Making Effective Use of Modern Development Tools

BUTLER COX
P,E,P

PEP Paper 10, May 1989

BUTLER COX

P,E,P

Making Effective Use of Modern Development Tools

PEP Paper 10, May 1989
by Kevan Jones

Kevan Jones

Kevan Jones is a consultant with Butler Cox in London, with experience in various areas of information technology, but with particular expertise in systems development. He has been involved in the PEP programme since its inception, carrying out research into the systems issues that are of concern to members, and analysing members' systems development productivity for the PEP programme. He also helped with the research for the Butler Cox Foundation Report, *Network Management*, and is currently doing research for a forthcoming Foundation Report on future technologies.

During his time with Butler Cox, he has also participated in a wide range of consulting assignments. Recent projects in which he has been involved include assessments of the effectiveness and efficiency of the systems development departments of several large clients, a definition of the systems requirements of a network-management system for a country-wide motoring services organisation, and an assessment of the process-control systems of a computer-integrated manufacturing installation for a major oil company.

Prior to joining Butler Cox, Kevan Jones was a member of the systems research team at Thorn EMI Electronics, where he worked on systems analysis, design, and implementation of large projects for the Ministry of Defence.

He has a first class honours degree in mathematics and computing and is an Associate Fellow of the Institute of Mathematics and its Applications.

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

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

BUTLER COX

P,E,P

Making Effective Use of Modern Development Tools

PEP Paper 10, May 1989
by Kevan Jones

Contents

Chapter 1

Exploiting the potential of modern development tools

Vendors claim that modern development tools will provide large increases in productivity

Most vendors claim that development teams will achieve very large increases in productivity by using modern development tools in place of traditional development tools, such as third-generation languages, because modern development tools are easier to learn, reduce the amount of code and testing that is required, are self-documenting, and produce code that requires less maintenance. In many cases, they are also quite modular in nature, with the various modules independent of one another. This, again, makes the maintenance task easier. Analysis of data submitted by PEP members indicates, however, that while most members are achieving some reductions in time and effort, few are consistently achieving the level of improvements that should be possible if the true potential of modern development tools were being realised.

Few PEP members are achieving the level of improvements that should be possible

The benefits of modern development tools are not being fully exploited because systems development departments have failed to understand that the selection and introduction of such tools into the development environment is a very different, and much more complex, process than it was with third-generation tools, which could be used to develop most types of application. There are scores of modern development tools on the market, but each has different design objectives, and each is appropriate for different types of application. Modern development tools cannot therefore be selected in isolation; they must be considered in the context of the wider development environment in which they will be used. What is needed is a clearly defined procedure, which those responsible for selecting, implementing, supporting, and using modern development tools can follow, to ensure that the potential benefits are consistently achieved.

HOW DO WE DEFINE MODERN DEVELOPMENT TOOLS?

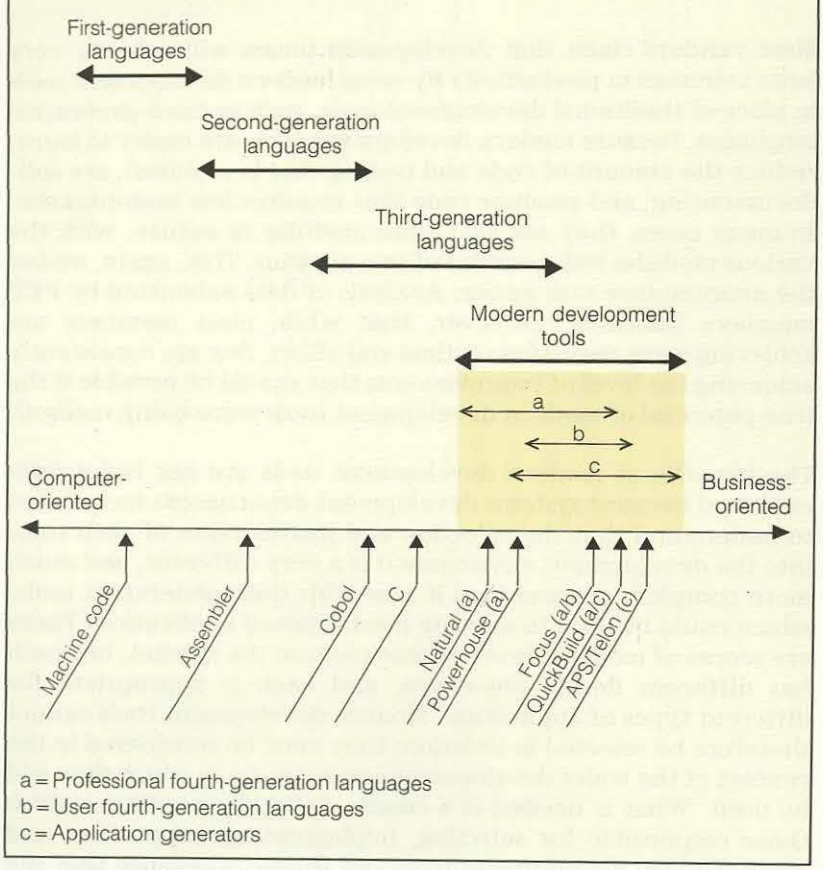
Modern development tools is the generic term for fourth-generation languages and application generators

Since our definitions of terms may not correspond precisely with those of other specialists, or of vendors, it is important to clarify them here. In this paper, the term 'development tools' is used to cover both modern development tools and programming languages. Thus, Cobol, Assembler, Natural, and Telon are all development tools. The term 'modern development tools' includes both fourth-generation languages and application generators, both of which include a wide range of products.

A *fourth-generation language* is a syntax-based programming language in which an application can be written. Fourth-generation languages differ from older languages, such as Cobol, in being more concise (that is, the commands are more powerful), and in not requiring the developer to have detailed knowledge of the underlying computer systems. In fact, each successive generation of programming languages has become more business- or

application-oriented. This evolution is illustrated in Figure 1.1. Most fourth-generation languages require the developer to specify explicitly the order in which operations are to be performed — that is, they are 'procedural' languages.

Figure 1.1 Each generation of programming languages has become more business-oriented



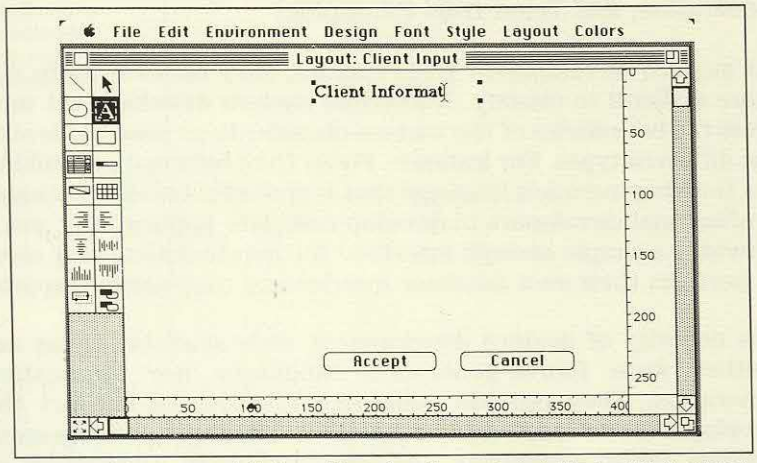
There are two categories of fourth-generation language

Fourth-generation languages can be split into two categories depending on the experience of the intended developer and the complexity of the development interface. The first category contains the most powerful tools, which are used to develop complete applications. These will normally be used by professional developers and will have a complex development interface. Examples of professional fourth-generation languages are Powerhouse from Cognos, and Application Master from ICL. The second category contains development tools that usually permit only access to data and the generation of simple reports, although some do provide additional and more powerful facilities. These will typically be used by end users and have a less complex development interface. An example is Query Master from ICL.

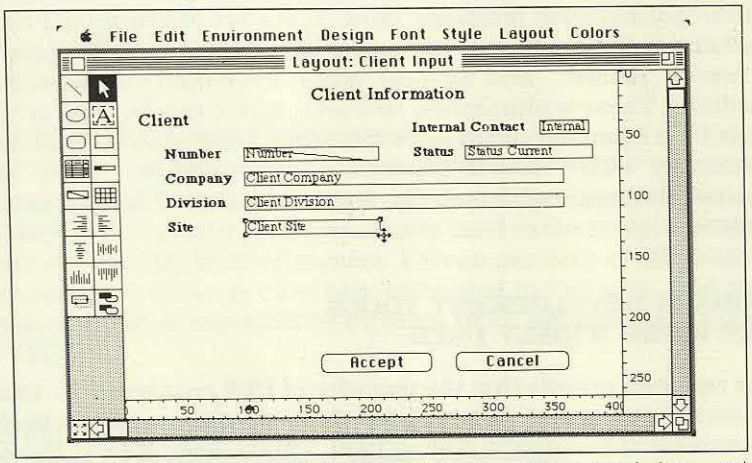
An *application generator* is a screen-based development tool with which applications are developed by interacting with the screen rather than by writing statements of code. The interaction with the screen may be via icons, pull-down menus, screen painters, and so on. Figure 1.2 shows three stages in the development (painting) of an input screen with the application generator, Fourth Dimension, from Analyses Conseils Informations (ACI) UK.

Figure 1.2 Application generators provide developers with numerous facilities, including screen painters

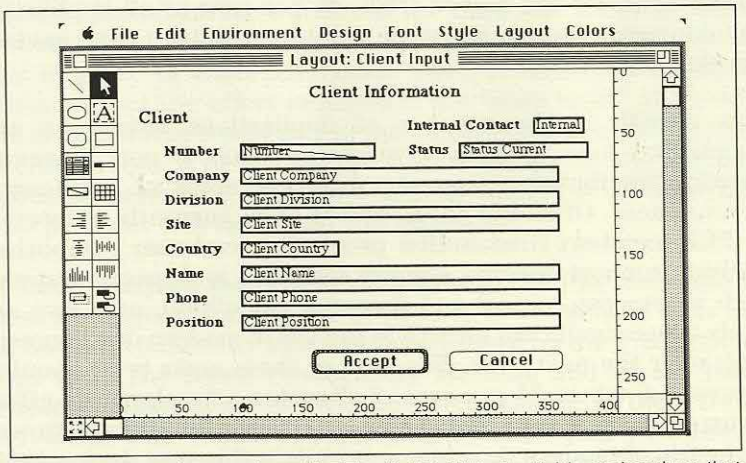
These layouts were created using the screen-painting facility provided by the application generator, Fourth Dimension, from ACI UK.



The control buttons are initially positioned on the screen and the text title is added.



Each data field can then be defined on the screen, using pop-up windows, and the appropriate text is then added.



The finishing touches can be added by highlighting the fields and options that are important.

Application generators are not always purely screen-based; some do have an embedded coding language, such as a fourth-generation language. Application generators are largely non-procedural — that is, the generator rather than the developer decides in what order the defined operations are to be performed. Examples of application generators are APS from Software Generation, and Telon from Pansophic.

As modern development tools mature, they become more and more difficult to classify. Numerous modern development tools cross the boundaries of the various classifications used to identify the different types. For instance, Focus from Information Builders is a fourth-generation language that is powerful enough to enable professional developers to develop complete applications, yet it provides a simple enough interface for non-technical end users to perform their own database queries and to generate reports.

Modern development tools are difficult to classify

The majority of modern development tools available today are neither pure fourth-generation languages nor application generators. They provide additional facilities to support the development of applications, such as database management systems, report generators, and screen painters. An example of this is QuickBuild from ICL. This started life as Application Master, a fourth-generation language. Now, it is a set of integrated tools that also includes an end-user query language, a data dictionary, a screen painter, and several other development tools and facilities. These sophisticated tool sets, which can be used as the basis for a computer-aided software engineering environment, are commonly called fourth-generation environments. They are beyond the scope of this study, but are discussed briefly in the appendix.

MODERN DEVELOPMENT TOOLS ARE BEING WIDELY USED

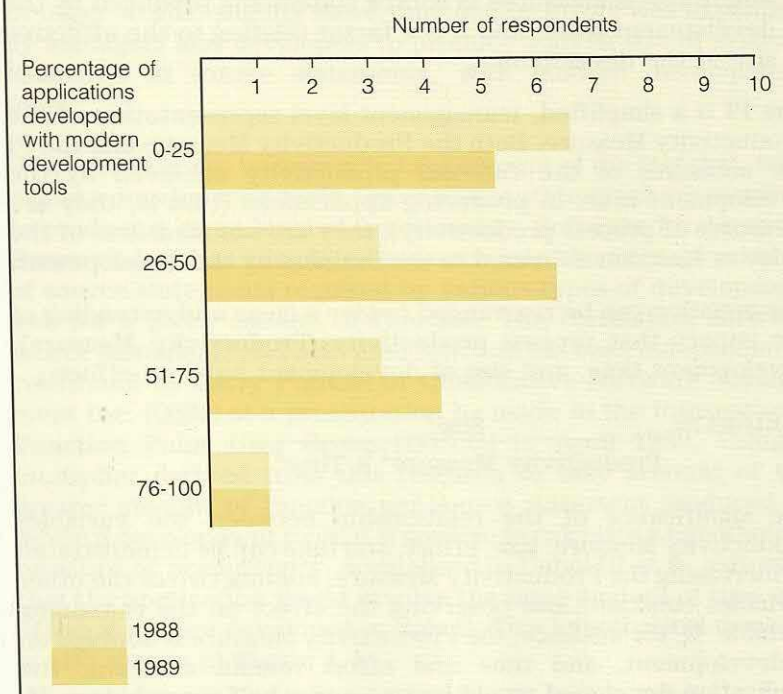
Our research reveals that the majority of PEP members are using modern development tools and that their use of these tools is likely to increase substantially during 1989. This is illustrated in Figure 1.3. Predicted growth in the number of applications developed and maintained with modern development tools this year is over 30 per cent. By the end of 1989, 45 per cent of all application developments and maintenance will be carried out with modern development tools.

The use of modern development tools is increasing significantly

This growth in the number of applications developed and maintained with modern development tools is not, however, equally distributed across the different areas of application development. Of all the development areas currently of interest to PEP members (transaction processing, end-user computing, decision-support systems, systems software, and special systems, such as process control and financial modelling), only two are likely to see significant growth in the use of modern development tools over the next year. The first of these areas is transaction processing, where 75 per cent of all systems development effort is currently expended. PEP members expect a 40 per cent growth in the use of modern development tools in this area during 1989, although their perceived level of success in using modern development tools for transaction-processing applications has

The use of modern development tools is most common in transaction-processing and decision-support applications

Figure 1.3 PEP members predict significant growth in the use of modern development tools during 1989



(Source: Butler Cox survey of PEP members)

varied widely. The second area is decision-support systems, where modern development tools are already used extensively, and with a perceived high level of success. Eleven per cent of all systems development effort is currently expended in this area, and this level of effort is expected to increase to 20 per cent by the end of 1989.

DEVELOPMENT TIME AND EFFORT HAS GENERALLY DECREASED WITH THE USE OF MODERN DEVELOPMENT TOOLS

Most PEP members have reduced the time and effort involved in development by adopting and applying modern development tools in place of third-generation tools. A bank, for example, estimated that the effort required in the 'main build' phase of a new development would be 210 man-days using its existing third-generation tool. It developed the system using a newly acquired application generator, Telon, and built the system in 150 man-days — 30 per cent less time than the estimate.

We have analysed the project data held in the PEP database to assess the impact that the use of modern development tools has had on the Productivity Measure as expressed by the Productivity Index (PI). The terms Productivity Measure and PI are basic measures employed in PEP. A detailed explanation of these terms is given in PEP Paper 5, *Managing Productivity in Systems Development*. For the purpose of this paper, it is sufficient to say that the Productivity Measure is calculated from the time, effort, and size of a development project, using the following equation:

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

where size is measured in source statements produced by the development team, and B is a factor relating to the effective size of the development.

The PI is a simplified, management-level representation of the Productivity Measure. Both the Productivity Measure and the PI are measures of the *internal* productivity achieved by the development team in producing applications (that is, they are measures of process productivity); they are *not* measures of the value or function delivered to the business by the development.

The Productivity Measure and the Productivity Index measure the internal productivity achieved by the development team

The equation can be rearranged to give a clear understanding of the impact that process productivity (Productivity Measure), development time, and size of development have on effort:

$$\text{Effort} \simeq \frac{\text{Size}^3}{\text{Productivity Measure}^4 \times \text{Time}^4}$$

The significance of the relationship between the variables Productivity Measure, size, effort, and time can be demonstrated by increasing the Productivity Measure, holding two of the other variables constant, and observing the effect on the remaining variable. If, for instance, the Productivity Measure is doubled on a development, and time and effort remain constant, the application developed would be two-and-a-half times bigger. If, however, time and size remain constant, one-eighth the amount of effort would be required to develop the application. If effort and size remain constant, the application would be developed in about half the amount of time. Typically, a combination of the above would apply.

The average PIs achieved by PEP members when using third-generation tools, modern development tools, and a combination of these development tools are shown in Figure 1.4. The figure shows that projects developed with modern development tools achieved a somewhat lower PI than those developed with third-

Figure 1.4 The hypothetical measure, normalised to Cobol, reveals the potential of modern development tools

Language type	Average PI	Hypothetical measure*
More than 60 per cent third-generation languages	15.4	15.4
Mixture of third-generation languages and modern development tools	14.9	17.7
More than 60 per cent modern development tools	14.6	19.4

* The hypothetical measure takes account of the fact that applications developed with modern development tools require fewer statements, and hence less effort, to produce the same functionality as applications written in a third-generation language

generation tools. This means that for those PEP members included in this analysis, modern development tools performed less well, in terms of *process* productivity, than third-generation tools. In other words, slightly more time and/or effort was expended by managers and developers to produce a given size of system, measured in source statements, with modern development tools.

The hypothetical productivity measure takes account of the fact that modern development tools require fewer statements

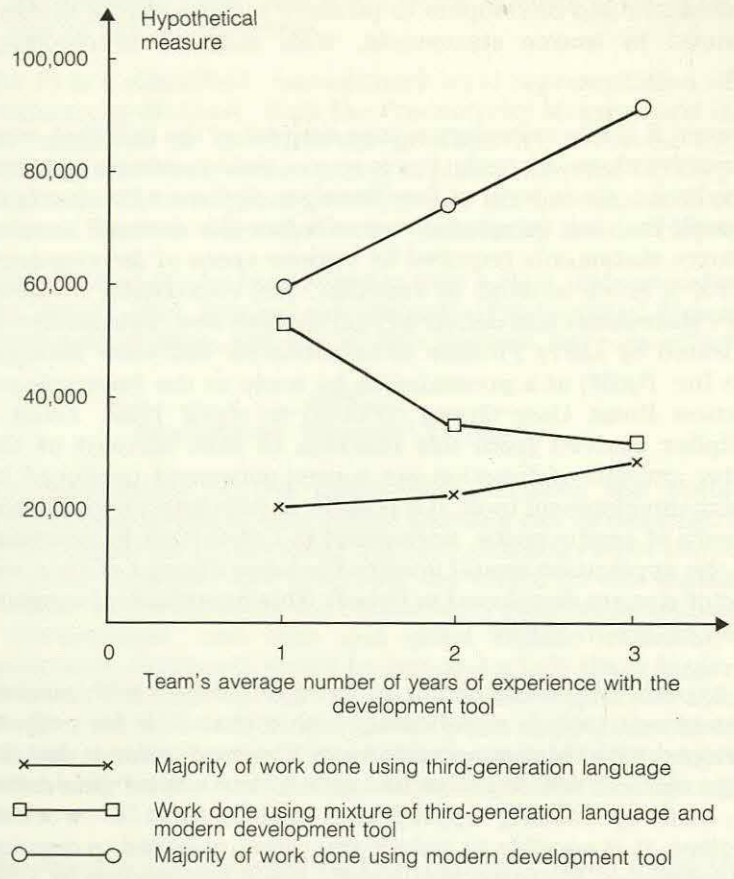
However, it is also necessary to take account of the fact that, with modern development tools, fewer source statements are required to produce a given level of functionality. Software Productivity Research Inc. has completed research into the average number of source statements required by various types of development tool for a given amount of function. The correlation between source statements and delivered function has been independently confirmed by Larry Putnam of Quantitative Software Management Inc. (QSM) at a presentation he made to the International Function Point User Group (IFPUG) in April 1989. Using a multiplier derived from this research to take account of the greater amount of function per source statement produced by modern development tools, it is possible to calculate a hypothetical measure of productivity, normalised to Cobol (that is, assuming that the application would involve the same amount of time and effort if it were developed in Cobol). This hypothetical measure is also shown in Figure 1.4.

The hypothetical measure for projects developed with modern development tools is significantly higher than it is for projects developed with third-generation tools. The implication is that, by using a modern development tool rather than a third-generation tool, and by making appropriate modifications to working practices, it is possible to reduce the effort required to develop an application. Research is currently being undertaken by QSM and Butler Cox to provide a proven and more effective measure of the increase in the functional value, as seen by the end user, that can be made by using modern development tools. This measure will eventually be included in PEP assessments, together with the PI.

The hypothetical measure generally increases as the experience of the development team increases

Our analysis indicates that the hypothetical measure generally increases as the development team's experience with the tool increases. Figure 1.5 overleaf shows how the hypothetical measure for applications that were developed with modern development tools is not only higher than that for applications developed with third-generation tools; it also increases at a faster rate. A development team with an average of one year's experience of using a modern development tool has twice the hypothetical measure of a team with three years' experience of using a third-generation tool. However, the analysis of data in the PEP database, shown in Figure 1.5, indicates that the hypothetical measure for applications developed partly by modern development tools and partly by third-generation tools actually decreases as the team becomes more experienced. We believe that this trend is due to the fact that the most experienced members of the team were those using third-generation tools. At the time of publication, information to confirm this was not, however, available.

Figure 1.5 The hypothetical measure, normalised to Cobol, increases dramatically as experience with modern development tools increases



(Source: Project data submitted by PEP members)

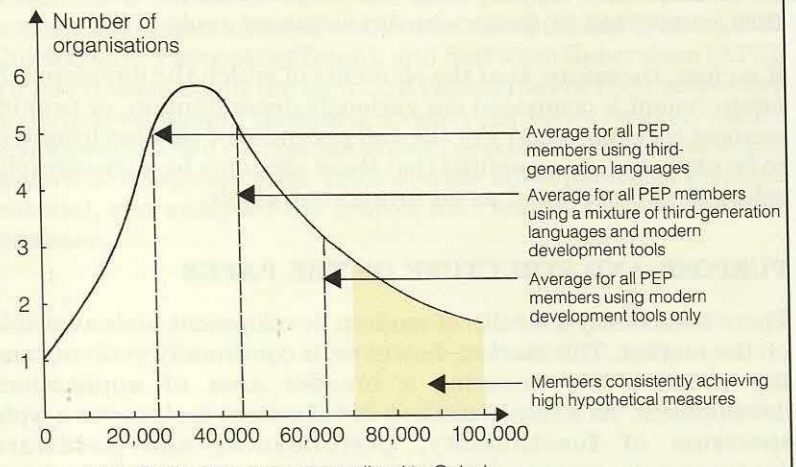
FOR BEST RESULTS, MODERN DEVELOPMENT TOOLS MUST BE INTEGRATED WITH THE DEVELOPMENT ENVIRONMENT

While most PEP members using modern development tools have achieved some increase in the hypothetical measure, only 20 per cent are consistently achieving much higher measures (see Figure 1.6). That is to say, only 20 per cent of members are using modern development tools to reduce substantially the time and effort required to develop an application. Detailed analysis of the project data submitted by PEP members showed that the wide variations in the hypothetical measure did not relate directly to length of experience with the development tool. We believe that the hypothetical measures achieved with modern development tools are simply less consistent than they are with third-generation tools, and responses to our questionnaire confirmed this.

Only a few PEP members are achieving high hypothetical measures

If they are to be used really effectively, modern development tools must be integrated into the development environment — that is, they must be used in conjunction with the right development approach, systems development techniques, and methods, and their capabilities must be matched with the particular require-

Figure 1.6 Only 20 per cent of PEP members are consistently achieving high hypothetical measures



(Source: Project data submitted by PEP members)

There is a complex set of relationships between the elements of the development environment

ments of the application. The complex set of relationships between these elements of the development environment must also be understood.

The term 'development approach' describes the complete life cycle, phases, and activities of the development of an application. Most organisations will be using at least one of the following development approaches:

- The traditional, or conventional, approach, where progress is achieved by proceeding in a linear fashion through each successive phase of the life cycle.
- The iterative approach, where several passes are made through one or more of the phases of the life cycle, with additional functionality and detail being added each time.
- The small-systems approach, where new, small systems or small enhancements to existing systems are required. This approach is typically a variant of one of the major approaches, with smaller project teams and less stringent management techniques.

The development approach used determines the appropriate development techniques

Other approaches, such as the package approach and the end-user-computing approach, may be used by some organisations, but are not widely adopted. The development approach used will, to a large extent, determine the development techniques that can be used, because for each approach, only particular techniques will be appropriate.

The term 'systems development technique' is used to describe the procedures on which systems development methods are based. Examples of systems development techniques are data analysis, functional decomposition, and prototyping. Systems development methods are commercialised systems development techniques. In other words, a systems development method is a way of implementing, in practice, the ideas embodied in a systems

development technique. Examples of systems development methods are LSDM/SSADM from Learmonth and Burchett Management Systems, and Prism, from the Hoskyns Group. The various systems development techniques and methods are, in turn, supported by particular development tools.

The various systems development techniques are supported by particular development tools

It is clear, therefore, that the elements of which the development environment is composed are variously dependent on, or provide support to, each other. For the full potential of this environment to be exploited, it is essential that these elements be very carefully selected and managed as an integrated whole.

PURPOSE AND STRUCTURE OF THE PAPER

There is currently a wealth of modern development tools available on the market. This market, however, is continually evolving and the products are covering a broader area of applications development. As a result, modern development tools cover a wide spectrum of functionality, performance, and hardware environments, making it a very complex task to select and use them effectively.

The purpose of this paper is therefore to offer guidance on how to choose and make effective use of modern development tools, primarily in the areas of transaction processing, both batch and online, and decision-support systems. It is intended for managers charged with selecting and implementing such modern development tools, and for managers and technical staff responsible for their support and use.

In Chapter 2, we explain how an organisation should identify and select the modern development tools that will be available for use by the systems development department. Most organisations will require a set of development tools to develop the full range of applications required by the business. We therefore suggest a suitable procedure for selecting modern development tools to add to a company's resources, including guidelines for compiling the criteria used for selecting them.

In most organisations, the introduction of a modern development tool will require changes to methods, standards, team roles and responsibilities, training, hardware, and the organisation structure. In Chapter 3, we discuss how these changes can successfully be brought about. Failure to plan for the introduction of modern development tools can have an adverse impact on the productivity of the development teams using them.

In Chapter 4, we define a procedure to ensure that the best development approach, techniques, and tools are selected for each individual application, be it maintenance or a new application.

RESEARCH SOURCES

To help us to identify the main areas of concern, we asked PEP members to complete a brief questionnaire. Thirty-six members, forming a representative sample, responded. We subsequently held a one-day focus group to which we invited a cross-section of PEP members to discuss their concerns, successes, and failures. We conducted 21 telephone interviews to gather further

information, and several face-to-face interviews with selected PEP members to gather more detailed information on their use of particular modern development tools. To ensure that we presented a balanced view, we interviewed four vendors of modern development tools — Information Builders (Focus), ICL (QuickBuild), Pansophic (Telon), and Software Generation (APS). We asked them about the technical capabilities of their respective modern development tools, and sought their views on likely future developments. We carried out an independent survey of over 230 modern development tools, reviewed the latest published research material, and analysed the project information held on the PEP database.

Chapter 2

Identifying and selecting modern development tools

As organisations continue to develop a greater variety of applications, they will require a more powerful range of development tools. Selection of the right modern development tools to meet an organisation's current and future application needs is vitally important, but it is becoming more and more complex as more modern development tools are introduced, and their abilities continue to be extended. Every organisation should regularly review the set of development tools that it has available to ensure that it continues to meet the evolving needs of users.

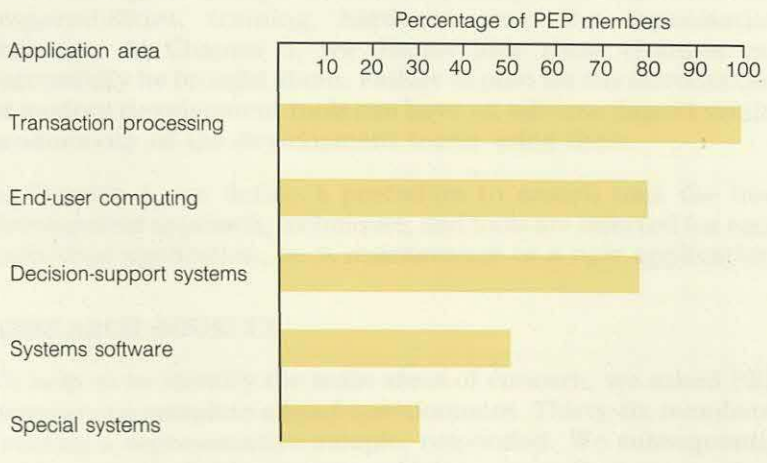
Selection of the right modern development tools is a complex process

RECOGNISE THAT A SET OF TOOLS WILL BE REQUIRED

The role played by the systems development department within most organisations is changing. No longer is the majority of the effort spent developing large transaction-processing systems, with a single, third-generation language. The applications required by users today range from very large and complex transaction-processing applications to very small and simple reporting applications. As a consequence, the systems development department in most organisations is now expected to develop and maintain a growing range of applications using a mixture of modern development tools and third-generation tools. The application areas being supported by PEP members are shown in Figure 2.1. All members are supporting transaction processing, and 80 per cent support end-user computing and decision-support

The demands on systems development departments are continually growing

Figure 2.1 PEP members are supporting a wide range of application areas

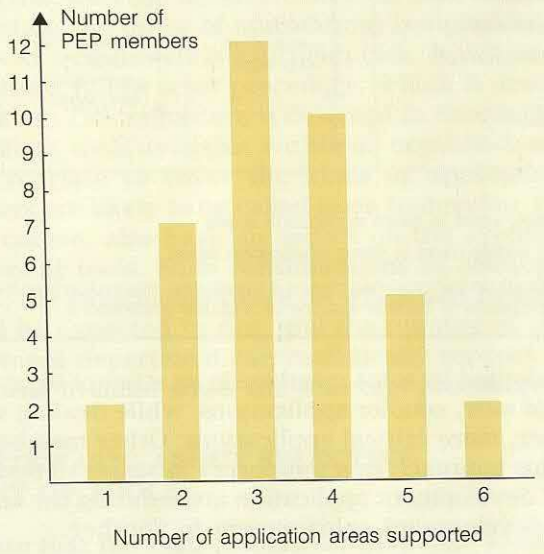


(Source: Butler Cox survey of PEP members)

systems. Our research showed that nearly 80 per cent of PEP members are supporting three or more different application areas (see Figure 2.2). To support these application areas, there are over 230 modern development tools available in the United Kingdom today. Approximately 30 per cent of them are compatible with DEC hardware, 25 per cent with IBM mainframe hardware, and about 8 per cent each with ICL, Hewlett-Packard, and Prime hardware. Thirty per cent are compatible with microcomputers (mainly IBM and IBM-compatibles), a market that has grown dramatically in the last three years. Several of the development tools are able to run on a range of machines from one supplier, or on the machines of different suppliers.

Over 230 modern development tools are available in the United Kingdom today

Figure 2.2 Each PEP member supports an average of three different application areas



(Source: Butler Cox survey of PEP members)

Modern development tools have different design objectives and address different application areas

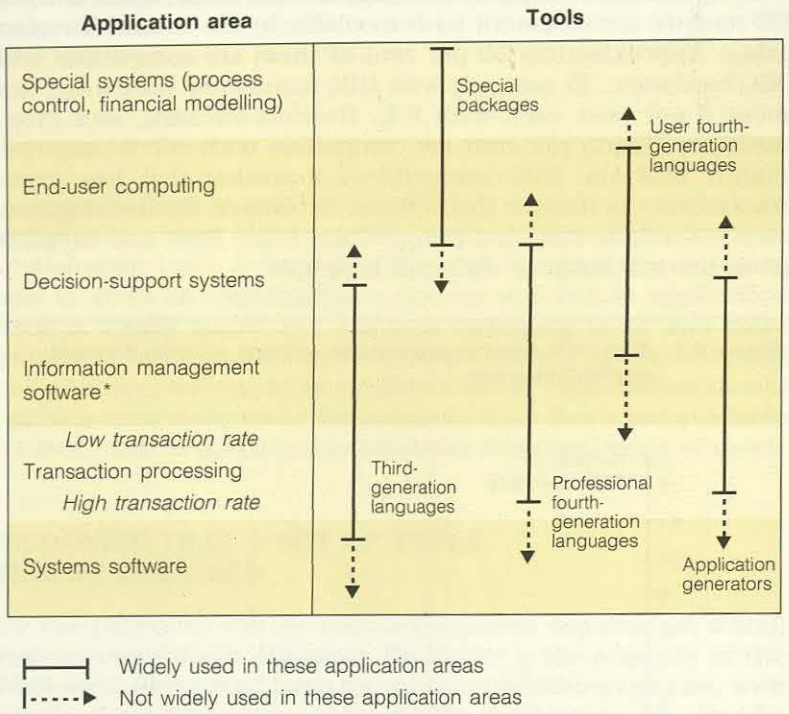
In many respects, this multiplicity of development tools is beneficial; they cover a wide range of application areas and operate on many types of hardware. The problem is that the different modern development tools have different design objectives and address different application areas. Figure 2.3 overleaf shows how third-generation languages, fourth-generation languages, application generators, and special packages (such as financial modelling) provide coverage across the main application types. For most organisations, therefore, no single modern development tool will cover all the application areas in which they have to maintain and develop their applications. To gain full coverage of the range of applications required by today's users, most organisations will need to acquire a set of development tools.

A set of development tools will usually be required

In view of this situation, PEP members have adopted one of three different approaches to selecting and using development tools:

— Restrict the types of applications to those that can be developed with the current development tools. This approach

Figure 2.3 Development tools cover a wide range of application areas



*Covers the applications that provide a means of transferring information from transaction-processing applications to decision-support or end-user applications.

has been adopted by some members in an attempt to try to stem the flood of new, smaller applications, while dealing with the much larger, more critical applications. Other members have adopted this approach as a temporary measure to restrict the number of development application areas during the transition from one development environment to another.

- Use the current development tools for all applications, even if some of them are not ideally suited to a particular application area. This approach is usually adopted by members who have standardised on one development tool. It may be successful if the single development tool enables all the required applications to be developed effectively and efficiently, and it has the advantage of concentrating development skills. It is, however, likely that there has been a compromise between the ability to develop all the required applications, and the effectiveness and efficiency of the department.
- Carefully select a set of development tools that provides coverage for all areas of application development, review the set of development tools regularly, and upgrade it when necessary. This approach ensures that the development tools are available to develop all user requests in the most efficient and effective manner, but it does mean that skills are less concentrated.

A simple way of viewing these three approaches is to consider the systems development department as a carpenter who has been commissioned to build some furniture. The specification requires that some of the joints will be nailed, and others screwed. The three approaches would produce the following results:

— The carpenter refuses to have anything to do with the screws because he has only a hammer. The customer may give him that part of the job that can be done with nails and a hammer, and give someone else the job of putting in the screws. More probably, he will give the whole job to someone else.

— The carpenter accepts the job, even though he has only a hammer, and puts in both the nails and the screws with the tool that he has. This is somewhat clumsy and inefficient, but the job is completed. The customer reluctantly accepts the job, but is not pleased as it does not precisely match his specification, and it took longer than he expected.

— The carpenter has the right tools and accepts the job. He puts the nails in with the hammer and the screws in with the screwdriver. The job is finished according to the specification, with minimal effort and fuss. The customer readily accepts the job.

Ensuring that the right development tools are available to develop and maintain the range of applications required by the systems department's customers is a difficult task. It will be simplified if organisations follow a set procedure, which is described in the next section. This procedure is designed to ensure that the set of development tools available within an organisation is adequate and appropriate to cover the kinds of applications that the developers are likely to be called upon to develop. Other factors will, of course, also have an impact on the eventual choice of development tools. Such considerations as company policy on vendors, the existing hardware on which any new development tool will be expected to run, and the number of tools that the development department can realistically support will all need to be borne in mind when the selection procedure is initiated.

Organisations should adopt a procedure to choose an appropriate set of development tools

BE SYSTEMATIC ABOUT SELECTING THE SET OF TOOLS

To ensure that the right modern development tools are selected to develop and maintain their current applications, organisations should adopt the following procedure:

— Assess how well the current set of development tools covers the types of application likely to be developed.

— Identify and select new development tools to fill any gaps in the coverage, or to improve the current set of development tools. Remove those development tools that are no longer required or that duplicate coverage. Clearly, this will be a policy decision; a development tool cannot be removed if it is still needed for maintenance or operational running.

— Review the coverage provided by the tools at regular intervals, especially if several applications have run into problems, or if an application has attracted an unusual number of complaints.

ASSESS THE APPLICATION COVERAGE OF THE CURRENT SET OF DEVELOPMENT TOOLS

To assess how well the current set of development tools covers an organisation's needs, the past, current, and future applications must be defined and characterised. The capabilities of the current development tools also need to be defined.

The characteristics of the applications will vary from one organisation to another. However, we recommend that, for the purposes of this exercise, the applications are characterised with respect to the requirements that may be met by the use of modern development tools. For the majority of applications, between 10 and 20 characteristics can be derived, along the lines of those listed in Figure 2.4. These can be amended and supplemented to suit an individual organisation. Each characteristic should be defined in a clear and simple manner so that every reader will have the same understanding of it. Characterising applications in this manner will simplify the task of ensuring that the modern development tools available cover all likely applications.

The characteristics of typical applications need to be defined

Figure 2.4 To select a set of development tools, an organisation needs to define the characteristics of the applications it develops

Application type	What type of application is it — for instance, transaction processing, end-user computing, decision-support systems, and so on?
Level of integration	What level of integration is likely to be required between the application types, databases, and machine environments?
Performance requirements	How critical is the response time of the application?
Urgency	How urgent is the development of the application, and how fixed are the deadlines for installation?
Hardware	On what hardware will the application be expected to operate?
Security	What level of security must the application provide for access to the application itself and to the data?
Volume of data	What is the likely total volume of data?
Size	What is the likely total size of the application?
Complexity	How difficult will the application be to develop, in view of its complexity?
Compliance with standards	What essential application standards must the application meet?
Interface with end user	What degree of familiarity will the users have with the system?
Flexibility	What is the likely extent and frequency of change?

An adequate set of definitions can be compiled by characterising all new and maintenance projects from the past two years. The types of application that are likely to be required in the future can best be assessed with reference to the current backlog, and in discussions with users about trends they see developing in their business area over the next two years.

The strengths and weaknesses of the various development tools should be listed, with reference to the particular applications for which they are used. In doing this, it may be necessary to add application characteristics that were originally overlooked. By matching the strengths and weaknesses of the current development tools with the various application characteristics,

it will be possible to identify where there are weaknesses in the application coverage, where there may be inadequate coverage in the future, and where there are areas of overlap. Such weaknesses, lack of coverage, or areas of overlap may be due to numerous factors, such as the inability of the development tool to provide the required function, or the poor productivity gained with the development tool in a particular application area. The eventual result of assessing the coverage provided by the current set of development tools may be to add new modern development tools, but it is equally likely that existing development tools might be removed, where they duplicate coverage, or where they cover an area of application that is no longer of concern to the user and that therefore requires no further maintenance.

The strengths and weaknesses of the current development tools should be assessed

ADOPT A TWO-PHASE APPROACH TO SELECTING NEW MODERN DEVELOPMENT TOOLS

To ensure that the modern development tools are fairly assessed and that the selection procedure is completed in a consistent and timely manner, we recommend that a plan be drawn up, specifying the budget and timescale for the exercise, and allocating responsibilities for each aspect of it. Initially, the plan will contain broad guidelines only; as the evaluation proceeds, the details can be refined and incorporated into the plan.

A plan should be drawn up, including budget, timescales, and responsibilities

We recommend that the selection procedure for modern development tools be carried out in two phases. The initial list may contain scores of possible products, especially if an organisation is using IBM or DEC hardware. A phased approach will ensure that all potential products are assessed systematically, with a minimum of effort.

Phase 1: Create a shortlist

Application of the criteria will reduce the shortlist to a manageable number

The objective of the first phase is to reduce the initial list of potential modern development tools to a shortlist of two or three to be retained for further investigation in Phase 2. The selection criteria are those that the modern development tool *must* satisfy. For example:

- The modern development tool broadly provides the required functionality or will enable the required applications to be developed. (A more detailed assessment will be carried out in Phase 2.)
- The modern development tool supports the development approach.
- The modern development tool runs on the required hardware and operating system.
- The modern development tool's user/developer interface matches the intended user/developer profile.

If any one of these criteria is not met, that modern development tool will be eliminated from further consideration.

The most cost-effective way of evaluating the large number of modern development tools in the initial list is for an organisation to discuss its requirements and its current development environment with the vendor of each tool. Only when the vendor has a clear understanding of a customer's requirements can he comment on the ability of his modern development tool to meet

them. If his tool is not suitable, he may be able to suggest products from other vendors which, combined with his, could provide complete coverage for a particular type of application. In turn, the vendor should supply references, ideally of organisations that are using the same hardware and developing similar applications with his modern development tool. Either by visiting those organisations, or by discussing the modern development tool with developers there, it should be possible to decide whether or not it meets the criteria and whether the vendor's claims can be supported.

The vendor should be involved, wherever possible

Phase 2: Evaluate the shortlisted products

Phase 2 is a more detailed analysis of the modern development tools identified in Phase 1, with greater emphasis on assessing their suitability for a particular development approach and a particular area of application. This assessment should be made with reference to such factors as the cost of the modern development tool, its shortcomings and special features, its likely impact on the development environment, and its future prospects. Figure 2.5 lists some of the more detailed questions that need to be resolved as part of this assessment. This detailed analysis should be coordinated and managed to ensure that the modern development tools are fairly judged. Any response from the vendor about the capabilities of the modern development tool, his viability, the customer base, support arrangements, and so on, should be in writing.

More detailed analyses will identify the best development tool for a particular organisation

Figure 2.5 The shortlisted products need to be assessed with reference to a range of factors

What are the costs associated with the modern development tool:

- For application development?
- For the production, operational, or end-user environment?
- For maintenance and enhancements?

What are the deficiencies and additional features provided by the modern development tool, in terms of:

- Functionality?
- Processing speed?
- Interfaces to other tools?
- Interfaces to databases?

What impact will the modern development tool have on the development environment? Will it require:

- Additional hardware?
- Changes to standards?
- Training?

What gains can be expected with the modern development tool, in terms of:

- Reduced documentation?
- Increased productivity?
- Higher-quality products?
- Reduced maintenance?

What are the prospects of the tool and the tool vendor:

- Is the vendor an established company or a market front runner?
- How many years has he been in the market and what is the user base?
- How has the tool advanced over the last year?
- What future enhancements are proposed?
- Does the hardware vendor recommend the tool?

Where possible, the shortlisted products should be evaluated concurrently

Depending on the resources available, and on the urgency of the need for the modern development tool, the analyses of the shortlisted tools can be carried out concurrently, or one after the other. We recommend that, wherever possible, they be carried out concurrently. This approach does require a higher peak level of resources, but will cost the same, and results will be achieved more quickly. Furthermore, different teams will be involved in the assessment of each tool, and one member of the team will usually become its champion.

Regardless of the approach selected, the subsequent stages should be well planned and managed to ensure that the most beneficial modern development tool is selected:

- Identify a contact in each vendor organisation. These people should be involved in the trials conducted to evaluate the modern development tools, and should introduce them to the analysis teams. Each team should consist of up to three experienced developers or analyst/programmers from within the organisation. Before the trials are carried out, the analysis teams must be knowledgeable about the appropriate modern development tools.
- Carefully define a small but realistic trial application that will test the abilities of the various modern development tools.
- Use the modern development tools to construct the trial application. Assess the selection criteria and collect the information required to compare the tools. This stage of the trial will show how the tools will perform in the development of new applications and in rewrites of existing applications.
- Define and implement one small and one large enhancement to the trial application. Again, monitor the performance of each tool and collect the information required to compare them. This stage of the trial will show how the tools will perform during enhancements and maintenance.
- Measure the operational performance of the developed application in the environment in which it will be expected to run.

The cost of trials is acceptable in view of the risks involved in choosing the wrong modern development tools

Ideally, these trials should be carried out in the organisation's own development environment, or alternatively, using the vendor's training facilities. The objective of the trials is to gain a better understanding of the use and capabilities of the modern development tools with respect to an organisation's eventual needs. These trials may seem a costly exercise in terms of time and effort, but compared with the cost of selecting incorrect modern development tools, it is an acceptable price to pay. If it is impractical to conduct such a series of trials, a tool can be selected on the basis of the organisation's current knowledge of those on the shortlist. A trial could then be conducted on the selected modern development tool. Provided that there were no major problems, that particular modern development tool would then be adopted.

If the series of trials has been completed, each modern development tool should be ranked. A representative from each analysis team should be available to answer any detailed questions. The strengths and weaknesses of the tools that have been on trial should be discussed. At the end of the meeting,

one modern development tool should be selected for addition to the organisation's resources. If there is a tie between several modern development tools, a small trial should be set up to resolve the issue. If it is still not clear which is the best tool, the one with the most users in the United Kingdom, or the one supplied by the vendor with the best support and help-desk facilities should be selected.

REVIEW COVERAGE AT REGULAR INTERVALS

The set of development tools available for use within the systems development department should continually evolve to match the required applications and support the development approaches adopted. It is therefore necessary to review the set of development tools at intervals appropriate to the rate of change within a particular organisation. Reviews should also be carried out at other times if several developments have run into difficulty, or if the application areas change.

The set of tools should be reviewed at intervals appropriate to an organisation's rate of change

One PEP member, for example, found that one of its financial applications, for which results were required daily, was taking nearly 24 hours to run. This was found to be the result of redeveloping several modules with a modern development tool. At the time of redevelopment, only a small amount of information had to be processed by the system. As the amount of information increased, however, the execution time increased dramatically. After receiving numerous complaints about the application from users, the development department identified the problem as the modern development tool's code, which required enormous machine resources. A more efficient modern development tool was adopted to replace the existing one. The critical processing modules have now been rewritten. The whole application is gradually being rewritten with the new tool, replacing modules as and when required.

CONTROL THE SIZE OF THE SET OF TOOLS

While it is likely that a set of development tools will be needed in most organisations, care must be taken to manage it and control its size. Adding yet another development tool may enable applications to be developed more effectively and efficiently, but it will also dilute the skill levels within the department. At one time, one PEP member had 15 development tools in use. It is now reducing this to three, and at an appropriate time, plans to redevelop each of the existing applications with one of the reduced number of development tools.

The appropriate number of development tools available for use within an organisation depends on the number and type of development environments and the hardware that it has. We recommend that most organisations should have:

The appropriate number of development tools for an organisation depends on the development environments and the hardware that it has

- A third-generation language, in order to develop applications for which their current modern development tools are unsuitable, or to maintain existing applications.
- At least one modern development tool, in order to improve development productivity for new applications. Some organisations will need several modern development tools — for example, one fourth-generation language for prototyping and a different fourth-generation language for development.

— A set of end-user tools, such as a query language and a report writer.

Once an organisation has decided on the set of development tools it needs to provide adequate and appropriate coverage for the kinds of applications that it is likely to require, it will need to assess the implications of introducing them into the development environment. Some changes will doubtless need to be made within the organisation to ensure that modern development tools are integrated smoothly into the development environment, and that the benefits to be derived from them are fully realised, with the minimum of disruption. How the necessary changes should be identified and managed is the subject of Chapter 3.

Chapter 3

Planning for the introduction of modern development tools

It is, of course, impossible to produce a single plan that can be used to introduce all types of modern development tools into all types of organisation. Some organisations are generally quick to react to change; others less so. They will therefore also be likely to adopt new development tools at different rates. Moreover, the type and complexity of the development tool will have a bearing on how long it takes to integrate it into the development environment. The level of planning and the amount of work required to implement each stage of the plan will therefore vary from one situation to the next, but we strongly believe that the introduction and subsequent integration of a modern development tool must be carefully planned and managed. The advice that we provide in this chapter is intended primarily for those organisations that are contemplating the introduction of modern development tools, but it will also be applicable where such tools have already been installed, especially where changes are having to be made to the development environment to exploit and support the tools.

The introduction of a modern development tool must be carefully planned and managed

Without careful planning and management, the benefits of a modern development tool may never be fully realised. One PEP member, for example, decided to introduce QuickBuild, but failed to get the commitment of the development staff at the outset. As a consequence, developers felt that they were forced, rather than encouraged, to use the tool, and continually found reasons not to use it. The problem was eventually overcome, but it need never have arisen if the introduction of the tool had been properly planned and managed. Another PEP member believed that his own in-house expertise was adequate to conduct a pilot application, and did not therefore involve the vendor. Again, problems arose that were expensive and time-consuming to solve, and which could have been avoided with careful planning and management.

Otherwise, its full potential may never be realised

An effective plan for introducing a modern development tool should consist of the following four stages:

- *Stage 1: Internally market the implementation plan.* This stage is designed to ensure that all the staff involved with the modern development tool know exactly what the implementation plan is, what their responsibilities are, and how it will affect them.
- *Stage 2: Initiate changes to exploit and support the modern development tool.* In this stage, changes are made to the development environment so that the modern development tool can be optimally supported and exploited. These changes may be of a 'one-off' nature, such as reducing team sizes, or they may be continuing changes, such as defining and creating a 'cook book' (described later in this chapter).

— *Stage 3: Implement a pilot application.* In Stage 3, the ability of the modern development tool is tested on a pilot application. If the correct tool has been selected, and Stages 1 and 2 have been completed correctly, the pilot application will succeed. Areas may well be identified where changes can be made to make better use of the tool — for example, areas where the documentation may be reduced, or where changes in the procedures may be introduced.

— *Stage 4: Modify the development environment in the light of the pilot application.* In Stage 4, any recommendations resulting from experience with the pilot application are implemented.

Stages 1, 2, and 3 should be planned in detail, at the outset. Stage 4 is dependent on the results of the pilot application. We recommend that, throughout the planning process, close contact be maintained with the vendor. While the vendor may not have detailed knowledge of a particular organisation's environment, he will have a wealth of experience in implementing the particular modern development tool. The advice of other organisations with experience of using the modern development tool should also be sought.

INTERNALLY MARKET THE IMPLEMENTATION PLAN

Staff need to understand the implications of the introduction of a modern development tool

The objective of this stage is to ensure that all the staff involved with the modern development tool are aware of the implementation plan and of its effects on their working environment, their roles and responsibilities, job security, market value, and so on. In PEP Paper 7, *Influence on Productivity of Staff Personality and Team Working,* we identified the staffing factors that are important to productivity; managers should pay particular attention to these factors when considering the introduction of a modern development tool.

The commitment of senior systems managers is essential

The commitment of senior systems department managers to the use of the modern development tool, and the development approach in which it will be used are also vitally important to the successful introduction of the tool. To win their support, it will be necessary to demonstrate that the money spent on the modern development tool and the changes made to the development environment will increase the development department's productivity. In PEP Paper 5, *Managing Productivity in Systems Development,* we demonstrated how to calculate the return on investment and the cash savings associated with new developments.

A group should be created to 'market' the modern development tool to the rest of the systems development department. This group should comprise the proposed technical expert (who will probably be one of the people who carried out the initial trial), a technical expert from the vendor, a sales representative from the vendor, and a senior project manager from within the systems department. This group will be responsible for introducing the modern development tool to development staff. We recommend that all staff involved with the tool — developers, managers, and user managers — attend a one-day presentation. Half the day should be spent introducing the modern development tool, and

half should be spent discussing the plan for introducing it into the organisation. This should encourage a positive attitude towards the modern development tool and its use.

The vendor should take the lead in the first half-day session, providing general background information on the tool, showing how it will be used in the proposed environment, describing the types of applications it will be used for, and providing details of the benefits that it can provide. In-house members of the group will participate in a supporting role.

A one-day presentation should be made to introduce the tool

In the second half-day session, the in-house members of the group will take the lead, clearly defining each stage of the plan and indicating the proposed timescales. All questions should be answered either during, or shortly after, the meeting. If outstanding issues are left unresolved, they may become stumbling blocks at a later stage. The pilot application should be described, the expected time scales should be made clear, and the members of the development team who will work on it should be announced.

INITIATE CHANGES TO EXPLOIT AND SUPPORT THE MODERN DEVELOPMENT TOOL

To maximise the benefits obtained from using modern development tools, various changes may need to be made to the development department. These changes could affect any aspect of the development department, from the computer hardware to the roles of the staff. To ensure that the potential benefits of modern development tools are achieved, most organisations will need:

- To introduce the role of analyst/programmer.
- To introduce the role of technical expert.
- To provide appropriate training.
- To reduce the size of project teams.
- To prepare for higher levels of user involvement.
- To increase the level of hardware support, both for development and operational work.
- To introduce prototyping.
- To reduce the amount of documentation.
- To use the tools' standard facilities and defaults.
- To define and implement a 'cook book' and a 'tool-limitations list'.

INTRODUCE THE ROLE OF ANALYST/PROGRAMMER

One of the advantages of modern development tools is that they enable applications to be constructed without the developer requiring detailed knowledge of the hardware environment on which the applications will run. By introducing analyst/programmers to take advantage of this feature, the communication problems that have commonly existed between analysts and programmers can be removed. A survey of PEP members carried out in late 1988 showed that just over 75 per cent of all development staff are now classified as analyst/programmers.

The introduction of analyst/programmers will make it easier to exploit modern development tools

The remainder are classified as either analysts or programmers, using traditional tools for maintenance or development. Care must be taken during the transition period, however. Not all analysts wish to get involved in programming, and not all programmers are interested in dealing with users. While analyst/programmers should be involved in the whole development process, they should also be allowed to concentrate on those parts of the process where their strengths lie. This will result in a better exploitation of their skills.

INTRODUCE THE ROLE OF TECHNICAL EXPERT

The technical expert is the focus of all enquiries

Whenever any new development tool is introduced, it is good practice to designate a technical expert as a focal point for all enquiries. The trials carried out during the selection process should provide several of the developers with a reasonable knowledge of the modern development tool. One of them would be the obvious choice to become the technical expert for the chosen tool. The technical expert will also be responsible for keeping up-to-date on the latest enhancements to the tool and for resolving any problems that arise.

PROVIDE APPROPRIATE TRAINING

All staff involved with the modern development tool should be trained. Different types of staff will have different training requirements:

- *Systems development managers* need to understand the capabilities of the modern development tool and how it should be used, as a basis for planning resources to support the development of an application.
- *Analyst/programmers* need to understand how to develop applications with the tool and what its capabilities are.
- *Technical support staff* need to have detailed knowledge of the modern development tool, both in terms of how it will be used for development and for end-user access, and of its limitations.
- *End users*: Need to understand the facilities provided for them by the tool. Users and user managers who will be involved in the development of applications should have a good understanding of the capabilities of the modern development tool and of the development approach within which it will be used. User managers, especially, need to realise the limitations of the tool and of the development approach.

Vendors are a good source of training

Most vendors of modern development tools provide very good technical training for all levels of development staff and users. Initially, the members of the pilot team should be sent on the vendor's course. If they find it effective, other staff should also attend. Alternatively, in-house courses should be run. Some vendors, such as ICL with QuickBuild, will help to set up such courses. The advantage of in-house courses is that they can be designed for a particular environment, for particular applications, and for particular standards and procedures. They can also be tailored to match the level and experience of those attending, and to utilise the most appropriate training methods and media, such as tutorials, videos, computer-based training, and so on.

There are advantages to running courses in-house

Some of the more established vendors hold forums at which technical experts can question product managers and swap

experiences with each other. Information Builders, for example, has established such a meeting for UK users of its product, Focus; this is called the 'Top Gun' event.

According to PEP members, the only area not covered adequately by vendors is training traditional third-generation programmers to think differently when using modern development tools. Programmers still tend to use modern development tools to write applications in a third-generation-language style, and thus make less than effective use of the modern development tool. Many third-generation-language programmers are not aware of the facilities provided by the modern development tools at their disposal. Many of these tools, especially application generators, for instance, allow the developer to specify the application at a higher level than was possible with third-generation languages; many have re-usable application elements and capabilities for defaults; many facilitate incremental construction and prototyping, which is checked with the user in stages. Programmers who are not properly trained can therefore spend a lot of time doing work that would not be required if the modern development tool were being properly exploited. Some vendors are aware of this problem and are introducing training programmes with a stronger emphasis on how to make the best use of the modern development tool, rather than on its capabilities and facilities.

The need for a change in programming style with modern development tools is not addressed in training programmes

REDUCE THE SIZE OF PROJECT TEAMS

Most modern development tools reduce the effort required in the construction phase of development — that is, coding, documentation, and testing. Thus, the teams required during the construction phase can be smaller. Analysis of the PEP database confirms this trend. During 1987, applications developed with modern development tools had an average peak staff level of 7.4. This is 22 per cent less than the corresponding peak staff level for applications developed with third-generation tools. During 1988, the average peak staff level fell to 6.9 — 39 per cent less than the corresponding peak staff level for applications developed with third-generation tools. As a result, there will be fewer channels of communication, less management effort, and more productive developments.

With modern development tools, the construction phase of a project requires fewer development staff

PREPARE FOR HIGHER LEVELS OF USER INVOLVEMENT

With modern development tools, which provide the ability to prototype applications, it is common for users to be involved at the design stage of the application. The use of prototyping with modern development tools is discussed further on page 28. Analysis carried out in mid-1988 for PEP Paper 6, *Managing Contemporary System Development Methods,* showed that user involvement increased two- to four-fold when modern development tools, rather than traditional tools, were used. In addition, as there is a greater throughput of applications and as more new developments are possible with the use of modern development tools, user involvement at the design stage will increase still further. All these points are discussed in detail in PEP Paper 6.

With modern development tools, users will be more involved at the design stage of an application

To maximise the benefits of user involvement with applications development, analyst/programmers will need good business awareness and people-related skills as well as technical knowledge. Their

ability to communicate well with users is as important as their ability to use the modern development tool. All analyst/programmers dealing directly with users should have their business awareness, communications skills, and technical knowledge assessed, and if necessary, be provided with training to improve them.

Users must play an appropriate role

Some organisations have seen the emergence of so called 'power users' — business people who not only use the modern development tools, but push them to their limits, and build all, or substantial parts, of the applications that they require. The majority of PEP members, however, still have problems getting users committed to new developments. On the one hand, user departments often underrate the importance of their involvement to the success of the development; on the other, developers report that it is often difficult to gain access to the right users.

Several actions can be taken to ensure that users do play an appropriate role:

- Draw up guidelines defining the roles and responsibilities of user representatives in a development, stating how their performance will be evaluated. The evaluation should be carried out by the project sponsor or the systems department, not the users' line managers.
- Persuade the manager sponsoring the development to assign to the team the staff who will have the greatest contribution to make to the project, not those who are perceived as the most dispensable within the department.
- Ensure that the user representatives perceive the work as critical and as an integral part of their job, not as a task that they have unfortunately been picked to do. A positive attitude from the user representatives is essential.
- Ensure that the users' other (non-systems) work is done by someone else, so that they can concentrate on the development project.
- Ensure that when the user representatives finish their development-project responsibilities, there are non-systems jobs for them to return to, so that they will be willing to contribute again, if required.
- Reward user representatives who are effective and helpful, in order to encourage others.

The importance of such actions with respect to user representatives is not confined to their role in the use of modern development tools, but they are especially critical when users are involved with development projects. Unless user representatives are controlled and managed correctly when they are involved in development work, the full benefit of their contribution will not be realised.

INCREASE THE LEVEL OF HARDWARE SUPPORT

More processing capacity may be required once a modern development tool is introduced

The initial trials should make it clear whether additional hardware resources will be required to handle the processing requirements of the new modern development tool. There will also be a greater requirement for terminal access when a modern development tool is adopted, and even more so if prototyping is adopted. Ideally, there should be a terminal on each developer's desk.

Several PEP members found that they needed more processing capacity once a modern development tool had been introduced, both to run the modern development tool and to run the new applications being generated by it. If the usage of the system is regularly monitored and the capacity required for all new developments is assessed, the need for additional resources can be anticipated. If this requirement is not anticipated, the resulting bottlenecks may reduce the productivity of development teams, and provide poor performance for the users.

Some modern development tools, such as Focus, permit development work to be offloaded onto microcomputers or minicomputers, thus reducing the pressure on mainstream computing resources. Systems integration and testing would usually still have to be carried out in the operational hardware environment, however.

INTRODUCE PROTOTYPING

One of the strengths of most, although not all, modern development tools is their ability to build prototypes. This strength should be utilised wherever possible. Prototyping is particularly suitable for the development of applications where the user is willing to help with the definition, where the application is not very well defined, or where the application is not too complex. Prototyping can also reduce the amount of effort required to train users. With certain applications, it may be possible to use one of the prototypes as a computer-based training aid, thus providing hands-on experience without incurring the risk of corrupting the live application.

The ability of modern development tools to build prototypes should be exploited wherever possible

Two types of prototyping are in common use — prototyping for iterative development and throwaway prototyping. Prototyping for iterative development, sometimes known as evolutionary prototyping, should be considered when the application meets the criteria defined above and when the modern development tool being used enables the application to be fully developed. Throwaway prototyping should be used to help with requirements definition when the application meets the criteria defined above and when the development tool selected for the application does not permit prototyping.

Two types of prototyping are in common use

A new type of prototying, known as experimental prototyping, is the development of small trial systems to assess whether a particular modern development tool is able to perform as required, or to try out several alternative designs or new technical features to determine their flexibility or performance characteristics. It can save a considerable amount of time and effort, and add to knowledge of the modern development tool. This type of experimental prototyping is normally carried out by the technical expert.

Although prototyping is a very effective technique when used in conjunction with a modern development tool, its use must be carefully controlled. We have seen instances where users have demanded that the first prototype (which was, in fact, only a mock-up of the user interface software) be implemented, where only parts of the application have been fully developed, and where enormous amounts of time have been spent on throwaway

The use of prototyping must be controlled

prototyping. Care should be taken to prevent such situations as these occurring. One American company, which uses Application Factory from the Cortex Corporation, uses a development approach in which the iterative prototyping phase is strictly limited to no more than 90 working days. This approach has been so successful that the company guarantees to build the required application for a fixed price within a fixed time period. If the delivered system fails to meet the user's requirements, the user pays nothing.

REDUCE THE AMOUNT OF DOCUMENTATION

Modern development tools will reduce the amount of documentation required for an application

Applications developed with traditional tools usually require detailed documentation to explain the meaning and the structure of the code. Some modern development tools make this traditional style of program documentation virtually redundant, by highlighting the program text that will help to clarify the meaning of the code, and automatically indenting code to indicate its structure. The total amount of documentation required for an application will thus be significantly reduced. Once the application has been developed, listings showing the relationships between the data fields, programs, and screens can also be generated automatically. Some modern development tools provide good documentation facilities for maintenance purposes, such as automatically generating cross-reference listings and menu-structure charts from the code and data structures. Fourth Dimension from ACI UK is an example of this type of modern development tool. User documentation is not, however, generated automatically. The procedures and aids required by the user will still have to be produced in the traditional manner.

It should be remembered, however, that a poorly documented application is difficult to maintain, whatever language it is written in. While modern development tools reduce the amount of documentation required, they do not preclude the need to produce it in a careful and timely manner.

USE THE TOOLS' STANDARD FACILITIES AND DEFAULTS

The standard facilities provided by modern development tools should be used as a matter of course

Most modern development tools provide many standard facilities, such as default screen formats, report layouts, input patterns, edit masks, validation modules, and check-digit routines, which organisations often ignore because they already have their own standards. Very often, however, the existing standard screens and layouts were defined to suit the programming language that was used at that time, and are not best suited for use with modern development tools. With some modern development tools, the default options can be modified to match an organisation's current standards. Otherwise, we recommend that the standard defaults provided by the modern development tool should be used as a matter of course.

DEFINE AND IMPLEMENT A 'COOK BOOK' AND A 'TOOL-LIMITATIONS LIST'

Cook books and tool-limitations lists help developers avoid the problems that commonly arise

Several PEP members use what they call a 'cook book' to help resolve problems that arise in using a modern development tool. Figure 3.1 is an extract from one PEP member's cook book, specifying how a screen-based system should be developed with Focus. Normally, compilation of the cook book is the responsibility of the technical expert.

Figure 3.1 A cook book advises users and developers on the use of a modern development tool

Specifying a screen-based system

Do:	***Keep things simple*** Do you need flashy formatting? The more colours, special formatting, and highlighting you use, the more complicated the code becomes. Determine the functions of each screen as you would for a third-generation language — the more the functions are broken down, the simpler the coding.
Do:	***Be precise*** Document the validation required behind each screen, for each field which requires validation. Be clear on screen processing — do not be afraid to use program design language to define the program flow for the screen sequence in pseudo-English — this is as important as it is with third-generation-language specifications.
Do:	***Be careful with PFKEYS*** When using PFKEYS to navigate through a system, use the default keys when possible. If other keys are required for special functions, use PF5, 6, 7, 8, 9, 10, 11.
Do:	***Use painter*** When designing the screens, use Focus painter. This helps you design, and saves the programmer time. You will know that the screen can be used in Focus.
Do:	***Issue your own information messages*** When the user presses an invalid PFKEY, or invalid data is entered, issue meaningful error messages. When an action has been taken (for example, job submitted or record deleted), issue a confirmation message.
Consider:	***Response times*** A fourth-generation language may be quicker to code, but will be slower than a third-generation language to respond. Is this critical to your system?

A similar aid is the 'tool-limitations list'. This contains detailed information on the limitations of the various development tools currently being used and is particularly helpful when deciding which development tool to use for a particular application. (This topic is discussed in detail in Chapter 4.) An example of part of a tool-limitations list used by a PEP member is shown in Figure 3.2. Again, responsibility for compiling the tool-limitations list normally lies with the technical expert.

IMPLEMENT A PILOT APPLICATION

Before a modern development tool is made available for general use, one or more pilot applications should be developed. The experience gained will be used to refine the use of the modern development tool and the development methods. If the selection procedure has been followed correctly, and if appropriate changes have been made in the development department to support the tool, no major problems should arise with the pilot application. It will simply confirm that the modern development tool can develop the required applications, and increase the confidence of the development department in its ability to do so.

A pilot application will indicate where the use of the modern development tool needs to be refined

The pilot application should be carefully selected since it is an important step in gaining acceptance of the modern development tool. We recommend that the application is:

Figure 3.2 The tool-limitations list specifies the limitations of a particular modern development tool

Limitations of Focus from a PEP member's tool-limitations list.*
Focus cannot update any file except a Focus or a VSAM file. These files cannot be read by any other language except Focus or Cobol programs making use of Focus Host Language Interface. However, Focus files can easily be created from QSAM or VSAM files, or DL/1 databases. Similarly, QSAM files can easily be created from Focus files.
Without central database control for simultaneous users, only one user can update a file at a time. Theoretically, a maximum of 128 simultaneous users is possible, but Information Builders indicates that about 20 is a more realistic limit. The operational range is between five and 20 users; typically, 15 users are supported.
Focus has no facility for automatic forward recovery (which is available in IMS). It is possible to code your own back-up logging and recovery routines in Focus.
Alternatively, frequent back-up copies of files can be taken. In the event of an irretrievable corruption of the database, the back-up copy would be restored and the user would have to re-enter his updates from the time the copy was taken.
No audit trail is provided for external files, except for IMS trace (which can be very large). Limited audit information is available for Focus file modification.
The 'non-procedural' nature of the Focus language makes complex processing difficult to achieve. Cobol subroutines should be used for complex logic and calculations whenever necessary.
The 3800 (laser) printer format character sets (that is, boxes and lines) are not available using Focus.
* These are the limitations of the version of Focus that one PEP member has experienced in his particular environment. Information Builders informs us that the current version of Focus overcomes most of these limitations.

— A real business application — that is, an application required by users — but not one that is critical to the success of the business. It is advisable to become reasonably experienced with a modern development tool before using it to develop critical business applications.

— Typical of the type of application for which the modern development tool was selected.

— Small — that is, an application that can be developed fully in two to three months. If it takes much longer than this to produce results, developers will lose sight of the overall development life cycle and the impact of the modern development tool.

— As far as possible, in the normal development environment.

Extra effort will, of course, be required to monitor the project, to collect detailed information about its progress, and to document any difficulties that were experienced. This effort should not, however, be taken into account in measuring the performance of the modern development tool as it will not be incurred in a normal project. The vendor should also be involved in the first pilot application. This may be expensive, but in the majority of cases, very productive. One PEP member who failed to do this had to abandon the first project that a modern development tool was used for. At the outset, the development department made a very inaccurate estimate of the machine resources that would be required by the modern development tool. It lost control of the application, as users demanded more and more functionality at the prototyping stage, and it failed to delegate responsibility

to the user department, where it would have been appropriate to do so. All these factors contributed to the failure of the project, and all could have been avoided.

On completion of the pilot application, the whole project should be assessed to identify any changes that might enhance the use of the modern development tool. The information gathered can also be used to produce guidelines for estimating the cost and effort likely to be involved in future development projects.

MODIFY THE DEVELOPMENT ENVIRONMENT

The development environment may need to be changed in some manner to facilitate the introduction of improvements identified as a consequence of the pilot application. For example, it may be necessary to modify standards, to reduce the level of documentation, or to increase the level of processing capacity. Such changes should be assessed and, if required, implemented. However, the temptation to make continual changes should be resisted. We recommend that suggested changes be fully documented and reviewed at regular intervals — say quarterly — to decide whether they are applicable, and to assess the costs and implications of implementing them.

Suggested changes to the development environment should be reviewed at regular intervals

With all the administrative and organisational changes implemented, the organisation is now in a position to use its set of development tools to best advantage. The only outstanding problem that it might now face is knowing which of the development tools available for use is the most appropriate for a particular application. Ensuring that the most appropriate tool is used for a particular application is a far more complex task than choosing a third-generation programming language to use on a project. Even so, many organisations insist on continuing to follow the same procedure. In Chapter 4, we point out what the pitfalls are, and offer some advice on how to avoid them.

Chapter 4

Ensuring that the most appropriate modern development tool is used

Use of the wrong modern development tool is a common cause of development failure

We have seen that modern development tools can significantly improve development productivity, but most PEP members were also able to quote at least one disaster and numerous problems associated with their use of modern development tools. Two factors were common to nearly all of these development failures — poor management of the use of contemporary systems development methods, and the inability of the selected modern development tool to develop the required application fully.

Managing the use of contemporary systems development methods is the subject of PEP Paper 6. In that paper, we suggested how the management problems associated with such methods might be overcome. Here, we are concerned with the second problem commonly faced by systems development managers — selecting the most appropriate modern development tool to use for a particular development project.

Members cited many problems deriving from the inability of the selected modern development tools to develop the required applications fully. The most common were:

- The developed application could not process the necessary information within the time required.
- The modern development tool lacked the functions required to develop the application.
- The developed application could not interface with other applications and/or databases.

Modern development tools must be chosen to match the other elements of the development environment

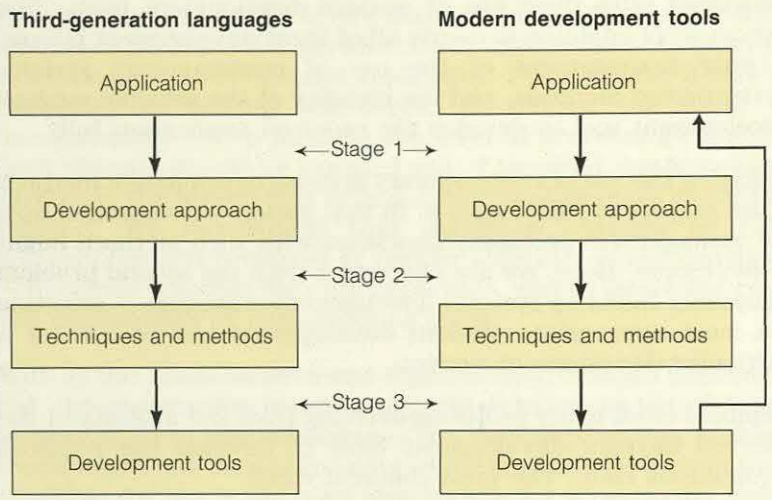
Either the project was completed, but at the expense of increased effort and cost, or it was abandoned, an appropriate development tool was identified, and the project was totally redeveloped. Either way, it is a costly exercise, and the blame is usually attributed to the shortcomings of the modern development tool. This is usually an incorrect diagnosis. The majority of projects that fail do so because the wrong tool was chosen for the application. A modern development tool cannot be selected in isolation. It must be considered in the context of the wider development environment in which it will be used. In this chapter, therefore, we suggest a procedure to use for ensuring that the most appropriate development environment, including modern development tools, is chosen, and recommend how that procedure should be put into practice for each development project.

DEFINE THE PROCEDURE FOR MATCHING THE DEVELOPMENT ENVIRONMENT AND THE APPLICATION

The problem of selecting the right development tool for an application rarely arose with third-generation languages because they could be used to develop most of the types of application

required. The traditional procedure for selecting an appropriate third-generation language is illustrated in the first column of Figure 4.1. The modern development tools currently available cannot be used for all types of applications, however, and the range of applications required is increasing. It is therefore essential to match the development tool with the characteristics of the application, if the full potential of the tool is to be realised for a particular development project.

Figure 4.1 The procedures for selecting traditional and modern development tools vary



Stage 1
The type of application determines which development approach should be adopted

Stage 2
The development approach is supported by various techniques and development methods

Stage 3
The techniques and methods are supported by various development tools

Stage 1
The type of application determines which development approach should be adopted

Stage 2
The development approach is supported by various techniques and development methods

Stage 3
The techniques and methods are supported by various development tools, whose capabilities match the characteristics of the application

An analogy can be made here with house building. Using third-generation tools was equivalent to building a house as a traditional craftsman would do, designing and building each component from basic materials. Using modern development tools is equivalent to building a house by using prefabricated components, such as windows, doors, and wall panels, as the basic elements. As with modern house-building techniques, modern development tools certainly enable the final product to be built much more quickly, but unless the right set of components is selected, it will not be possible to build the application according to the original design.

Choosing the right modern development tool is a complex process

The problem of selecting appropriate modern development tools arises because, in most organisations, the relationships between the application, the development approach, the systems development techniques and methods, and the modern development tools, are not well understood. Most organisations are using the procedure that was developed for third-generation languages to choose which

modern development tools to use for an application. In our analogy, this is comparable to selecting the prefabricated components without considering what type of building is to be constructed — a flat, a house, an office block, or a hospital.

A proper selection procedure should ensure that the modern development tool not only supports the systems development techniques and methods, but that it is also able to develop the required application. It is contrasted with the traditional procedure in Figure 4.1. We believe that this formal procedure should be adopted for selecting all development tools. If it is implemented correctly and updated regularly, it will guarantee the most effective match between the development environment and the application. Combined with the effective management of contemporary methods, this should ensure that projects developed with modern development tools are successfully completed.

Tesco Stores Ltd is in the process of implementing just such a procedure. Tesco uses three main development tools — Telon, Focus, and SDT, a fourth-generation language from McCormack and Dodge. The company has clear guidelines for deciding which development tool should be selected for a particular application. These guidelines, accompanied by detailed instructions, are issued to developers in a document entitled, *The Development Language Selection Criteria.* This document gives the reasons for selecting a particular language, and two diagrams, one for new applications and one for maintenance, indicating which development tools will be appropriate for applications with certain characteristics. An example of the diagram for new applications is given overleaf in Figure 4.2.

In the rest of this section, we explain the basic elements of the procedure, and describe how it should be documented so that modifications can be incorporated for future reference.

DEFINE THE ELEMENTS OF THE DEVELOPMENT ENVIRONMENT

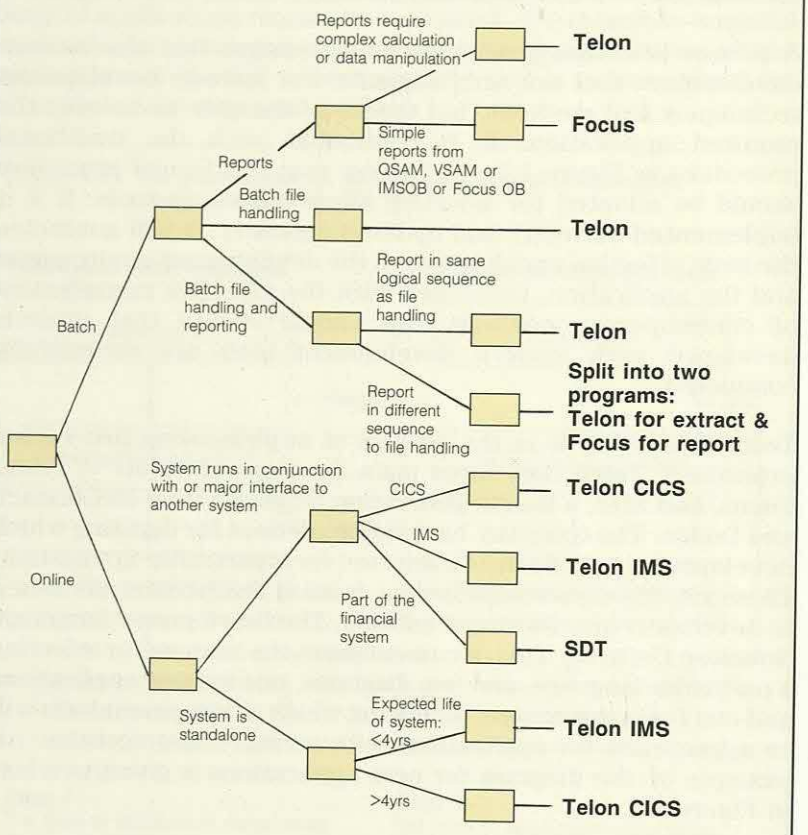
Elements of the development environment need to be defined and documented

Before the procedure for selecting the application development environment can be implemented, the elements need to be defined and the definitions documented so that everyone involved in the selection procedure has the same basic understanding of the application development environment. This document is used whenever a development project, either maintenance or new, is started. We have grouped the elements that need to be defined into four categories:

- Development approaches.
- Systems development techniques and methods.
- Development tools.
- Application characteristics.

The level of definition and the number of definitions in each category will vary from one organisation to another. The definitions should be reviewed on a regular basis to take into account the evolutionary changes in the development environment, and the introduction of new types of applications,

Figure 4.2 Tesco has a procedure for selecting appropriate development tools for each new application



(Source: Tesco Stores Ltd)

development approaches, and systems development techniques, methods, or tools. (These terms were defined in Chapter 1.)

Most PEP members probably already have much of this information available in some form, and are using it as the basis for selecting the approach, techniques, methods, and tools for the development of each application. It is, however, important that the selection procedure be rigorously applied and that the procedures be modified as lessons are learnt. Otherwise, mistakes will continue to be made. One company we know of, for example, was persuaded by a member of the project team to develop a critical customer-control system with a new fourth-generation language. This person had experience with this language, and believed that it was appropriate for the application in question. The selection procedures were ignored. The fourth-generation language could not, in fact, provide all the facilities required by the application, and after several months of effort, the project was in disarray.

The procedures can be modified as lessons are learnt

Development approaches

All the major development approaches used within the organisation should be briefly described, with an explanation of the objectives and the actions required at each phase. It should be clear to the reader how the phases flow from one to another.

Systems development techniques and methods
All the systems development techniques and methods available within the organisation should be briefly defined, and associated with the development approaches that they support. Several development approaches may be supported by a single technique or method, and several techniques and methods may support one phase of a development approach.

Development tools
All development tools should be identified and described in a similar manner to the systems development techniques and methods, and associated with the various methods and techniques that they support. Again, there may be multiple associations.

Application characteristics
To ensure that the development tool that is selected will facilitate the efficient and successful development of the application, the nature of the applications developed by an organisation needs to be clearly understood. The nature of an application can be defined in terms of a set of characteristics. Each new application can be described in these terms, and hence, be defined in a consistent manner. Most applications can be defined for this purpose in terms of between 10 and 20 characteristics, relating primarily to the development approaches and the development tools currently used by a particular organisation. Examples of the kinds of characteristics of an application that will determine whether or not it is a suitable candidate for a particular development approach are listed in Figure 4.3. Examples of the kinds of application characteristics that will determine whether a particular development tool is appropriate for it or not are listed in Figure 4.4 overleaf. These lists can be amended and supplemented to suit an individual organisation.

The list of application characteristics used as the basis for selecting a tool is similar to the one used in Chapter 2, and the list of characteristics compiled for that purpose can serve as the basis for this list. However, where the earlier list was compiled with

Figure 4.3 Every application has characteristics that will determine the suitability of using particular development approaches

Scope of the impact	A measure of the impact of the application throughout the organisation. This could range from company-wide applications to personal systems.
Clarity of the definition of users' requirements	This could range from well defined and easy to understand, to poorly defined and difficult to understand.
Urgency	A measure of the urgency of the development of the application, and of the deadline for installing it.
Number of locations	The number of geographical locations or sites.
Complexity	A measure of how difficult the application will be to develop, in view of its complexity.
Security	The level of security that the application must provide for access to the application itself and to the data.
Audit requirements	The level of audit that the application must provide. This could range from none, to very high (for financial systems).

Figure 4.4 Every application has characteristics that will determine the suitability of using particular development tools

Application type	Definitions should reflect the type of application rather than the business area — for instance, transaction processing rather than financial systems.
Level of integration	A measure of the level of integration expected between this application and other application types, databases, and machine environments. This could range from none, to numerous and very complex.
Performance requirements	A measure of the required performance of the application. Some may require instant response times; for others, response times will be less critical.
Type of development	An indication of whether the application is a modification, an addition, or an enhancement.
Level of portability	A measure of the portability of the developed application. This could be across different machine configurations, or across the machines of different manufacturers.
Likelihood of enhancements	The expected time from first installation to the first major enhancement.
Volume of data	An estimate of the total volume of data.
Security	The level of security that the application must provide for access to the application itself and to the data.
Complexity	A measure of how difficult the application will be to develop, in view of its complexity.
Size	An estimate of the total size of the application.
Expected life	An estimate of the life of the application.
Interface with end user	The degree of familiarity that the users will have with the system.
Flexibility	A measure of the likely extent and frequency of change.

a view to selecting a modern development tool to add to an organisation's existing set of development tools, this list will serve as the basis for selecting a particular development tool for a specific application — in other words, it will contain certain application characteristics for which a third-generation language may be more appropriate than a modern development tool. For example, a requirement for a high level of portability is more likely to be met by a third-generation language than by a modern development tool.

The characteristics must be clearly defined so that they will be consistently interpreted by different readers. They should not be too detailed or too technical, because they need to be kept to a manageable number. The lists should be amended as the development environment evolves. They provide the basis for the preparation of the selection tables described in the next section.

PREPARE THE SELECTION TABLES

Two tables need to be prepared to serve as the basis for matching the development approach and the development tools with the application. Their structure and the kinds of information they should contain are described in this section. Once prepared, the

The selection tables are prepared only once, and are used for all development projects

tables can be used for any development project. They are based on the lists of characteristics described above, with input from experts in the areas of development concerned. They will, of course, need to be updated periodically to reflect changes in the development environment.

Both tables are organised in a grid format and used in a similar manner. The first is used to select the development approach for a specific application. The second is used to ensure that the development tools selected will enable the required application to be developed.

An example of part of the table for selecting development tools is shown in Figure 4.5. The application characteristics are listed on the left-hand side of the table; the types of development tools are listed across the top. The application characteristics should be listed, as far as possible, in order of importance. A maximum score should be assigned to each of the application characteristics to indicate their relative importance — say, 20 for the most important characteristic, and five for the least important. Each application characteristic is broken down into a range of options, each of which receives a score. In Figure 4.5, for example, the 'expected life' of the application is considered one of the most significant characteristics and is given a maximum score of 20. This is broken down into three options — 'less than one year', 'between one and three years', and 'over three years'.

Figure 4.5 Use of the development-tool selection table ensures that appropriate development tools are used for each application

Application characteristics	Development tools available		
	Cobol	Focus	Telon
Expected life (20)			
Less than one year	10	15	20
Between one and three years	7	10	20
Over three years	5	15	20
Performance requirements (20)			
Very high (interactive)	20	15	15
High (time-critical)	15	15	15
Medium	10	20	20
Low (not time-critical)	10	20	20
Volume of data (15)			
Less than 5 megabytes	10	15	15
More than 5 megabytes	12	5	13

The numeric value entered onto the grid is an indication of the ability of the tool to develop an application that supports that option. If a particular development tool can fully support that option, it receives the maximum score for that characteristic. If it provides adequate support, it receives a lower score. If it provides no support, it scores zero. If, for example, the expected life of the application being considered is over three years, the application needs to be developed bearing in mind the continuing support that the tool will be able to provide, and the ease of maintenance of the application over the longer term. The capability of each tool to develop such a system is considered in turn. Cobol scores five as it is not the strategic development tool

for this organisation, and it produces applications that are not the easiest to maintain. Focus scores 15; although it is not the strategic development tool either, it does produce applications that are easy to maintain. Telon is the strategic development tool and produces applications that are easy to maintain; it scores the maximum of 20.

The procedure for compiling the selection tables is summarised on the left-hand side of Figure 4.6. This part of the procedure is carried out only once, before the process is initiated. Once the selection tables have been finalised, they should be tried out on several recently completed developments. This should reveal any errors in the selection tables, and also demonstrate how well the developments were supported by the development approach, systems development techniques, methods, and tools chosen. Modifications to the selection tables should be made as and when appropriate.

PREPARE DOCUMENTATION

The definitions and the selection tables should be fully documented

We recommend that these elements of the selection procedure be fully documented, and regularly updated. The documentation should consist of:

- The definitions of, and relationships between, the various development approaches, systems development techniques, methods, and tools.
- The approach and development-tool selection tables and the instructions for their use.

All the comments and decisions made during the process should also be documented. If a development should subsequently fail, the appropriate part of the definitions or selection tables can be amended by referring to the documentation. In this way, mistakes will not be repeated.

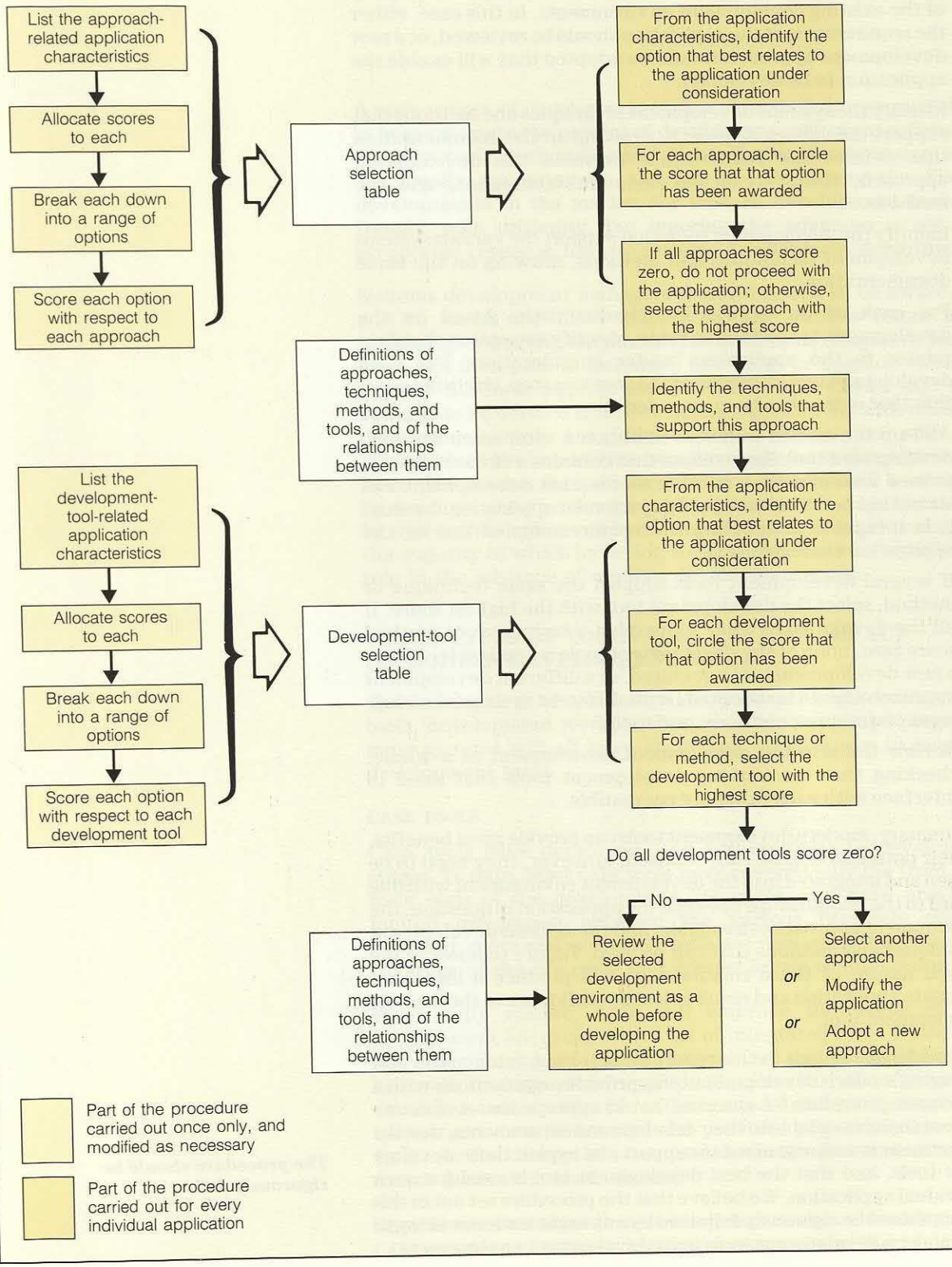
ENSURE THAT THE PROCEDURE IS USED FOR EVERY APPLICATION

The second part of the procedure is applied for each new project

The rest of the selection procedure, illustrated on the right-hand side of Figure 4.6, should be carried out at the beginning of each development project and is best done at a meeting attended by one or two users, the internal technical experts, and several of the systems development department's project managers, all of whom will contribute from their experience, and one of whom will manage the development. Everyone present should be acquainted with the definitions and the selection tables. This part of the procedure is described below:

- For each application characteristic listed on the approach selection table, identify the option that best relates to the application under consideration for development. For each approach, circle the score that that option has been awarded.
- Add up the circled numbers to obtain a total score for each approach. Any column that contains a circled zero will score a total of zero — in other words, that approach should not be considered for this particular application because it is incapable of meeting the requirements of one of the application characteristics.

Figure 4.6 Elements of the development environment should be selected according to a formal procedure



- The development approach with the highest score is the one that will enable the application to be developed most effectively, providing that the development tools available also support the application. If all the development approaches score zero, the application should not be developed, as it is not supported by any of the existing development environments. In this case, either the requirements of the application should be reviewed, or a new development approach should be adopted that will enable the application to be developed.
- Identify the systems development techniques and methods that support the chosen approach, drawing on the documentation that defines the relationships between the development approach, the systems development techniques, and the methods.
- Identify the development tools that support the various systems development techniques and methods, drawing on the same documentation.
- For each of the application characteristics listed on the development-tool selection table, identify the option that best relates to the application under consideration. For each development tool identified in the previous step, circle the score that that option has been awarded.
- Add up the circled numbers to obtain a total score for each development tool. Any column that contains a circled zero will score a total of zero — in other words, that development tool should not be considered for this particular application, because it is incapable of meeting the requirements of one of the application characteristics.
- If several development tools support the same technique or method, select the development tool with the highest score. If all the development tools supporting a technique or method score zero, none of the development tools is applicable. Either a new development tool is required, or a different development approach should be adopted, with different systems development techniques, methods, and tools.
- Review the selected development environment as a whole, checking that the various development tools that need to interface with each other are compatible.

In summary, modern development tools can provide great benefits. If their potential is to be fully realised, however, they need to be chosen and integrated into the development environment with due regard to the relationships between the application in question, the development approaches available, and the systems development techniques and methods that can be used. Failure to observe the critical nature of these relationships will produce a less-than-adequate application and result in loss of confidence in the modern development tool.

The guidelines set out in this report for selecting, integrating, and managing modern development tools provide organisations with a systematic procedure for ensuring that an appropriate set of development tools is available to their development departments, that the department is well organised to support and exploit those development tools, and that the best development tool is used for each individual application. We believe that the procedure set out in this report should be rigorously followed by any organisation seeking to use modern development tools in its development environment.

The procedure should be rigorously followed

Appendix

Preparing for the future

A wide variety of development tools and methods is now available to help automate the process of developing applications software, and new tools, with higher and higher levels of automation, are continually appearing on the market. Predicting the likely developments in the market for modern development tools is fraught with difficulty, but ignoring the advances is not an acceptable response for any systems development department.

Systems development managers therefore need to be aware of, and to assess the likely impact of, the various forces and technologies shaping the market for modern development tools. They can then plan to introduce new tools or upgrade existing ones at the most appropriate time. This will ensure that the benefits to be derived from developing applications with modern development tools are fully exploited today, but with an awareness of the impact that new development technologies will have. The market for modern development tools will be shaped by developments in other types of application development tools, by new technologies, and by the development of other concepts, the majority of which have not, to date, played a very significant role in the advance of modern development tools.

OTHER TYPES OF TOOLS WILL HAVE AN IMPACT ON MODERN DEVELOPMENT TOOLS

The development of computer-aided software engineering (CASE) tools, development workbenches, and object-oriented programming systems will all have an impact on advances in modern development tools.

CASE TOOLS

CASE tools have developed, and will continue to develop, in stages. To date, most CASE tools have covered only the front and back ends of the development life cycle, and have provided limited integration between the various life-cycle stages. However, more and more CASE tools are now providing extended coverage of the life cycle and greater levels of integration.

When fully mature, CASE will embrace the whole of the development life cycle, with a set of integrated tools. Currently, there is no integrated-CASE (I-CASE) tool set that covers the entire life cycle, manages all information (data models, data dictionary, databases, and so on) with an integrated database management system and data dictionary, and provides project-management facilities. The concept of I-CASE is also sometimes described as the fourth-generation environment.

Several vendors of modern development tools are working with CASE vendors to provide better coverage of the life cycle and

better integration. An example of this is the recent announcement by Pansophic and Cadre Technologies of their intention to produce a CASE system, Teamwork/Telon. (Telon is an application generator supplied by Pansophic.)

DEVELOPMENT WORKBENCHES

Development workbenches support and help to integrate various development tools and provide a development environment that enables applications to be designed, compiled, linked, and tested. Other facilities provided by workbenches include word processing, online documentation, and program-testing facilities. Some examples of development workbenches are Maestro from Philips, ISPF from Mackinney Systems, and TSO from Morino Associates.

OBJECT-ORIENTED PROGRAMMING SYSTEMS

Object-oriented programming systems (OOPSs) encourage the developer to reuse design and code rather than re-invent it for every application. They also provide a means of producing systems that can be readily comprehended by someone with business knowledge of the application. Object-oriented applications differ from traditional applications in that they directly model the application, as opposed to the data flow. The developed application can thus mirror the real requirement, not only in its operation but also in its design and construction.

In well chosen areas, OOPSs facilitate faster development than conventional systems. They also provide for the development of much more complex applications — including systems that support image, graphics, voice, and video — than conventional programming environments. OOPSs are, however, at an early stage of development, and current systems have numerous disadvantages. For example, developers are not familiar with the basic concepts, they have difficulty cataloguing and retrieving required objects (of which major systems contain hundreds of thousands), and OOPSs require considerable hardware resources. Examples of OOPSs are C++ from AT&T Bell Laboratories, and Smalltalk-80 from Xerox's Palo Alto Research Center.

NEW TECHNOLOGIES WILL INCREASE THE CAPABILITIES OF MODERN DEVELOPMENT TOOLS

Amongst the new technologies, expert systems, data dictionaries, and intelligent workstations will have a particular impact on modern development tools as their capabilities are combined to create even more powerful tools.

EXPERT SYSTEMS

Expert systems are computer systems containing organised knowledge, both factual and heuristic, that concerns some specific area of expertise, and are able to produce inferences for the user. Several suppliers of modern development tools are currently incorporating expert systems with their tools. Information Builders are integrating Focus, a fourth-generation language, with Level 5, an expert-system tool. This will provide an integrated data and knowledge-base system. The advantage of using expert systems with modern development tools is that the expert system

can be used to establish rules and carry out reasoning where the modern development tool lacks this ability.

If properly exploited, expert systems can provide many benefits in the area of systems development. For instance, rules could be established to help in the maintenance task by defining the scope of the impact of an enhancement or a change to an existing application. Expert systems also have great potential in the area of requirements definition — helping with the structured and logical questioning of users. Expert systems could also provide valuable aids in the testing area — providing test information and improving confidence in the various tests performed.

DATA DICTIONARIES

The central role of the data dictionary in applications development is becoming widely recognised. Data dictionaries provide a means of storing and retrieving information on the data, design, and tools associated with the particular applications under development. Various development tools can exchange and consolidate application design information via the data dictionary, which acts as the complete design database.

Currently, IBM is developing a dictionary product, and international standards for dictionaries are being agreed. Two data dictionary products that are currently available and well integrated with application development tools are the Data Dictionary System from ICL, and SQL*Design Dictionary from Oracle.

More and more dictionary products are moving towards becoming information repositories. These hold not only design information, but also information needed for implementation, and information on the network configuration. The information repository is a very powerful tool; when combined with expert systems, it can help to optimise the design process and the actual performance of the application. When the information repository is used in a distributed or cooperative manner, care must be taken to ensure that gaining access to the dictionary does not become a bottleneck, with one designer locked out while another is working. Careful control of the data dictionary is also essential, if various parts of the development are carried out on independent machines, to ensure that the data and the design maintain their integrity across the various machines.

INTELLIGENT WORKSTATIONS

More and more modern development tools can be used on the full range of hardware (PC to mainframe) available from a particular manufacturer. This means that more and more applications development work can be done on hardware that is independent of the main development or production mainframe. As the capabilities of intelligent workstations increase — that is, as they acquire more powerful processors, greater memory, and better screen resolution — and as their costs continue to fall, the scope for using them to develop applications becomes greater and greater.

NEW CONCEPTS WILL INFLUENCE THE USE OF MODERN DEVELOPMENT TOOLS

Reverse engineering and reusable code and designs will both play a role in extending the use of modern development tools.

REVERSE ENGINEERING

Reverse engineering will enable existing applications to be redeveloped with the latest development tools. The amount of automation provided by the reverse-engineering tools available today varies greatly. Some of the tools simply extract design information from the existing application, and the extracted design is then used to develop the new application. Re-engineering tools, however, also automatically construct the new application. Both reverse engineering and re-engineering are discussed in detail, with case histories of their use, in PEP Paper 8, *Managing Software Maintenance.*

There are several reverse-engineering tools available for analysing existing programs written in Cobol, to extract design information. The design information may need to be supplemented before the application can be rewritten with the latest modern development tool. Examples of this type of tool are Via/Insight from Viasoft and Recoder from Language Technology.

Re-engineering tools enable existing applications to be redeveloped with the latest development tools automatically, with minimal effort. The re-engineering tool analyses the existing application and derives design information from the existing code. This design information can then be used with the modern development tool to reconstruct the application. There are fewer re-engineering tools available today than reverse-engineering tools, and their abilities vary widely. Two examples are Bachman from Bachman Associates, and PSL/PSA from Meta Systems. The Bachman re-engineering tool currently provides facilities:

- To design databases using an expert-system-style dialogue with the database administrator.
- To optimise an existing database design by the above method, along with knowledge of the existing design.
- To convert from one database management system to another.

In the last two cases, Bachman is as yet unable to modify program code to take advantage of the new design.

REUSABLE CODE AND DESIGNS

Reusable code is not a new concept; it has been used very successfully for application development in the scientific and military fields. As the majority of applications developed contain very similar if not identical chunks of code, the concept of reusable code is to use existing common modules instead of writing the identical code for each application.

When the application is being designed, the common modules are identified. During development, the developer can then reference the library of common modules, extract the required module, and include it in the application.

The natural progression from reusable code is to reusable design. One company that supplies a tool to aid in reusable design is Oracle, whose SDW tool provides the developer with the ability to build conceptual models of the required application. Oracle also supplies various SDW dictionaries, which contain the basic designs for various types of application, such as asset management, purchase ledger, personnel management, and so on. The developer can modify the basic design in the dictionary and use it to develop the required design.

BUTLER COX

P,E,P

Butler Cox

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

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

The services provided include:

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

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

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

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

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

P E P

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

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

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

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

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

Forthcoming PEP Papers
Staffing the Systems Development Function
Trends in Systems Development among PEP Members

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

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

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

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

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

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

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

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

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

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

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