender statement
Contact details for tender processing
Confidentiality statement
Completion and submission of proposal
Criteria used to assess responses

Background information
Overview of the company
Overview of the current system
Scope of the proposed system
Essential requirements of the new system
General systems requirements and constraints

Summary of current systems procedures
Main functions of current system
Responsibilities for performing main functions
Control procedures
Specific areas of difficulty

Systems requirements
User requirements
Technical requirements
Operational requirements

Required tender content and format
Management summary
Application software
Hardware
Operating systems software
Support
Implementation summary
Contractual terms

Producing a detailed invitation to tender and ensuring that suppliers respond using the specified format is very useful, particularly for larger, more complex applications. The effort of matching product functionality to requirements is borne by the suppliers, rather than by the package purchaser. Commitments given in the chosen supplier's response should be incorporated into the final contract, to ensure that the supplier has a legal obligation to meet the claims he has made. The supplier, in turn, can identify any modifications that may be required to the package, advise on the most appropriate ways to make these modifications, and estimate the costs and timescales involved.

Stage 5: Weight and score. The next task is to agree on the weightings for each *desirable* requirement. Many organisations carry out this task before the invitation to tender is prepared. We believe, however, that it is more efficient to agree on the weightings while waiting for the suppliers to respond to the invitation to tender. Requirements should be classified into different types and each type should also be weighted. For example, for an accounting system, requirements may be classified into financial accounting, management accounting, and so on. This will enable the importance of individual requirements to be reflected, along with the importance of different types of requirements.

The advantage of defining weightings for the different requirements when suppliers are preparing their tenders is that it saves time, because the invitation to tender can be issued before the weightings are agreed. The process of defining the weightings focuses the project team's attention on the completeness of the requirements and helps to identify any missing requirements.

Close contact with suppliers will ensure that the score given to each selection criterion is accurate

When the tender responses have been received, the first task is to eliminate any product that cannot meet all the *essential* requirements. For the remaining products, each of the weighted criteria should be scored to produce a total score for each product. Each member in the project team should produce his own scores, and these should be discussed, to agree on a score for each criterion. Close contact with suppliers will be essential during this stage to ensure accuracy in the scoring. Figures 3.6 and 3.7 (overleaf) show sample forms for recording the classification and weighting of requirements and the scores given to each package being evaluated. Total scores for packages can be compared to identify the preferred products.

Stage 6: Obtain additional information. In this stage, additional information about the preferred products and their suppliers is gathered from visits to, and discussions with, the suppliers and users of the packages, and from company searches, and so on. This should serve to clarify any issues arising from the tenders, to confirm the experience of current users of the preferred products, to verify the financial stability of the supplier, and to confirm the costs, resources, and timescales required to implement the packages.

Figure 3.6 Each requirement, and type of requirement, should be weighted

	Application package selection checklist **Package name:**				
	Essential or Desirable indicator (E or D)	**Score**	**Weighting applied**	**Weighted score**	**Comments**
Requirement type 1 — Requirement 1 — Requirement 2 — Requirement 3 — Requirement 4 — Requirement n					
Total score for Requirement type 1			Weighting for Requirement type 1	X	
Requirement type 2 — Requirement 1 — Requirement 2 — Requirement 3 — Requirement 4 — Requirement n					
Total score for Requirement type 2			Weighting for Requirement type 2	Y	

Total score for package = X + Y

Figure 3.7 Agreed scores should be applied to the weighted criteria to produce a total score for each product

	Weighted scores by requirement and type of requirement			
	Package 1	**Package 2**	**Package 3**	**Package 4**
Requirement type 1 — Requirement 1 — Requirement 2 — Requirement 3 — Requirement 4 — Requirement n				
Total score for Requirement type 1				
Requirement type 2 — Requirement 1 — Requirement 2 — Requirement 3 — Requirement 4 — Requirement n				
Total score for Requirement type 2				
Total score for package				

Checklists should be used for obtaining additional information

It is important that the project team has a clear view about what is to be achieved by obtaining the additional information. Checklists or questionnaires should be developed to ensure that all the appropriate issues are covered. These should be designed in such a way as to ensure that a fair view is gained about the issues under investigation. For example, if an organisation is trying to assess the product support provided by a potential supplier, it would be inappropriate to ask ''Do you provide comprehensive support for the package?'' because the answer is always likely to be ''Yes''. Questions, along the lines of those listed below, would give an accurate impression of the standard of support offered:

- What support do you provide for the package?
- Do you provide both user and technical support?
- Do you have a specific department dedicated to supporting the package?
- How many staff do you have providing support for the package?
- Do you have guaranteed response times for queries?

If Stage 5 revealed a product that is clearly preferred, Stage 6 should initially be undertaken with reference to that product only. If several products are closely scored at the end of Stage 5, emphasis should be placed on assessing those areas that will be most important in making the selection decision. Usually, it is the various aspects of implementation — costs, timescales, and so on — that are critical at this stage.

While these issues will usually have been considered during earlier stages, it is only at Stage 6 that both the supplier and the organisation are likely to have sufficient information to estimate accurately the costs and timescales of implementing the package. This information will be obtained in discussions between the supplier and the organisation, and by subsequent analysis. The high level of effort and the long timescales involved in implementing some packages can offset their apparent advantages. Typically, the more flexible and complex a package is, the more difficult it is to implement. Thus, while a product may have a very close fit with an organisation's requirements, its implementation may be either too costly, or take too long, to make selection of this product a viable business solution.

More flexible and complex packages are often more difficult to implement

Stage 7: Conduct detailed trial. For complex applications, when fast implementation is not of fundamental importance, a trial period is recommended, to gain a detailed understanding of an application package. This will confirm whether or not the product will offer a good operational solution. A trial should be undertaken only if it has a good chance of being successful. Most suppliers agree to trials, because it should make subsequent implementation easier.

Stage 8: Make decision and negotiate contract. Following a successful product trial, or the provision of appropriate additional information, a decision will be made to acquire the package. At this stage, enough information will exist to develop a detailed business case for the investment.

The final stage is to conclude the contractual negotiations. Statements made in the tender response should be included in the contract, together with other safeguards to protect the organisation's interests. Any important issues that arose during the selection exercise should be recorded in writing, and written responses from suppliers should also be included in the contract. Payment terms should be agreed, including penalties for poor performance by the supplier.

Penalties for poor performance by the supplier should be specified in the contract

Repeating some of the stages. Figure 3.4 showed that there are five points during the process when it may either be appropriate to go back to a previous stage or to abandon the package-development option:

— If the basic requirements agreed in Stage 2 are different in scope from the ideas when the project was initiated, it may be necessary to revise the project's terms of reference, planning, and management procedures.

— If Stage 3 reveals that the products available do not adequately meet the fundamental requirements, the project may have to be rethought. Alternatively, if it is not possible at this stage to find products that are likely to be suitable, bespoke development could be considered.

— At Stage 5, the weighting and scoring may lead to the conclusion that no package can meet the essential requirements. Bespoke development should then be considered, or the scope of the project could be changed.

- At Stage 6, if the additional information reveals that the supplier is not financially sound or has other shortcomings, it may be appropriate to obtain additional information about the other products (and suppliers) assessed, or to consider bespoke development.
- If the detailed trial (Stage 7) is unsuccessful, the supplier of the next-highest-scoring product should be investigated, and if appropriate, this product should also be subjected to a detailed trial. The final decision should usually be to contract for the supply of the application package that has been successful at trial. If no package is found to be satisfactory at the trial stage, bespoke development should again be considered.

A SUBSET OF THE STAGES MAY BE APPROPRIATE IN SOME CIRCUMSTANCES

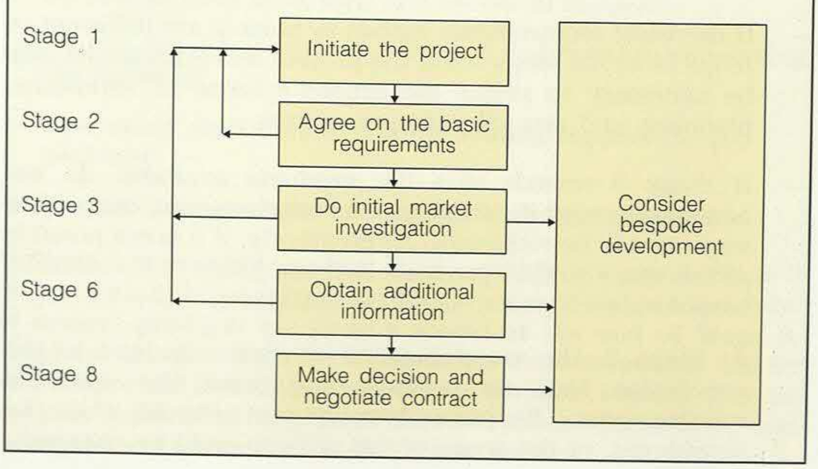
The approach we have described so far is the full approach that we recommend PEP members adopt when selecting packages for major, complex applications. The approach should, however, be used pragmatically. It is not necessary, for example, to undertake a detailed trial of a small, inexpensive product. At the project-initiation stage, the project manager will take a view on how the project will be carried out, which stages are necessary, and what emphasis and level of detail should be undertaken at each stage. Figure 3.8 illustrates a subset of the approach that is appropriate for a smaller, less complex application.

The recommended selection approach should be used pragmatically

For a small, simple application, the project team can decide during Stage 1 (when the project is initiated) to miss out Stages 4, 5, and 7. In the example shown in Figure 3.8, the project team has decided that it is not appropriate to produce an invitation to tender or to weight and score responses. Nor is it appropriate to undertake a detailed product trial. Missing out these stages exposes the project to more risk, but the risk is limited because of the controlled project framework. The decision about the preferred supplier will be taken during Stage 3, when the market investigation is undertaken. This means that this stage will probably need to be more extensive than in a project to select a

Some of the stages can be omitted for simple applications

Figure 3.8 The recommended approach may be modified for the selection of a small, simple application


Stage 1
Stage 2
Stage 3
Stage 6
Stage 8
Initiate the project
Agree on the basic requirements
Do initial market investigation
Obtain additional information
Make decision and negotiate contract
Consider bespoke development

larger, more complex application. Additional information will still be gathered to assess the preferred supplier's performance in predefined areas.

PROPRIETARY METHODS ARE WORTH CONSIDERING

Three proprietary methods for selecting application packages are available

An alternative to devising a formal method in-house is to acquire a proprietary method for selecting application packages. While several of the integrated development methods, such as Method 1, have sections that can be used for package-based development, proprietary methods that have been produced specifically to help with package selection are now becoming available. We have identified three such methods, each of which has been used successfully by several large organisations. The three products are SIIPS (Selection and Implementation of Integrated Packaged Software), supplied by KPMG Peat Marwick, Buy/Build, supplied by Hoskyns, and LBMS Package Evaluation, supplied by Learmonth and Burchett Management Systems. Summaries of each are given in Figures 3.9, 3.10 (overleaf), and 3.11 (on page 33). The three methods have many similarities:

— A detailed definition of requirements is produced.

— The options are progressively reduced from a long list, to a short list, to a preferred supplier.

— The supplier's involvement is extensive.

Figure 3.9 SIIPS, from KPMG Peat Marwick, is a comprehensive method that covers both the selection and implementation of an application package

The key aspects of this method are the emphasis placed on a user-driven project and the use of a detailed invitation to tender, using weightings of requirements so that different packages can be compared. The work is split into four phases.

Phase 1: A detailed definition of requirements is undertaken. This focuses on what is required, rather than how it is to be provided. During this phase, it is assumed that the ultimate solution may be either package-based or bespoke development; there is no difference in approach. Major users of the proposed application must agree on a report outlining the requirements, at this phase.

Phase 2: A conceptual design is developed for the application. This documents what the new system is to do. Current facilities are then reviewed and a 'shopping list' is produced of software components (for example, packages, fourth-generation languages, and so on) that are required to implement the new system. A 'long list' of potential suppliers is created and reduced to a short list by assessing both the suppliers' standing and product features.

Selection is based on a list of mandatory and desirable features (a MAD list) required to support the business-processing, data, and technical requirements. These requirements are incorporated into an invitation to tender. The responses from invited suppliers to each item on the MAD list are weighted and scored. Desk checking of the responses is backed up by visits to sites where the product is being used. Generally, the highest-scoring package is chosen.

Phase 3: This phase deals with the implementation of the chosen package. The SIIPS approach to implementation recognises that individual areas of a package can be designed, developed, and implemented independently of, and concurrently with, other areas. Other issues addressed within this phase include training, documentation, interfaces with other systems, testing, and any other activities required to get the system ready for live operation.

Phase 4: This phase provides a framework for carrying out a post-implementation review of the project and for measuring the benefits derived from the new system.

Figure 3.10 The Buy/Build method, from Hoskyns, treats the selection and implementation of an application package as a single project

Buy/Build is an integrated part of the PRISM (Professional Information Systems Management) development method. It is based on progressive management decisions being made to identify options, select suppliers, and establish the functionality of interfaces or other facilities that are implemented with the package. It is comprehensive and flexible, and can be tailored for the type of package being selected. This process is comparatively risk-free because checklists and charts are provided that outline the work and deliverables from each task. If tasks are undertaken in a different order, or not included in a project, it is clear what work and information will be missing. A project will normally comprise seven phases, each of which culminates in a major review and decision.

Phase 1: Strictly speaking, this phase of work falls outside the Buy/Build method; it is part of the Business Study module of PRISM. The business requirements and problems to be solved by the new system are identified and initial choices are made about the business and project objectives that the system should satisfy, and about whether it is to be package or bespoke development.

Phase 2: In this phase, the feasibility of a package solution is verified. The following tasks are undertaken:

— A search is initiated to identify potential suppliers.

— A request for information is issued, outlining fundamental requirements and constraints.

— A shortlist is compiled.

Phase 3: The preferred supplier is selected:

— Detailed user requirements are specified.

— A request for proposal for the new system is produced and issued.

— Evaluation criteria are agreed.

— Demonstrations of products are attended.

— A supplier is selected on the basis of the evaluation criteria.

Phase 4: In this phase, an understanding of the scope of required modifications to the preferred package is developed. Modifications are planned and undertaken, using the appropriate PRISM development module, to arrive at an overall project plan:

— Identify package scope.

— Identify build scope.

— Develop preliminary installation approach.

Phase 5: The installation of the new system is planned:

— Identify the conversion and interim maintenance and support requirements.

— Place contract.

— Review package and acceptance criteria.

— Prepare environment.

— Initiate training.

— Finalise installation preparations.

Phase 6: The new system is installed, tested, and integrated with other systems:

— Test design.

— Initiate and test package software.

— Integrate package and other software.

Maintenance arrangements are agreed.

Phase 7: Installation is completed with acceptance tests, followed by cut-over to the new system:

— Conduct acceptance test.

— Cut-over to new system.

— Finalise installation.

Figure 3.11 LBMS Package Evaluation covers the selection of an application package and planning for its implementation

LBMS Package Evaluation is structured into five activities, with three of the activities — requirements analysis, package selection, and package documentation — overlapping. This means that the list of potential packages can be progressively reduced as the requirements are defined in more detail. The method used by LBMS Package Evaluation places initial emphasis on the development of a technical understanding of the business requirements, followed by a detailed understanding of the potential packages. The two can then be matched. The five activities are as follows:

Activity 1: Project initiation
The scope of the project is defined, its cost is estimated, and resources are assigned. Following this, the market is 'trawled' for potential packages. An initial screening mechanism, called Strategic Requirements Comparison, based on defined and agreed mandatory requirements, is performed to provide a shortlist of likely packages. The aim of this activity is to identify those packages that are suitable for further, more detailed, analysis.

Activity 2: Requirements analysis
This activity is based on standard analysis steps, which allow the current system to be investigated, requirements for the new system to be defined, and 'business system options' to be produced. The details of the required system form the basis of the work in Activity 3.

Activity 3: Package selection
In this activity, the potential packages are compared, and a shortlist is compiled:

— 'Functional comparison' identifies the packages that are best able to meet the basic functionality requirements, and cope with volumes or transactions and file sizes. A shortlist of up to three packages is compiled and each is scored.

— 'Data comparison' ensures that the shortlisted packages can handle the organisation's required data structures, content, and format. Scores are again allocated to each package.

— 'Process comparison' is a more detailed check that the required processing is present in the potential packages.

All three steps identify, document, and specify the modifications that will be required to the packages, and their impact (both technical and organisational).

Activity 4: Package documentation
This optional activity consists of two steps and should be carried out only if the documentation provided by the supplier is insufficient to complete Activity 3. The first step, which should be undertaken only if absolutely necessary, provides a mechanism for deriving the logical data structure to be used by the package (used in the 'data comparison' step of Activity 3). In the second step, the project team produces its own package documentation for the 'process comparison' step of Activity 3.

Activity 5: Implementation planning
This activity assesses the full implications of procuring and implementing the package in terms of:

— Hardware and system software needs.

— Customisation requirements.

— Organisational impact.

— Interface and data collection/conversion needs.

— Once-off and continuing costs.

— Implementation timescales and resources.

Collating and reviewing all such information ensures that the final decision to purchase a particular package can be taken with confidence.

Each method also has its unique features, however. These differences should be carefully considered by any organisation wishing to introduce a proprietary method. The main differences are tabulated in Figure 3.12, overleaf.

Figure 3.12 The three proprietary methods have many distinctive features

	Product		
Feature	**SIIPS**	**Buy/Build**	**LBMS Package Evaluation**
Level of user involvement	High	High	Low, except at requirements definition
Emphasis on requirements, user and technical	Primarily user	Primarily user	Business requirements defined in technical terms
Method of involving suppliers	Suppliers provide information and analysis in response to invitation to tender	Suppliers provide information in reply to request for information and request for proposal	Information provided by suppliers and interpreted by the project team
Decision-making process	Decision made by scoring of weighted requirements	Decisions made progressively as more information received	Quick move to detailed investigation of leading contender
Implementation coverage	Implementation covered in detail	Implementation integral part of project	Implementation planned
Links to structured methods	Can be used with any structured method, but based on KPMG's SDLC method	Links to other parts of PRISM	Links to LSDM
Consideration of bespoke development	A choice is made between bespoke and package development	Can be considered at end of each stage, if appropriate	Can be considered at end of each stage, if appropriate
Flexibility to assess different types of package	Suitable primarily for large, complex applications; would need modification for use with other types of system	Very flexible method that enables a project to be 'tailor made'	Detail of stages can be modified as necessary; some guidance given

Each of the proprietary methods is appropriate in different circumstances

Each proprietary method is appropriate in certain cases. If a PEP member needs a very flexible approach, Buy/Build would be the most appropriate choice. If the intention is to select mainly large, complex packages, SIIPS would be more appropriate. LBMS Package Evaluation is most appropriate for organisations that require a detailed technical understanding of packages and that have complex interface requirements.

In this chapter, we have looked at the ways in which application packages should be selected. The benefits of using a chosen application package can be realised, however, only if any modifications required are carried out correctly and if it is successfully implemented. We address these issues in the next two chapters.

Chapter 4

Modifying application packages may be a viable option

The application package that is selected in the process described in Chapter 3 should be a good 'fit' with the requirements of the business, but it is unlikely to be a perfect fit. In such cases, making modifications to the basic product may provide a better business solution. As with a bespoke system, however, modifying a package is more difficult than developing functionality from scratch. (We know from the PEP database that projects that involve changing or enhancing existing systems are typically less productive than new developments.) The viability of modifying a package will be determined by three main factors — company policy, the cost likely to be incurred, and the nature of the modifications that are required.

COMPANY POLICY MAY PRECLUDE MODIFICATION

Some organisations allow no modification to packages, insisting that a package will be used only if it has a high degree of 'fit' with the organisation's requirements; alternatively, the organisation may be prepared to modify its working practices to accommodate an application package. These organisations believe that greatest benefit derives from using packages in the form in which they are supplied rather than spending time and effort on modifying them.

Some organisations have experienced problems with modified packages

Several organisations we spoke to said that they now place less emphasis on modifying packages than they did in the past. One large financial services company told us that its previous policy had been to make all the modifications necessary to application packages to give the users exactly what they said they wanted. This policy had recently been changed, because of the extensive support problems that it had experienced with the modified products, and in particular, the cost of enhancing modified packages.

A variation of this policy is to persuade a package supplier to base his package on the organisation's specific requirements. Clearly, this option is not available to all organisations, but by developing this type of relationship with a package supplier, an organisation would never incur the costs of modifying a package and would have its specific requirements met.

Others will modify any of their packages

At the opposite end of the scale, some organisations have taken a strategic decision that any of their application packages may be modified. This policy is typically chosen by organisations that place the emphasis on providing users with a system that fully meets their requirements, regardless of cost. Such a policy might also be adopted in order to provide a standard user interface — for instance, always amending packages to provide screen layouts that are consistent with other applications.

THE TRUE COST OF MODIFYING A PACKAGE MUST BE CALCULATED

Assuming that the option of modifying application packages is not precluded by company policy, it is essential that the project manager has a clear idea of the true cost involved in modification, because he has to strike a balance between the need for a package to fit business requirements precisely, and cost. Some development managers believe that the effort needed to modify packages so that they can interface with other applications and can match users' needs is such that the total cost of the package and the modifications often makes it more cost-effective to develop the equivalent bespoke application. Many organisations that has invested heavily in application packages are incurring increasing costs and suffering slipping timescales because of the level of modifications that has to be carried out to deliver the required systems.

Modifications to packages can result in increased costs and time slippage

Without doubt, modifying application packages can be both expensive and time-consuming. Indeed, one PEP member who had taken a strategic decision to invest in application packages rather than bespoke system development had experienced so much difficulty and expense in modifying packages that he was considering reversing the decision and allowing only bespoke system development.

It is even more difficult to estimate the costs of modifying an application package than of developing a bespoke system, because the systems development department does not usually have a detailed understanding of the product to be modified, and consequently, cannot accurately estimate the work involved. The issue is complicated because the department is typically faced with the choice of undertaking the modifications using in-house resources, or getting the package supplier to make the necessary changes.

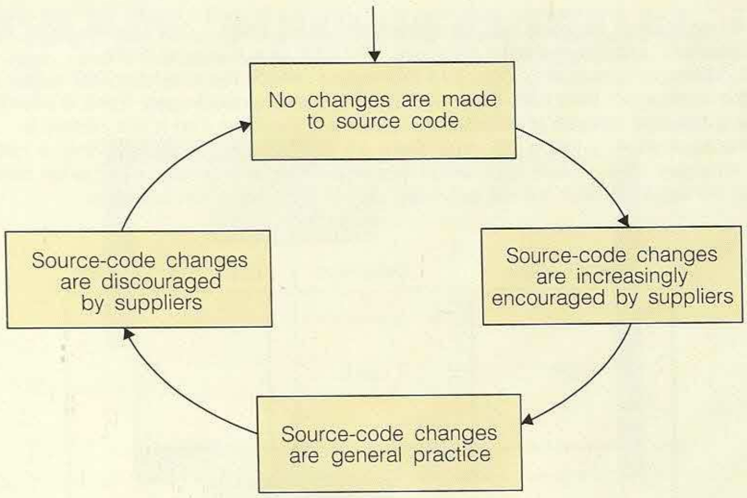
Several organisations we spoke to during our research make decisions about application packages with reference to a rule of thumb; if 80 per cent of the required functionality can be provided by the package, it is worth modifying to provide the remaining 20 per cent of requirements. Clearly, this rule does not accurately reflect the work or the cost involved in modifying a product, but applied pragmatically, it can be a useful basis on which to decide whether it is worth acquiring and modifying a package. Depending on the design of the package, it may be both costly and time-consuming to make relatively small changes. However, if the fit of the application to business requirements has to be high, and if the business procedures cannot be changed to conform with the package, modification can be a viable option.

Some organisations use an 80/20 rule of thumb to decide if it is worth modifying a package

MODIFICATIONS TO THE PACKAGE CORE SHOULD BE DISCOURAGED

The attitude of application package suppliers to modifications made to the source code of their products is changing, and in general, seems to have come full circle, as illustrated in Figure 4.1. This changing attitude is determined by an issue that is closely linked to package modifications — continuing product support.

Figure 4.1 Suppliers' attitudes to having their products modified have come full circle


No changes are made to source code
Source-code changes are increasingly encouraged by suppliers
Source-code changes are general practice
Source-code changes are discouraged by suppliers

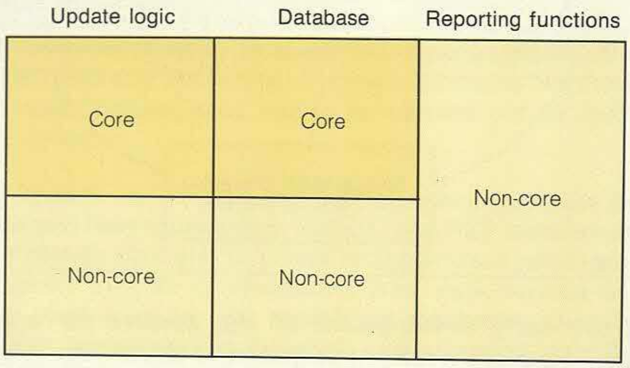
During the early development of the market for application packages, suppliers were not prepared to allow their products to be modified. Some subsequently realised that by either tailoring their products for particular customers, or allowing customers to change packages themselves, they could offer a product that provided a better fit with business requirements, and hence, their product gained a competitive advantage over those of other suppliers. To compete, other suppliers followed this trend. However, when package suppliers enhanced their product range by adding functionality in new product releases, they found that support problems increased for users who wanted to upgrade their non-standard versions of the products. This resulted in suppliers discouraging or forbidding changes to the source code of the package core. This trend has now been reversed. As suppliers have enhanced their products and begun to provide parameterised and flexible packages with advanced tailoring tools, they have stopped making specific amendments to their products for individual customers, and preventing customers from making their own.

First-generation packages have a large core component and a small non-core component

The concept of the package core is illustrated in Figure 4.2, overleaf. Application packages contain both core and non-core components, and the more advanced fourth-generation packages available today have a core that is smaller than the non-core components. First-generation packages have limited reporting facilities, with a database structure and processing logic that provides only the function necessary for the standard applications. Usually, a small number of parameters is available. Second-generation packages are similar to basic packages, except that they have several modules that provide additional functionality in areas other than the standard application. As with first-generation packages, a small number of parameters is usually available. Third-generation packages use parameters extensively to allow areas of the product that are not part of the standard application to be tailored to meet an organisation's specific needs. Fourth-generation packages allow all the components, other than those fundamental to the standard application, to be modified to meet an organisation's specific needs, using modern development tools.

Figure 4.2 The core of a package comprises key elements of the database and update logic

An application package can be divided into three areas — the update logic, the database, and the reporting functions. The core of a package is the fundamental processing logic and information held in the database that is key to the application. With both third- and fourth-generation packages, there is usually a substantial amount of information held in the database that is not central to the application. If this is the case, there will also be processing logic that is not necessary for the basic application. Reporting functions access information held in the database and are not generally part of the core of the package.



If an organisation's requirements are likely to change, it is therefore usually advisable to select a later generation of package, which can be changed in line with evolving requirements.

The later the generation of package, the easier it is to modify

Figure 4.3 illustrates the types of modifications that can be made:

- First-generation, basic packages can have their few parameters changed. In addition to this, the package core can be changed by modifying its source code, or an addition can be made to the package, which may affect either the core or the non-core components.

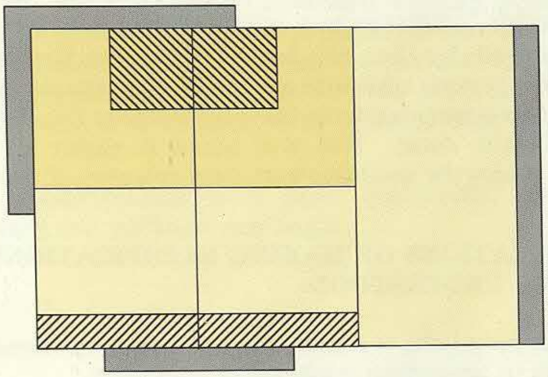
- Second-generation, modular packages can have their few parameters changed in either the main product or the accompanying modules. The core of both the main product and the modules can be changed by modifying the source code, and additions can be made to both the main product and the modules. With modular packages, it is possible to add a new module.

- Third-generation, parameterised packages have a relatively small core. The parameters can be changed and the core can be modified by amending the source code. Again, an addition can be made that affects either the core or the non-core components.

- Fourth-generation, flexible packages provide considerable scope for amending the non-core components by using modern development tools. The core can again be changed by amending the source code, and additions can be created that affect either the core or the non-core components.

Figure 4.3 Different modifications can be made depending on the generation of the application package

Note that the relative size of the core and non-core components varies according to the generation of package. First-generation packages have a large core component, whereas fourth-generation packages have a large non-core component.



■ Additions made to the package, affecting either the core or the non-core components. For second-generation, modular packages, an addition could take the form of a completely new module.

▧ Changes made to the source code of the package core.

▨ Modified non-core parameters, except in the case of fourth-generation packages, where the non-core parameters can be modified using modern development tools.

PEP members should avoid changing the source code of the package core

Our research shows that almost 90 per cent of PEP members had modified the source code of application packages, although such modifications are carried out infrequently by about 60 per cent of members (rather than occasionally or generally). We believe, however, that PEP members should avoid changing the source code of the core of a package. If the functionality provided by this fundamental part of an application package is in question, the organisation has probably chosen the wrong package. If changes to the core of a package are essential, however, they should be carried out by the package supplier because they will alter the very heart of the product. The supplier should always approve of the proposed modifications and should agree to support the amended product.

It is usually better for in-house staff to carry out non-source-code changes

The results of our survey also showed that many PEP members are willing to make changes that do not involve changing the source code of a package — that is, by changing parameters and making additions to packages. About 40 per cent of PEP members reported that they had made these types of changes. However, we suspect that this figure is understated because it does not include those instances where the setting of package parameters is carried out by users without the knowledge of systems staff. PEP members reported adopting several policies towards undertaking non-source-code changes to packages. By far the most popular was for the organisation's own staff to carry out this type of change. This is usually the best option for making additions to a package, because the organisation can control the development and is not relying on a third-party supplier.

Setting package parameters is usually fairly straightforward. Parameters usually fall into two types — those that change the way the package can be used, and those that affect the technical environment — file sizes, for example. Users should be encouraged to set the first type of parameters. However, security procedures should be established to prevent users accessing the second type of parameters, which should be controlled by systems staff.

Where possible, additions to a package should be structured as standalone modules that can access data from the proprietary product. Where such additions are made, the organisation should try to obtain a guarantee from the supplier that the data formats will remain the same. This will make it easier to integrate organisation-specific modules with new releases of the package.

THE IMPLICATIONS OF MAKING MODIFICATIONS SHOULD BE UNDERSTOOD

To help PEP members, we make the following recommendations with regard to amending application packages:

Time estimates for package modification are more important than cost estimates

— Always make early estimates of the modifications required and build them into the business case for the project. Time estimates are more important than basic cost estimates, as the main benefit of using an application package should come from reduced timescales for delivering an operational application.

— Get estimates from as many sources as possible, including, as appropriate, estimates from the package supplier, in-house estimates, and estimates from any consultants or other third parties who may be involved. This should provide a more reliable assessment of the work involved than a single source of information.

— Do not make source-code changes to the core of the package. If the core functionality is in question, and the basic way the product works needs changing, the package may well be unsuited to the requirements.

— Do not make changes to the source code of the non-core components unless they are absolutely necessary. If possible, get the package supplier to carry out any such source-code modifications, because this will avoid the need to spend time understanding the package before the changes can be made. Undertake source-code changes in-house only with the full knowledge of the supplier and with his commitment to future support.

With a modified package, the supplier should give assurances that he will not change the data structures

— Ensure that additions to packages are designed as individual modules to provide specific functions. Structure these modules so that they can use the predefined data structures and formats from the basic application. Involve the supplier, who should be able to give guidance about how best to design these additional modules. Seek assurances that the data structures will not be changed in future releases.

— Where possible, encourage users to make additions to packages. Many packages are now supplied with advanced tailoring tools, such as package-specific fourth-generation languages, screen painters, and report writers. As the users are the customers for the package, they should shape the way it works and looks. However, some of these tools are complex and more appropriate for use by systems development staff.

- Make users responsible for setting parameters that affect the way they use the system. Establish security procedures to restrict access to any system parameters that will affect the technical environment in which the package operates — for example, changing file sizes, and so on. These technical options should be controlled by systems staff.

Once the modifications have been completed, the task of implementing the package can begin.

Chapter 5

Application packages must be correctly implemented

An organisation may select the 'right' application package, but if it is implemented in an inappropriate way, the potential benefits, in terms of reduced timescales and costs, will not be realised. While about 40 per cent of PEP members have a formal method for implementing packages, the majority report that they use identical procedures to those used for implementing bespoke applications. There are, however, significant differences between package-based and bespoke developments, which mean that it is not usually appropriate to implement them in the same way. Nor is it appropriate to implement all types of application packages in the same way. A package-specific implementation method is required that is flexible enough to be adapted in line with the particular generation of package that has been selected.

A package-specific implementation method is required

DIFFERENT IMPLEMENTATION APPROACHES ARE APPROPRIATE FOR PACKAGES AND BESPOKE DEVELOPMENTS

The differences between an application package project and a bespoke system development project have significant implications for the way in which each is implemented. The five areas in which the two types of project differ significantly are:

- The emphasis of development work.
- Project team roles and responsibilities.
- The type of testing.
- The use of third-party services.
- Systems sign-off.

THE EMPHASIS OF DEVELOPMENT WORK

The work necessary to develop a system with an application package is considerably different from that involved in developing a bespoke application. In an application-package-based project, the emphasis of work is not on developing a system but on establishing the facilities already contained within the package.

Assuming that no modifications are necessary, the implementation stage in a package-based project is the equivalent of the design, development, and implementation stages of a bespoke system development. Like the design stage in a bespoke system development, the implementation stage in any application-package-based project comes after the requirements have been defined. The difference is that, with a package-based project, not only have the requirements been defined, but a further stage of work has been carried out to select the product that will be used.

Implementing a package is equivalent to designing, developing, and implementing a bespoke system

In both types of project, implementation will normally be followed by a systems-acceptance stage, and a post-implementation review.

When an application package is used as the basis of a system that has several functions, it is much easier (compared with bespoke development) to carry out the design and implementation stages for each function concurrently. This means that it is possible to reduce considerably the overall timescale for delivering the full system. On a large project, it may even be possible for separate teams to be responsible for providing each of the functions.

The package has to be established to provide the required functionality

One of the main benefits of using an application package is that expenditure on the main-build stage of development is replaced by payment of a fixed sum for a known amount of functionality, which can be delivered within known timescales. The package will, however, still have to be established in a way that provides an operational system. Systems management must be aware of what type of work will be necessary, and make sure that appropriate staff and other resources are available to undertake it. The organisation will have to decide how it is going to use the framework of potential facilities, and put these in place — in other words, it will have to design the new system. In addition, the technical environment within which the package will operate must be established. Examples of work in this area include establishing the necessary job streams, developing restore-and-recovery procedures, and configuring teleprocessing monitors.

PROJECT TEAM ROLES AND RESPONSIBILITIES

Users are more likely to want to play a leading role in package implementation than in a bespoke development, and we believe that this trend should be encouraged. However, difficulties can arise because users typically do not understand as thoroughly as systems staff how to undertake an implementation project. In addition, our research has shown that there is a marked tendency for users to under-estimate the commitment that they need to make while a package is being implemented.

While it is good practice for systems and user staff to work closely on any project, it is particularly important that a good relationship is established on application package projects, especially at the implementation stage. The project can then be effectively managed, realistically planned, and sensibly controlled.

THE TYPE OF TESTING

Several organisations we spoke to during our research do not test application packages; they simply assume that they work. The majority, however, do test packages, but because the package's code has been developed by the supplier and is used in other organisations, testing is often limited just to user testing.

Packages should be tested thoroughly

We believe that all application packages should be tested more thoroughly than this, for the following reasons:

— Suppliers have been known to make mistakes and to deliver the wrong versions of products.

- An organisation's technical architecture may not fully support a specific application package — for example, some organisations that use IBM plug-compatible mainframe computers have had difficulties trying to run leading packages developed for the IBM marketplace.
- Although packages are typically used in many organisations, each organisation will use the package in a different way. With complex products, it is difficult for suppliers to test every combination of parameters and all ways of working.

While testing simple packages should be a straightforward operation, the testing of more complex products should be more extensive. For example, with financial applications, it is not unusual for the new system and the existing system to be run concurrently for an extended period, and for full reconciliations to be undertaken. The implications of this type of testing must be taken into account when planning the implementation. If two systems are to be run concurrently, the workload may be such that additional staffing and machine resources will need to be made available. Overall, however, the time and effort required to test an application package should be significantly less than to test a bespoke system development.

THE USE OF THIRD-PARTY SERVICES

Training and support arrangements for a package are different from those for a bespoke system

The fact that an application package is supplied by a third party, who usually developed the system, almost inevitably means that an organisation will make different arrangements for support, training, and documentation from those that they would make for a system developed in-house. The use of these services should mean that the timescale for systems delivery is shorter than for a bespoke system development. However, the organisation will have to decide which services to use, and plan to make the most efficient use of them. For example, management will have to decide whether all training will be provided by the supplier, or whether to get the supplier's staff to train a small team of the organisation's staff, who will then train the remainder. The requirements for, and timing of, both technical and user training will be affected by the support arrangements that are agreed.

SYSTEMS SIGN-OFF

Systems sign-off is usually more complex and time-consuming for an application package. The involvement of a third party typically makes the process more formal.

All implementation information must be carefully recorded and filed, in case any contentious issues arise

Adequate operational performance of the package-based system will usually have been specified in the contract as a criterion for final payment for the package and for ratification of the contractual arrangements. The relationship with a third party means that all documentation relating to the implementation must be carefully recorded and filed. If any contentious issues arise during implementation, final systems sign-off can be a prolonged process, involving long negotiations between the parties.

Systems management should be aware of these potential problems and take early action to avoid them. For example, if problems occur, they should be notified to the supplier immediately so that appropriate action can be taken. This is preferable to holding back all the contentious issues until the end of the implementation stage.

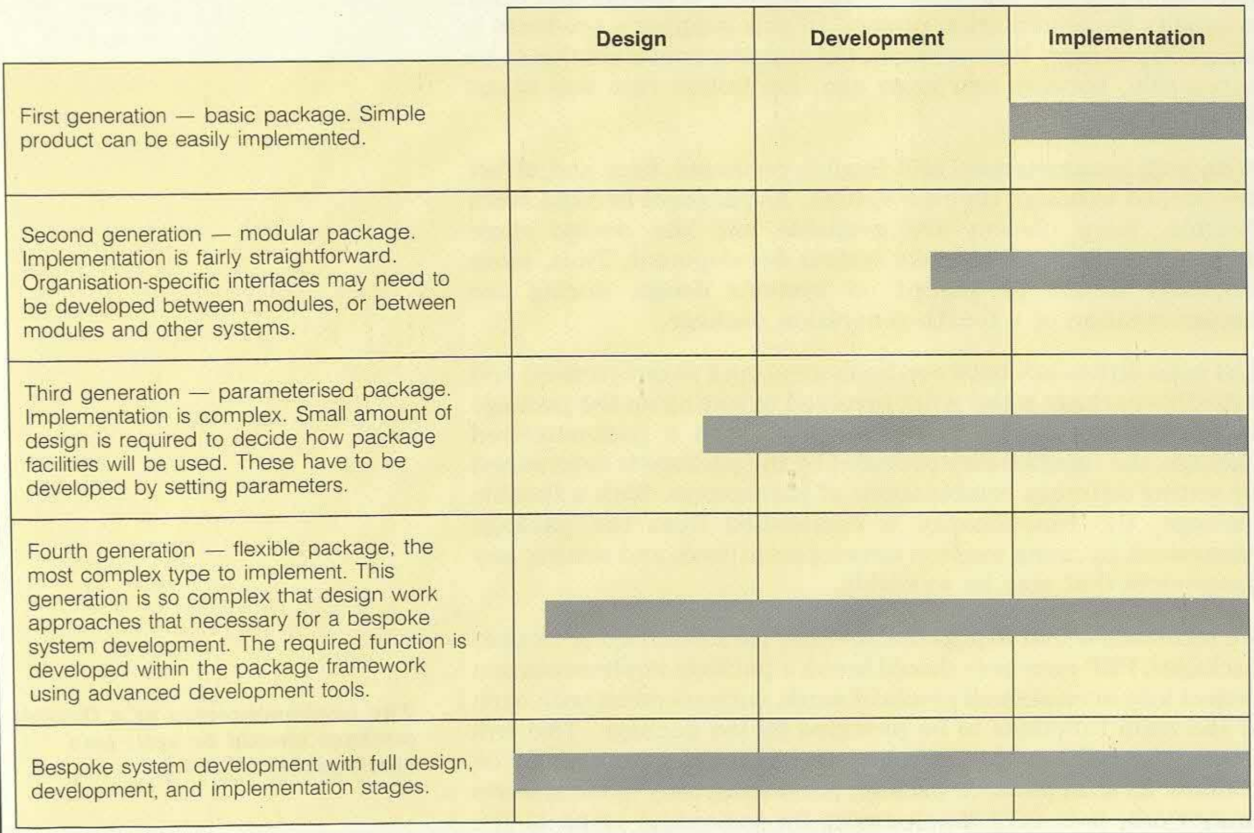
THE TYPE OF PACKAGE SELECTED WILL DICTATE THE APPROPRIATE IMPLEMENTATION METHOD

The generation of package being used determines the most appropriate way to implement it. As Figure 5.1 shows, the more recent the generation of package, the more closely the work involved in implementation approaches that of implementing a bespoke system development.

BASIC AND MODULAR PACKAGES

The implementation of both first- and second-generation packages should be straightforward, following a standard pattern, which consists of installing and testing the software, developing manual procedures, training users, loading or converting data, testing the system, signing off, and live running. The stages of work should be relatively short and simple because this type of package offers little scope for tailoring.

Figure 5.1 The more recent the generation of package, the more closely implementation approaches the development of a bespoke system



First generation — basic package. Simple product can be easily implemented.

Second generation — modular package. Implementation is fairly straightforward. Organisation-specific interfaces may need to be developed between modules, or between modules and other systems.

Third generation — parameterised package. Implementation is complex. Small amount of design is required to decide how package facilities will be used. These have to be developed by setting parameters.

Fourth generation — flexible package, the most complex type to implement. This generation is so complex that design work approaches that necessary for a bespoke system development. The required function is developed within the package framework using advanced development tools.

Bespoke system development with full design, development, and implementation stages.

Although basic packages may seem inflexible and outmoded compared with third- and fourth-generation products, the ease and speed with which they can be implemented often means that they can still provide an effective solution to an application need. If a basic package can meet an organisation's requirements, the benefits in terms of reduced costs and timescales for systems delivery can often be significantly greater than those to be gained from using a more flexible product. However, as we pointed out in Chapter 4, it is much more difficult to modify a basic package than a third- or fourth-generation package.

The speed with which a basic package can be implemented may make it a better buy than a more flexible product

The implementation of modular packages should follow the same basic pattern as that of basic packages. In addition, however, decisions have to be made about which modules of a package to use. The implementation of the modules can be treated as smaller implementations of the basic package. Very often, with modular packages, organisations find it necessary to develop their own interfaces between modules, or between modules and other applications.

PARAMETERISED AND FLEXIBLE PACKAGES

The implementation of both parameterised (third-generation) and flexible (fourth-generation) packages is more complex. One PEP member had experienced considerable difficulty implementing flexible packages. He claimed that he could have provided new systems within shorter timescales and at a lower cost by undertaking bespoke development. We also know of one supplier of sophisticated, flexible packages, who admits that customers find it very difficult to implement his products. We were told that the current failure rate (that is, the product is abandoned after a lengthy implementation project) of this supplier's products is about 30 per cent. Nevertheless, the supplier considers this to be acceptable, because two years ago, the failure rate was about 50 per cent.

The implementation of complex flexible packages is sometimes abandoned

With both parameterised and flexible packages, time and effort are needed to design the new system. As packages become more flexible, more choices are available and the design stage approaches that of a bespoke system development. Thus, more emphasis should be placed on systems design during the implementation of a fourth-generation package.

The main difference between implementing a parameterised and a flexible package is the work involved in setting up the package to provide the required functionality. With a parameterised package, the functionality provided by the package is determined by setting different combinations of parameters. With a flexible package, the functionality is established from the package framework by using modern development tools and setting any parameters that may be available.

We recommend that to implement either parameterised or flexible packages, PEP members should break a package implementation project into several small pieces of work, corresponding with each of the main functions to be provided by the package. This will enable the full benefits of using an application package to be realised. As an application package provides a ready-made systems framework, it is very much easier for individual areas of the

The implementation of a flexible package should be split into small pieces of work

system to be designed, developed, and implemented independently of, and concurrently with, other areas, thereby reducing the timescale for delivering the completed system.

Chapter 6

Getting the best from application packages

Figure 6.1 lists the actions that systems managers should take to ensure that they get the best from application packages.

Figure 6.1 Action checklist

Selecting a package

Assume that any new systems will be based on an application package, unless there are clear reasons for not doing so.

Adopt an informal approach where speed is of the essence, where there is no particular need for the package to fit closely with any existing business requirements, or where the package is small, microcomputer-based, inexpensive, and not strategically important.

Adopt a formal approach where the package is to provide a competitive-advantage, front-office system, where the package is large, mainframe-based, expensive to acquire, or of strategic importance to an organisation's business.

Consider the merits of proprietary package-selection methods before devising one in-house.

If the organisation's requirements are likely to change markedly, choose a more recent generation of package, which will be easier to modify.

Modifying a package

Make every effort to estimate the true cost involved in modifying a package.

Do not make source-code changes to the core of the package.

Where source-code changes to the non-core components are essential, get the supplier to make them, or at least get his commitment to support the changed version.

Make only those modifications that are appropriate for the particular generation of package being used.

Design additions to packages as individual modules to provide specific functions.

Encourage users to make the required non-source-code additions to packages.

Make users responsible for setting the parameters that affect the way they use the system.

Implementing a package

Recognise that a package-based approach to development is quite different from bespoke development, and choose a package-specific implementation method.

Do not assume that packages work: test them all; the more complex the package, the more extensive and more rigorous testing should be.

Do not segregate the design, development, and implementation stages — there is no need to complete the detailed design of the whole system before starting work on development tasks in a particular area, nor does development have to be complete before implementation can begin.

Encourage users to become involved in package implementation, but make sure that they understand the commitment that will be required from them.

Keep all documentation relating to the implementation, for resolving any conflicts with the supplier before final systems sign-off.

For modular packages, treat the implementation of the modules as smaller implementations of the basic package.

To implement either parameterised or flexible packages, create several small implementation projects, one for each main functional area of the package.

Negotiate support arrangements with the supplier.

Chapter 6 Getting the best from application packages

We have discussed in this report the quantifiable and demonstrable benefits to be derived from the use of application packages. These will not, however, be achieved unless packages are selected, modified, and implemented in a systematic manner, and with reference to the needs of each particular business.

The guidelines in this paper will help systems development managers to get the best from packages

Our research shows that the use of packages will continue to increase, as more suppliers make more sophisticated packages available, and as users realise that many of their traditional concerns about packages are unfounded. As packages begin to represent a significant proportion of organisations' portfolios of information systems, it is of paramount importance that the process of acquiring them and ensuring that they work well in a particular environment is efficient and comprehensive. The guidelines provided in this paper should ensure that systems development managers are in a position to do this with confidence.

Appendix

Questionnaire analysis

In this appendix, we summarise the responses to the questionnaire that we circulated to all PEP members about the use of application packages. Seventy per cent of PEP members responded. We have followed the format and numbering sequence from the questionnaire, omitting Question 1, which requested details about the respondent, and the specific project data supplied in answer to Question 5.4.

Question		PEP response
2.	**Use of application packages**	
2.1	Does your organisation use application packages?	98 per cent responded YES
2.2	What types of applications are packages used for?	The use of applications outlined in the questionnaire is shown in Figure A.1. In addition to the applications listed in the questionnaire, 39 per cent of respondents reported using industry-specific application packages.
2.3	For how long has your organisation used application packages?	
		Percentage of respondents
	— Less than two years	6
	— Two to five years	24
	— Five to ten years	37
	— More than ten years	33
2.4	How is the use of application packages changing in your organisation?	
		Percentage of respondents
	— Decreasing quickly	0
	— Decreasing slowly	2
	— Static	9
	— Increasing slowly	64
	— Increasing quickly	25
2.5	Which hardware environments are application packages used in?	
		Percentage of respondents
	— Mainframe	90
	— Minicomputer	67
	— Microcomputer	77

Figure A.1 Packaged software is used for a wide variety of applications

Application	Rank	Percentage of respondents using a package
Project management	1	84
Payroll	2	69
General ledger	3	65
Accounts payable	4 =	61
Personnel records	4 =	61
Pension records	6	49
Asset accounting	7	45
Purchasing	8 =	37
Financial planning	8 =	37
Inventory control	10	35
Accounts receivable	11	33
Telephone accounting	12 =	24
Fleet management	12 =	24
Executive information	14 =	20
Sales order processing	14 =	20
Invoicing	14 =	20
Bills of material	17	18
Production control	18	16
Mailing	19	14
Job costing	20	12
Route planning	21	10
Market research	22	8

2.6 What percentage of your application systems is package-based?

	Percentage of respondents
— Less than 25 per cent	65
— 25-50 per cent	26
— 50-75 per cent	7
— 75-100 per cent	2

2.7 What are the main reasons for using application packages? Please score the following on a scale from 1 (unimportant) to 5 (very important):

	Average of respondents' scores	Rank
— Cost savings	3.48	3
— Time savings	4.21	1
— Guaranteed quality	2.77	4=
— Need for user-controlled project delivery	1.88	6
— Do not want to reinvent the wheel	3.54	2
— Risk reduction	2.77	4=

Other (please specify)

The only other reason quoted more than once was to overcome staff and skill shortages.

2.8 What are the main reasons for not using application packages? Please score the following on a scale from 1 (unimportant) to 5 (very important):

	Average of respondents' scores	**Rank**
— Difficulty in amending package	3.33	3
— Lack of control over future of application	3.19	4
— Non-conformance to data architecture	2.88	5
— No suitable package that meets business requirements	4.27	1
— No suitable package that will run in the hardware environment	3.40	2
— Little knowledge of package market	1.35	6

Other (please specify) — Other quoted reasons included integration problems and doubts about continued supplier support.

2.9 Please tick the box that best describes your organisation's collective knowledge of the application package market:

	Percentage of respondents
— Many packages assessed during the last two years for a variety of applications	27
— Several packages assessed during the last two years for more than one application	71
— No assessment of packages made during the last two years	2

2.10 Has the supply of 'soft' application packages changed your organisation's policy towards buying application packages? — 37 per cent of respondents answered YES

If YES, is your organisation more likely to buy an application package?

All respondents who answered YES are more likely to buy an application package.

3. Application package selection

3.1 Do you have a formal method for selecting application packages?

47 per cent of respondents answered YES

3.2 Do you use external consultants/resources to help with selecting application packages?

43 per cent of respondents answered YES

3.3 Do you have explicit rules for deciding between using an application package and developing a bespoke system?

14 per cent of respondents answered YES

3.4 Do you estimate the full life-cycle cost and time schedule for a package-based application and compare them with the estimated full life-cycle cost and time schedule of the equivalent bespoke application?

53 per cent of respondents answered YES

3.5 Is the requirements-definition stage for a package-based application carried out as thoroughly as for a bespoke application?

65 per cent of respondents answered YES

3.6 Do you undertake a detailed analysis of potential package suppliers?

71 per cent of respondents answered YES

3.7 Do users undertake the following roles in application package projects?

	Percentage of respondents who answered YES
— Initiate	61
— Control	39
— Manage	45

Where YES, would systems staff normally take this role in a bespoke development?

		Percentage of respondents who answered YES to the first question
	— Initiate	40
	— Control	63
	— Manage	55
3.8	When selecting an application package, do you explicitly consider its fit to your organisation's technical architecture? If YES, please tick the box that best describes what you do:	90 per cent of respondents answered YES
		Percentage of respondents who replied YES to the initial question (Note: several respondents adopt more than one policy.)
	— Select only packages that comply with the architecture	36
	— Modify packages to comply with the architecture	14
	— Extend the architecture to include the database, languages, user interfaces, and so on, used by the package	50
	— Extend the architecture to include new hardware to support a package	25
4.	**Package implementation**	
4.1	Do you have a formal method for implementing application packages?	41 per cent of respondents answered YES
4.2	Does your approach to implementing application packages differ from that used for implementing bespoke systems?	41 per cent of respondents answered YES
4.3	Do you use external consultants/resources to help with application package implementation?	65 per cent of respondents answered YES
4.4	Do you compare times/costs for application package implementation with bespoke developments?	57 per cent of respondents answered YES

4.5 Has the source code of your application packages been amended/enhanced to meet requirements?

88 per cent of respondents answered YES

Have the changes been made by:

Percentage of respondents who replied YES to the initial question (Note: several respondents adopt more than one policy.)

— Your organisation	54
— Package supplier	83
— Other (please specify)	13 (usually consultants or agreed third parties)

Are source code changes made:

— Generally	12
— Occasionally	30
— Infrequently	58

How much has your organisation spent on amending the source code of application packages in the last two years?

Average £158,000 per organisation. (Note: the response rate to this question was less than 40 per cent, reflecting the difficulty of obtaining the information.)

What proportion of your systems development budget has been used on amending the source code of application packages over the last two years?

Insufficient data to report reliable average.

Have your staff, or staff from the package supplier, been most effective in achieving the following?

Percentage of respondents for each category:

	Own staff	**Supplier's staff**
— Reducing the risk of amending a package	57	43
— Reducing the cost of amending a package	37	63
— Reducing the associated support problems of amending a package	36	64

4.6 Do any of the changes made to your application packages not involve changes to source code?

42 per cent of respondents answered YES

Have these changes been made by:	**Percentage of respondents who replied YES to the initial question** (Note: several respondents adopt more than one policy.)	
— Your organisation	90	
— Package supplier's staff	48	
— Other (please specify)	4 (consultants or agreed third parties)	
Are these types of changes made:		
— Generally	40	
— Occasionally	42	
— Infrequently	18	
How much has your organisation spent on these types of amendments to application packages in the last two years?	Average of £69,000 per organisation. (Note: the response rate to this question was less than 40 per cent, reflecting the difficulty of obtaining the information.)	
What proportion of your systems development budget has been used making these types of changes in the last two years?	Insufficient data to report reliable average.	
When making these types of changes, have your staff, or staff from the package supplier, been most effective in achieving the following?	**Percentage of respondents for each category:**	
	Own staff	**Supplier's staff**
— Reducing the risk of amending a package	54	46
— Reducing the cost of amending a package	92	8
— Reducing the associated support problems of amending a package	79	21
4.7 Has the supply of 'soft' application packages changed your policy towards enhancing/amending packages?	31 per cent of respondents answered YES	

If YES, do you now have less need to change your application packages?

87 per cent of respondents who answered YES to the initial question also answered YES to this question.

5. Statistics

5.1 Please give your estimate of the cost savings your organisation has achieved by using application packages rather than bespoke development:

	Percentage of respondents in each category	Rank
— Less than 10 per cent	38	1
— 10-25 per cent	34	2
— 25-50 per cent	19	3
— Greater than 50 per cent	9	4

5.2 Please give your estimate of the total cost savings that your organisation has achieved during the last two years by using application packages.

Average of £536,000 per organisation. (Note: only 22 per cent of respondents replied to this question.)

5.3 Please give your estimate of the time savings that your organisation has achieved by using application package rather bespoke development:

	Percentage of respondents in each category	Rank
— Less than 10 per cent	25	2
— 10-25 per cent	34	1
— 25-50 per cent	19	4
— Greater than 50 per cent	22	3

BUTLER COX
P,E,P

Butler Cox

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

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

The services provided include:

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

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

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

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

Education
Through the Cranfield IT Institute (CITI), educating systems specialists, IT managers, line managers, and professionals to understand more fully how to apply and use today's technology.

PEP

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

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

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

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

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

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

Previous PEP Papers
1 Managing User Involvement in Systems Development
2 Computer-Aided Software Engineering (CASE)
3 Planning and Managing Systems Development
4 Requirements Definition: The Key to System Development Productivity
5 Managing Productivity in Systems Development
6 Managing Contemporary System Development Methods
7 Influence on Productivity of Staff Personality and Team Working
8 Managing Software Maintenance
9 Quality Assurance in Systems Development
10 Making Effective Use of Modern Development Tools
11 Organising the Systems Development Department
12 Trends in Systems Development Among PEP Members
13 Software Testing
14 Software Quality Measurement
15 Application Packages

Forthcoming PEP Paper
Project Estimating and Control

BUTLER COX

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

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

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

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

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

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

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

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

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

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