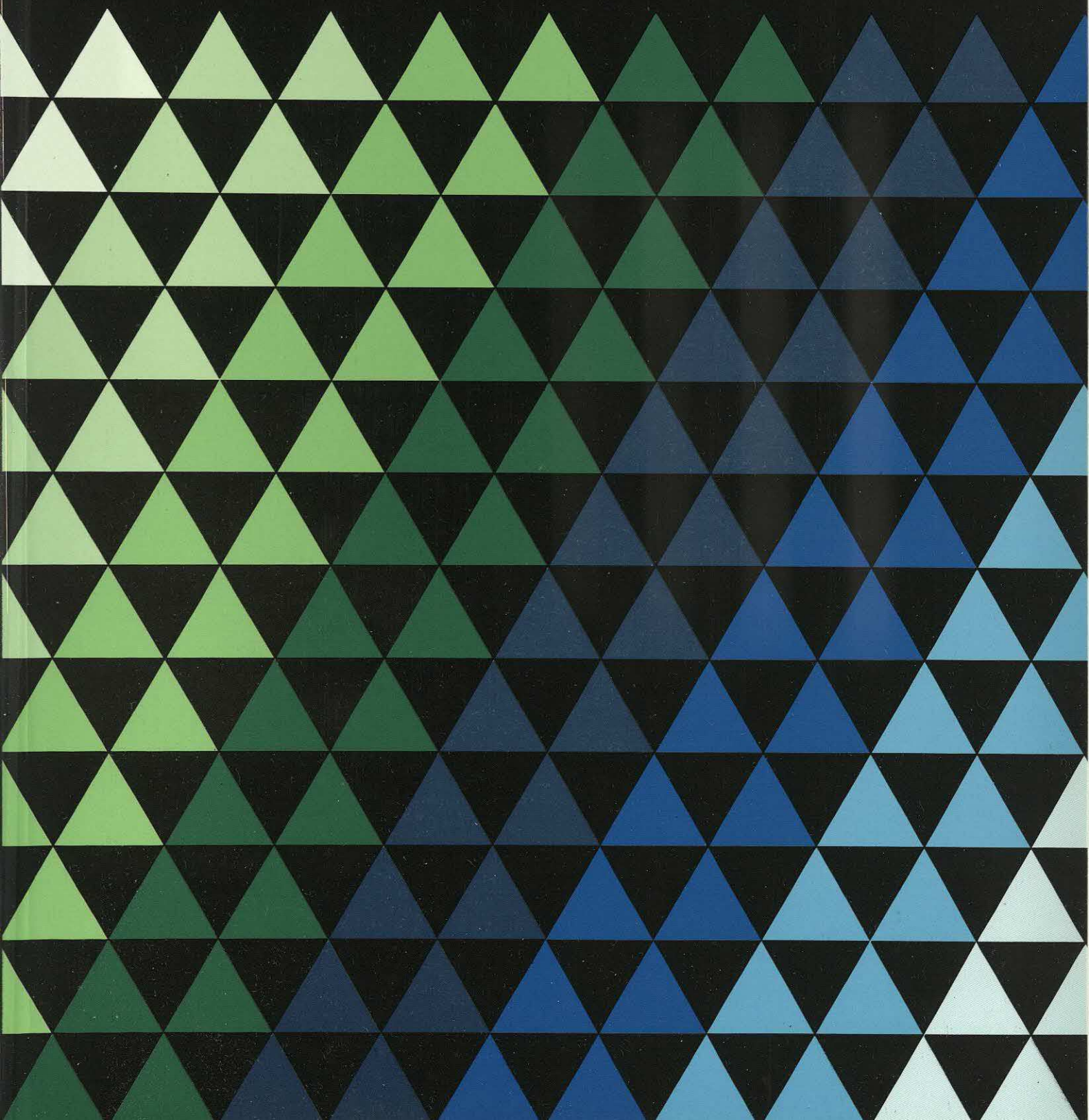
The Future of System Development Tools

BUTLER COX FOUNDATION

Research Report 74, March 1990



BUTLER COX FOUNDATION

The Future of System Development Tools

Research Report 74, March 1990

Butler Cox plc

LONDON
AMSTERDAM MUNICH PARIS

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

Availability of reports

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

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

BUTLER COX FOUNDATION

The Future of System Development Tools

Research Report 74, March 1990

Contents

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

Report synopsis

This report considers the new types of system development tools that are beginning to appear. These include I-CASE, reverse-engineering, rule-based, and object-oriented tools, all of which will, in the long term, make the development department's task much easier. Many of these are still at an early stage of development, and the claims made for them should be treated with caution. However, benefits can be gained if they are adopted selectively and with regard for the future. In summary, our advice is: continue to exploit fourth-generation languages; always consider package solutions; plan to migrate to I-CASE, without yet making a full commitment to it; use re-engineering tools to maintain existing applications and as a means of preparing them for an I-CASE environment; adopt rule-based tools as they mature in the 1990s; and monitor developments in other new types of tools as they emerge, particularly tools for exploiting object management, parallel computers, multimedia tools, and tools for developing advanced user interfaces.

Chapter 1

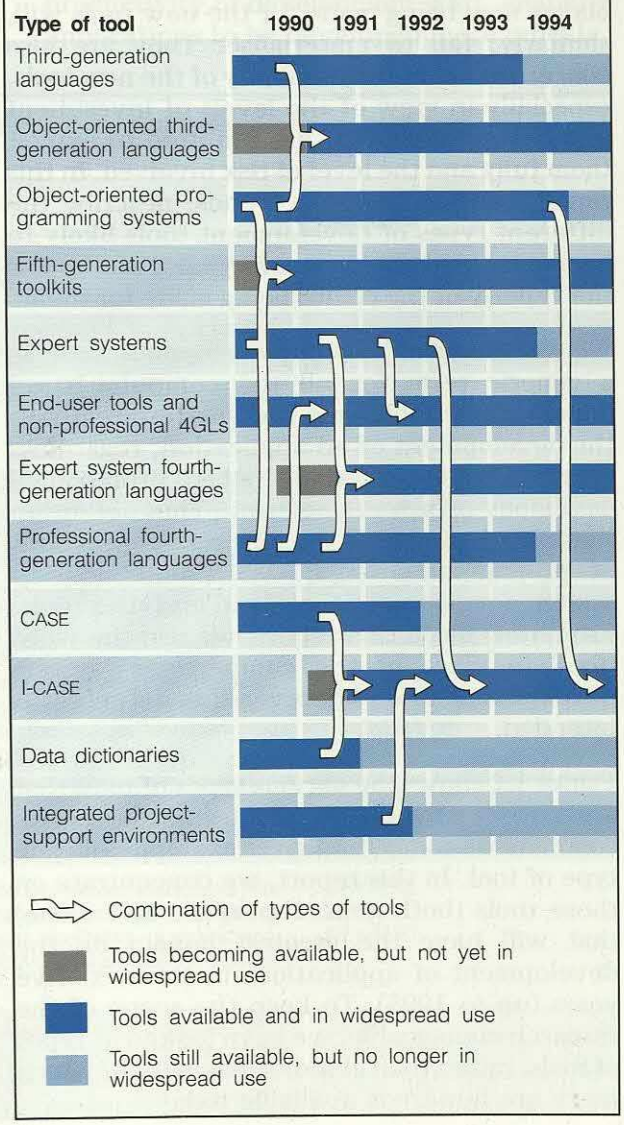
Treat the promises made for new system development tools with caution

All businesses are increasingly dependent on computer applications. As a consequence, systems departments are under growing pressure to produce more efficient applications that contain more functionality, that are more reliable and more flexible, and that provide access to greater amounts of information. Furthermore, development managers are expected to achieve this with fewer resources and reduced timescales. To keep pace with these growing demands, systems departments rely heavily on the use of development tools, the capabilities of which have continually advanced in response to business demands. These advances, however, have created some significant problems for the development department. Not only are some of the tools that are currently being used to develop applications out of date (because they have reached the limits of their capability but are retained to maintain old applications), but there is little compatibility between different types of tools.

Suppliers of development tools are responding by promising improved tools for use both by professional systems development staff and by business users who will use the tools to develop their own applications. The suppliers claim that the new tools will solve many of the problems currently faced by the development department in providing a service that is both reliable and flexible, and in building applications that have greater functionality. The improved tools, claim the suppliers, will be easier to use, which means that many business users will find it easier to use the tools to develop their own applications. Indeed, many of the tools will be positioned as end-user tools. In the development community, however, there is great uncertainty about the form that these future tools might take, about their impact on the development process, and about the most appropriate timing for their introduction.

These concerns are not unfounded. As Figure 1.1 shows, the situation is already complex, and will become more so in the

Figure 1.1 **Over the next five years, existing types of development tools will combine to form new ones**



mid-1990s, with various types of development tools combining to form enhanced tools and new types of tools emerging. The tools available in the mid-1990s will be more integrated with each other and will provide more powerful facilities, thereby increasing productivity and quality and permitting the development of more complex applications.

In the past, the introduction of a new type of development tool was accompanied by promises about the benefits it would bring. Rarely, however, have these promises been realised. Third-generation languages, for instance, were supposed to facilitate 'programming in English' (Cobol), and 'automatic programming' (Fortran). Some fourth-generation languages were supposed to remove the need for programmers. Foundation members are concerned that the claims now being made for the new tools will, similarly, fail to materialise. They are also concerned about the durability of the new tools, especially in view of the levels of investment and commitment that are required to exploit them fully and the level of risk involved. In this report, we examine the trends, describe the different types of development tools likely to be available in the next five years, and assess the validity of the claims being made for them.

We use the term 'system development tool' as a generic term for all the computer-based development tools that contribute directly to the development of an application, regardless of whether the user of the tool is a professional developer or a business user. This definition therefore includes all the different generations of programming tools, CASE tools, expert-system development tools, and end-user tools. From now on in this report, we use the word 'tool' in place of the cumbersome 'system development tool' wherever this definition is intended.

Figure 1.2 shows the various types of tools that are covered by the above definition and in common use today, and gives examples of each type of tool. In this report, we concentrate on those tools (both general-purpose and niche) that will have the greatest impact on the development of applications in the next five years (up to 1995). To keep the scope of the research manageable, we have looked at types of tools, rather than at individual tools, of which there are hundreds available today.

Project-management, planning, and estimating tools are mostly excluded from the scope of this report because they are management tools rather than development tools. An exception is made in Chapter 2, however, because management tools are an integral part of the future integrated development environment. Where advances in particular tools, like CASE tools, depend on advances elsewhere — for instance, in methods or standards — we have commented on these other advances, where appropriate.

There have been continuing advances in tools

Because of the increasing demands placed on the development department by the business, and because of the increasingly serious shortage of skilled systems development staff, tool suppliers have concentrated much of their effort on the area in which the greatest gains are likely to be made — the main-build (or programming) stage of the development process. This is the stage at which the greatest amount of development time and effort is currently spent. (The maintenance stage of the applications life cycle may take up to 60 per cent of the time and effort, but this stage can be considered to be iterations of the earlier stages, of which the programming stage is one.) As a result, advances in programming tools have resulted in significant productivity improvements at the main-build stage.

There has not, however, been a smooth transition from one major advance to the next. As Figure 1.3 on page 4 illustrates, when a new generation of tool is first introduced, the overall level of benefit initially falls, until development staff become familiar with the new tool, and procedures and methods are established. The level of benefit thereafter increases to a point above the level achieved with the previous tool. As experience is gained with the new type of tool, its limitations are recognised and the increase in benefit levels off. Eventually, the type of tool reaches the limits of its capability, and a new, more advanced generation is introduced. This pattern can be clearly traced in the progress of programming tools since the late 1940s:

— Initially, there was no choice other than to use first-generation languages. They were, however, extremely difficult to use because

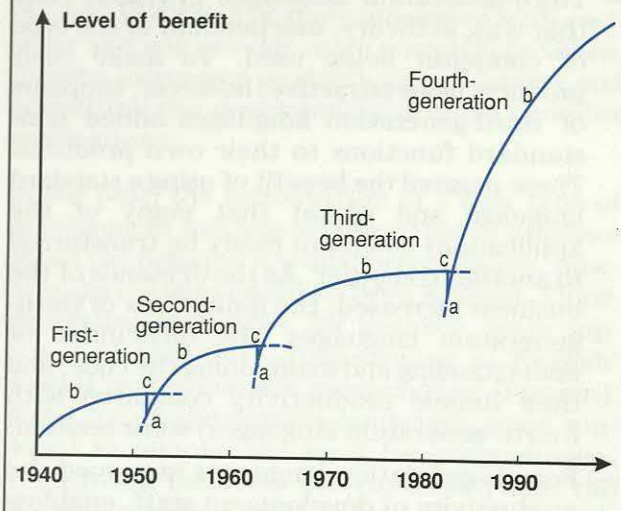
of the cryptic codes that represented instructions and storage locations. As the benefits of using computer systems were realised, programmers recognised that developing large applications with first-generation languages was not only complex, but also very costly.

- Second-generation languages provided a more productive means of developing larger and more complex applications, with mnemonic symbols replacing the cryptic codes, and a wider range of facilities. Second-generation languages were later extended by adding labour-saving facilities such as macro functions, code libraries, and subroutine structures. As the number of computers within an organisation increased, however, it became clear that the applications written in a second-generation language for one type of computer could not be transferred to another type, and could not satisfy the growing needs of the business.
- Third-generation languages provided code that was, in theory, independent of the type of computer being used. To make their products more attractive, however, suppliers of third-generation languages added non-standard functions to their own products. These negated the benefit of using a standard language and meant that many of the applications could not easily be transferred to another computer. As the demands of the business increased, the limitations of third-generation languages (the difficulties of understanding and maintaining the code, and their limited productivity compared with fourth-generation languages) were realised.
- Fourth-generation languages improved the productivity of development staff, enabling applications to be developed more quickly.

Figure 1.2 The term system development tools includes all generations of programming tools, CASE tools, expert-system development tools, and end-user tools

Type of tool	Description	Examples
First-generation languages	Programming tool that uses a binary format to express the application in the form of basic processor instructions (machine code).	ICL 1900 machine code Motorola 6800 machine code
Second-generation languages	Programming tool that uses symbols and mnemonics to express the application in the form of processor instructions (assembly language).	System 360 Assembler PLAN (ICL 1900)
Third-generation languages	Programming tool that uses a subset of natural language to describe how the computer is to implement the solution.	Cobol Fortran PL/1
Fourth-generation languages	Programming tool that uses a subset of natural language to specify 'what' the computer is to do to implement the solution. This type of tool can be divided into professional and end-user fourth-generation languages. (Unless otherwise specified, the term fourth-generation language refers in this report to a professional tool.)	Professional: FOCUS Oracle Application Master End-user: FOCUS Mapper
Computer-aided software engineering tools (CASE)	Tools that automate the techniques on which systems development methods are based. These tools typically provide some level of support for at least one stage of development. These tools include analyst workbenches, application generators, and system generators.	Oracle*CASE IEF FOUNDATION IEW
Re-engineering tools	Tools that enable existing applications to be improved by providing facilities to redocument, analyse, restructure, and/or regenerate applications in a standard and consistent manner.	Recoder Via/Insight
Expert-system tools	Include a wide range of tools all based on the techniques associated with knowledge engineering. These tools permit the development of applications based on the knowledge or know-how of experts, specialists, or technicians. Compared with most of the tools mentioned above, they are used to express the problem, not the solution.	CRYSTAL LEVEL5 NEXPERT
End-user tools	Include a wide variety of tools, ranging from simple spreadsheets and data-access and reporting tools to simple development and modelling tools. There is an increasing overlap between end-user tools and some of the other categories mentioned above.	Excel dBase

Figure 1.3 Programming tools have all followed a characteristic life cycle



Once again, however, fourth-generation languages were often specific to a particular hardware architecture, which meant that it was still difficult to transfer applications from one type of computer to another. Although most fourth-generation-language suppliers are working to overcome this difficulty, some systems departments are still recovering from the uncontrolled use of a variety of fourth-generation languages.

Today, the term fourth-generation language covers many types of tool, from Telon, a powerful screen-based development tool used by professional development staff, to FOCUS, a development tool and database management system that can be used equally well by professional developers and by users, and Mapper, which has been used very successfully by many users, some of whom have used it to develop very large applications. All that these tools have in common is that they can be used to produce applications more concisely and rapidly than traditional third-generation languages, and provide a better support environment than third-generation languages.

As one generation of tools has followed another, other advances have been made in associated development aids such as compilers, editors, debuggers, database management systems, and code libraries. The net result is that development departments now have a wide range of tools and aids to choose from. Most organisations are currently using both third- and fourth-generation languages (see Figure 1.4). Some are also using second-generation languages in specialised areas, such as realtime systems, where operational efficiency is critical or where there is a need for special hardware.

These advances have led to increased costs

The benefits deriving from each generation of tool have not been achieved without actual and potential costs being incurred by systems departments. The actual costs arise from the increasing costs of adopting a new generation of tool and from the resulting increased complexity. The potential costs arise from the risks associated with adopting any new tools when business success depends so heavily on the applications developed with them.

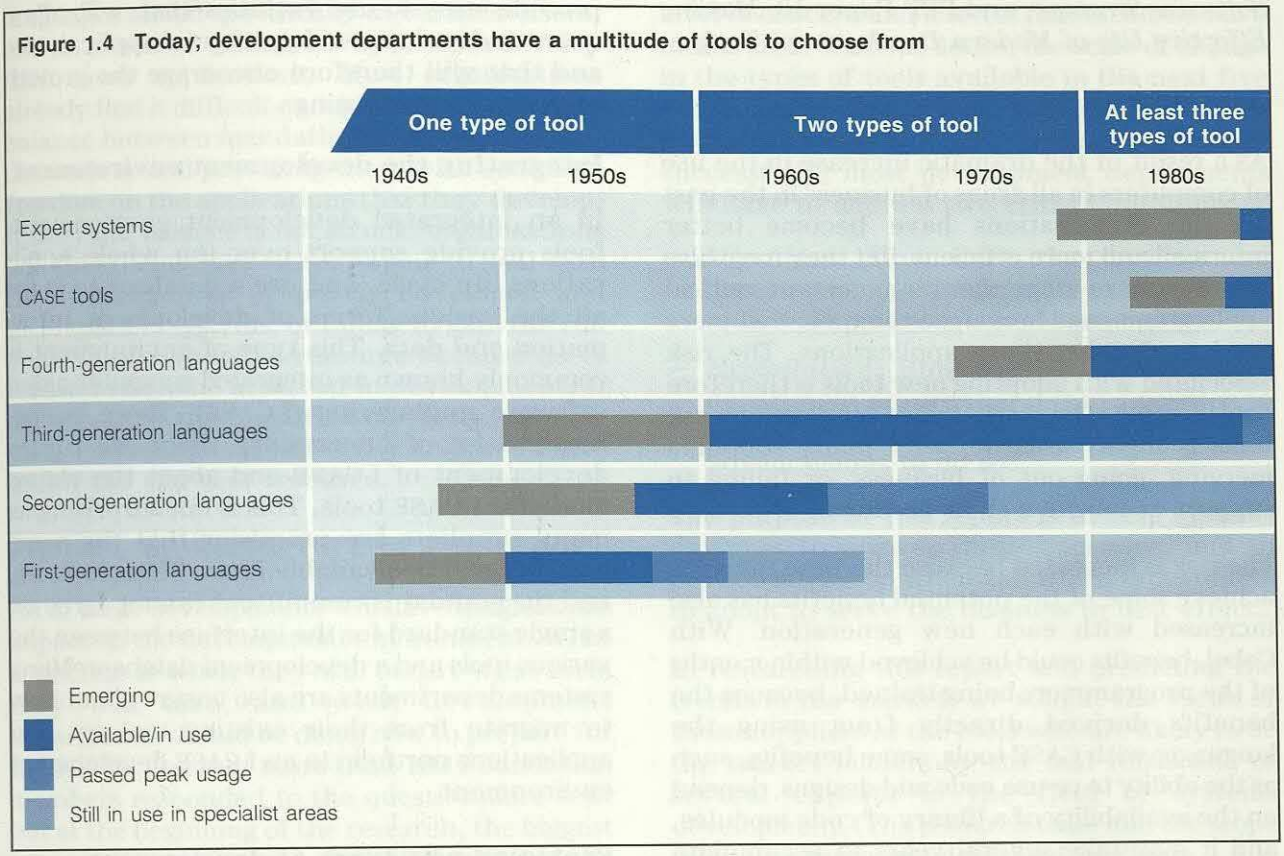
Increasing costs

Although the cost of any single type of tool has increased over time, the cost of adopting the latest generation of tools has increased dramatically, from hundreds of dollars in 1960 for a Cobol development environment (editor, compiler, and linker) to hundreds of thousands of dollars in 1990 for a CASE development environment. Even if inflation is taken into account, the cost of adopting a new generation of tools today has increased 100-fold since the 1960s. Furthermore, each new generation of tools has typically required an increasing amount of additional hardware, training, and consultancy. The cost of these has also increased substantially.

Managing increased complexity

Although each new generation of tools duplicates most of the functions provided by its predecessor, it is usually incompatible with its predecessor. The newer tools are therefore first used to develop new applications, while the older tools continue to be used to maintain existing applications. Most organisations, for example, will maintain Cobol code for at least two years (and possibly 10 or more years) after they stop using it as the main development

Figure 1.4 Today, development departments have a multitude of tools to choose from



language. Because of this, many organisations now use a mixture of third-generation languages, fourth-generation languages, some elementary CASE tools (being used as main-line development tools), and a few other development tools, such as assembler languages and expert-system shells, which are used for specialised applications.

This complex situation was confirmed by the results of a recent survey of members of Butler Cox's Productivity Enhancement Programme (PEP). Thirty-eight PEP members used over 60 different fourth-generation languages and application generators (tools that generate applications, typically in Cobol, from a series of definitions and screen-based forms), 36 of which were used by only one organisation. The most popular tool — FOCUS, from Information Builders Inc — was being used by only seven organisations. In most cases, using a wide variety of old and new tools is the only way the systems department can meet the demands of the business for new applications, but controlling such a development environment places a heavy burden on systems management.

The management task is further complicated by the lack of standards for linking either the tools or the applications produced with them, and by the overlap or gaps in the functionality provided by different tools. The problems caused by the lack of standards are particularly acute for applications written in fourth-generation languages. Although there are many fourth-generation languages available for a particular development environment, there are often very limited facilities for linking the applications written in each language. The gaps in the functionality provided by different tools means that it is not possible to use a set of tools that covers the complete applications life cycle.

It is the responsibility of the systems department to ensure that the tools available within the development department are carefully selected to minimise the gaps and overlap, and that they are used to best effect. Advice on these aspects of systems management has been provided in previous Foundation Reports and PEP Papers — notably Report 47, *The Effective use of System Building Tools*, Report 67, *Computer-Aided Software Engineering (CASE)*, Report 69,

Software Strategy, and PEP Paper 10, *Making Effective Use of Modern Development Tools*.

Accepting business risks

As a result of the dramatic increase in the use of computers in all areas of business in the past decade, organisations have become better informed and more efficient, but they have also become more dependent on certain critical applications, and hence, on the tools that were used to develop those applications. The risk associated with adopting new tools is therefore considerable. The marketplace for development tools is highly volatile, with many suppliers merging, going out of business, or failing to support or improve their products.

The risk is increased because the time taken to achieve some of the potential benefits has also increased with each new generation. With Cobol, benefits could be achieved within months of the programmers being trained, because the benefits derived directly from using the language; with CASE tools, some benefits, such as the ability to re-use code and designs, depend on the availability of a library of code modules, and it may take several years to accumulate such a library. There is therefore a risk that a new (and better) tool will emerge before the cost of the existing one has been fully recovered.

There is still much uncertainty about the future of development tools

As in the past, the new generation of tools now coming onto the market is promising to resolve current problems. The difference is that several types are emerging, all promising to address a much wider range of specific problem areas. Some suppliers are starting to integrate their tools both with their existing products, and with those from other suppliers, to create an integrated development environment, which has the added advantage of eliminating the need for an organisation to depend on a single tool supplier. Other suppliers are working on products that are not part of an integrated development environment, but that promise to provide facilities for developing applications to exploit emerging technologies, such as parallel computing. At the same time, increasingly powerful end-user tools are emerging. These will provide easy-to-use facilities that will allow users to develop a wider range of applications, and that will therefore encourage the growth of end-user computing.

Integrating the development environment

In an integrated development environment, tools provide support over the whole applications life cycle, and use a database to store all the various forms of development information and data. This type of environment is commonly known as integrated computer-aided software engineering (I-CASE). There is currently a lot of uncertainty about the future development of I-CASE and about the claims made for I-CASE tools. This is not surprising as many suppliers are promising that the same benefits will be available from different tools, and the standards committees cannot agree on a single standard for the interface between the various tools and a development database. Many systems departments are also unsure about how to migrate from their existing tool set and applications portfolio to an I-CASE development environment.

Providing new types of development facilities

Outside the I-CASE development environment, many other advances in tools are promising facilities to meet current needs for sophisticated decision-support systems, powerful user-friendly interfaces, and easily maintainable applications. One advance that has recently received considerable attention is the object-oriented approach to development and the associated tools and application development methods. It promises not only to increase productivity by making it easier to re-use code, but also to reduce the effort involved in maintaining applications (because the 'objects' represent things in the real world). The object-oriented approach to development requires staff who are skilled in identifying objects and the relationships between them. Its applicability to business-computing applications is still to be proven, however.

Enhancing the capabilities of end-user computing

Advances in end-user tools have been hastened by the introduction of fourth-generation

languages aimed specifically at business users, and an increased willingness by users to develop their own applications. Most organisations already find it difficult enough to strike the right balance between mandating what users do for themselves and providing them with complete freedom on the applications that they develop. If the right balance is not struck, organisations will find themselves either having to deal with the consequences of poorly developed and incompatible systems, or failing to gain the full benefits of a valuable resource because users refuse to become involved in development. If implemented and supported correctly, the new tools will result in very effective and productive use of the end-user resource.

Purpose and structure of the report

All of these developments will have a significant impact on the development environment. What is unclear is when they will occur, what form they will take, and what development departments should be doing now to prepare for them. The fact that more than 180 Foundation members responded to the questionnaire sent out at the beginning of the research, the biggest response ever received, is an indication of the level of concern and interest that members have in this area. Without doubt, the scale of change in the types of tools available in the next five years, and in the benefits achievable, will be greater than ever before. To manage this change successfully, most development departments will have to address two critical questions:

- *Migration:* How does the development department migrate from its existing set of tools and applications to the emerging tools?
- *Exploitation:* How does the development department ensure that it is in a position to exploit the emerging tools as soon as it is advantageous to do so?

The purpose of this report is to help systems directors to answer these questions, and to provide advice on when and which types of tool to adopt to serve the business to best effect.

In researching this report and predicting the trends in the market, we sought the views of those suppliers of the tools who are likely to be the market leaders by the mid-1990s and of several experts in the field of systems development. (The research team and the scope of the research are described in Figure 1.5.)

Figure 1.5 Research team and scope of the research

The research for this report was led by Kevan Jones, a consultant with Butler Cox in London, who specialises in systems development and has experience in the use of modern development tools. The work for this report included a worldwide research programme, which spanned the period from April to December 1989. Kevan Jones was assisted, in particular, by:

- Simon Forge, a principal consultant in Butler Cox's Paris office, with considerable experience in the field of systems development.
- Rob Moreton, an associate of Butler Cox, who is a principal lecturer at the City of Birmingham Polytechnic, and who specialises in systems development issues.
- Lothar Schmidt, a senior consultant in Butler Cox's Munich office, with extensive knowledge of the European software market.

Further research was carried out by John Cooper (Australia), Antonio Morawetz (Italy), and Per Hansen (Sweden), all associates or consultants with Butler Cox, and John Schmidt, a senior manager from Ernst & Young in the United States.

We conducted an extensive review of the published literature and carried out detailed interviews with tool suppliers in Europe and the United States. More than 180 Foundation members replied to the questionnaire sent out at the beginning of the research, and we subsequently held a series of workshops in the United Kingdom and France to gather more detailed information on particular aspects of advances in the development-tool market.

In addition, we conducted interviews with various experts in the field in France, Germany, Austria, Italy, Sweden, Australia, and the United Kingdom. We should like to offer our special thanks to Russell Jones, editor of the Systems Development Monitor, David Gradwell, founder of Data Dictionary Systems Limited and Rapporteur for the International Standards Organisation Committee for Information Resource Dictionary Standards, and Dr Gilles Kahn, a Research Director at INRIA, France, who specialises in development aids.

We also drew on the experience of experts within Butler Cox, on our consultancy work in the systems development area, and on the Butler Cox Productivity Enhancement Programme (PEP) database, which is now the most comprehensive source of information in Europe about the characteristics of systems development projects.

From our research, it is clear that one of the most important of the advances is the I-CASE development environment. This promises to provide a stable and flexible development environment that will allow systems departments to mix and match tools from different suppliers. The adoption and implementation of an I-CASE development environment will have a major impact on the development department and will provide many benefits. It will not, however, be the solution to all development problems. Currently, it is not clear what form an I-CASE development environment will take, who the major suppliers will be, or to what standards they will conform. In Chapter 2, we look at the major suppliers and players in the developing I-CASE market and provide guidance on the gradual adoption of I-CASE products.

For most development departments, an I-CASE development environment will provide an effective means of developing many of the applications required, but such an environment will not be fully available for several years. In Chapter 3, we look at the steps that organisations can take in the meantime, without creating barriers to the subsequent smooth migration towards an I-CASE development environment.

Significant advances in tools are also occurring outside the I-CASE development environment, and these should not be ignored because they, too, will provide benefits. In Chapter 4, we describe these advances and identify the tools that should be exploited.

Many of the advances occurring in the field of development tools will mean that the tools will be easier to use, both by professional developers and by business users. Chapter 5 is concerned with the very important area of development tools that can be used by business users. The potential growth in end-user computing, and the benefits that can be gained from it, are enormous. End-user computing is already well established in some organisations. In others, however, there are often barriers preventing it from being fully exploited. In this chapter, we show how advances in tools that can be used by business users can overcome these barriers and encourage the growth of productive end-user computing, provided that the tools are properly introduced and supported.

Chapter 2

Plan for the future with I-CASE in mind

Most development departments are currently operating in a far-from-ideal situation. They are under increasing pressure from the business to provide more sophisticated applications, with improved development productivity, but the tools that they have available do not provide full coverage of the applications life cycle, nor are the different tools usually either compatible or integrated with each other (see Figure 2.1, overleaf). The lack of an integrated set of tools means that applications may be of uncertain or poor quality and that considerable effort may be needed to provide automated support for the whole of the applications life cycle. I-CASE (integrated computer-aided software engineering), a development environment that supports a set of integrated tools, promises to provide a solution to these problems.

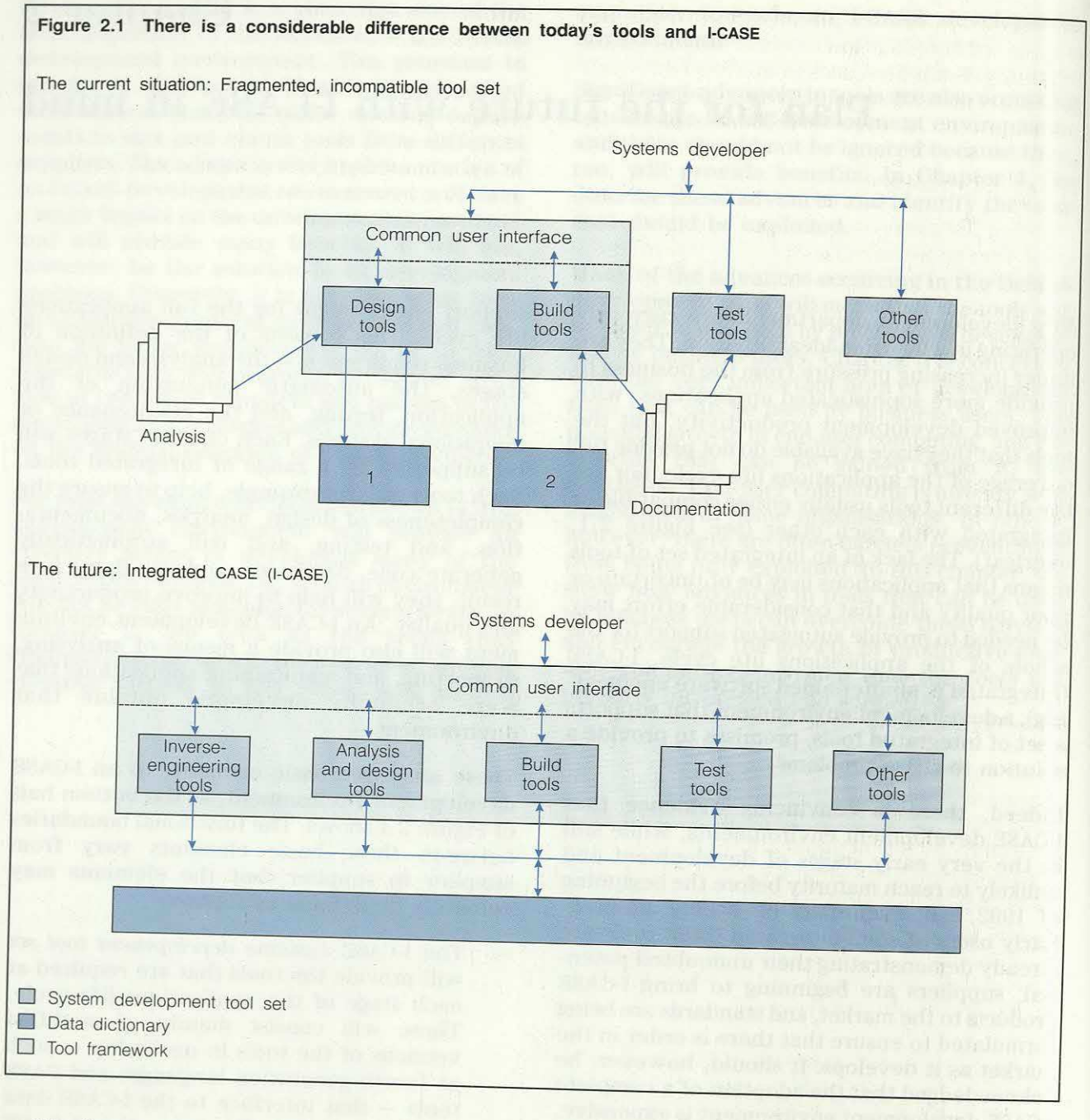
Indeed, there is convincing evidence that I-CASE development environments, while still in the very early stages of development and unlikely to reach maturity before the beginning of 1992, will eventually be widely adopted. Early users of semi-integrated CASE tools are already demonstrating their undoubted potential, suppliers are beginning to bring I-CASE products to the market, and standards are being formulated to ensure that there is order in the market as it develops. It should, however, be acknowledged that the adoption of a complete I-CASE development environment is expensive, and risky while the market is still in an embryonic stage of development. Furthermore, it requires a high level of commitment if the potential benefits are to be realised. Until I-CASE matures and becomes more firmly established, it is therefore prudent to plan for migration towards it, not making a full commitment to it yet, but leaving the path clear for its eventual adoption.

The term I-CASE is used to describe a set of tools that provides an integrated and semi-automated support environment for the full applications life cycle. This consists of the definition of business requirements, the analysis and design stages, the automatic generation of the application, testing, and the maintenance of operational systems. Each of these stages will be supported by a range of integrated tools. Such tools will, for example, help to ensure the completeness of design, analysis, documentation, and testing, and will automatically generate code, databases, and test data. As a result, they will help to improve productivity and quality. An I-CASE development environment will also provide a means of analysing, supporting, and maintaining applications that were originally developed outside that environment.

There are three basic elements to an I-CASE development environment, as the bottom half of Figure 2.1 shows. The functional boundaries between these basic elements vary from supplier to supplier, but the elements may generally be defined as follows:

- The I-CASE *systems development tool set* will provide the tools that are required at each stage of the applications life cycle. These will consist mainly of modified versions of the tools in use today — such as fourth-generation languages and CASE tools — that interface to the I-CASE data dictionary and fit into the I-CASE framework (see below).
- The I-CASE *framework* will provide both a standard interface to which all the tools must conform, and a means of supporting the project team at every stage of development via a common user interface. The framework can be either a standard data dictionary interface, or a layer of software that provides the interface between the tools and the data dictionary.

Figure 2.1 There is a considerable difference between today's tools and I-CASE



The framework is sometimes also known as the integrated project-support environment (IPSE).

— The I-CASE *data dictionary* will be the store for all development information, for all stages of the applications life cycle. It will therefore support the flow of information between the various life-cycle stages. The data dictionary is also known as the repository, encyclopaedia, or information resource dictionary system (IRDS).

The potential benefits of I-CASE are considerable

Although there are no complete I-CASE development environments available today, the potential benefits are considerable. I-CASE promises to improve the quality of applications, to increase productivity and usability, and to reduce dependence on individual tool suppliers. Experience of using CASE tools, which will form part of the I-CASE development environment,

shows that they are already improving the quality of applications, and hence, their reliability. Such increases in quality will result in improvements in productivity, particularly in the maintenance stage of the life cycle.

Improved quality

The data dictionary lies at the heart of I-CASE and serves as the means of passing information automatically from one tool to another. By using the dictionary in this way, the errors often introduced when an analyst or programmer changes from one tool to another, or uses a tool to implement a paper-based specification, are eliminated.

An I-CASE development environment also provides an effective means of communication between development staff and users. This is particularly so at the analysis and design stages, where systems developers and users can work together to develop the specification, using the interactive screen-based analysis and design tools. This will result in problems being resolved at the design stage, not at the programming stage. In turn, this will result in applications that are a better fit with the users' needs.

The use of the I-CASE data dictionary as the central store of all development information also means that common definitions and routines can be used by different applications, thus improving quality (and productivity) because it will be easier to re-use existing code, and ensuring greater compatibility between different applications.

Increased productivity

Increased productivity in the maintenance of applications will result from improved quality, because less time will be spent removing errors introduced at the main-build stage. Productivity at the main-build stage will also increase because it will not be necessary to respecify information already stored in the dictionary, thereby reducing the opportunities for introducing new errors. Once information has been stored in the dictionary by one of the I-CASE tools, developers can be certain that that information can be used without modification by any of the other tools. Increased productivity in the analysis, design, and development of applications will result from increased automation and greater re-use of design information.

The I-CASE data dictionary will also increase productivity by providing development staff with access to all past applications. If this information is well organised, developers should easily be able to locate designs or code that are similar to those required for a new application. Re-use of designs or code in this way will obviously improve productivity.

Some automated code and application generators have been available for several years. During the first two years of the 1990s, more will be introduced and they will become more sophisticated, making it possible to generate applications and test data automatically from the output of the analysis and design process. They will be used to construct applications, reports, and screen layouts, largely removing the need for traditional programming.

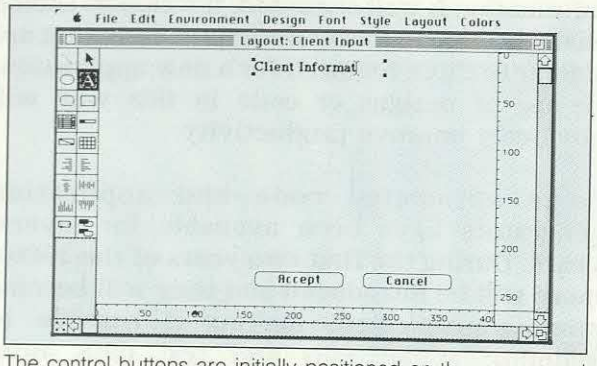
Report-generator and screen-painting tools will make it possible for a developer, or a business user with knowledge of the application, to construct complex screen layouts with icons, buttons, and pull-down menus, in a matter of minutes. These report generators will form part of the tool set in the I-CASE development environment, dramatically improving the productivity of constructing applications. Figure 2.2, overleaf, shows the stages in the construction of a screen layout, using Fourth Dimension from Analyses Conseils Informations (ACI), which took three minutes to construct. Fourth Dimension also allows screen layouts and reports based on various standard formats and data definitions to be generated automatically. Standard-format screens can be generated in seconds with this option.

Greater ease of use

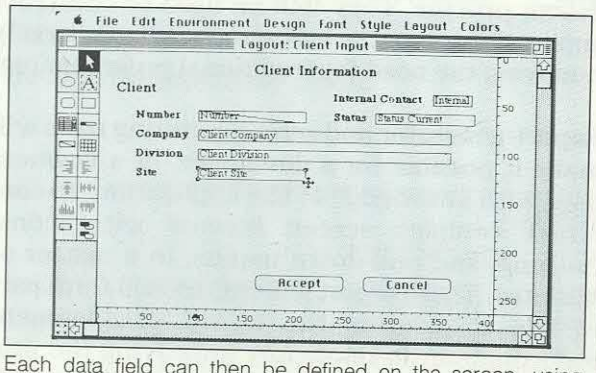
While the use of tools in the I-CASE development environment will automate many of the more mundane or repetitive tasks involved in development, the developer will still have to make complex decisions concerning the use of the development methods and the appropriate tools. The I-CASE development environment will provide appropriate guidance to the developer and easy-to-use interfaces to the various tools that support the whole life cycle.

Figure 2.2 Powerful I-CASE automation tools will improve productivity

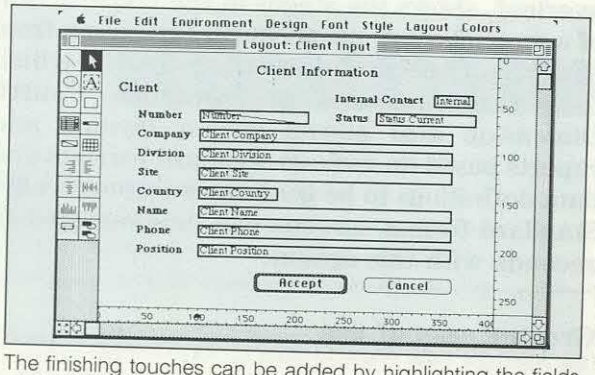
These layouts were created using the screen-painting facility provided by the application generator, Fourth Dimension, from ACI.



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.

Expert systems embedded in the I-CASE development environment will provide access to a knowledge base about the development process and will provide guidance or advice, when required. These 'smart' I-CASE development environments will also have the ability to add to, or modify, their knowledge base, with the assistance of the user, to reflect the latest methods or procedures. Using such an environment to automate a development task will reduce the number of human errors introduced into an application and thus improve the quality of the delivered application. Smart I-CASE tools will also help to improve the operational performance of applications by providing developers with advice about the most efficient data structures and coding.

The I-CASE development environment will also provide a consistent user interface for development staff, probably based on windowing techniques, which is independent of the tool being used. This will reduce the time it takes to become familiar with new tools, and thus increase development productivity. It is important to remember, however, that these advantages are not restricted to an I-CASE development environment.

Reduced dependence on tool suppliers

In the future, an I-CASE data dictionary will store all data relating to past, current, and future projects in a standard manner (and independent of most of the tools used), which means that organisations will be less dependent on a particular tool supplier. In effect, it will be possible to plug tools into the I-CASE tool framework, or indeed, unplug them, as and when required, with little impact on the application base. The exception is where the tool stores proprietary information in the data dictionary, such as non-standard code. To achieve true independence, however, will require all tool suppliers to conform with the same standard for storing information in a data dictionary.

Suppliers are beginning to commit to I-CASE

While a fully functional I-CASE development environment is unlikely to be available before the beginning of 1992, many tool suppliers are already offering semi-integrated CASE tools (a set of CASE tools that provide coverage over a large portion of the applications life cycle). These include Information Engineering Facility (IEF) from Texas Instruments and James Martin Associates, FOUNDATION Integrated Environment for Software Engineering from

Andersen Consulting, IEW from Ernst &Young and Knowledgeware Inc, and CASE* from Oracle Corporation. As products like these mature, they will form the basis of I-CASE development environments. Some organisations are already using these tools and are starting to reap the benefits. EBES, a Belgian utility company, implemented the IEF tool set in 1988. Its experience is described in Figure 2.3.

Suppliers are, however, taking different approaches to developing the basic components of I-CASE, and currently fall into one of three groups. The first group comprises hardware and software suppliers who are cooperating to produce software for an I-CASE development environment. The second group comprises independent software suppliers, and the third group comprises suppliers of development methods, who are working to provide the improved versions of their products that are essential for the effective use of I-CASE.

Cooperating hardware and software suppliers

Several of the major hardware suppliers have realised the importance of an I-CASE development environment (in particular, its potential to sell additional hardware), and have announced their commitment to it. Rather than compete with the established CASE suppliers, they have promised to provide a data dictionary and a 'standard' framework in which to integrate the CASE tools and other tools that are available today.

IBM

In September 1989, IBM announced its I-CASE development environment, called AD/Cycle, which provides a standard development environment that complies with IBM's Systems Application Architecture (SAA). Within this development environment, there will be a DB2

Figure 2.3 Users of integrated CASE tools are reporting early successes

EBES

EBES is a privately owned company that produces electricity and distributes electricity, gas, water, and cable TV to the northern part of Belgium. It has a systems development department with a total of 100 staff. Of these, 40 are primarily involved in developing applications (financial, personnel, materials handling, and customer information systems) and are organised by business area. The other 60 provide support to the users of the 2,000 terminals in an IBM mainframe environment. In the past, EBES suffered from the same problems as most development departments; it had a large maintenance load and an applications backlog that was continually growing.

A company-wide strategic-planning study drew attention to the need for more rigorous information planning, and recommended the introduction of a computer-aided tool to improve the applications-development process. In March 1988, after evaluating various products, EBES decided to adopt the Information Engineering Facility (IEF) tools set and its associated Information Engineering Method (IEM), from Texas Instruments/James Martin Associates.

After some success with a limited pilot project, EBES realised the full implications of adopting such a tool set. To achieve the potential benefits in quality and productivity, the systems department would have to change its development methods, the development approach, hardware, and the organisation structure itself.

EBES therefore carried out a further study to define an environment in which the methodology and its associated I-CASE tools could be effectively and efficiently used. It resulted in the development of a model of the company, which grouped the business areas into 14 natural 'clusters'. These are the basis on which the long-term architectures of the company's information systems, its hardware strategy, and the organisational implications of such changes, are assessed.

One year after completing the pilot-testing phase, the company has developed three applications with the IEF tool set and methods. They have taken about the same amount of time and effort as applications using traditional development tools and methods. EBES believes, however, that the learning phase is now complete, and that future developments using the new tool set will take much less time and effort. Furthermore, the applications developed with IEF are of high quality because the developers were able to concentrate on the business aims of the applications, rather than on their technical features. Several complete areas of the business have been analysed in preparation for using IEF to develop further applications, and ambitious plans have been made in many areas.

Although EBES spent only a matter of months learning how to use the tool set, it took a further year to understand the full implications of introducing such a fundamentally new approach to development. EBES has shown that, if correctly planned and managed, the introduction of such tool sets will result in immediate benefits in terms of quality, and in productivity benefits within six to 12 months. The tool set will not, however, provide all the benefits on its own; changes also have to be made in development methods and the development approach, and in various aspects of the organisation structure.

repository (data dictionary), a tool framework (the Repository Manager), and numerous tools supplied by IBM and other software suppliers. Thirty-five suppliers have so far committed to providing tools that will be compatible with AD/Cycle. The three suppliers providing the major elements of the tool set are Knowledgeware, Index Technologies, and Bachman Information Systems Inc.

Although the repository is far from complete, IBM has very ambitious plans to have a working version of AD/Cycle available by July 1990. Experts in the field, however, do not expect to see AD/Cycle appear until well into 1991, and they expect the initial version to be a very basic product.

Digital

Digital has an I-CASE strategy similar in some respects to that of IBM. Several of the basic products are already available and integrated with Digital's data dictionary, Common Data Dictionary Plus (CDD/Plus), which was released at the end of 1988. Digital's main thrust, however, is with its object-oriented interface between the tools and the data dictionary. This interface, known as 'a tools integration standard', is expected to be available in mid-1990 and will provide an integrated project-support environment. Currently, DECdesign, a front-end analysis and design tool, can be used to load high-level design information into the data dictionary, but it is still necessary for a programmer to translate the design information into detailed implementation logic for Digital's Vax Cobol generator. There are plans, however, to improve the level of integration in the near future. As with IBM, there is considerable collaboration with other tool suppliers — Digital currently has agreements with 20 different tool suppliers.

ICL

Although ICL is not a major international hardware supplier, its products in the I-CASE area are very advanced. ICL has an established set of integrated tools in the form of QuickBuild, a set of cooperating products for the management of information and the rapid development of applications. The basic product consists of Data Dictionary System (DDS), a fourth-generation language (Application Master), an application and database generator suitable for use in the early stages of development (Automatic System Generator), a query language (Query Master), a reporting language (Report Master), various other development aids, and a development method. DDS is the most technically advanced data-dictionary product available on the market.

Over the next few years, ICL plans to enhance the QuickBuild set of products so that it includes a three-tier data-dictionary structure, as shown in Figure 2.4. This structure includes the Advanced Development Dictionary (ADD), from the Sema Group, which complements ICL's own DDS, providing a distributed data-dictionary system across both ICL and Unix hardware. One of the benefits of such a configuration is that high-performance transaction-processing applications can exploit the IDMSX database on a mainframe, and management information systems or distributed applications can be based on either a mainframe or minicomputers exploiting Ingres. ICL expects the basic facility of the ADD, acting as a local dictionary supporting the DDS corporately, to be available in the middle of 1990. In addition, ICL is currently talking to several independent tool suppliers with a view to integrating various tools with its I-CASE development environment.

Independent software suppliers

Independent software suppliers are also developing tools that will 'fit into' an I-CASE development environment or than can evolve into full I-CASE products. Many of the suppliers of existing CASE tools or semi-integrated CASE tools are already well established, with their products being used by many organisations. Some of the most prominent are Oracle Corporation, Computer Associates, Cincom, Software AG, Cortex, Ernst & Young, and Texas Instruments and James Martin Associates. Figure 2.5, lists some of these, with details of their products, the hardware on which they run, and the coverage they provide over the applications life cycle.

Oracle Corporation offers an 'open' type of semi-integrated CASE product, called CASE*. This can be used on a wide range of hardware, from IBM 370-type mainframes down. The data dictionary, CASE*Dictionary, is based on an Oracle database and can be accessed via a variety of predefined routines or by using IBM's SQL, the industry-standard database-access language. There is also a variety of tools, such

Figure 2.4 ICL is aiming to provide a hierarchy of data dictionaries

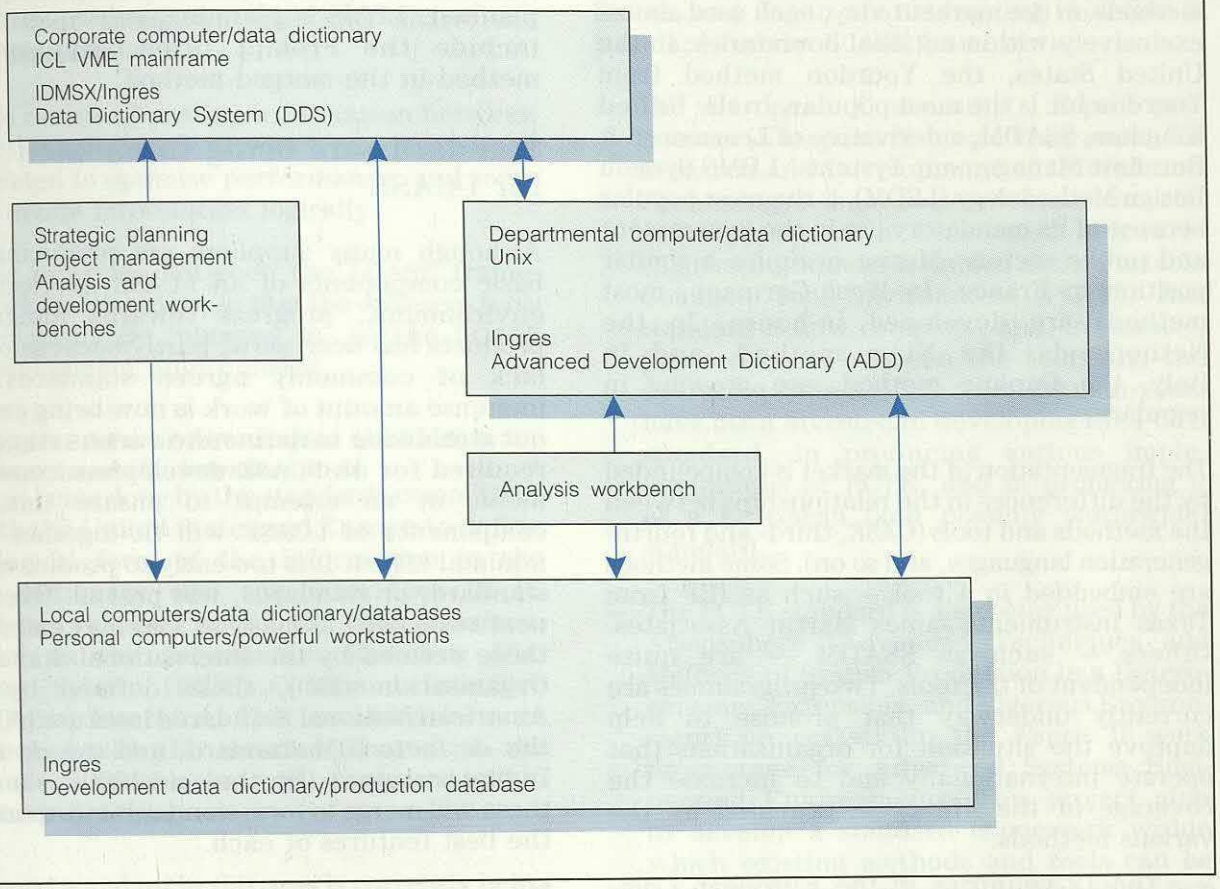


Figure 2.5 Established suppliers are already offering semi-integrated CASE products

Supplier	Product	Hardware	Coverage of the applications life cycle: Feasibility study	Analysis/ design	Main build	Maintenance	Project management
Oracle Corporation	CASE*	Many mainframes and PCs		■	■	■	
Cortex	CorVision	DEC Vax, DEC PC, IBM PC		■	■	■	
Ernst & Young	IEW	IBM mainframes and PCs	■	■	■		
Texas Instruments/ James Martin Associates	IEF	IBM mainframes and PCs	■	■	■	■	

as CASE*Designer and CASE*Generator, that provide the developer with a flexible and powerful development environment, and interfaces to a wide range of third- and fourth-generation languages. The Oracle CASE* product set supports various structured development methods including CASE*Method and IEF.

Methods suppliers

There are many applications-development methods on the market today, each used almost exclusively within national boundaries. In the United States, the Yourdon method from Yourdon Inc is the most popular. In the United Kingdom, SSADM, a derivative of Learmonth & Burchett Management Systems' LBMS System Design Methodology (LSDM), is the most popular because of its mandatory use in the government and public sectors. Merise occupies a similar position in France. In West Germany, most methods are developed in-house. In the Netherlands, the Niam method, and in Italy, the Daphne method, are growing in popularity.

The fragmentation of the market is compounded by the differences in the relationships between the methods and tools (CASE, third- and fourth-generation languages, and so on). Some methods are embedded in a tool — such as IEF from Texas Instruments/James Martin Associates. Others — such as SSADM — are quite independent of the tools. Two programmes are currently underway that promise to help improve the situation for organisations that operate internationally and to increase the coverage of the life cycle provided by the various methods:

- The 12 countries in the European Community are discussing the possibility of producing a Euro-method. It would probably be a combination of SSADM and Merise, with elements of Niam. Although this is unlikely to appear for several years, it will be of crucial importance to pan-European organisations, and of increasing interest to all organisations as 1992 approaches.
- The SSADM Research Centre in the United Kingdom has set up a scheme whereby tool suppliers can assess the compliance of their products with SSADM. The results of the compliance tests will be published. If this scheme is a success, it will provide potential buyers with guidance on which tools comply with the method, and it may encourage suppliers of other methods to do likewise.

Early in 1990, LBMS acquired Michael Jackson Systems Ltd. LBMS is now working towards merging its method, LSDM, with Jackson Structured Programming. This will enable the realtime version of LSDM to be brought to the market 12 months earlier than originally planned. LBMS is planning, subsequently, to include the Prompt project-management method in the merged method.

Standards are being formulated for I-CASE

Although many suppliers are producing the basic components of an I-CASE development environment, progress towards integrated products has been slow, partly because of the lack of commonly agreed standards. An immense amount of work is now being carried out worldwide to define the various standards required for an I-CASE development environment, in an attempt to ensure that the components of I-CASE will fit together with minimal effort. It is too early to predict which standard, or standards, will prevail. Over the next two years, as many as four may emerge — those defined by the International Standards Organisation (ISO), those defined by the American National Standards Institute (ANSI), the *de facto* IBM standard, and the *de facto* Digital standard. By the mid-1990s, some of these will merge to form standards incorporating the best features of each.

Most potential I-CASE suppliers are already releasing statements of intent of future compliance with standards, but some data dictionary products and CASE tools are being introduced that do not comply with any of the standards. If a full I-CASE development environment is to emerge, suppliers and users must back the standards committees by contributing to the development of the standards, and adopting them when they are defined. Failure to do so could result in the creation of several I-CASE development environments, based on proprietary standards controlled by the suppliers of the software or hardware. Development tools will then be restricted to a particular proprietary I-CASE environment, thus limiting the potential benefits to be derived from I-CASE.

Standards need to be created in three areas

Standardisation is needed in three areas before an 'open' I-CASE development environment can be achieved:

- The logical form of the information held in the data dictionary, and the means of accessing and manipulating it, so that different tools can use and share the information.

- The means of passing information between dictionaries, some of which may have been created to optimise performance, and some to divide information logically.

- The interface between the I-CASE framework and the tools, so that the different tools can easily be 'plugged in' to the I-CASE development environment.

The logical form of information in the data dictionary and the interface to it

Work is being done by the standards committees in both the United States and Europe to define the logical form of the information in the data dictionary and the interfaces to it. ANSI has released its X3.137-1988 standard, known as the Information Resource Dictionary System (IRDS) Command Interface and Panel Interface. This standard is more concerned with the interface than with the form of information that will be required within the data dictionary.

A second standard in this area is currently being defined by the ISO. Also known as IRDS, it is concerned with an overall set of standards for the interface to the data dictionary, as well as standards for the forms of information that will be held within it. A working draft of this standard is due for release in 1991/92.

The means of passing information between dictionaries

Through its IRDS Export/Import project, the ISO is also developing standards that will make it possible to move the contents of one data dictionary, and its definitions, to another. These standards will overcome the difficulties associated with multiple data dictionaries, and make it possible to use a hierarchy of dictionaries, in the form illustrated earlier in Figure 2.4. Work is also being done in this area by the Electronic Design Interchange Format (EDIF)/CASE group in the United States. To date, the EDIF/CASE group has provided only an outline of the functions that such a standard would require.

The interface between the I-CASE framework and the tools

Many research projects in Europe, the United States, and Japan are attempting to define a standard for the I-CASE framework:

- The European Community's Esprit programme (European Strategic Programme for Research and Development of Information Technology) has concentrated on the development of a Unix-based software-engineering standard, known as the Portable Common Tool Environment (PCTE). This standard is now complete, and, in essence, forms a layer between the tools and the operating system. More than 500 man-years have been invested in developing the PCTE standard, in producing various implementations of PCTE, and in promoting it as a European (and possibly worldwide) standard.

- The PCTE standard is also being used by the Atmosphere (Advanced Techniques and Methods of System Production in a Heterogeneous, Extensible, and Rigorous Environment) project within the Esprit II work programme for Advanced Systems Engineering Environments. This project aims to develop a standard framework within which existing methods and tools can be integrated, and is expected to take about five years to complete. It currently involves 38 contractors from 13 nations. The main partners are listed in Figure 2.6.

- The Alvey programme was set up in 1983 by three UK government departments, British industry, and academia, in response to increasing overseas competition, and in

Figure 2.6 There are seven main partners in the Esprit Atmosphere project aimed at producing a standard development framework

Organisation	Country
ASSOC CAP SESA INNOVATION	France
Bull SA	France
GEC	United Kingdom
Nixdorf Computer AG	West Germany
Philips Gloeilampenfabrieken NV	Netherlands
Société Française de Génie Logiciel (SFGL)	France
Siemens AG	West Germany

particular, to the Japanese Fifth-Generation Computer Project. The programme had the objective of stimulating British IT research through collaborative projects. In January 1988, the Alvey Directorate was subsumed into the new Information Engineering Directorate of the Department of Trade and Industry. Many Alvey projects are now complete, and several are still underway. Between 1983 and 1987, £33 million ($52.8 million) was committed to research and development into various aspects of software engineering, particularly integrated project support environments (IPSEs). Several commercially available products have been developed as a result of early work on two IPSE projects, Aspect and Eclipse. The IPSE 2.5 Project is due to finish in early 1990. Some of the research from this and other projects will be carried forward and exploited by the European Software Factory project as part of the Eureka programme, which encourages industry-led projects with European Community and other European partners.

— A project in the United States, sponsored by the US Department of Defense, has developed the Common APSE (Ada Programming Support Environment) Interface Standard (CAIS). CAIS was first developed in 1982 to resolve the incompatibility between Ada development environments in the American Army and Navy. Although CAIS has some very highly regarded features, it is judged by the PCTE community to be three years behind the work being done in Europe.

— In Japan, the Sigma (software industrialised generator and maintenance aids) project was established in the mid-1980s and involves more than 190 software companies. Its aim is to develop a common support environment for developers all over Japan, thereby improving productivity and quality and increasing the sharing of information among development staff. The project is due to finish in the middle of 1990. Sigma has two distinct elements: a standard software development environment and a networked system for information exchange. In the context of I-CASE, the former is of greater importance, although it is not as comprehensive as other developments in the United States and Europe. The development environment is based on a Unix workstation running a standard set of 50 integrated tools. The Sigma products may be available in Europe before the end of 1990. (Delegates on the Foundation's Study Tour of Japan in 1986 met with Professor Ohno, Chairman of the Sigma system development committee and one of the founders of the project. Details of his presentation can be found in the 1986 Study Tour Presentation Summaries.)

Both PCTE and CAIS have their origins in the technical computing field and were developed for Unix environments. So far, they have had little impact in the commercial computing field, but this will change as Unix becomes more popular in the commercial field and as more suppliers and organisations see the benefits of these standards. Currently, the PCTE and CAIS communities are assessing the possibility of merging the two standards to form one common standard, taking the best features from each.

Most suppliers intend to comply with the standards

Hardware suppliers are issuing statements of intent of future compliance with the various I-CASE standards, and software suppliers are promising to conform with those hardware suppliers' products. We were, however, unable to identify any group or organisation responsible for verifying the extent to which suppliers are, in fact, conforming to these standards.

IBM, Digital, and ICL (the three major hardware suppliers who have developed, or are in the process of developing, I-CASE products) have already committed to conforming to some degree to either the ANSI and/or the ISO IRDS standard:

— IBM has acknowledged that its Repository Manager/MVS (the AD/Cycle data dictionary) does not fully conform to the ANSI IRDS standard. It does, however, provide most of the services and capabilities required for compliance with the standard. In fact, the functions provided by the

entity-relationship model exceed those required by the ANSI standard.

- To try to ensure the 'openness' of its product, Digital has submitted its tools integration standard to the various standards organisations as a proposed standard for all I-CASE development environments. The term 'open' means that the Digital I-CASE environment will support other tools and the development of applications for other hardware and software environments.
- ICL's Data Dictionary System already fulfils the general requirements of the ANSI IRDS standard and has facilities in areas covered by the first draft of the ISO IRDS framework.

Most independent software suppliers are releasing statements of intent to conform to at least one of the hardware suppliers' data-dictionary or I-CASE framework standards. Oracle, for instance, will conform to whichever standards lead the market (which may result in its products complying with more than one standard), and IEF will conform to the IBM standards for AD/Cycle.

The levels of cost, risk, and commitment associated with I-CASE are high

Although the potential benefits of a complete I-CASE development environment are considerable, the levels of investment required are significant, and as with any area of IT in the early stages of its growth, the risks are high. In addition, to exploit the full potential of I-CASE, organisations need to make a substantial commitment of time and effort, both to cope with the changes required to incorporate I-CASE into the development environment, and to manage the complex environment that will be created by having to use I-CASE as well as traditional development methods and tools.

I-CASE will be costly and will require additional hardware

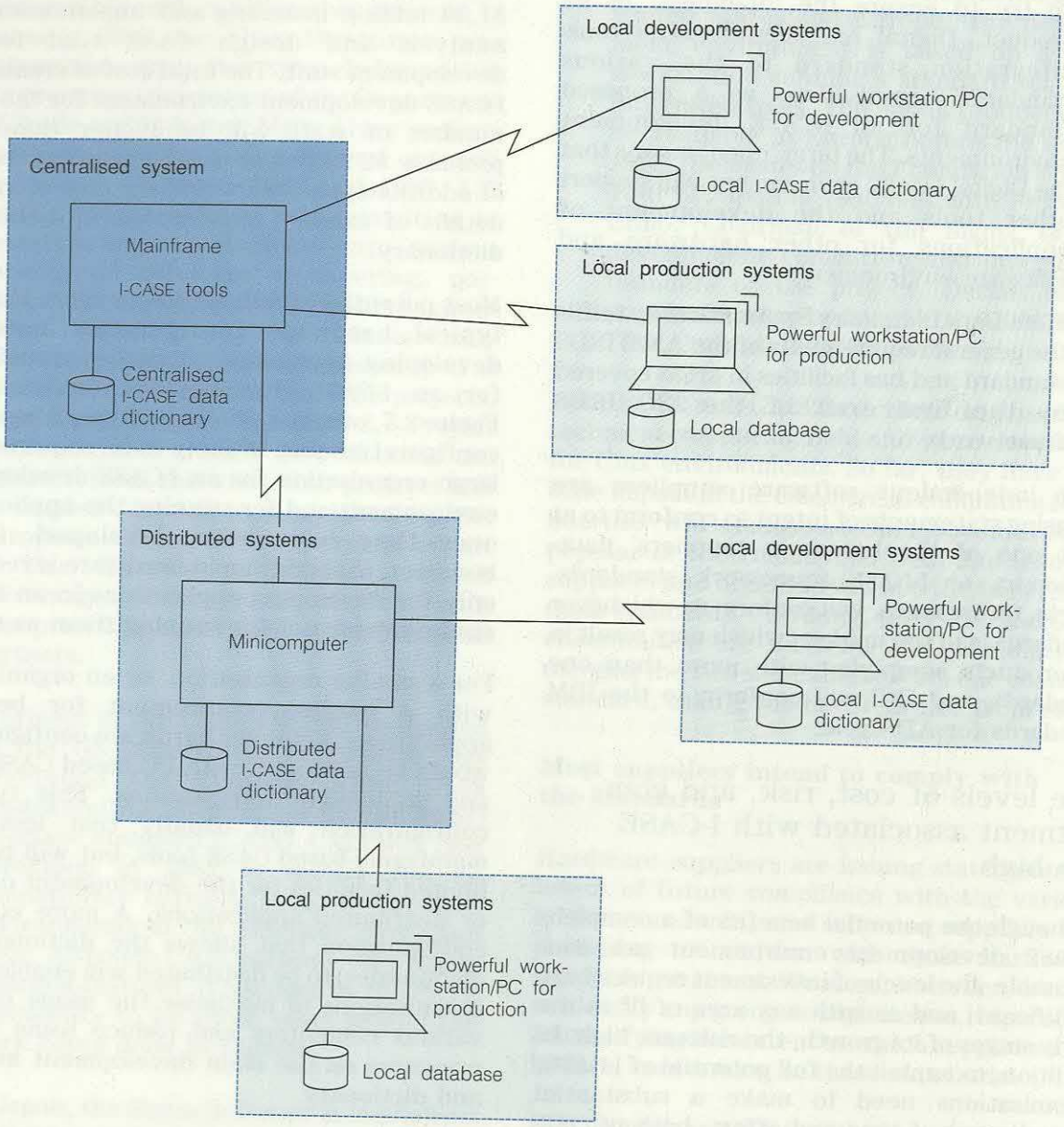
Although no complete I-CASE development environment yet exists, the cost of implementing currently available CASE tools indicates the level of investment that is likely to be required. In Report 67, *Computer-Aided Software Engineering (CASE)*, we quoted the experience of one organisation that had spent $1.34 million installing and implementing an analysis and design CASE tool for 30 development staff. The total cost of creating an I-CASE development environment for the same number of staff will be higher than this, probably $2 million or more, excluding the cost of additional hardware and the cost of loading details of existing applications into the data dictionary.

Most potential I-CASE suppliers agree that the typical hardware configuration used for developing applications today is insufficient for an I-CASE development environment. Figure 2.7, overleaf, shows the typical hardware configuration that is likely to be required by a large organisation for an I-CASE development environment and for running the applications once they have been developed. (Note, however, that additional hardware is required only for developing applications in an I-CASE environment, not for running them as well.)

For a smaller organisation, or an organisation with a smaller requirement for bespoke applications, a simpler hardware configuration would be adequate, with PC-based CASE tools and standalone workstations. This type of configuration will usually cost less than mainframe-based CASE tools, but will provide limited facilities for the development of large or distributed applications. A more complex configuration that allows the dictionary (or dictionaries) to be distributed will enable larger organisations to maximise the usage of their various computers and reduce some of the pressures on the main development machine and dictionary.

The introduction of I-CASE will hasten the trend towards powerful networked workstations. Today, CASE-tool suppliers perceive the personal computer, rather than the dumb terminal, as the device that will increasingly be used by development staff, but the basic PC available today cannot provide the facilities that will be required for a full I-CASE development environment. A typical developer using an I-CASE tool will require a large, high-resolution colour screen, access to fast, high-resolution printers and plotters, sufficient processing power and memory (at least 4 megabytes) to

Figure 2.7 The I-CASE development environment requires a complex hardware configuration for developing and running applications



drive these devices, and probably at least 100 megabytes of disc storage. The workstations will also need to be networked so they can access a centrally controlled dictionary and so that developers can intercommunicate with each other. This type of workstation is available today (the PS2/70, for instance), but costs several times as much as a basic PC.

In practice, most organisations are likely to use powerful networked workstations for the I-CASE development environment and PCs for other non-I-CASE development tools. PCs, rather than powerful workstations, will also be used as the user interface for most of the operational systems developed in the I-CASE environment, because the processing power, storage, and screen resolution available with PCs will be perfectly adequate for most applications.

An immature I-CASE marketplace leads to high risks

Adopting I-CASE today is risky because the cost of the products (and of implementing them) is high, the products are unproven, and standards are not, as yet, well defined. Until the standards issues are resolved, organisations have two options. Those who purchase hardware from a single major supplier who has issued a statement of intent to conform to a particular standard can select tools that conform to the same standard. For others, choosing the 'open' CASE tools and other non-CASE tools that are emerging will be the least risky approach, because these will not commit them to a particular I-CASE development environment.

Most of the semi-integrated CASE products available today, which could evolve to become I-CASE products, have at least one of the technical shortcomings discussed below. While these are not necessarily equally important to every organisation, the shortcomings of today's products will need to be overcome before I-CASE can become established as a really effective development environment.

Lack of full life-cycle support: The full benefits of I-CASE cannot be achieved unless the tools provide full support over the whole life cycle. If full coverage is not available, it will be necessary either to transfer information to and from tools outside the I-CASE environment, or to use non-automated methods to fill in the gaps. There are still, for instance, very few methods or tools for automatically converting logical designs into physical designs, few tools with powerful testing facilities, and few with flexible reporting facilities based on windowing techniques and graphical displays. In each case, it will not be possible to use the data dictionary to ensure consistency and accuracy, thereby increasing the probability of introducing errors.

Difficulty of a distributed development environment with a centralised data dictionary: In any I-CASE development environment, several developers will need to interact with the data dictionary at the same time, and the hardware and software must allow this to happen with reasonable response times. Problems may arise if the dictionary is held centrally on one computer, and many believe that a distributed data dictionary is the only way of overcoming these problems. The precise way in which the dictionary is distributed will depend on the individual organisation. The critical issue, however, is how the dictionary is updated and how information is controlled as it is passed from one part of the distributed dictionary to another.

Lack of control facilities for a distributed data dictionary: With a multi-user distributed data dictionary, there must be control facilities to prevent the same information being updated simultaneously, by several users, and to provide adequate recovery, integrity, and file-locking features. These issues are being addressed by the IRDS Export/Import facilities and the work of the EDIF/CASE group, but the techniques will need to be improved to facilitate the use of multiple versions of multiple dictionaries stored in a variety of hardware and software environments. Existing CASE tools provide only a limited level of version control, if any, for distributed data dictionaries.

Lack of support for different computer systems: Most organisations will need to run applications on different computer hardware and/or in different software environments. Even if an organisation currently has a single hardware supplier, it should consider 'open' I-CASE tools, so that the choice of tools does not preclude the use of other hardware in the future. None of the promised I-CASE environments will provide complete 'openness'. Even some of the 'open' tools available today provide only a very limited choice of hardware with which they can be used.

Lack of support for group working: Most CASE tools available today provide very good support for individual developers. The real benefits of CASE and I-CASE, however, will arise from large projects involving teams of developers. The tools must therefore provide good support for groups of people, in particular to facilitate intercommunication between members of a development team.

I-CASE will require a high level of commitment

Adopting and implementing an I-CASE development environment will have a significant impact on the development department because the I-CASE tools provide automated support for,

and thus have an impact on, the whole applications life cycle. Adopting an I-CASE development environment will probably result in changes not only to the set of tools used, but also to the level of management required, to the organisational structure of the development department, and to the development approach, methods, and standards used within the development department, as these are all inextricably linked. A decision to adopt an I-CASE development environment should therefore not be made without recognising the high level of commitment that will be required to make these changes.

Most experts expect that an I-CASE development environment will make it easier to develop core corporate and departmental applications because such applications have large areas of commonality. The process of managing the whole development environment will, however, become more complicated because it will still be necessary to support, manage, and control the tools and applications that cannot be included in the I-CASE environment. In general, it will be necessary to establish different procedures for the I-CASE and non I-CASE environments.

One of the major differences of an I-CASE development environment will be in the life cycle of an application. Figure 2.8 shows the difference between the typical life cycle used today and that required by an I-CASE development environment. The traditional life cycle is supported by various methods and tools. However, the information that is passed from one stage to the next is typically in the form of a document that has to be manually loaded into any tools that support later stages. Also, with the traditional life cycle, small enhancements and the maintenance of applications tend to be treated as different activities, not necessarily following all the stages of the life cycle.

The changes in the life cycle that will be brought about by an I-CASE development environment are significant. The data dictionary will act as the medium for passing information from one tool or stage to the next, and maintenance and small enhancements will have to follow the same life-cycle stages as the original design and build. This will have the effect of changing the traditional development life cycle from the 'waterfall' model (shown in the top half of Figure 2.8) to the cyclic process shown in the bottom half of the figure.

The change in the life cycle required by I-CASE will have an impact both on the development approach and on the methods used. We have already seen how the methods suppliers are enhancing their existing products so they can provide full coverage of the applications life cycle, and we can expect this process to continue as I-CASE development environments become established.

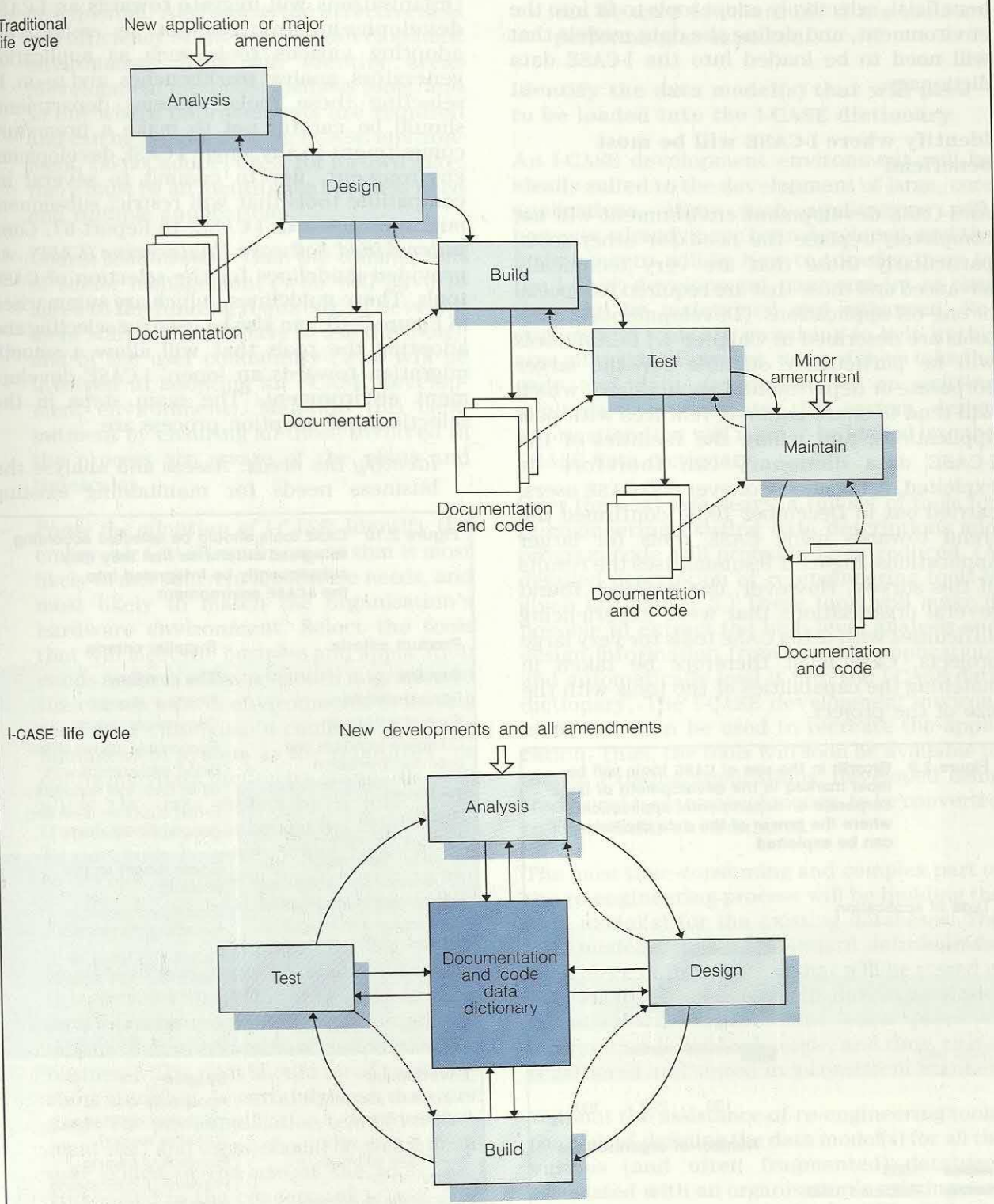
The changes to the life cycle brought about by I-CASE will also result in better control of the maintenance process, which in turn, will result in the development of more manageable applications. With the traditional life cycle, development departments may have good intentions to keep the documentation of the various applications up to date. However, other pressures usually result in changes made to program code not being reflected in the analysis and design documentation. Moreover, the changes are often not fully tested. This not only makes the application more difficult to maintain in the future, because the documentation is out of date, but it can create a need for further maintenance, because the broader implications of the change — those that can be determined only by assessing the original analysis and design documentation — can be missed.

In the I-CASE life cycle, all work, whether it is new development, enhancement, or maintenance, will follow the analysis, design, build, and test cycle, and will be enforced by the development approach. The documentation will be updated automatically at each stage, resulting in more manageable applications.

It would be wise to migrate slowly towards I-CASE

We have seen that while an I-CASE development environment promises considerable benefits, the costs, risks, and level of commitment required are also very significant, because I-CASE is still in the very early stages of its development. Since no complete I-CASE development environment will be available until 1992 at the earliest, we recommend that organisations begin to migrate towards I-CASE, taking care not to create barriers to its subsequent adoption, but not yet making a full

Figure 2.8 The development life cycle in an I-CASE environment differs markedly from the life cycle that is common today

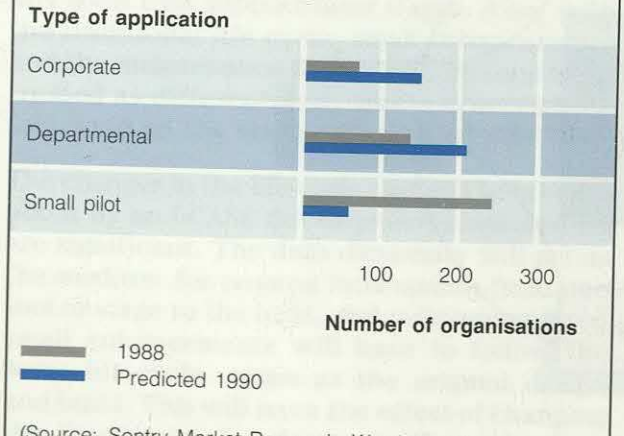


commitment to it. In migrating towards an I-CASE development environment, organisations should identify where I-CASE will be most beneficial, selectively adopt tools to fit into the environment, and define the data models that will need to be loaded into the I-CASE data dictionary.

Identify where I-CASE will be most beneficial

An I-CASE development environment will not completely replace the need for other tools, particularly those that are very technically advanced and those that are required for special or one-off applications. (Developments in such tools are described in Chapter 4.) I-CASE tools will be particularly suitable for the larger corporate or departmental applications, which will tend to share common features with past applications, and where the facilities of the I-CASE data dictionary can therefore be exploited. A US survey of over 650 CASE users, carried out in December 1988, confirmed the trend towards using CASE tools for larger applications. Figure 2.9 summarises the results of this survey. However, our research found several organisations that were experiencing difficulties with using CASE tools for very large projects. Care must therefore be taken in matching the capabilities of the tools with the size of the project.

Figure 2.9 **Growth in the use of CASE tools will be most marked in the development of large corporate or departmental applications, where the power of the data dictionary can be exploited**



(Source: Sentry Market Research, Westborough, Massachusetts)

Selectively adopt tools to fit into the I-CASE environment

Organisations will migrate towards an I-CASE development environment by selectively adopting various tools such as application generators, analyst workbenches, and so on. In selecting these tools, systems departments should be careful not to make a premature commitment to any single I-CASE development environment, nor to commit to several incompatible tools that will restrict subsequent migration towards I-CASE. In Report 67, *Computer-Aided Software Engineering (CASE)*, we provided guidelines for the selection of CASE tools. These guidelines, which are summarised in Figure 2.10, can also be used for selecting and adopting the tools that will allow a smooth migration towards an 'open' I-CASE development environment. The main steps in the selection and adoption process are:

— *Identify the needs*: Assess and analyse the business needs for maintaining existing

Figure 2.10 **CASE tools should be selected according to agreed criteria so that they may subsequently be integrated into the I-CASE environment**

Product criteria	Supplier criteria
General Proven reliability Ease of installation Complete technical and user documentation	**The company** Financial strength Commercial stability Reasonable market share Good relationships with other CASE-tool suppliers Broad customer base and geographic coverage
Technical Fast response Appropriate method — single or multimethod Consistency and integration within stages of the life cycle Consistency and integration between stages of the life cycle Graphics support for some types of techniques Simple-to-use graphics facilities	**Commitment to the product** Established record of marketing the product Appropriate levels of expenditure on research and development Existence of specific plans to develop the product
Environment Support of acceptable hardware bases Ability to work within acceptable software environments Appropriate multi-user support Ability to interface with other environments	**Support** Acceptable level of manpower devoted to customer support Provision of training Provision of customising support Good response to problems and queries

applications and developing new applications. This will identify critical business areas where an I-CASE environment could be exploited. Also assess the effectiveness and efficiency of the existing development environment. This will identify areas (development methods, analysis tools, and so on) where improvements are required and can be provided by an I-CASE environment. Do not adopt I-CASE for its own sake; there must be an identifiable business need and suitable applications.

- *Obtain commitment*: Gain the commitment of senior management (who will have to agree to the funding required), of development staff (who will have to use the tools), and of the user community (who will be involved in adopting an I-CASE development environment). Maintain this commitment by ensuring all those involved in the process are aware of the plans and timescales.

- *Phase the adoption of I-CASE*: Identify the embryonic I-CASE environment that is most likely to meet current and future needs, and most likely to match the organisation's hardware environment. Select the tools that will meet the business and application needs and will allow a smooth migration to the chosen I-CASE environment. Selecting tools that either use a common database management system as the repository, or conform to data dictionary standards, will allow the data stored by a tool to be transferred to, or accessed by, other tools. To start with, however, it may be necessary to develop in-house software to overcome a lack of compatibility between the different tools.

- *Implement and assess the tools*: Tools should first be used on a pilot project to develop an important and realistic application (but not one that is critical to the success of the business). The pilot should be of relatively short duration — certainly less than one year. The pilot-application team (development staff and users) should be well trained and skilled in the use of the tools, and committed to the concepts of I-CASE. The results of the pilot should be assessed so that ways of improving the use of the tools can be identified, and the future use of the tools be encouraged. When implementing the tools, it is also important to assess them carefully by identifying the measures that can be used to determine if the tools are performing as expected.

Identify the data model(s) that will need to be loaded into the I-CASE dictionary

An I-CASE development environment will be ideally suited to the development of large, core applications. Many such applications will, however, already have been developed, and the main concern will be how to transfer them to the I-CASE development environment so that they can be maintained and enhanced. Re-engineering tools are promising to help in this area although, at present, none of them take the code and data descriptions of an existing application and create the system design information that will need to be loaded into the I-CASE data dictionary.

Over the next two years, a full set of tools for re-engineering existing data descriptions and program code will probably be introduced. (A detailed description of re-engineering tools is given in Chapter 3.) These tools will make it possible to extract the high-level analysis and design information from existing applications and automatically load it into the I-CASE data dictionary. The I-CASE development environment can then be used to recreate the application. Thus, the tools will soon be available to enable existing applications, developed using traditional tools and techniques, to be converted to the I-CASE environment.

The most time-consuming and complex part of the re-engineering process will be building the data model(s) for the existing databases. The data model(s) give the standard definition for each piece of information that will be stored in the data dictionary. Correctly defining the data model(s) will ensure that each piece of information is held only once, and thus, that it is gathered and stored in a consistent manner.

Without the assistance of re-engineering tools, the task of defining the data model(s) for all the various (and often fragmented) databases associated with an organisation's existing core applications will be exceedingly difficult. In the meantime, however, organisations should begin to identify the applications that will need to be

transferred to the I-CASE development environment, should start to define the high-level data model(s) that will need to be stored in the data dictionary, and should work to achieve some conformity between the various databases. The task of defining the high-level data model(s) is very time-consuming, however. By starting now, organisations will develop a good understanding of the various data model(s), and once the I-CASE development environment has been fully implemented, the effort of loading the data model(s) into the I-CASE dictionary will be minimised.

Organisations should prepare for the introduction of I-CASE now by following the advice given in this chapter. Nevertheless, they should recognise that I-CASE will not, by itself, provide all the benefits and increases in productivity required to satisfy business needs in the next two to three years. In the meantime, most organisations will therefore need to continue to exploit their existing tools, make effective use of the more advanced tools now becoming available, and encourage and make effective use of end-user computing. These areas are covered in Chapters 3, 4, and 5 respectively.

Chapter 3

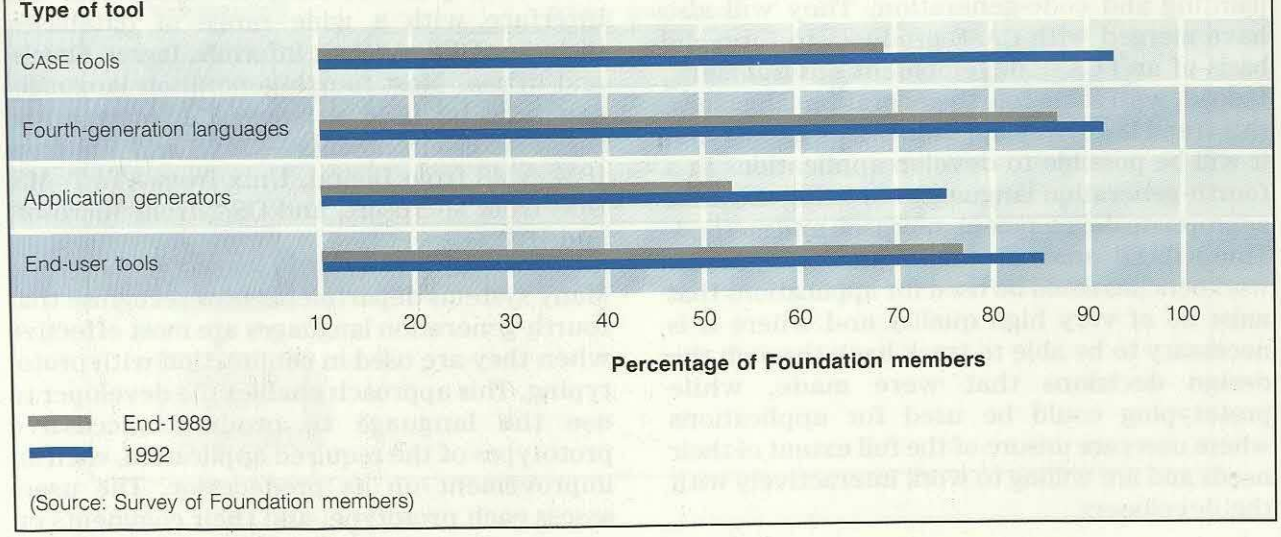
Continue to exploit existing tools

We demonstrated in the previous chapter that the adoption and use of CASE tools is part of the migration path towards an I-CASE development environment. A survey of Foundation members at the end of 1989 revealed that just under 70 per cent are currently using CASE tools, and over the next three years, over 90 per cent expect to be using them (see Figure 3.1). This is an indication of the increasing level of confidence that members have in the ability of CASE products to provide significant benefits.

CASE tools will not, however, provide the productivity improvements that are required in the short term to deal with the pressures facing development departments. First, most CASE tools are specifically aimed at the development of new applications, while most of the current workload is for the maintenance of existing applications. Second, the CASE tools currently available are simply not as productive as some fourth-generation languages. (Recent analysis of the details of nearly 350 projects submitted by members of the Butler Cox Productivity Enhancement Programme revealed that projects on which CASE tools such as analyst workbenches and report and enquiry generators were used had a lower productivity rating — that is, a lower rate of code production — than projects developed without CASE tools.)

This places development departments in a very awkward situation. The tools that they need to migrate towards a better development environment in the medium to long term are unable to provide the level of productivity that they need to cope with short- to medium-term development pressures. To overcome this problem, development departments will need to continue to exploit existing, well proven, fourth-generation languages in developing new applications, start to use the newer re-engineering tools to

Figure 3.1 Over 90 per cent of Foundation members expect to be using CASE and fourth-generation-language tools by 1992



(Source: Survey of Foundation members)

maintain and enhance their existing applications, and assess the potential of application packages as a means of reducing the development workload.

Continue to use well proven fourth-generation languages

For several years, there was uncertainty about whether fourth-generation languages had a long-term future. This uncertainty arose because of the problems associated with early fourth-generation languages — such as their lack of flexibility and the operational inefficiency of the applications developed with them. There was also a fear that emerging CASE tools and application generators would supersede fourth-generation languages. Now, however, the early problems have largely been overcome. Fourth-generation languages are well established in most development departments. As Figure 3.1 shows, over 85 per cent of Foundation members are currently using fourth-generation languages, and this is expected to increase to over 90 per cent in the next three years. Fourth-generation languages have also proved to be highly productive in terms of the rate at which functionality is delivered — often providing three times more function points per man-month than third-generation languages, and requiring almost 50 per cent less time and effort than equivalent Cobol developments.

By 1993/94, many of today's fourth-generation languages will have evolved from coding-based tools to automated tools that support screen-painting and code-generation. They will also have merged with CASE products to form the basis of an I-CASE development environment. Indeed, with some fourth-generation languages, this trend is already apparent. This means that it will be possible to develop applications in a fourth-generation language, but using the most appropriate development approach and method. Thus, CASE analysis techniques and design workbenches could be used for applications that must be of very high quality and where it is necessary to be able to track back through the design decisions that were made, while prototyping could be used for applications where users are unsure of the full extent of their needs and are willing to work interactively with the developers.

Many fourth-generation languages are now available as part of an integrated product set that includes a variety of CASE tools and application generators. QuickBuild from ICL in the United Kingdom, and Oracle from Oracle Corporation in the United States, are two examples. This trend will continue as tools become available to automate more development activities and more stages of the applications life cycle.

Like other tools, fourth-generation languages have their strengths and weaknesses. Both must be understood so that an appropriate choice of tool can be made for any particular development project. Many fourth-generation languages are not, for example, appropriate for developing very complex transaction-processing applications. There is also a lack of standards, which means that applications developed with a particular fourth-generation language are usually tied to that language and its supplier, because the code has a unique syntax. One exception to this is dBase from Ashton-Tate Corporation; other suppliers provide tools that compile or interpret dBase source code.

On the positive side, however, fourth-generation languages provide facilities not commonly available with other tools. Several are very open products, in that applications developed in them can run on a range of hardware and can work with a range of database management systems. FOCUS, from Information Builders Inc, is a good example. Applications written in FOCUS can run on a broad range of hardware architectures (including IBM, Digital, Hewlett-Packard, and various implementations of Unix) and can interface with a wide range of databases, including DB2, Adabas, Informix, Ingres, Oracle, and dBase. Most fourth-generation languages can now be used in several hardware and software environments — MVS and VM from IBM, VMS from Digital, Unix from AT&T, MS-DOS from Microsoft, and OS/2 from Microsoft and IBM.

Many systems departments now recognise that fourth-generation languages are most effective when they are used in conjunction with prototyping. This approach enables the developer to use the language to produce successive prototypes of the required application, each an improvement on its predecessor. The users assess each prototype, and their comments on

how it needs to be changed to meet their requirements are included in the next prototype. This process ensures that the delivered application closely matches the users' requirements. Most development departments have produced their own in-house prototyping methodologies for use with fourth-generation languages; all are based on similar principles. One of the most successful is described in Figure 3.2.

Some organisations are using fourth-generation languages in conjunction with various CASE tools. CASE tools can be used to carry out the business analysis and produce the business specifications that are an essential starting point for the first prototype. Fourth-generation languages and the prototyping methodology can then be used to develop the application.

To meet the increasing need for new applications in the short to medium term, therefore, the development department should continue to use fourth-generation languages whenever possible, and should use proven CASE tools either in conjunction with the fourth-generation

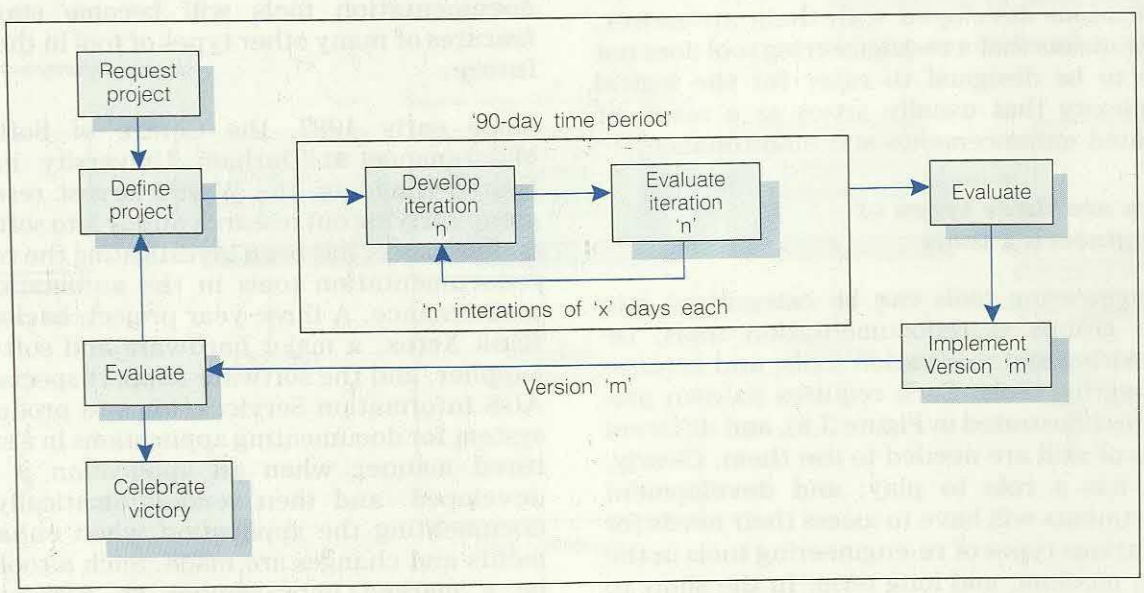
Figure 3.2 Prototyping with fourth-generation languages can bring great gains in productivity and quality

Information Engineering Associates

Information Engineering Associates (IEA) is a subsidiary of DuPont, a major chemical company based in the United States. IEA uses Application Factory, a fourth-generation language from Cortex, with its own in-house prototyping methodology, to develop applications for DuPont's Textile Fibres Division. It has been so successful that it now promotes its services outside DuPont.

Scott Schultz, head of IEA, believed that the traditional development life cycle had many shortcomings. In an attempt to overcome them, he and his team adopted Application Factory and developed a method known as Rapid Iterative Production Prototyping (RIPP). At the heart of RIPP is a classic, iterative prototyping method, which is restricted to a 90-day time period. (The RIPP process is illustrated in the diagram below.) Before the start of the 90-day period, a high-level specification is defined in business terms and becomes the basis for the first prototype. Once the prototyping period has started, the application will go through several iterations, with the users collaborating with the systems developers. Each iteration moves the application closer to the users' requirements. At the end of the 90-day period, the application is delivered to the user. If the application proves too big for one time period, it is broken down into smaller applications, each of which is allocated its own 90-day time period, and the whole process is repeated. When the project is complete, the developers and users get together to celebrate their joint achievement.

In 1988, DuPont estimated that, for 15 applications, it had saved over $2.3 million by using RIPP, and delivered applications of higher quality. Scott Schultz believes that the greatest advantages of the RIPP approach is that it brings the IT and business functions closer together.



languages, or to carry out development activities not supported by them.

Use re-engineering tools to help manage old applications

As in many areas of business computing, the field of re-engineering has been subject to over-eager claims by suppliers, and is described in confusing new terminology. Re-engineering tools will, nevertheless, provide many development departments with a means of making great savings in two areas. First, they can be used to reduce the effort required to maintain the existing applications portfolio, which on average, consumes at least 60 per cent of the development department's effort. Second, they can be used to prolong the usefulness of some of the older applications, in which most organisations have invested many thousands of man-days.

Before the end of 1991, tools for re-engineering existing Cobol applications will be available, providing a real opportunity for many organisations to break free of the legacy of applications originally developed many years ago. By 1992 or 1993, tools for re-engineering applications written in the more common fourth-generation languages will also be available. In fact, it will be easier to develop these re-engineering tools because fourth-generation languages generally have fewer syntactical constructs than Cobol, and the applications developed with them are newer, which means that a re-engineering tool does not have to be designed to cater for the logical complexity that usually arises as a result of repeated enhancements and maintenance.

There are three types of re-engineering tools

Re-engineering tools can be categorised into three groups — redocumentation tools, restructuring and renovation tools, and inverse-engineering tools. Each requires its own procedures (illustrated in Figure 3.3), and different levels of skill are needed to use them. Clearly, each has a role to play, and development departments will have to assess their needs for the various types of re-engineering tools in the short, medium, and long term. In the short to medium term, they can be used to simplify the maintenance task, to move to a new technology such as a relational database, to provide links that can integrate existing applications or databases, or to facilitate the use of more advanced architectures such as powerful workstations that use windowing techniques and are based on a client-server configuration. In the longer term, their importance will increase as their role in transferring the existing applications portfolio to an I-CASE development environment is recognised.

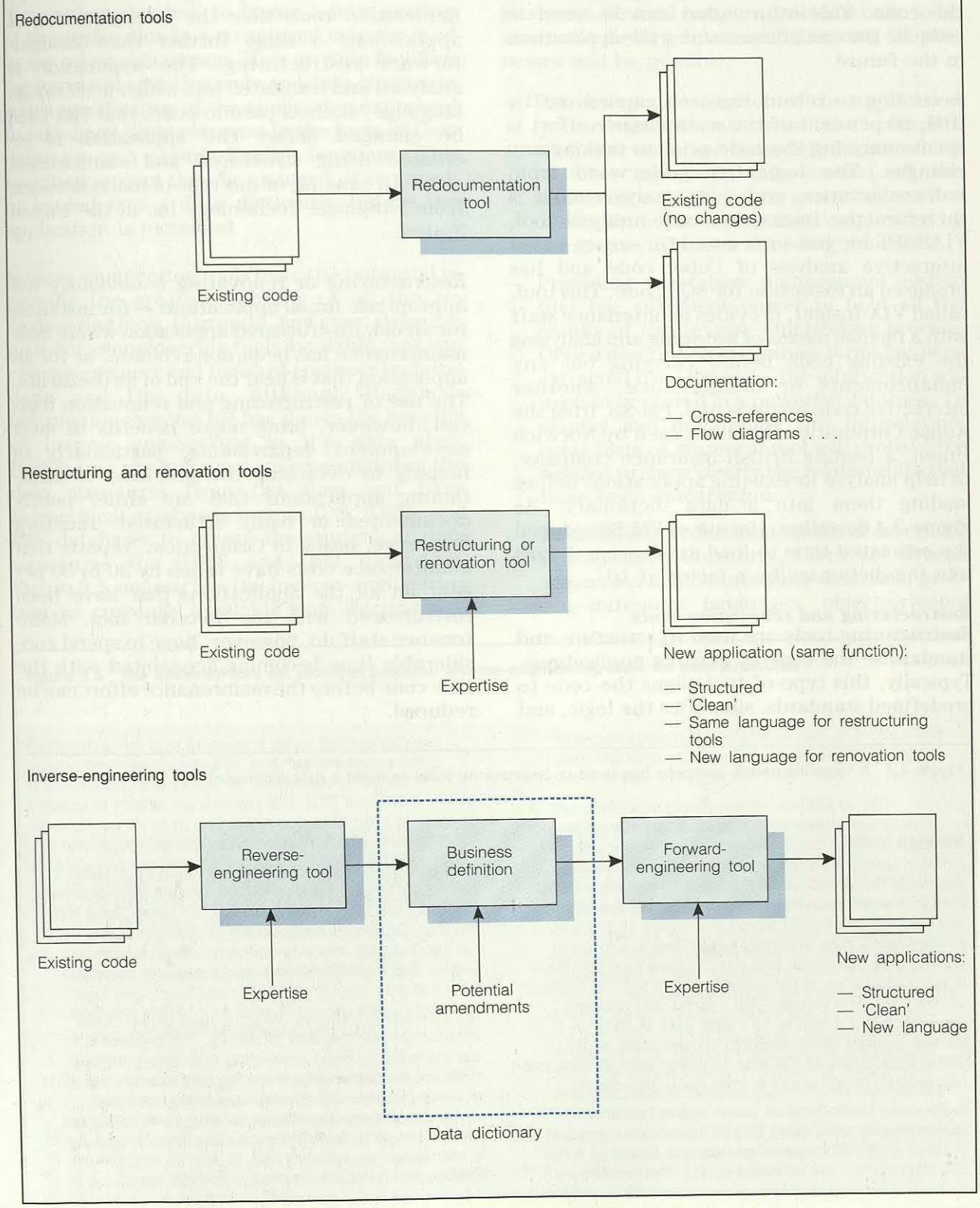
Redocumentation tools

Redocumentation tools provide a means of automating the various tasks involved in documenting an application. Typically, they produce process flow charts and cross-reference listings directly from the source code and data definitions. Examples are the 4DxRef tool that produces documentation of applications developed in Fourth Dimension, from Analyses Conseils Informations (ACI) in France, and Abstract, a tool from Advanced System Concepts in the United States that documents RGP III code. The outputs from such tools enable developers to assess the likely impact of a change and to estimate the effort required with greater accuracy.

Redocumentation tools have been available for some time. Early versions of these tools tended to produce large quantities of printed output, which the developer had to go through manually. More recent tools provide interactive screen-based access to the documentation. Redocumentation tools will become standard features of many other types of tool in the near future.

Since early 1987, the Centre of Software Maintenance at Durham University in the United Kingdom, the West's largest research group carrying out research purely into software maintenance, has been investigating the role of redocumentation tools in the automation of maintenance. A three-year project, backed by Rank Xerox, a major hardware and software supplier, and the software-support specialists, AGS Information Services, aims to produce a system for documenting applications in a structured manner when an application is first developed, and then semi-automatically redocumenting the application when enhancements and changes are made. Such a tool will be a marked improvement on current redocumentation tools.

Figure 3.3 The different types of re-engineering tools require different procedures, different levels of support, and different technical knowledge



Code-analysis tools also fall into the re-documentation category. These tools provide information on how well structured the code is, and assess the maintainability and testability of the code. This information can be used to estimate the cost of maintaining the applications in the future.

According to recent research carried out by IBM, 50 per cent of the maintenance effort is spent analysing the code prior to making any changes. The logical step forward from redocumentation and code-analysis tools is therefore the interactive code-analysis tool. VIASOFT Inc has such a tool for screen-based interactive analysis of Cobol code and has promised an extension for SQL code. This tool, called VIA/Insight, provides maintenance staff with a flexible means of accessing and analysing the existing code before carrying out any enhancements or maintenance. Another interactive code-analysis tool, PM/SS, from the Adpac Corporation, has been used by Norwich Union, a leading British insurance company, to help analyse its existing applications before loading them into a data dictionary. As Figure 3.4 describes, the use of PM/SS reduced the estimated time to load application details into the dictionary by a factor of 10.

Restructuring and renovation tools

Restructuring tools are used to structure and standardise the code of existing applications. Typically, this type of tool aligns the code to predefined standards, simplifies the logic, and removes any redundant code. Examples of restructuring tools are Retrofit and DataTec, from Peat Marwick Main and Co.

Renovation tools take the re-engineering of applications a stage further than straightforward restructuring. The application is analysed and translated into a high-level design language, such as pseudo code, that can then be changed before the application is re-created using structured and standardised code. An example of this type of tool is Recoder, from Language Technology Inc in the United States.

Restructuring or renovating is obviously not appropriate for all applications — for instance, for an old, unstructured application where little maintenance has been done to date, or for an application that is near the end of its useful life. The use of restructuring and renovation tools can, however, bring major benefits to most development departments, particularly in helping to overcome the problems of maintaining applications that are either poorly documented or badly structured. Hartford Insurance, based in Connecticut, reports that maintenance costs have fallen by 20 to 50 per cent on all the applications that have been restructured with the Recoder tool. Maintenance staff do, however, have to spend considerable time becoming acquainted with the new code before the maintenance effort can be reduced.

Figure 3.4 A large insurance company has used re-engineering tools to build a data dictionary

Norwich Union

Norwich Union is a large insurance company, ranked second in the UK life-funds market. The company found itself in a situation common to many that have a lot of incompatible information stored in a variety of systems. It realised that a data dictionary would enable it to standardise data definitions, and thus, to save time and effort in the maintenance of old systems and in the development of new ones. The problem that it faced was daunting, however; it had a total of 18 million lines of code to transfer — eight million lines of in-house code, some dating back to the 1970s, and 10 million lines of code that had recently been bought in from the United States.

In February 1988, Norwich Union began two pilot dictionary exercises, using PM/SS from Adpac. The first involved about 30 in-house applications, including some 200 data items. "We achieved a 10:1 improvement — it took only 10 per cent of the time it would have taken previously to load the applications into the data dictionary. We used this as our cost justification", stated Steve Kirby, the then Dictionary Administrator at Norwich Union. On the second pilot, the team took one month to complete a project that they estimated would previously have taken 14 months.

Norwich Union still uses PM/SS to load details of old applications into the data dictionary. This process is carried out only when an existing application requires sufficient maintenance or enhancement to justify the effort of loading details of the whole application into the dictionary. All new developments have to be consistent with the existing data dictionary. Currently, Norwich Union is assessing the use of its data dictionary, and the use of PM/SS, as an impact- or change-analysis tool.

Inverse-engineering tools

Inverse-engineering tools start by reverse-engineering an application and its associated databases to a stage where they can then be 'forward-engineered' to create a new version of the application in a structured manner, and in any selected language. They provide facilities for translating existing code and data structures into a specification of the application expressed in high-level business terms in the form of flow diagrams or a pseudo-English language. This specification can then be changed, if required, or loaded into a data dictionary before the application is recreated.

Inverse-engineering tools have the potential to provide the greatest benefits of all the re-engineering tools. Currently, however, there are no inverse-engineering tools available that can reverse-engineer and forward-engineer both the code and the data, although Bachman Information Systems Inc has shown the benefits of inverse engineering in the data area. Bachman's products can inverse-engineer the data structures from flat files and from hierarchical databases such as IDS, IDMS, and IMS databases to create the equivalent data structures for IDMS and DB2 databases. Figure 3.5 summarises the inverse-engineering tool set currently available from Bachman.

Bachman, which clearly leads in this field, admits that the development of a tool to carry out the inverse-engineering of data is easier than developing one to inverse-engineer processes. Several complex problems have to be overcome before inverse-engineering of processes will be possible:

- Extracting a description of the application in business terms from the complex computer code and data definitions, which in most cases, do not contain all the information required to construct such a description. The task is also complicated by the great variations in coding practices.
- Providing a database management system that has sufficient power to hold all the results of the reverse-engineering process. Once a description in business terms has been extracted from the code, the information will need to be stored in a powerful database. To display and manipulate this information, some form of graphical tool will probably be needed to show clearly the relationships that have been constructed.
- Establishing a consistent means of describing applications. Currently, there are several means available (data-flow diagrams, problem-statement languages, object-oriented

Figure 3.5 The Bachman tool set provides powerful inverse-engineering tools for data structures

Bachman's aim is to produce a set of tools capable of inverse-engineering (reverse- and forward-engineering) both data and processes. Its current set of tools is capable of inverse engineering IMS, IDS, and IDMS schemas and/or data definitions embedded in Cobol code into fully optimised DB2 statements. The total cost of the software and hardware is around £30,000 ($50,000). The tool set consists of four basic components — the capture facility, the data analyst (DA), the database administrator (DBA), and the expert advisor:

- The capture facility provides guidance on loading the Bachman Repository with details of IMS, IDMS, and Cobol files. It applies rules to decide how the definitions should be stored in the repository and builds a complete conceptual data model.
- The DA is an entity-relationship diagramming tool that is used to: tidy up the data models produced by the capture facility; enhance existing DA models; normalise DA models reverse-engineered from the DBA; and develop an entity-relationship model either from scratch or from models transferred from other tools such as IEW, Excelerator, and so on. It also uses the expert system in an iterative way to prompt and coach the data modeller as he develops the logical data models.
- The DBA creates fully optimised DB2 or IMS definitions either from scratch or from the logical models produced by the DA. It also reverse-engineers DB2 and IMS definitions to a logical definition acceptable to the DA tool. Again, the system prompts the designer, using expert-system techniques.
- The expert advisor provides advice to users of the other three components of the tool set. The knowledge of three 'gurus' — Charlie Bachman, Chris Gane (pioneer of data-flow diagramming), and Chris Loosely (designer of the DB2 access mechanism) is embedded in the expert advisor. There are four levels of interaction. At the highest level, the expert advisor provides the novice with very precise and detailed guidance, which enables him to learn quickly. At the lowest level, it automatically takes many of the simpler decisions, so that an expert can use the tool without being bombarded with constant advice.

representations), but each has its limitations. Standards on how to manipulate such information on a screen will need to be created.

We expect that, by the beginning of 1992, inverse-engineering tools will be available to process both the processes and the data. These tools will need the assistance of an expert but will be very effective for loading details of existing applications written in third- and fourth-generation languages into an I-CASE data dictionary. The leaders in developing products in this field are Bachman and Language Technology Inc.

Over the next few years, inverse-engineering tools will evolve to the stage where they can analyse about 80 per cent of the existing code; the remaining 20 per cent will require human intervention. These tools will probably use some form of knowledge base — a database that contains information and rules — to help extract a description of the application in business terms. When a new problem is encountered, the user will add new rules to the knowledge base, enabling the tool to resolve a similar problem automatically when it occurs again.

Re-engineering tools will change the applications life cycle

In the future, re-engineering tools will change the way in which applications are developed and maintained. The life cycle of the I-CASE development environment, described in Chapter 2, will be extended to include re-engineering, as shown in Figure 3.6. Many of the existing core applications will be transferred to the I-CASE environment, with the aid of re-engineering tools, by analysing the applications, restructuring them, and reverse-engineering them. This process is described in more detail below.

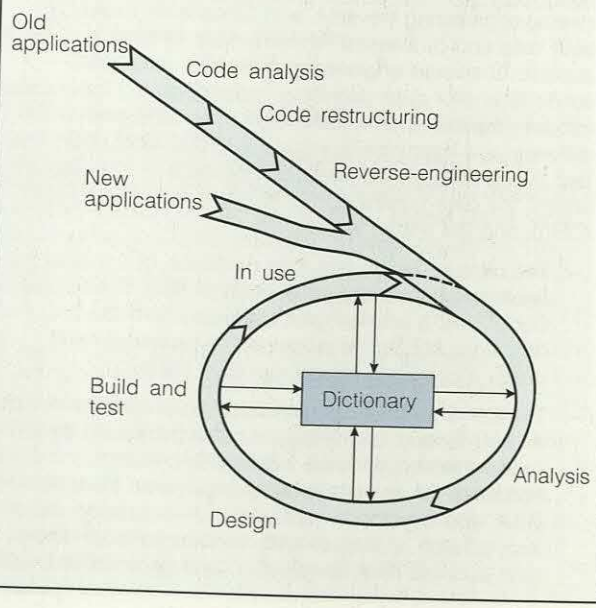
Before any decision is made to load the details of an existing application into the I-CASE data dictionary, code-analysis tools should be used to determine the condition of the application. This analysis will serve as a basis for estimating the effort required to load the application into the dictionary, and for assessing whether it is worthwhile.

Once the size of the task is known, and the coding and data-naming conventions have been defined and standardised, restructuring and renovating tools can be used to structure the code to ensure that it is easier to understand. Standardising the data, however, may be more difficult and very time-consuming as it will probably need to conform with existing data in the data dictionary or other applications. Report 64, *Managing the Evolution of Corporate Databases*, discusses the issues surrounding database integration.

Reverse-engineering tools can then be used to extract a description (in business terms) of the application from the code and data structures. This description will be transferred to the I-CASE development environment to forward-engineer the application.

Clearly, this will not be an easy process and will require considerable human intervention. However, it will permit existing applications to be transferred to the I-CASE development environment in a semi-automated and cost-effective manner, and will reduce the high proportion of development effort spent on maintenance.

Figure 3.6 The life cycle of the I-CASE development environment will be enlarged to include re-engineering, so that existing applications can be transferred to the I-CASE environment



Evaluate the potential of application packages

In the past, a considerable amount of effort was required to modify packages so that they could interface with other applications and match users' needs. The total cost of the package and the modifications often made it more cost-effective to develop the equivalent bespoke application. This is no longer always the case, because software packages are now available as integrated sets of modules rather than as monolithic packages, and they are becoming extremely flexible. In Report 69, *Software Strategy*, we gave several examples of 'soft' packages that can be modified to meet users' needs. Such packages are typically sold with a set of tools to help with modification and implementation.

In Report 69, we also showed that if a suitable package is available, it is usually a more cost-effective solution, and therefore a better investment, than a bespoke development. These three trends in the package market — the growing flexibility of packages, the increased availability of sets of integrated packages, and their growing cost-effectiveness — mean that the development department should always evaluate packages as a means of meeting a particular application requirement.

Soft packages will provide flexible solutions

The inflexible packages of the 1970s are now being superseded by the flexible 'soft' packages of the 1990s. Today, suppliers such as SAP in Germany and Computer Associates in the United States provide packages that can be tailored to match individual requirements. Their only disadvantage is that they require development staff to learn how to use the tool set that will be used to modify a package for a particular application. Some package suppliers are now addressing this problem by supplying a combination of package and tool set that permits the development of bespoke applications as well as tailoring of the package.

As the different I-CASE development environments become established, package suppliers will begin to supply the basic building blocks that can be added to an I-CASE environment, so that the package can be customised in-house to create a bespoke application. We expect that, by the late 1990s, packages will be supplied as a set of related 'objects' that can be loaded directly into an organisation's data dictionary and be used to develop an application. (Object-oriented concepts are described in Chapter 4.)

These developments will mean that, as I-CASE is established in the mid- to late-1990s, the distinction between bespoke development and package solutions will start to become blurred. Over the last 20 years, an increasing amount of the functionality on bespoke developments has been constructed from basic building blocks that form part of the software infrastructure. (It is no longer necessary, for example, to develop database-access routines on an application-by-application basis.) The trend in packages is the reverse — they are now supplied in ways that allow them to be broken down into smaller and smaller building blocks. In the 1990s, development departments are therefore likely to be occupying the middle ground, assembling well integrated medium-sized building blocks both from the software infrastructure and from packages to form an application that is a good fit with users' needs. This trend for packages and bespoke developments to merge is illustrated in Figure 3.7, overleaf.

Integrated packages will serve both industry-specific and common business areas

Several major hardware and software suppliers (such as IBM and Oracle) are now offering integrated industry-specific packages. This trend is likely to continue into the 1990s. Late in 1989, Oracle Corporation announced that it was expanding into the package market, with a range of products from word processing to manufacturing packages. Figure 3.8, on page 37, summarises the features of Oracle's core suite of manufacturing packages. IBM has a set of more than 50 software products aimed at integrating the plant-floor, design-operations, and production-planning areas of manufacturing.

These integrated packages will provide interfaces to the most commonly used database management systems, making integration with existing applications much simpler. Most suppliers will provide packages that cover

Figure 3.7 Packages and bespoke developments for common applications will merge in the 1990s

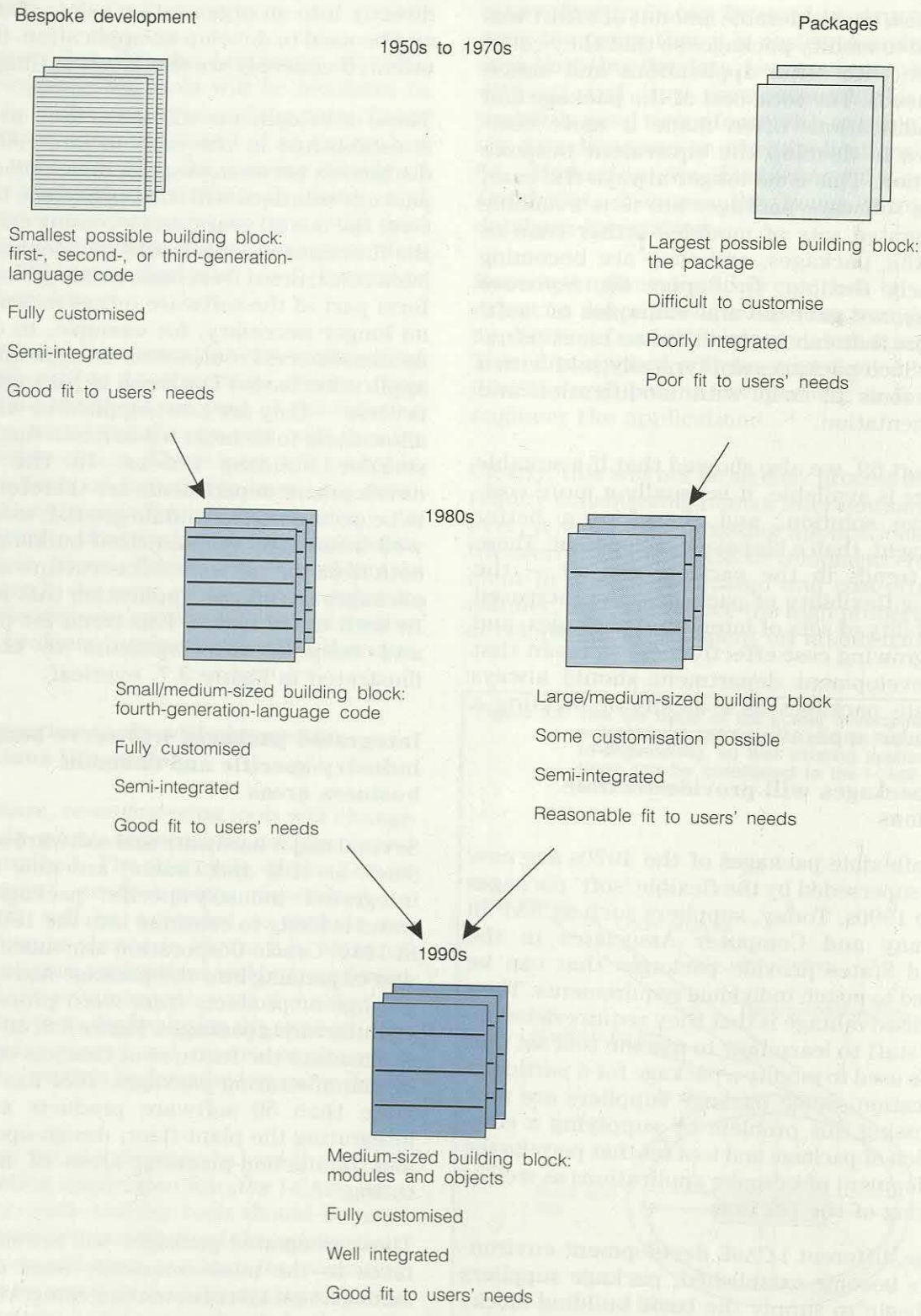


Figure 3.8 Integrated suites of packages will replace large areas of common applications

Oracle Inc is one of the world's largest software houses, with revenues of $584 million in 1988. Its main product is the Oracle database management system and its associated tool set. It recently announced an expansion to its product range — the Core Manufacturing suite of packages.

Core Manufacturing provides a set of full-featured, portable, decentralised products designed to support distributed manufacturing. This product will run on all the hardware and software environments supported by Oracle and is designed to permit very easy access to the packages via a Macintosh-style set of menus, pop-up windows, graphics, online help, and other facilities, all of which reduce the number of keystrokes required. There are five basic packages in the integrated suite:

- Bill of materials, which provides engineers with a tool to configure products quickly and accurately.
- Work in progress, which enables production to be scheduled for maximum throughput and to facilitate just-in-time manufacturing techniques.
- Master scheduling, which tightly links customers' delivery requirements to production schedules.
- Manufacturing resource planning, which improves control over production cycles and minimises inventory levels.
- Order entry, which gives sales departments immediate access to accurate price data and delivery commitments.

This product can also be linked with Oracle's financial/personnel set of packages to provide a tightly integrated manufacturing, accounting, and personnel system.

common business areas, such as accounting, personnel, and so on, as well as industry-specific areas, like manufacturing, construction, and distribution.

Even with I-CASE, package solutions will remain cost-effective

Package suppliers will tailor their products so they can be used within an I-CASE development environment. This means that it will be possible to modify a package within an I-CASE environment and integrate it with existing applications. A package solution will therefore be a better investment for most applications, even as organisations migrate towards an I-CASE development environment.

By continuing to exploit existing tools, most organisations will be able to meet the requirements of users, and migrate towards an I-CASE development environment. Over the next five years, however, there will be further advances, both in current tools and in new forms of tools that are outside the I-CASE environment. Exploiting these more advanced tools will place systems departments in an even better position to meet the growing demands being placed upon them. The emerging, advanced tools that Foundation members should now be assessing are the subject of the next chapter.

Chapter 4

Use emerging tools to develop more advanced applications

So far in this report, we have seen how advances in tools will produce an integrated development environment (I-CASE) that is appropriate for large corporate or departmental applications. We have also seen how existing tools can be exploited to support systems development until I-CASE matures. I-CASE is not, however, the only advance that will be made in development tools during the first half of the 1990s.

Other advances will make it possible to develop more sophisticated applications than those that are feasible today. Initially, these more advanced tools will be available only outside the I-CASE development environment. When they become widely adopted and proven, I-CASE standards will be expanded to accommodate the new tools. Most organisations will, however, need to adopt these new types of tools over the next few years; they cannot afford to wait until they become integrated into I-CASE.

Several emerging tool technologies are promising to bring benefits to the development environment in the next five years. They are listed in Figure 4.1. Object-oriented and rule-based tools, in particular, promise a wide range of major benefits, and in several other areas, advances are producing new tools or adding new features to existing ones:

- Object-oriented tools enable applications to be developed in a way that models the real world. Applications developed with object-oriented tools consist of various 'objects' that have certain attributes and relationships in common with the real world. Object-oriented tools promise to provide significant increases in productivity by making code re-usable and easier to understand, improving the user interface, and providing support for hypermedia (multimedia) applications — that is, applications that support more than one medium — for instance, data, voice, image, and video. (Hypermedia was discussed in detail in Report 73, *Emerging Technologies.*)
- Rule-based development tools have been available for several years, but over the next five years, they will become easier to use and their capabilities will increase significantly. In particular, they will increasingly be able to integrate rule-based applications with other types of application. They will encourage the development of rule-based applications that can tackle complicated problems, and they will be used to improve the automatic assistance given to users by embedding 'guidance on use' in the application.
- In other areas, advances in development tools are making it possible to execute existing applications faster, to develop multimedia applications, to exploit computer resources better, or to integrate applications in areas that are currently isolated.

Obviously, not all the tools mentioned in this chapter will be applicable to all development departments, but we recommend that members consider how and where it might be possible to benefit from using them.

Object-orientation will be an effective development approach by 1994

Object-orientation is not a new concept; it was used extensively in artificial intelligence work in the 1960s. Yet, over the last few years, it has received so much publicity that it is now often taken to mean anything with an icon or a

Figure 4.1 Future advances in tools outside the I-CASE environment will permit the development of more sophisticated applications

Benefits	System development tools					
	Object-oriented tools	Rule-based tools	Parallel tools	Multimedia/ interfacing tools	Distributed processing	Integrated tools
Significant increase in development productivity	✓	✓		✓		
Easier interpretation of the application	✓	✓		✓		
Support for greater logical complexity	✓	✓				
Support for hypermedia applications	✓			✓		
Support for better user interfaces	✓			✓		
Integration with non-data processing systems	✓					✓
Improved guidance to the user		✓		✓		
Improved performance			✓		✓	
Improved utilisation of resources					✓	

window-based interface. Object-orientation, in fact, provides a means of developing applications that emulate the way in which objects in the real world relate to each other. Objects are arranged in a hierarchy, with each object building on the attributes of other objects higher up the hierarchy. Representing objects in this way has several advantages, such as ease of understanding and reduced coding and maintenance.

Development techniques and tools specially designed to provide the basic elements of object-orientation are already available. Because the potential benefits of full object-orientation are so significant, more and more suppliers are examining ways of introducing object-orientation to existing tools, or of developing new tools based on object-oriented concepts. Once the initial barriers to using such a radically different development approach have been surmounted, early users of object-oriented tools have gained considerable benefits, and the take-up of object-oriented concepts is set to develop quickly in the first half of the 1990s, as described overleaf in Figure 4.2.

Object-orientation reflects the objects and activities of the real world

With the object-oriented approach, applications are constructed from software objects that, in most cases, represent objects in the real world, and from messages that represent commands or actions to be carried out by objects. Each object has two basic components — the form of the data or information associated with the object (its *state),* and the procedures associated with the object (its *methods).* The combination of both data and procedures within an object is known as *encapsulation.* Examples of an object and its state and methods, taken from a banking application, are shown in Figure 4.3, on page 41. In the top left-hand corner of this figure, the 'Account' is the object, and its state contains four elements: account number, account owner, balance, and interest rate. Three methods are associated with the object: deposit of cash, withdrawal of cash, and processing of a cheque.

Messages sent to an object are interpreted according to the object's state and methods. Thus, the same message may be sent to several

Figure 4.2 Object-oriented concepts and tools will be in use within most organisations by 1994

1990	Several organisations use object-oriented programming systems and object-oriented database management systems to develop complex applications. Organisations start introducing window- and icon-based interfaces to in-house applications.
1991	Tool suppliers release more comprehensive object-oriented tool sets and database management systems for a wider range of hardware and software environments. Users place pressure on the development department for more object-oriented interfaces similar to the windowing facilities available in high-street products.
1992	Most in-house applications have a standard object-oriented-type interface. Object-oriented facilities appear in several tools and in application areas such as office automation and process control. Organisations assess the viability of using object-oriented concepts for a wider range of developments.
1993	Early users of object-oriented development tools start to reap the benefits from the library of objects that will have been built up. Organisations extend their use of object-oriented tools for a wider range of complex applications.
1994	I-CASE suppliers start supplying object-oriented tools and database management systems. Packages containing commonly used objects become available; and these will be incorporated into an organisation's library of objects. Object-oriented development becomes established as one of the approaches available to the systems department to aid in the task of developing applications.

different objects, each of which may interpret it differently. This is known as *polymorphism*. In our example, the message 'withdraw $100 from account number 123 (Mr Jones)' may be sent to both the objects, 'Savings' and 'Bonus'. If sufficient funds are available, the withdrawal will be made from the 'Savings' account. Funds would be withdrawn from the 'Bonus' account only if the money had been in the account for a predefined period of time.

Related objects can be grouped into classes and arranged into a hierarchy. On the lower left-hand side of Figure 4.3, the various types of accounts are arranged into a hierarchy. The object 'Account' is the basic form of account. 'Savings', 'Normal' and 'Bonus' are all specialised forms of account. 'Account' is therefore both an object and a class. The objects 'Savings', 'Normal', and 'Bonus' are special forms of 'Account' and are thus sub-classes of 'Account'. All the attributes (state and methods) associated with 'Account' also apply to 'Normal', 'Savings', and 'Bonus', and thus are defined only once.

If necessary, the attributes associated with any object can be modified, or new ones can be added, but only the attributes that are different need to be defined. For instance, in our example, the method 'cheque' is redefined to exclude cheque withdrawals from the savings account. This arrangement of objects into a hierarchy, where objects transfer attributes down the hierarchy, is known as *inheritance*.

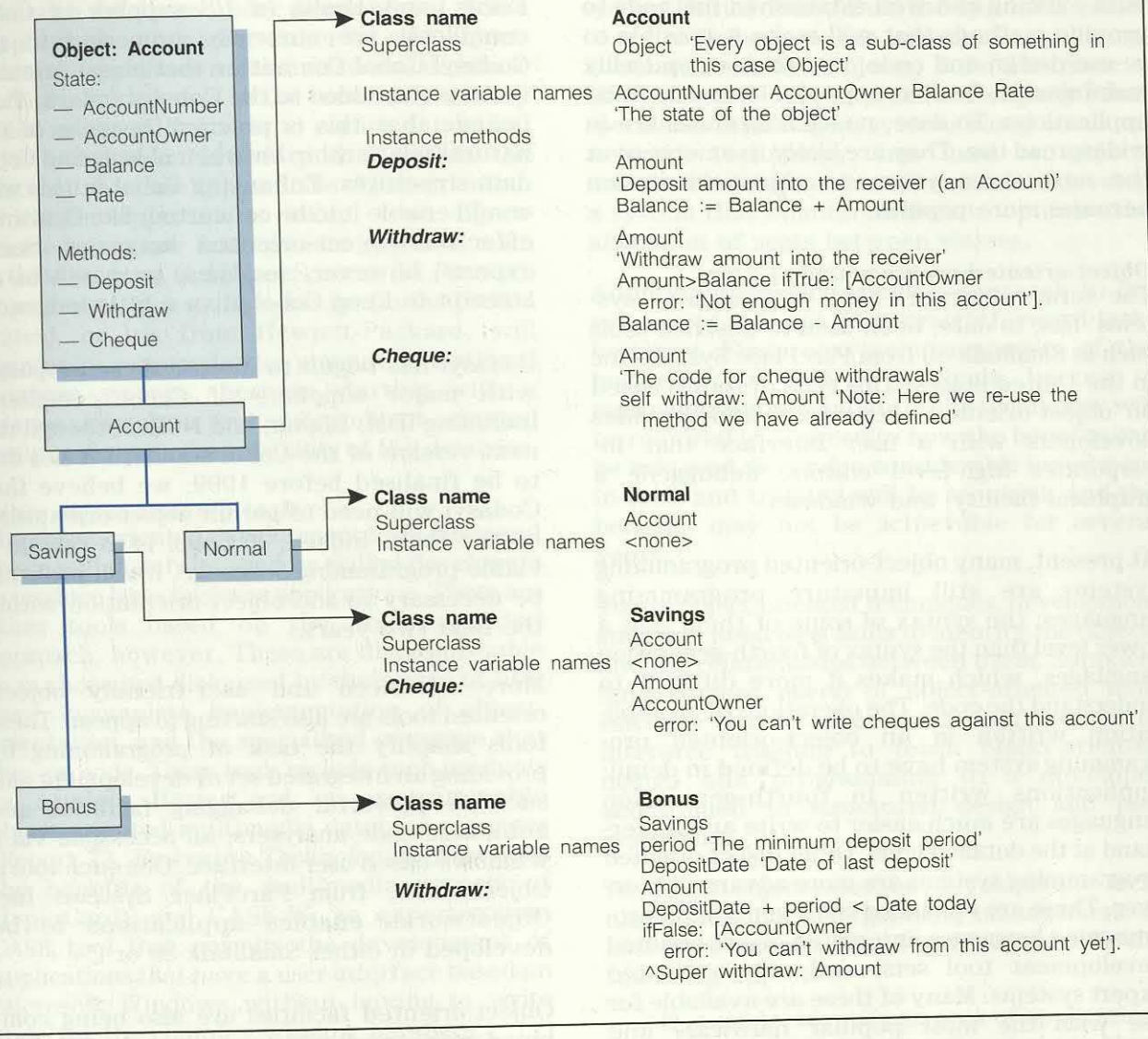
Object-orientation with encapsulation, polymorphism, and inheritance provides a means of developing applications that are easier to understand, require less maintenance, and are easier and quicker to construct:

- Applications developed by using object-oriented techniques are easier for business people to understand because the descriptions of the high-level objects and the actions resulting from the messages reflect the business processes.
- They require less maintenance because small changes in the real world should result in small changes in the corresponding objects. Equivalent applications developed using more traditional techniques would typically require a sizeable change, with a correspondingly large amount of recoding.
- Applications developed using object-oriented techniques are easier and quicker to construct because existing objects can be reused in other applications. In addition, the inheritance property of objects means that less coding is required.

Object-orientation is being introduced into many development aids

Over the past few years, object-orientation has gradually been recognised as a very productive approach to the development of computer applications. Several development methods and tools based on the object-oriented approach are already available. These can be divided into three main areas — object-oriented analysis and design methods, object-oriented programming systems, and object-oriented database management systems. However, many of the object-oriented programming and database tools available today are primitive. They will need to be enhanced to provide many of the basic

Figure 4.3 Each object has two basic components — the form of the data associated with the object (its state) and the procedures associated with the object (its methods)



features (such as a high-level language, and data recovery and back-up) required for commercial computing before the object-orientation approach can be widely adopted. The larger suppliers of tools are all working to incorporate object-oriented concepts into their products, and the current shortcomings will undoubtedly be overcome.

Early in 1989, the Object Management Group was established in the United States to set a standard for the emerging methods in object-oriented development and to promote the concept of object management. By September 1989, there were 18 members, including Data General, Hewlett-Packard, Sun Microsystems, Unisys, Canon, Philips, and 3Com. In December 1989, the Object Management Group and the X/Open Group announced an agreement under which they will cooperate in areas of mutual interest, with X/Open adopting standards approved by the Object Management Group, where appropriate, and incorporating the specifications into future releases of its X/Open Portability Guide Book.

Object-oriented analysis and design methods

Object-oriented analysis and design methods exploit the underlying object-oriented programming and database management systems

and aim to overcome some of the deficiencies of existing development methods and tools. They will be created by merging new concepts with existing and well established methods to provide methods that will make it possible to re-use design and code, and to develop easily maintainable, flexible, expandable, and modular applications. To date, no such methods are in widespread use. They are likely to emerge over the next three years, as object-orientation becomes more popular.

Object-oriented programming systems

The term 'object-oriented programming systems' has, to date, been used to describe tools such as Smalltalk-80 from ParcPlace Systems Inc in the United States. This type of tool is based on object-oriented techniques, and provides developers with a user interface that incorporates high-level editors, debuggers, a graphical facility, and windows.

At present, many object-oriented programming systems are still immature programming languages; the syntax of some of them is at a lower level than the syntax of fourth-generation languages, which makes it more difficult to understand the code. The operations of an application written in an object-oriented programming system have to be defined in detail; applications written in fourth-generation languages are much easier to write and understand at the detailed level. Other object-oriented programming systems are more advanced however. These are enhanced third-generation programming languages, integrated object-oriented development tool sets, and object-oriented expert systems. Many of these are available for use with the most popular hardware and software environments, although by far the most common are those available for Unix and MS-DOS environments.

One of the more popular object-oriented programming systems is actually a third-generation language that has been enhanced to include object-oriented concepts. This is C++ from AT&T, an enhanced version of the C programming language. The benefits provided by enhancing third-generation languages in this way are reduced learning time, relatively easy conversion of existing applications, and the maturity (and hence, stability) of the third-generation language.

Some suppliers are now attempting to enhance Cobol by providing it with object-oriented programming facilities. Hewlett-Packard, Micro Focus, and Realia (a US supplier of Cobol compilers) are currently proposing to the Codasyl Cobol Committee that object-oriented features be added to the Cobol standard. They believe that this is practical because of the natural relationship between objects and Cobol data structures. Enhancing Cobol in this way would enable it to be converted, like C, into an effective object-oriented language. Some experts, however, see these proposals as an attempt to keep Cobol alive a little longer.

Codasyl has begun to discuss these proposals with major suppliers of Cobol compilers, including IBM, Digital, and NCR. Although the next version of the Cobol standard is not due to be finalised before 1999, we believe that Codasyl will need to permit object-orientation before then. Indeed, if Cobol is to remain a viable programming language, it will probably be necessary to add object-orientation within the next two years.

More advanced and user-friendly object-oriented tools are also starting to appear. These tools simplify the task of programming by providing an integrated set of development aids such as powerful debugging facilities and automatic code analysers, all accessible via a windows-based user interface. One such tool is Objectworks, from ParcPlace Systems Inc. Objectworks enables applications to be developed in either Smalltalk-80 or C++.

Object-oriented facilities are also being combined with expert systems to provide tools that will enable very complex applications to be developed and easily maintained. The Aion Development System (ADS), from Aion in the United States, is one such example. In Report 73, *Emerging Technologies*, this type of tool was classified as a fifth-generation toolkit, as it provides clear improvements in capability and productivity over the earlier generations of tools.

Object-oriented database management systems

Object-oriented database management systems provide facilities for holding objects in a database and loading them into memory when they

need to be executed. They are more relevant to commercial computing than object-oriented programming systems, and many tool suppliers have such databases under development. Some suppliers are promising either to add object-oriented capabilities to their relational database management systems in the same way as they added relational features to earlier databases, or to provide an interface that makes the relational database appear to be object-oriented. Whichever approach is taken, true object-oriented database management systems, such as G-BASE from Graphael Soretas (in France), Vbase from Ontologic Systems Inc (in the United States), or Iris from Hewlett-Packard, will eventually supersede the amended relational database systems, because starting with a relational database and adding object-oriented features restricts the capability of the database.

Other object-oriented tools

The object-oriented development tools discussed above will mainly be used by skilled developers to produce large business applications. There are other tools based on the object-oriented approach, however. These are distinguishable from those just discussed by their ease of use, their incomplete implementation of object-orientation, and the specialised coverage that they provide. These tools include such products as Apple's HyperCard, a programmable object-oriented multimedia database manager (Report 73, *Emerging Technologies*, describes the benefits of the multimedia aspects of HyperCard), and CASE:W, an expert-system CASE tool that permits the development of applications that have a user interface based on Microsoft Windows without having to write code for the complex window handlers.

Initial problems should not discourage the adoption of object-oriented tools

Although object-oriented technology is still relatively immature, several companies have already used the object-oriented approach to develop various applications and have found it to be very productive. Cadre Technologies, a large US developer of computer-aided software-engineering tools, claims that the amount of new code needed for an application has been reduced by 80 per cent since it introduced object-oriented tools, because so much of the existing object-oriented code can be re-used.

In Canada Wild Leitz, part of the Swiss manufacturing group, used an object-oriented approach to develop a geographical information system. It has found that the system is easier to maintain because the impact of any change is isolated to a few objects. BehavHeuristic, a US software development company that specialises in intelligent systems based on neural network technology, has used an object-oriented development environment to develop a system that enables airlines to optimise the allocation of seats between classes.

Adopting an object-oriented approach to development will not be a straightforward task, however. First, convincing managers of the business case will be difficult, because a reasonable understanding of the technology will be required to appreciate how the benefits can be achieved. Second, a considerable investment in tools and training will be required, and the benefits may not be achievable for several years.

To use object-oriented techniques, development staff will need new skills to identify the objects and the relationships between them. Suppliers (and current users) of object-oriented tools estimate that it takes about four months of intensive training to teach object-oriented design and programming to a developer accustomed to structured design and programming.

To adopt object-oriented development techniques without antagonising management or users, we suggest that organisations take the following approach:

- If possible, identify an area of applications development that is well suited to object-oriented techniques and that has a relatively high profile — for example, an application that deals with objects in the real world, like stock control, or process control. In such areas, it is usually easy to identify the objects and the relationships between them, but the application would be difficult or expensive to develop with existing tools. Success with object-oriented techniques in one area will arouse interest and encourage further developments in other areas.
- Standardise on an object-based user interface. Both existing and new applications

can be made considerably more user-friendly with object-oriented techniques such as window- and icon-based interfaces. One of the first computers to use such techniques was the Macintosh. Object-oriented user interfaces are being used on more and more computers — NextStep, the interface for the NeXT workstation being a good example. NextStep, the Open Software Foundation's Motif, and IBM's Presentation Manager are all examples of how the WIMPS (windows, icons, menus, and pointing devices) interface is being transferred to mainstream computing.

Rule-based technology will emerge in several forms

Rule-based technology has been widely used to build systems that capture human expertise, and forms the core of expert-system shells and languages, as explained in Foundation Report 60, *Expert Systems in Business*. During the first half of the 1990s, there will be a phenomenal increase in the use of rule-based technology, not only embedded in the tools used for development, but also as part of the application being developed. This technology will provide a means of developing rapidly changing, complex applications that would previously have been too expensive or have resulted in an impossible maintenance task.

The main advances in rule-based technology will be in the following four areas:

- *Integration of expert-systems tools with current tools* — Many suppliers of existing tools will integrate their products either with an existing expert-system tool or with one that they have developed themselves. The enhanced tool will facilitate the development of data processing applications that have embedded expert systems that can either provide advice and help to the user, or be used to code complex parts of applications that could not be solved with conventional programming techniques.
- *Fifth-generation toolkits* — There are several fifth-generation toolkits currently in use that have proved to be very effective for the development both of expert systems and of other types of application. These tools, which combine rules and objects with older development techniques, will enable very sophisticated and complex applications to be developed and easily maintained.
- *Application-specific tools for building expert systems* — Rule-based tools (and even knowledge bases) will increasingly be available for specific application areas. These tools will, in some cases, enable developers with no experience of knowledge engineering to develop knowledge-based applications.
- *Tools with embedded expert systems* — An increasing number of tools used to help develop applications will have expert systems embedded in them. They will provide expert guidance to developers and will help to improve the quality and productivity of applications development.

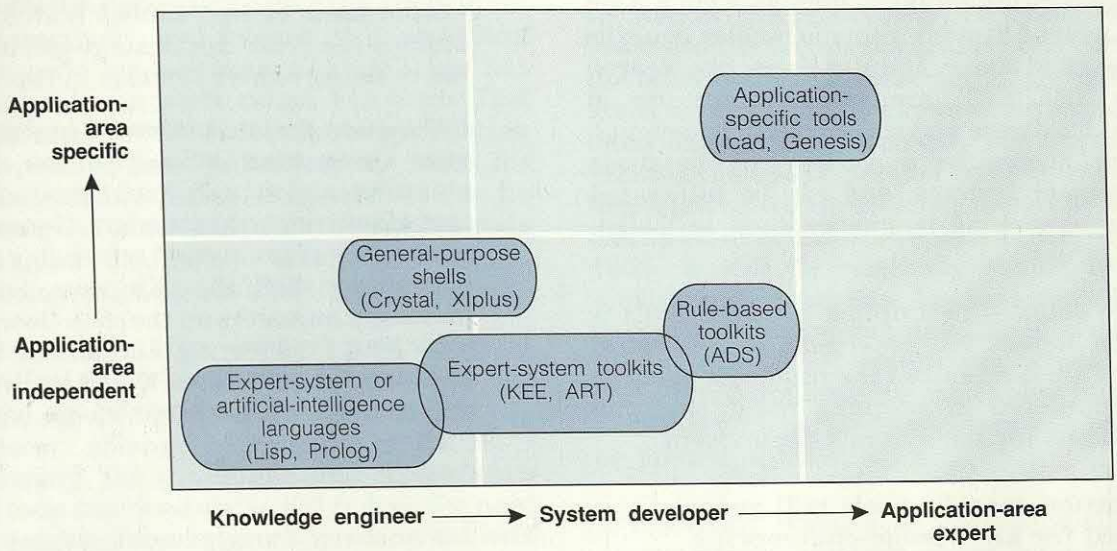
These advances will help organisations overcome many of the barriers that have limited the take-up of rule-based or expert systems to date, such as the need for knowledge-engineering expertise, problems experienced by early attempts to exploit expert systems in a business environment, and the inability to integrate rule-based systems with existing systems. Figure 4.4 shows how these advances make rule-based applications and expert systems more accessible to developers, with the loss, in some cases, of some flexibility. As rule-based tools become more readily available and are more widely used, the successful development of expert systems will become more common.

Ideally, rule-based tools should be capable of being used in different hardware and software environments, provide access to other databases and languages, and allow business users to define new rules, and carry out appropriate testing while the application is being used. This ideal is many years away, but the current advances are steps in the right direction.

It will be possible to integrate expert systems and conventional applications

One of the reasons for the slow take-up of expert systems is the difficulty of integrating them with conventional data processing applications. Most successful expert systems have been standalone applications, used in very specialised areas. Several banks and financial

Figure 4.4 There is a trend towards less flexible but more usable rule-based development tools

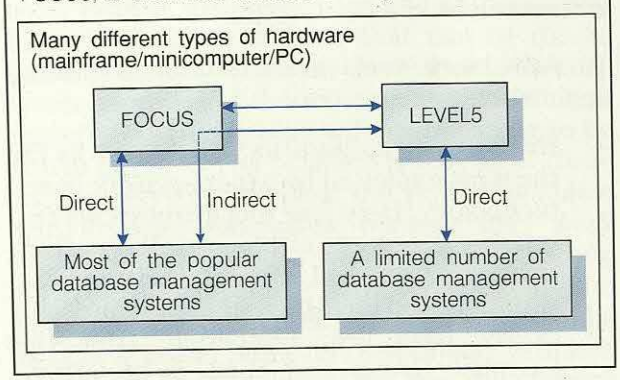


institutions, for instance, have demonstrated the benefits of using expert systems, but these are largely separate applications because of the difficulties of integrating the expert system with conventional applications. Many leading tool suppliers are now beginning to work on integrating the tools and development environments used for building expert systems with existing tools and databases and thus improve the level of integration between expert systems and conventional applications.

One supplier who has been working on this problem for several years is Information Builders. It is best known for its fourth-generation language and database management system, FOCUS, which is used worldwide by approximately 600,000 people. In 1987, Information Builders acquired Level Five Research, the supplier of the LEVEL5 expert-system development tool, and is currently working on linking database and expert-system technologies by embedding LEVEL5 within FOCUS. Figure 4.5 describes how Information Builders intends to integrate these two technologies and thereby provide access to the wide range of existing applications and databases running a variety of computer systems already implemented with FOCUS. Early in 1990, Information Builders will be releasing a development tool, LEVEL5/OBJECT. This tool will enable the development of integrated business applications, using a combination of advanced artificial intelligence techniques and an object-oriented development environment.

Figure 4.5 Information Builders provides links to an expert system from FOCUS

LEVEL5, the expert system tool, is available today from Information Builders as a standalone product for building and running expert systems. Information Builders has also linked the LEVEL5 product with its FOCUS database management system, as illustrated in the diagram below, so that LEVEL5 applications can be called from within FOCUS in the same way as subroutines can be called. Not only will this enable FOCUS users to call expert-system-based routines from within any FOCUS application, but it will enable LEVEL5 to be used on the wide range of computer systems supported by FOCUS and gives access, via FOCUS, to enormous stores of existing data.



Fifth-generation toolkits will increase productivity in developing complex applications

Fifth-generation toolkits, such as Aion's Aion Development System, provide similar benefits to those cited above. Many of them, like Neuron Data's NEXPERT, are available for use in various hardware and software environments, provide access to a variety of database management systems, and can be integrated with programming languages such as Cobol, Pascal, and C.

A more detailed description of these tools is given in Report 73, *Emerging Technologies*. Report 73 also describes the problems likely to be encountered in adopting them, and the implications for the systems department.

Application-specific tools will reduce the need for knowledge-engineering skills

At the beginning of 1989, a new type of application-specific expert-system tool began to appear. This type of tool provides all the basic building blocks and facilities required to construct an expert system, and contains embedded general knowledge of the application area. The users of such a tool are 'shielded' from the complexities of knowledge engineering by a framework that helps them to develop the application. In some cases, so little knowledge-engineering experience is required that users can make significant contributions to the development of an expert system, or even build it themselves.

To date, such tools are available for three application areas:

- *Engineering diagnostics:* This area is by far the most exploited by expert-system-based technology. Here, the tool analyses several inputs, received either remotely from sensors or entered directly, assesses the situation, and identifies where the problem or the fault lies. TestBench, from the Carnegie Group and Ford, is an offline consultative tool for tackling large and complex engineering-diagnostic problems.
- *Process control and management:* Tools in this area are aimed primarily at the computer-integrated manufacturing market, although they are also used in network management, and alarm analysis. Typically, they will need to respond in realtime to the analysis of any situation. G2, from Gensym Corporation in the United States, is an online, realtime, application-specific tool for building expert systems in this area.
- *Production design, planning, and scheduling:* These tools help with the design, planning, and scheduling of products in a manufacturing environment. Genesis is a production-planning and scheduling toolkit from Sira in the United Kingdom, designed for use by engineers on the shop floor. Icad, from Icad Engineering Automation Ltd of Coventry in the United Kingdom, is a tool that integrates a design knowledge base and CAD techniques to provide 'intelligent CAD'.

The move away from general-purpose rule-based tools that can be used only by staff with knowledge-engineering skills towards more applications-specific tools that require staff with little or no skill in the knowledge-engineering area will certainly increase the use of expert systems. In the early 1990s, the use of application-specific tools will spread to other areas where rule-based technology can be exploited.

Embedded expert systems will provide guidance to the user

Earlier, in Chapter 3, we said that knowledge bases will be embedded in inverse-engineering tools used to unscramble existing applications code, and that they will help developers to follow complex development methods. Examples include VIA/Insight, the interactive Cobol code analyser from VIASOFT Inc, which is based on a knowledge base, and TOP-ONE/CONVEX, which converts Cobol from ICL to IBM environments, and is based on an expert system. There is a growing trend for tools to have a knowledge base embedded in them to help developers carry out tedious or complex tasks.

Embedded expert systems will be available in the near future to support:

- *The analysis process:* Helping to gather facts and information, analysing this

information and assessing its completeness, and generating the specification of requirements.

- *The testing process:* Testing the application interactively, and interactively analysing the code for mistakes. VIASOFT Inc has released a product called VIA/SmartTest that analyses the code, and an application knowledge base containing the rules for analysis and testing. This product can be combined with VIASOFT's interactive code analyser, VIA/Insight, to form VIA/Center, a comprehensive set of maintenance tools.
- *Decision making:* Providing guidance at critical points during development on the best use of the tools available.

In summary, the continuing advances in rule-based tools described above will reduce the need for specialist knowledge-engineering skills to develop rule-based applications. By the mid-1990s, the development and use of rule-based applications will be commonplace, and application-specific tools will enable most users to develop their own rule-based applications.

Other advances in tool technology will be valuable in particular areas

Concurrently with advances in object-oriented and rule-based tools, there will be changes in the capabilities of existing tools, and additions to them that will enable different types of applications to be developed. The most significant advances will be:

- The introduction of tools to exploit parallel computers.
- The use of multimedia tools to develop new types of applications.
- The use of powerful tools to develop greatly improved human/machine interfaces.
- The use of tools to facilitate distributed processing and thus improve the use of computer resources.
- The use of various tools to integrate office automation and data processing applications.

New tools will enable existing applications to exploit parallel computers

In Report 73, *Emerging Technologies*, we discussed the trends in parallel computers and the benefits that can be achieved, and described areas in which parallel computing has already been exploited in business. The barriers inhibiting the use of parallel computers and the implications for the systems department were discussed in detail. We reported that, by about 1993, it will be relatively simple to write applications that can be transferred from uniprocessors to computers based on parallel processors. The extent to which these transferred applications can exploit the power available with parallel computers will, however, be limited, in most cases.

It is already possible to transfer applications directly from a uniprocessor computer to a parallel computer if the computers have compatible operating systems. Although it may be possible to run several of the transferred applications in parallel, thereby increasing throughput, the execution time for individual applications is not likely to decrease by much. Even so, the operational improvement achievable in this way will be adequate for most of today's applications.

There are, however, several kinds of specialised tools that can be used to enable existing applications to make better use of the processing power available with parallel computers. Some tools analyse the application and automatically identify sections of code that can be run in parallel; others permit the developer to identify whole modules of the existing application and to define the code that will enable these to be run in parallel. The problem is that all of these tools rarely fully exploit the power available with parallel computers, because they work with sequential code, which means that any inherently parallel characteristics of the application are not taken into account. Over the next few years, however, reasonably efficient tools will become available for transferring existing (sequential) applications into a parallel environment. Since few existing applications will need to be redesigned and rewritten to exploit the full power available with parallel computers, these tools will allow most

organisations to exploit parallel computers with the minimum of rework.

Multimedia tools will enable new types of applications to be developed

Windowing techniques not only make the human/machine interface look more interesting, but also enable the user to be more productive. This is also true for hypermedia (or multimedia) systems, which were described in detail in Report 73, *Emerging Technologies*. In that report, however, we warned against the use of multimedia systems just to add special effects to the user interface; multimedia systems must also provide some business benefits.

We expect that, by about 1993, tools will be available to enhance windows-based interfaces to provide a multimedia capability, both for existing and new applications. The major limiting factor on the development of multimedia applications is the immaturity of multimedia database management systems. Although several suppliers are currently offering such systems — Informix Software's Informix-Online database, and Graphael Soretas's G-BASE, for example — it will take time before they are widely adopted. Early examples of this type of tool are starting to have an impact in the area of expert systems, however. Several suppliers of expert systems are using Apple's HyperCard as an interface to their products, and Intellisoft has successfully combined hypertext software with its expert system, Knowledge Pro.

During the mid-1990s, products like NextStep from NeXT, and NewWave from Hewlett-Packard will evolve to form a powerful PC- or workstation-based interface to applications. These developments will provide facilities for developing extremely powerful and easy to use human/machine interfaces that use windows, multitasking, multimedia, and object-oriented facilities. Figure 4.6 shows examples of the types of user interfaces that will be available in the 1990s.

The use of these types of interfaces will have a profound effect on future workstations. As more power is required in the terminal both to drive the user interface and for applications processing, today's dumb terminals and PCs will be replaced by or upgraded to powerful workstations with a WIMPS interface.

Future tools will make it possible to develop greatly improved human/machine interfaces

At present, applications developed in-house rarely have powerful, user-friendly human/machine interfaces, primarily because the time and costs constraints on in-house developments are usually so severe. Over the next few years, the introduction of powerful tools will make it possible to develop window-based interfaces for many in-house applications. We have already mentioned how tools such as CASE:W can be used to generate all the basic windowing functions needed for an application, without the need for detailed programming. This type of tool will become much more common once the current copyright disputes over the various 'look-and-feel formats' are settled. Until these disputes are settled, tool suppliers will be reluctant to invest heavily to provide products that support a particular windowing style.

Extensions to current tools will encourage better use of computing resources

The trends in user interfaces and workstations are associated with the trend towards distributed data processing. Distributed processing, in its broadest sense, allows applications, or parts of an application, to be executed on various processors connected through a network. The decision about what to execute where is made either at compile or at execution time, the aim being to make more efficient use of the resources available and hence improve cost-effectiveness and response times. The ability to offload some (or all) of the processing onto cheaper computers makes distributed processing a very attractive proposition.

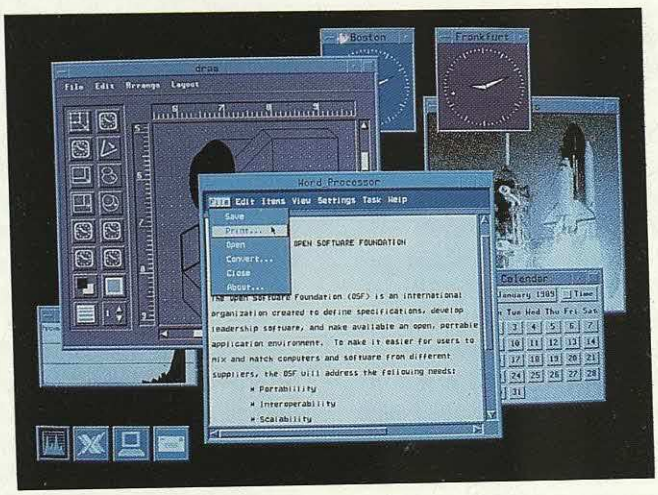
Distributed processing ranges from the automatic distribution of predefined programs or modules across various processors, as in parallel computing, to complete applications that are distributed across a network of computers. The most common form in use today is that of client/server systems, where the client machines (that is, workstations) contain the user interface, while the server machine holds the

database. Requests for data are passed across the network from the client to the server, and only the appropriate data is returned to the client machine.

Much work has been done in providing tools to support distributed data processing. IBM, for example, provides strong support with facilities such as the Transparent Computing Facility (TCF) clustering service for its AIX operating system. This facility enables all the machines in the cluster to cooperate with one another in a transparent manner — that is, the user need not be concerned with which machine in the cluster is currently being used. Digital is also providing support for distributed data processing with its Vaxcluster approach. Although a Vaxcluster system can contain a wide range of Vax processors, storage controllers, and peripheral devices, and can vary greatly in size and components, it functions as a single entity.

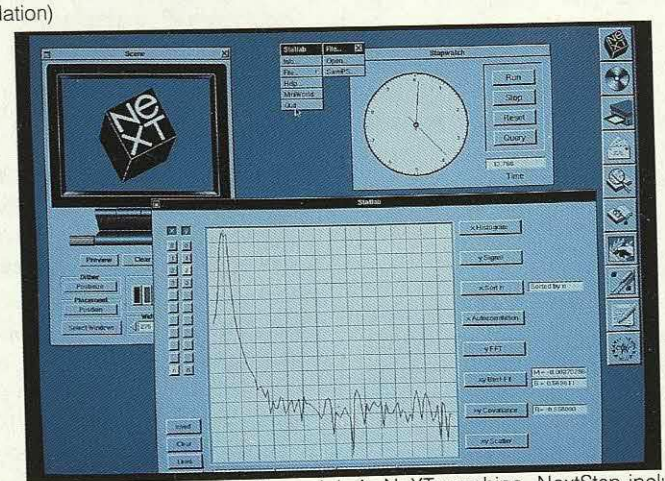
Several of the major tool suppliers are also seeking to provide their products with facilities to support distributed processing. Information Builders, with FOCUS, and Must Software

Figure 4.6 In the 1990s, users will have access to extremely powerful human/machine interfaces



The photograph shows Motif, the graphical user interface designed by the Open Software Foundation, Motif combines Digital's XUI and Hewlett-Packard's X Widgets with a Presentation Manager look and NewWave's three-dimensional windows in an X-Windows environment.

(Source: Open Software Foundation)



The photograph shows NextStep, the user interface for Steve Jobs's NeXT machine. NextStep includes a set of tools for object-oriented programming.

(Source: Paul Avis Photography Inc)

International, with Nomad, have adopted phased approaches to ensure that, by the early 1990s, their respective fourth-generation languages can be used to develop distributed applications. This trend will be followed by many other suppliers. We expect that, by 1993, most development departments will have the opportunity to exploit their organisation's computing resources by using tools that support the development of distributed applications, providing, of course, that their computers are networked.

Future tools will be able to integrate office automation and data processing applications

To date, office automation applications and data processing applications have, by and large, been treated as two separate areas. Many organisations have recognised the advantages of integrating these two types of application, and tool suppliers have begun to provide tools that will make this possible.

Oracle, for instance, has announced an electronic mail system, Oracle*Mail, that is compatible with the various other Oracle tools. The eventual aim is to provide a database that supports all information, regardless of its form or function. Thus, the Oracle database will support word processing information, facsimile information, and data processing information, regardless of its origin or business area.

The advances in tools described in this chapter (and earlier in the report) will not only enable the development department to provide more sophisticated applications, but they will also enable users to make better use of the computing resources available within the organisation. As the facilities available with the new tools begin to emerge, the development department will have to manage the expectations of its users. Many business managers will be aware of the potential of the new tools and facilities and will begin to press for applications that take advantage of them. If the development department fails to respond in an appropriate way, business managers will start to experiment with the tools themselves. Indeed, they will be encouraged to do so by the suppliers, who will position many of their products as end-user tools.

The systems department must therefore reappraise its attitude towards end-user computing in the light of the new tools that are becoming available. In particular, it must strike the appropriate balance between uncontrolled development by users and stifling the initiative of users, thereby jeopardising the enormous benefits that can be gained from end-user computing. In the final chapter of the report, we offer advice on how the systems department can provide support and guidance to users so that they can exploit the end-user tools effectively, and so that the systems department can prevent future chaos with end-user computing.

Chapter 5

Encourage and expand end-user computing

In most organisations, some applications development is being carried out outside the development department. Systems departments may or may not be aware of this work, and may or may not be supporting it. As a consequence, there is no consensus on either the role of users in applications development, or on the scope of end-user computing. For the purposes of this report, we therefore define end-user computing as the development of a 'program' in which a user (a member of staff not directly attached to, or working within, the development department) plays a major role. A 'program' in this context can range from a simple procedure to retrieve and analyse data to a complex, multi-user application.

The tools designed specifically for end-user computing are relatively immature compared with traditional development tools. From the early 1980s onwards, most systems departments have provided their users with various tools and support in their use. Successful use of these tools has been the exception rather than the rule, and it is therefore still common for systems departments to consider end-user computing as a 'second class' form of computing. Nevertheless, the trend towards end-user computing is unlikely to be reversed. The tools suitable for use by business users are becoming more sophisticated, enabling a growing number of them to develop advanced applications. If correctly supported, guided, and managed, end-user computing will provide business benefits because users will be able to fully exploit the computing resources available. Well developed and useful end-user applications can also relieve some of the pressures on the development department.

End-user tools have met with varied success in the past

Since users began to get involved in developing their own applications in the early 1980s, they have been obliged to use tools that were mainly designed for professional systems developers, rather than business users. Today, however, they have access to tools such as FOCUS, which can be used as a flexible data-access tool and as a tool for developing quite complex applications. Users have developed relatively large applications with Mapper from Unisys, and HyperCard from Apple Computer Inc, and the macro facilities of products such as Symphony and Excel are being used more and more to produce very complex spreadsheet applications. The problem is that these tools do not automatically include vital features such as recovery, audit, and back-up, and it is unrealistic to expect most users to have the expertise to provide such features themselves.

The experience of most organisations with end-user computing has therefore been mixed. Furthermore, as the number of users involved in developing their own applications grows, and the business significance of their applications increases, the risk to the business also increases. One company we spoke to used a complex spreadsheet, developed by an accountant, to calculate budgets for the following year. As a result of one mistake in the spreadsheet, all the budget figures were underestimated by 10 per cent, with the result that insufficient funds were reserved for the business for the following year.

Our survey of Foundation members confirmed that problems with end-user computing are very common. Most members quoted difficulties in at least one of the following areas over the last few years:

- A lack of control and coordination of development effort.
- Insufficient understanding by users of the meaning of the information being accessed

and of systems issues such as security and back-ups.

- Limited availability of suitable tools for end-user computing.
- Problems with maintaining applications developed by users.

Development departments therefore tend to be very wary of attempts by users to develop applications themselves. Any attempt by systems departments to prevent users developing their own applications, however, is doomed to failure. Business users will be increasingly able and willing to use computers as an aid to their work. The systems department's role should be to encourage and support end-user computing in a way that allows business users sufficient freedom whilst also providing standards and guidelines in the areas where these are necessary for corporate consistency. Indeed, because of the current pressures on development departments, they can no longer ignore end-user computing.

Appropriate tools are now making end-user computing a valid option

In the last two years of the 1980s, there have been marked improvements in end-user computing tools. With the growth in the use of PCs, a wide range of user-oriented tools has become available, with improved user interfaces, automatic validation of information, and powerful commands making them easier to understand and use. This trend is likely to increase as advances continue to be made in both fourth-generation languages and the user interface, particularly windowing techniques. Some features, such as security and back-up, are still not generated automatically by most end-user tools, but such features have become less critical in the last few years because the latest hardware and network technologies now often incorporate them automatically. Furthermore, users now tend to be offered much better support and guidance both by the tools and by the systems department. These trends will result in an increase in the number of business users who are prepared and competent to develop their own applications and who will regard end-user computing as an effective use of their time.

In our survey, just over 75 per cent of Foundation members said that they provided business users with tools that enable them to access data and to develop applications. Very few, however, had any idea, other than estimates based on installed equipment or machine usage, of the extent to which the tools were actually used, what they are used for, or how the level of use is likely to rise in the future. This is partially due to the fact that measuring the use made of PCs that are not networked is extremely difficult. Nevertheless, the majority believed that there would be a significant and steady increase in end-user computing over the next five years. (This trend was confirmed by the organisations we interviewed that did keep records.)

Renault, a French car manufacturer, provides business users with microcomputers and mainframe tools and packages. Most users only need database query facilities, provided by IBM's QMP/DB2, and enhanced graphical and statistical facilities, provided by the SAS package from SAS Institute Incorporated. More complex applications are developed using SAS or Nomad 2 from Must Software International. Experienced users have developed microcomputer applications with Ashton Tate's dBase, Borland's Paradox, and the Multiplan and Excel spreadsheets. Renault expects the number of users to increase from 1,000 in 1989 to 10,000 in 1996.

At British Airways, an international airline, business users have produced over 10,000 procedures that are retained for future use, and countless others that have been used and discarded. If users had not carried out this development work, some of it would never have been done and some of it would have been added to the development department's applications backlog. Figure 5.1 shows the growth in end-user computing within British Airways. Such levels of growth indicate the importance of end-user computing to an organisation.

The rate of growth in end-user computing will clearly vary from organisation to organisation, but it will be encouraged by four major factors:

- *Technology*: More powerful and easier-to-use end-user tools are emerging, cheaper PCs and intelligent workstations are becoming available, and it is becoming possible for users to access computing

resources via a network without having to concern themselves with the physical configuration of the hardware or databases. The technological advances that are likely to have the greatest effect on the growth in end-user computing are described in Figure 5.2.

- *Business pressures*: Within many organisations, there is an increasing need for applications to be produced rapidly, in response to business pressures. The majority of these applications tend to be small, and to have a relatively short life. Many of these 'throw away' systems can be developed by users.

- *Education and awareness*: There is a growing number of computer-literate business staff who have received some formal training in computing and who are willing to use computer-based technology. This is leading to an increasing level of awareness of the capabilities of computer applications, particularly in the areas of flexible yet sometimes complex information access and report generation (sometimes known as executive information systems), and of flexible information access, analysis, and reporting (sometimes known as decision-support systems).

- *Frustration*: It is becoming increasingly difficult to attract and retain good development staff. This has resulted in a staff shortage that has increased the applications backlog and thus extended the time it takes to develop a new application. The response of users to an unacceptable timescale will be to do more of the work themselves, or to commission contract staff or a software-development company to do the work.

Ensuring that the most effective use of end-user computing is achieved will become more and more important for all organisations as the demand for new applications continues to outstrip the development department's ability to produce and maintain them. In the United Kingdom, for example, the average backlog now amounts to about 30 months. The demand for new applications is growing at about 15 per cent

Figure 5.1 End-user computing at British Airways has grown dramatically

British Airways employs about 49,000 staff. In 1984, it replaced its existing end-user tool, ADI/ADRS, with FOCUS. At that time, there were 122 staff with the ability to use FOCUS. Since then, the FOCUS user population has increased to over 2,000 — a 17-fold rise in less than five years. British Airways is now the biggest user of FOCUS in the United Kingdom. In the 16-month period prior to our meeting with British Airways, the number of staff using FOCUS on a regular basis had also risen dramatically — about 30 per cent per annum.

Since staff began to develop applications in FOCUS, over 10,000 procedures have been written and retained. These range from very basic general enquiry and access applications to very complex applications that actually write other FOCUS applications and then execute them.

During the interview, Chris Bell, from the FOCUS Support Information Management Group, demonstrated the potential benefits of using a powerful end-user tool such as FOCUS. He was able to use simple one- or two-line commands to extract information on the use of FOCUS, by date, development area, and so on, without leaving his desk or using the telephone. This demonstrates that if they are correctly implemented, and supported with the appropriate training and guidance, end-user tools can be effective aids to everyday work.

Figure 5.2 Advances in technology will make tools easier for users to exploit

Increased guidance: In the short term, most data-access and retrieval tools will have in-built help facilities to provide guidance on the use of the tool, but they will provide no guidance on the meaning or use of the data. In the medium to longer term, as expert and object-oriented systems mature, the more advanced end-user tools will be easier to use and will contain knowledge not only of what the data means, but of how it can be used. End-user tools that contain an embedded knowledge base will start to emerge in the early 1990s and tools based on object-orientation will follow in the mid-1990s.

Better facilities for transferring information: Other less powerful or less flexible end-user tools, such as spreadsheets, financial-modelling packages, and word processors, will be able to accept and process information regardless of its source. Such a facility is already available with some computers — the clipboard facility on the Apple Macintosh allows data, text, graphics, and images to be transferred from one tool to another in a consistent manner.

Improved user interface: Many end-user tools will be based on PCs or powerful workstations. Initially, the PCs may well be standalone, although the requirement for access to large databases will require many of them to be networked. Powerful networked workstations will be used where more computing power is required. These machines, combined with the end-user tools based on them, will provide a very flexible and powerful human/machine interface, using icons, windows, pull-down menus, and so on.

a year, whilst the number of qualified development staff is growing only at about 5 per cent a year. It is clear from these numbers that the gap between the demand for applications and the development department's capacity to provide new applications will continue to widen, even if there are substantial improvements in development productivity.

Provided with the appropriate tools, training, and guidance, users could do much of the maintenance work currently being carried out by the development department. A survey of 24 members of Butler Cox's Productivity Enhancement Programme, carried out in late 1988, showed that about 40 per cent of all maintenance work, primarily generating new reports, could be carried out by users. They could also access and analyse information without needing to have specific software written by the development department, thus contributing to a reduction in the applications backlog.

It is clear, however, that the advances in end-user tools will not, by themselves, overcome all the problems described above or result in all the improvements. Nevertheless, with help and guidance from the systems department, users will be able to make better use of the available computing resources and make a considerable contribution to reducing the development workload faced by most organisations. Effective end-user computing can produce significant business benefits. It can also result in a better relationship between the systems department and the user community, a better understanding by that community of the development issues facing the department, and a reduction in the workload of the development department. The challenge that faces the systems department is to provide an appropriate level of guidance and support to maintain some control without discouraging the enthusiasm of users and stifling their creative use of the emerging end-user tools.

It is critical to provide appropriate guidance and support

The advances described earlier in this report will make tools easier to use and more business-oriented. More and more users will therefore be able to play a constructive role in ensuring that the organisation's computing resources are used for the maximum benefit of the business. The resources of the systems department are, however, limited, and therefore need to be allocated carefully to ensure that they are used to the greatest possible effect.

We recommend that systems departments start by categorising the different types of user so that each category can be provided with the level of support, guidance, education, and tools that will enable business users to make the most effective use of the computing resources available to them. Without such a categorisation, it will be difficult to allocate resources in the most effective way and to plan for the growth of end-user computing.

Categorising staff in the way described below should not determine absolutely what tools may be used, or what level of service an individual is entitled to. Individual organisations are likely to have their own policies and constraints when deciding on the tools and support that should be provided. The categorisation will, however, enable the systems department to get a better understanding of the extent of end-user computing, and hence of the levels of guidance, support, and technology (hardware, software packages, and tools) that it is appropriate to provide in order to encourage end-user computing throughout the organisation.

The systems department should also set guidelines for different types of application, and encourage users to seek the development department's 'seal of approval' for each application. Encouraging users to have their developments approved will prevent the proliferation of poorly documented applications.

Identify categories of user

There are four main ways in which systems departments can classify different types of business user: by their role, by the type of data they access, by their department, or by their need for or use of applications and tools. We recommend that the latter is used, which is commonly broken down into five categories:

— Category 1: Potential users, who at present do not use any computer-based applications.

- Category 2: Those who have a need to use or who only use applications and packages that have been written for a specific task that requires them to input data — for example, an accounts application.
- Category 3: Those who have a need to use or who use enquiry and analysis tools to access databases and analyse the data.
- Category 4: Those who have a need to develop or who use end-user tools to develop small applications, primarily for personal use.
- Category 5: Those who have a need to develop or who develop applications that may be used by many other users.

Each category of user is, in effect, an expansion of the one prior to it. Users tend to move through Categories 1 to 5 when first introduced to end-user computing, and regress through the categories as they move into the higher managerial roles. Each category refers to the use (actual or potential) made of end-user tools rather than to the type of tool used. Therefore, someone using a spreadsheet simply to add up a list of figures would be in Category 2, a user loading data into a spreadsheet from a database and analysing it would be in Category 3, and a user writing macros and developing a spreadsheet for a specified task would be in Category 4 or 5.

Staff can be assigned to the appropriate category by means of a simple questionnaire that assesses their use of tools as well as their needs. There will, of course, need to be some mechanism for re-assessing at regular intervals the category to which an individual is assigned, because neither his needs nor the technologies used will be static. Once staff have been categorised in this way, the appropriate level of support and resources can be allocated in the most effective manner. Figure 5.3 suggests how the various types of support and resources — tools, training, help, guidance, and so on — might be allocated. In this figure, the tools shown in the cells on the first row have been classified as follows:

- *Fixed-processing tools*: These are the applications and packages used to support the daily work of the users. Most of the applications will have been developed in-house or bought as packages. All of these tools carry out a fixed processing task on specified information.
- *Flexible processing tools*: These are the packages, such as spreadsheets and financial modelling packages, that allow users to process data in a predefined manner.
- *Data-access tools*: These enable data to be accessed and retrieved from centralised or corporate databases. They generally permit

Figure 5.3 Allocating appropriate resources to the different types of end user will encourage growth in, and improve, end-user computing

	Category of user (Relationship with development department)				
Type of support needed	1. Potential user (None)	2. Current user (Weak)	3.Data-access user (Medium)	4. Personal developer user (Strong)	5. User developer (Very strong)
Tools	—	Fixed/flexible processing tools	PLUS Data-access tools	PLUS Report-generation tools	PLUS Fourth-generation tools
Machine (access)	—	Dumb terminal/PC	PC	Powerful workstation	Powerful workstation
Training	IT awareness	Use of tools	PLUS Basic data processing	PLUS Basic development	PLUS Best practice for data processing
Help (you telephone us)	—	Permanently staffed help desk	Permanently staffed help desk	Telephone	Telephone or face-to-face
Guidance (we advise you)	—	Hardly at all	Very little	Hand-holding	Hand-holding and directing

'read only' access and the data is transferred to a local machine if it is to be amended or modified. These tools use a simple programming language or a pseudo-English language syntax.

— *Report-generation tools:* These generally enable reports to be generated from a local or centralised database. Again, they tend to use a simple programming language or a natural-language syntax.

— *Fourth-generation tools*: These are used to develop applications (sometimes with the cooperation of the development department and sometimes without) that tend to be run on PCs or powerful workstations.

By way of illustration, an individual classified in Category 2 would normally be provided, as a minimum, with access to either a dumb terminal or a PC. A dumb terminal would be adequate for someone who required access only to fixed-processing tools — that is, applications that were already fully developed, and that only required data to be input. A PC, however, would be needed by someone who required access to the more advanced flexible-processing tools such as spreadsheets. Such staff would usually need to attend a standard training course on the use of the tools. Support would be provided via a permanently staffed help desk because this type of user typically requires immediate assistance. There would be little need for any further guidance other than that provided by the training course.

Issue guidelines for different types of application

Guidelines for end-user applications should be defined to avoid constraining users. Applications should be classified by size, the number of users, the type of data they access, and so on. The classification can also serve to determine the level of inspection required to attain the systems department's 'seal of approval', discussed below.

An example of a matrix that can be used to classify end-user applications and to define the guidelines for their development is shown in Figure 5.4. In this example (which is based on

Figure 5.4 A matrix can be used to classify applications and define the guidelines for their development

	Data attributes	Application attributes	Project attributes	Associated guidelines
Class A (simple spreadsheet or database query)	• Personal • Non-strategic • Low-volume • Independent	• Personal • Standalone • Low complexity	• One to five workdays • No formal project management warranted	• Obtain authorisation • Use password • Back up data • Use common sense • Document as appropriate • Label the application and output reports
Class B (spreadsheet used on regular basis or database reporting program used by more than one person)	• Departmental • High-volume • Used by other programs	• Corporate • Used by more than one person	• Six to 20 workdays • Some project approval/project management warranted	Class A guidelines + • Do recommended control analysis • Document • Get 'seal of approval' for system security and so on
Class C (micro-based DBMS application, or complex spreadsheet, or simple spreadsheet used for critical decision-support)	• Strategic or sensitive • Used to update corporate database	• Complex • Uses non-recommended technology	• 21 to 40 workdays • Formal project approval/project management warranted • More than 40 days — system development standards apply	Class B guidelines + • Do compulsory control analysis • Do feasibility and cost-benefit analysis • Get agreement from development department

(Source: Based on an example from the Software Management Institute)

work done at the Software Management Institute), all end-user applications are classified into one of three classes, according to their attributes. An application is always allocated to the highest possible class. If, for instance, it had data and application attributes in Class A, and project attributes in Class C, it would be considered as a Class C application. Examples of the application types that might fall into each category are included in the matrix.

The guidelines associated with that class of application are then applied, to ensure that the user is not unnecessarily restricted. For the development of a simple spreadsheet, for example, categorised as Class A, the following guidelines would apply:

- Obtain appropriate authorisation to develop the application. Professionals often have implicit authorisation by virtue of their job level; clerical staff may have to request it from a supervisor.
- Use passwords to restrict access to the application.
- Always back-up both the data and the application.
- Document the application and procedures for using it.
- Label the application and any output it produces as 'Class A'.

For Class B and Class C applications, the guidelines would become progressively more stringent because the scope of such applications is wider and the risks are therefore greater. Classifying end-user applications in this way will ensure that they are evaluated prior to development, that appropriate development guidelines are followed, and that future users are aware of the standards to which each application was developed. In some organisations, however, it will not be practical to classify all applications, and the guidelines should be aimed at the riskier Class B and Class C applications.

Encourage users to seek the systems department's 'seal of approval'

Systems departments should encourage users to regard the concept of the 'seal of approval' as the equivalent of the acceptance testing they carry out on applications developed by systems staff. In providing its approval, the systems department should be looking for good documentation, comprehensive testing, consistent use of data, and so on. The systems department will also have the opportunity to add security, back-up, or systems features that the user may not have considered. Once the applications have been approved, any subsequent maintenance and enhancements can be carried out in a controlled manner either by users or by the development department.

Clearly, not all end-user applications will require the same level of inspection. Indeed, some will need none at all. If users are required to submit major applications for inspection, however, and if the process is conducted effectively, the end-user development environment can be effectively managed.

Provide guidance and support for end-user computing

Business users will adopt a growing variety of development tools to help them develop their own applications. As a result, the role of the development department will change in time. It will still be responsible for developing those applications that are too complicated for users to develop themselves, and those that span business functions, and for supporting users as they carry out their own developments.

In addition, it will be the systems department's responsibility to guide and support end-user computing. In Report 71, *Staffing the Systems Function*, we described in detail the skills required for this critical area. Further advice on providing support and guidance for end-user computing, from both within and outside the systems department, is given in the Butler Cox Foundation Position Paper *Information Centres in the 1990s*, published in February 1990. Moreover, users will demand better human/machine interfaces, based on windows and icons, and better advisory and help facilities to be built into applications. All this will place additional demands on the development department. By exploiting the emerging tools identified in this report, these demands should be well within its capabilities.

BUTLER COX FOUNDATION

The Butler Cox Foundation

The Butler Cox Foundation is a service for senior managers responsible for information management in major enterprises. It provides insight and guidance to help them to manage information systems and technology more effectively for the benefit of their organisations.

The Foundation carries out a programme of syndicated research that focuses on the business implications of information systems, and on the management of the information systems function, rather than on the technology itself. It distributes a range of publications to its members that includes Research Reports, Management Summaries, Directors' Briefings, and Position Papers. It also arranges events at which members can meet and exchange views, such as conferences, management briefings, research reviews, study tours, and specialist forums.

Membership of the Foundation

The Foundation is the world's leading programme of its type. The majority of subscribers are large organisations seeking to exploit to the full the most recent developments in information technology. The membership is international, with more than 400 organisations from over 20 countries, drawn from all sectors of commerce, industry, and government. This gives the Foundation a unique capability to identify and communicate 'best practice' between industry sectors, between countries, and between IT suppliers and users.

Benefits of membership

The list of members establishes the Foundation as the largest and most prestigious 'club' for systems managers anywhere in the world. Members have commented on the following benefits:

- The publications are terse, thought-provoking, informative, and easy to read. They deliver a lot of message in a minimum of precious reading time.
- The events combine access to the world's leading thinkers and practitioners with the opportunity to meet and exchange views with professional counterparts from different industries and countries.
- The Foundation represents a network of systems practitioners, with the power to connect individuals with common concerns.

Combined with the manager's own creativity and business knowledge, Foundation membership contributes to managerial success.

Recent Research Reports

56 The Impact of Information Technology on Corporate Organisation Structure
57 Using System Development Methods
58 Senior Management IT Education
59 Electronic Data Interchange
60 Expert Systems in Business
61 Competitive-Edge Applications: Myths and Reality
62 Communications Infrastructure for Buildings
63 The Future of the Personal Workstation
64 Managing the Evolution of Corporate Databases
65 Network Management
66 Marketing the Systems Department
67 Computer-Aided Software Engineering (CASE)
68 Mobile Communications
69 Software Strategy
70 Electronic Document Management
71 Staffing the Systems Function
72 Managing Multivendor Environments
73 Emerging Technologies: Annual Review for Managers
74 The Future of System Development Tools

Recent Position Papers and Directors' Briefings

Information Technology and Realpolitik
The Changing Information Industry: An Investment Banker's View
A Progress Report on New Technologies
Hypertext
1992: An Avoidable Crisis
Managing Information Systems in a Decentralised Business
Pan-European Communications: Threats and Opportunities
Information Centres in the 1990s

Forthcoming Research Reports

Assessing the Value from IT
Systems Security
New Telecommunications Services
Using IT to Transform the Organisation
Electronic Marketplaces

Butler Cox

The Butler Cox Foundation is one of the services provided by the Butler Cox group. Butler Cox is an independent management consultancy and research company. It specialises in the application of information technology in industry, commerce, and government throughout Europe and the rest of the world. It offers a wide range of services to both users and suppliers of information technology.

Butler Cox plc
Butler Cox House, 12 Bloomsbury Square,
London WC1A 2LL, England
☎ (01) 831 0101, Telex 8813717 BUTCOX G
Fax (01) 831 6259

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

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

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

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

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

Ireland
SD Consulting
72 Merrion Square, Dublin 2, Ireland
☎ (01) 760088/762501, Telex 31077 EI,
Fax (01) 767945

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

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

Spain and Portugal
T Network SA
Núñez Morgado 3-6°b, 28036 Madrid, Spain
☎ (91) 733 9986, Fax (91) 733 9910