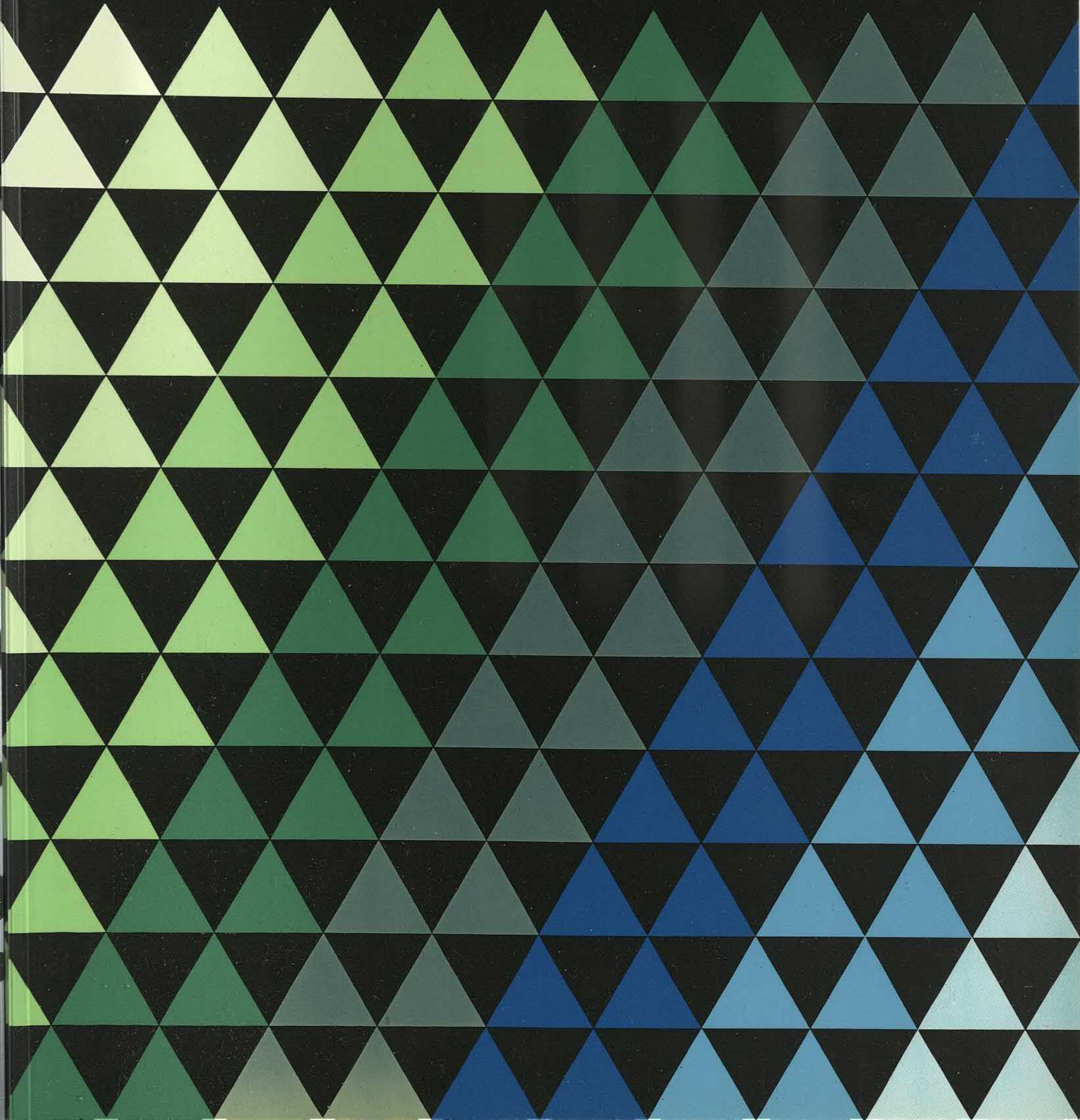
Computer-Aided Software Engineering (CASE)

BUTLER COX FOUNDATION

Research Report 67, December 1988



BUTLER COX
FOUNDATION

Computer-Aided Software Engineering (CASE)

Research Report 67, December 1988

Butler Cox & Partners Limited

LONDON
AMSTERDAM MUNICH NEW YORK PARIS

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

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

Computer-Aided Software Engineering (CASE)

Research Report 67, December 1988

Contents

Management Summary

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

Chapter 1

CASE: A new term to describe existing concepts

A recurring theme of Foundation Research Reports over the years has been to provide Foundation members with advice about how to improve the process of developing application systems. The last time we examined this subject in detail was in Foundation Report 57, *Using System Development Methods*, which was published in June 1987. In that report, we emphasised that no one development method was suitable for all types of development process. We also highlighted the fact that methods are an incomplete solution to systems development problems because, by themselves, they do not improve development productivity. We concluded that earlier report by saying that "Improvements in productivity come from using development tools to automate the activities required by the methods. Indeed, many methods are almost unusable without appropriate tools."

In the 18 months since Report 57 was published, that basic message has not changed. Since then, however, considerable media attention has been given to computer-aided software engineering (CASE), which is now being heralded as the solution to the application development problems that organisations have had for many years. At first sight, the concepts of CASE appear to be all-embracing and revolutionary. In theory, they are. The state of the art today, however, is much more modest.

Like so many 'hot' topics in the IT field, the term CASE has been used to describe something that is at present essentially simple and straightforward, but which has the potential to evolve into something much more wide-ranging. The temptation is to believe that the promised evolution is just around the corner. In fact, the evolution may not occur in the directions predicted and may take a lot longer than many pundits believe. An analogy is the development of 'office automation'. Ten years ago, grand theories of office automation were extrapolated from the basic word processing systems that were then coming into use. How many of those predictions have actually been fulfilled?

The reality today is that, by and large, the term 'software engineering' is used to describe development techniques that have been in common use for several years, and CASE tools are the development tools used to automate those techniques. Indeed, many of the tools described in Report 57 can now be described as CASE tools. At present, software engineering corresponds largely with structured analysis and design techniques, and CASE tools are the development tools used to automate those techniques.

This implies that many CASE tools (and the methods based on structured techniques that they support) apply only to a limited part of the applications software life cycle. As we pointed out in Report 57, development methods covering the whole of the life cycle (planning, analysis, design, programming, testing, implementation, and maintenance) do not yet exist. Thus, by definition, CASE tools covering the whole of the application life cycle do not yet exist either. However, the current generation of CASE tools is being extended, both to link back to the planning stage of the life cycle and to link forward to the programming stage. Thus, the output from analyst/designer workbenches (which are typical of mainstream CASE tools) can now be used as the input for code-generation tools. Although a few products are now beginning to address all stages of the life cycle, they do not yet support fully all the activities, and it will be several years before an integrated set of CASE tools exists, able to support fully the whole of the applications software life cycle.

DEFINITION OF CASE TOOLS

Although the term 'CASE tool' is today largely used to describe tools that support only the analysis and design stages of the software-development process, it is increasingly being used to describe any tool that supports any stage of the software life cycle. In this report, we use the term 'CASE tool' in this wider context. However, we specifically exclude programming languages, fourth-generation languages and other advanced system-building tools, and administration and project-management tools from the definition. Thus, tools with the following characteristics and facilities are included in our definition:

— Standalone tools based on PCs or engineering workstations. Typically, these tools provide graphics and text facilities for use by analysts and designers. In particular, the facilities are used to create and maintain the data-structure and activity diagrams required by structured development techniques.

— A dictionary that is used to hold the data and activity details about an application. Such a dictionary can be used in conjunction with a standalone analysis and design tool and will therefore be based on a PC or workstation. It may also be linked to a larger central dictionary and database management system that provides facilities for holding and managing the data for all stages of the software life cycle and for several separate, but related, applications. Often, the central dictionary and database management system will be based on a mainframe.

— Tools that ensure that the application design remains consistent as it progresses through the life cycle. Structured techniques require increasingly detailed diagrams to be developed. These tools ensure that no logical inconsistencies are introduced as the additional detail is added.

— Automatic code- and database-generation facilities that convert the output from the design stage to programs and database structures for the target hardware and software environment.

The colour plates on pages 5 to 8 show the types of graphical displays produced by analysis and design CASE tools.

A common feature of CASE tools is that they automate the techniques on which systems development methods are based. Thus, CASE tools are meant to be used by systems development staff — principally analysts and designers, but also to a lesser extent, by programmers.

It is important to distinguish between a CASE tool and an I-CASE or *integrated*-CASE tool. No full I-CASE tools yet exist, but when they do, they will cover the entire software life cycle, and will manage all information (data models, data dictionary, databases, activity models, and so on) for the entire applications portfolio. An I-CASE tool will use an integrated database management system and data dictionary to store and manage all of this information. It could also provide project-management facilities.

A BRIEF HISTORY OF CASE

The term CASE was first coined in the early 1980s, but has received general attention only from the mid-1980s. Although the acronym is new, the idea of automating parts of the systems development process is much older. Tools for generating code from decision and parameter tables have existed since the early 1970s, as has computer support for the production of textual descriptions of system specifications. Thus, the first generation of what are now known as CASE tools were (and still are) used to create program code from other pseudo languages. An example of one of these early tools is the Pacbase code generator, available from CGI Informatique, a French software supplier.

The next generation, and the first to be called CASE tools, appeared in 1984. These tools provide support for the analysis and design stages of the life cycle, in particular by providing graphics-based facilities for designing application systems. Initially, these tools provided little more than facilities for drawing and maintaining the charts and diagrams required by structured design techniques. The more sophisticated tools can now check the consistency of related diagrams. One of the earliest examples of this type of tool, and still one of the market leaders, is the Excelerator analyst/designer workbench, which is available from Index Technology Corporation.

It was at this point that methods suppliers became interested in providing CASE tools to support their development methods. Many of these have entered the market, including James Martin Associates with its IEF Information Engineering Facility, Arthur Young with IEW (Information Engineering Workbench), and Arthur Andersen with FOUNDATION Integrated Environment for Software Engineering. More recently, database software vendors have become aware of the potential for using their database and data-dictionary products as the basis for CASE tools, and they began, in 1987/88, to provide graphics-based interfaces to their products so they could be used in this way.

The first commercial integrated project-support environments (IPSEs) appeared at about the same time as second-generation CASE tools. To some extent, IPSEs fit uneasily into our definition of CASE tools. Rather than providing direct technical support, they provide a framework for project administration and management within which other CASE tools may be used. The majority of IPSEs available today support realtime and scientific software development and are distinct from IPSEs for business applications. The first business-oriented IPSEs appeared in the late 1970s; the most widely used today is Maestro from Softlab and Philips.

The third generation of CASE tools, which represents the state of the art today, is evolving from the two previous generations. In particular, automatic links are being built between analysis and design tools and code-generation tools, and a few products now provide a central dictionary used to store some information about the applications portfolio.

The characteristics of each generation, together with representative products, are shown in Figure 1.1.

THE GROWING DEMAND FOR CASE TOOLS

Our own survey showed that about 25 per cent of Foundation members are now using tools that correspond to our definition of CASE, and that a further 65 per cent are in the process of implementing CASE tools, or intend to introduce them by 1990. Different types of CASE products are supplied by different types of supplier. The more extensive tools, which cover several stages of the software life cycle, are often provided by partnerships between consultancies and specialist product developers. One example is IEF Information Engineering Facility, which was developed by Texas Instruments, a major US electronics manufacturer, and is supplied and supported by James Martin Associates.

Figure 1.1 Three generations of CASE tools

Characteristics of each generation	Typical products and their suppliers
First generation Appeared in the 1970s. Coverage — only one stage of the life cycle, usually programming. Simple and unsophisticated. Relatively low-cost. Mostly text input — little graphical support.	Pacbase code generator (CGI Informatique) Gamma code generator (Knowledgeware Inc) DELTA code generator (Delta Software Tools) VAX Cobol Generator (Digital Equipment Corp)
Second generation Products available from mid-1980s. Coverage — more than one stage of the life cycle, usually analysis and design. Originally simple graphical aids. Later, provided consistency checking within, and between, stages. Impact on quality of software. Usually single-user or single project/multi-user. Usually PC- or workstation-based; rarely fully mainframe-based. Little integration between stages.	Maestro (Softlab Inc and Philips Business Systems Ltd) Auto-Mate Plus (LBMS plc) Excelerator (Index Technology Corp) IEW (Knowledgeware Inc/Arthur Young Information Engineering Services) PDF (Michael Jackson Systems Ltd) SPEEDBUILDER (Michael Jackson Systems Ltd) ADT Yourdon Analyst/Designer Toolkit (Yourdon International Ltd)
Third generation State of the art in late 1980s. Moving towards linking front and back ends of the life cycle. First interface between analysis and design aids and code generators. First appearance of mainframe-based development dictionaries to act as repositories for applications data. Large investment and significant strategic impact.	IEF Information Engineering Facility (Texas Instruments/James Martin Associates) FOUNDATION Integrated Environment for Software Engineering (Arthur Anderson & Co Management Consultants) CASE* (Oracle Corp)

Database vendors are also increasingly entering the market for CASE tools, building on their experience of integrated data dictionaries and database technology and techniques for managing several logically linked data dictionaries. One example is Oracle Corporation, a major supplier of database management systems for DEC and IBM environments. This company has recently introduced a series of products (under the name CASE*) that cover several stages of the software life cycle.

As yet, the major computer manufacturers have not made a significant impact on the CASE-tool marketplace. Sometimes, as with IBM, this is because the market is still too small to attract them; sometimes, it is because they prefer to encourage third parties to provide products that fit into their hardware architectures. IBM is unlikely to produce a significant CASE product before 1990. Meanwhile, it is encouraging third parties to develop CASE tools that are consistent with its Systems Application Architecture (SAA).

DEC has been more active in this field and has already provided several products since the establishment of its CASE business centre early in 1987, in particular the Application System Development Environment, which focuses on Cobol generation, data dictionaries, and project management. DEC is also encouraging third parties to provide complementary products.

In general, the market for CASE tools is still relatively small, but it is growing rapidly. Many new suppliers are expected to enter the market in the next year or so. Rapid growth and the relative immaturity of the products will inevitably lead to rapid changes in the products available and in the structure of the supply-side of the market. Any organisation considering the implementation of CASE tools should consider the impact that these changes will have on its choice of CASE tools. (Our view on the most significant changes that are likely to occur is set out in Chapter 5 on page 33.)

PURPOSE OF THE REPORT

Clearly, at their present state of development, CASE tools are not the answer to every systems development problem. They are not appropriate for end-user development, where fourth-generation languages are more appropriate. They are not very suitable for maintaining existing systems developed without using CASE tools. And they provide only partial code-generation facilities, so high-level languages will continue to be used extensively for the foreseeable future. Thus, CASE tools will need to be used in addition to, and in conjunction with, other development tools.

Nevertheless, significant benefits in development productivity and software quality can be gained by implementing CASE tools successfully. The purpose of this report is therefore to answer the question "What are the real benefits of CASE tools and how can they be achieved?"

This report is aimed at managers within the systems function who are responsible for systems development productivity and software quality. It is relevant both to existing users of CASE tools and to those contemplating introducing them in the near future.

In Chapter 2, we describe the benefits that CASE tools can provide in terms of software quality and development productivity. These benefits will not be realised, however, unless the tools are used in conjunction with structured development techniques. CASE tools also need to be applied selectively. They are not suitable for all types of systems development. Chapter 3 shows how to decide when and where to use CASE tools to best effect. The procedures for selecting the most appropriate tools from those on offer are set out in Chapter 4. It is particularly important to select tools that match the existing development environment in terms of methods, hardware, and software. Furthermore, because different types of CASE tool address different stages of the software life cycle, it is important to select tools that provide support for the stages that cause most concern.

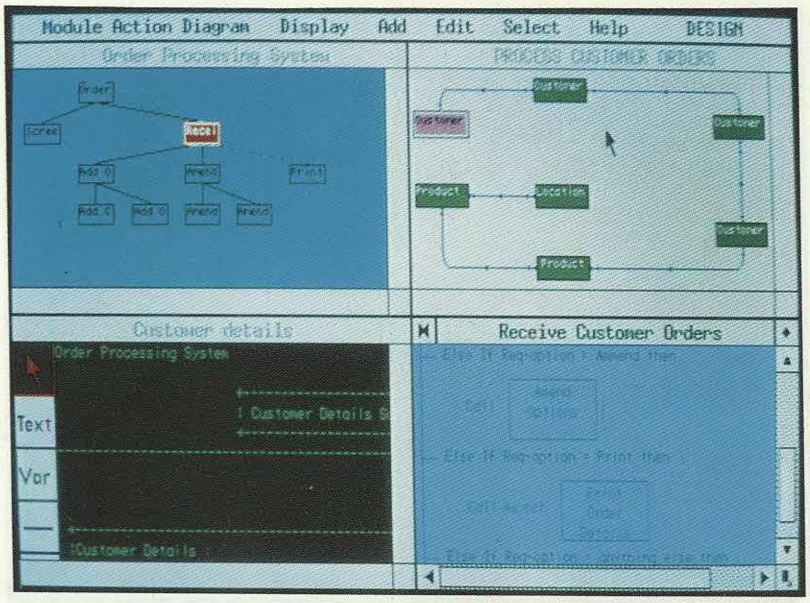
Finally, in Chapter 5, we provide guidance about managing the implementation of CASE tools. Not only is it important to introduce the tools in a controlled manner, it is also important to prepare for the future and to ensure that the significant investment made in methods and CASE tools will not be invalidated by impending technical developments.

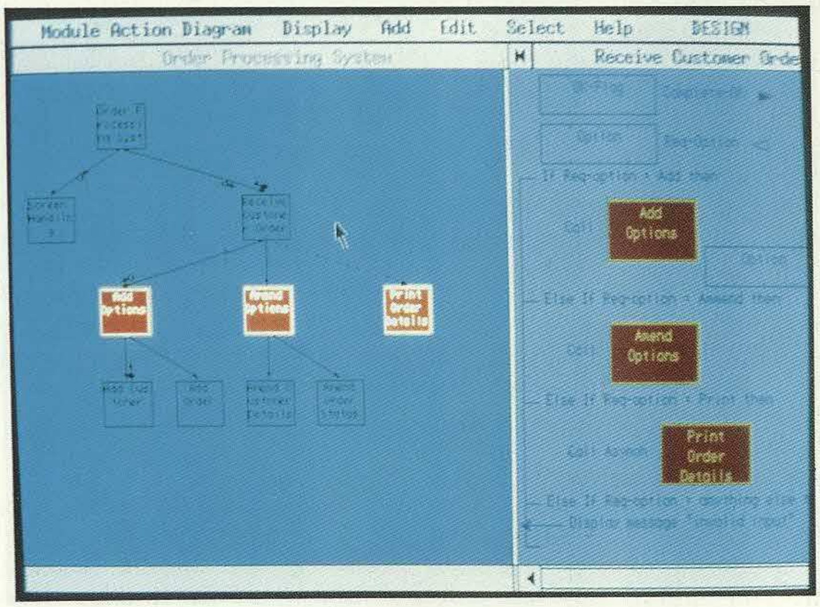
SCOPE OF THE RESEARCH CARRIED OUT

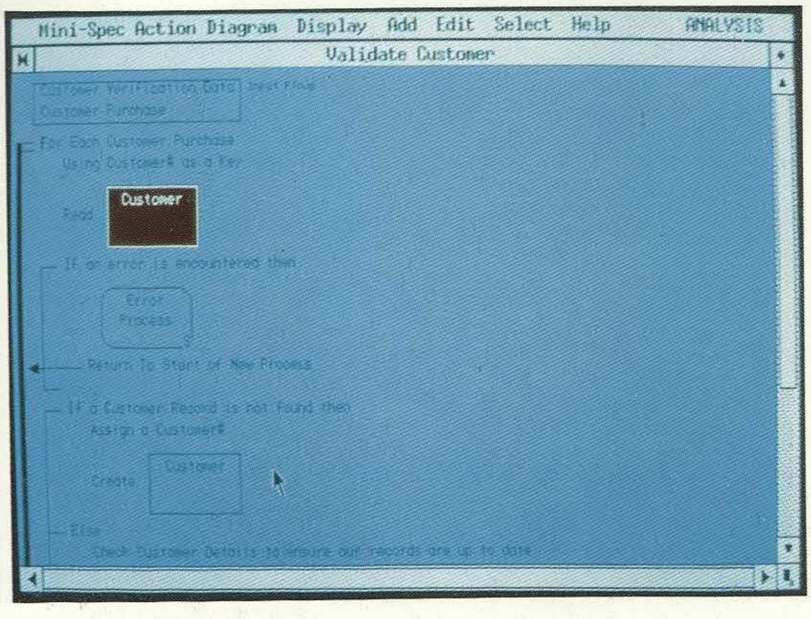
The report was drafted by Mary Cockcroft, head of Butler Cox's systems consultancy practice and an expert in systems development methods. The conclusions and recommendations of the report are the outcome of an extensive, worldwide programme of research carried out during the first half of 1988. Researchers included Richard Mugnaioni, a senior consultant based in Butler Cox's London office, who specialises in systems development issues, David Flint, a principal consultant and author of a previous Foundation report on advanced system-building tools, Frans Molhoek, manager of Butler Cox's office in Amsterdam, Lothar Schmidt, a senior consultant in the Munich office with extensive knowledge of the European software industry, and John Cooper, who runs the Foundation in Australia.

In order to understand fully the current state of the art in CASE tools and to assess the likely future developments, we conducted an extensive review of the CASE-tool marketplace. During this part of the research, we reviewed the published literature on CASE and software engineering in general. We also reviewed the product literature of 35 leading CASE-tool suppliers (both in the United States and in Europe) and conducted detailed interviews with each of these suppliers. We also sought the views of acknowledged experts on the subject, both through their publications and by interviewing a selection of them. In addition, the views and practices of Foundation members were gathered in face-to-face and telephone interviews in seven countries worldwide, and in the analysis of the responses from 150 Foundation members to the questionnaire sent out at the beginning of the research. This questionnaire asked Foundation members about their present use of, and future plans for, CASE tools.

Examples of graphical displays produced by analysis and design CASE tools

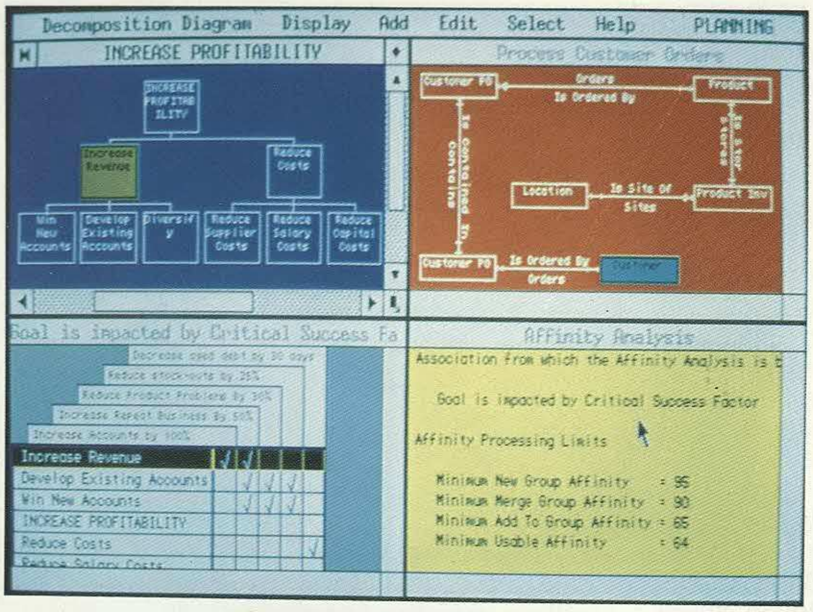





Mini-Spec Action Diagram Display Add Edit Select Help ANALYSIS
Validate Customer
Customer Verification Data
Customer Purchase
For Each Customer Purchase
Using Customer# as a key
Read Customer
If an error is encountered then
Error Process
Return To Start of New Process
If a Customer Record is not found then
Assign a Customer#
Create Customer
Else
Check Customer Details to ensure our records are up to date

(Source: Arthur Young's IEW)

Examples of graphical displays produced by analysis and design CASE tools

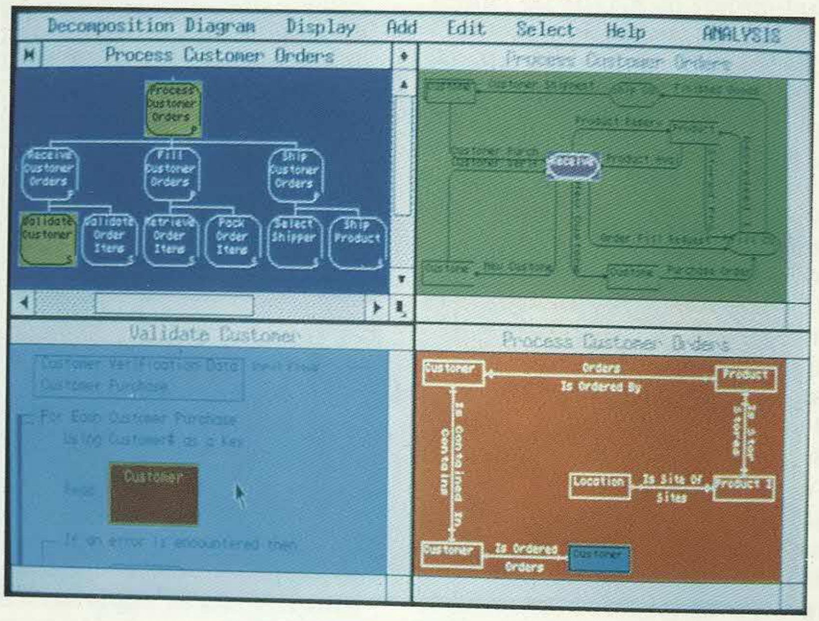


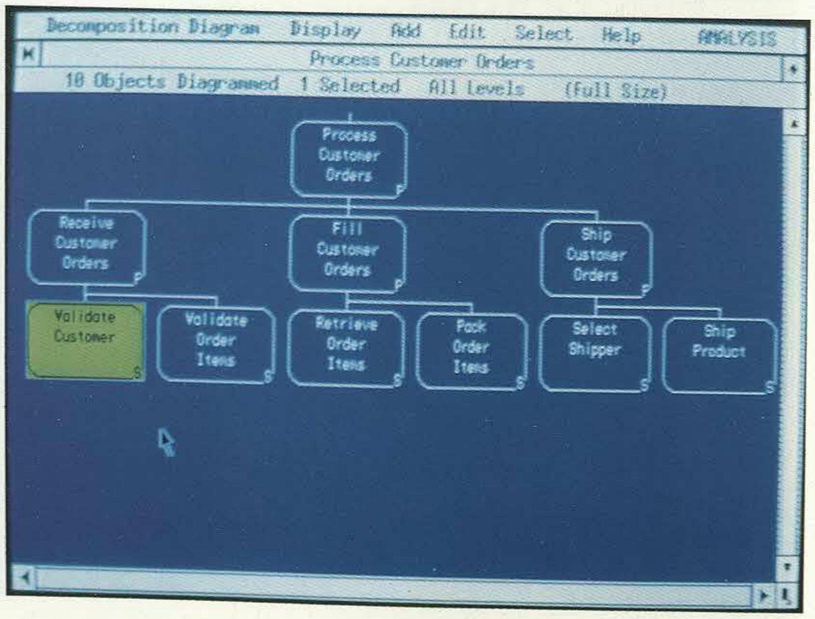
Association from which the Affinity Analysis is b

Goal is impacted by Critical Success Factor

Affinity Processing Limits

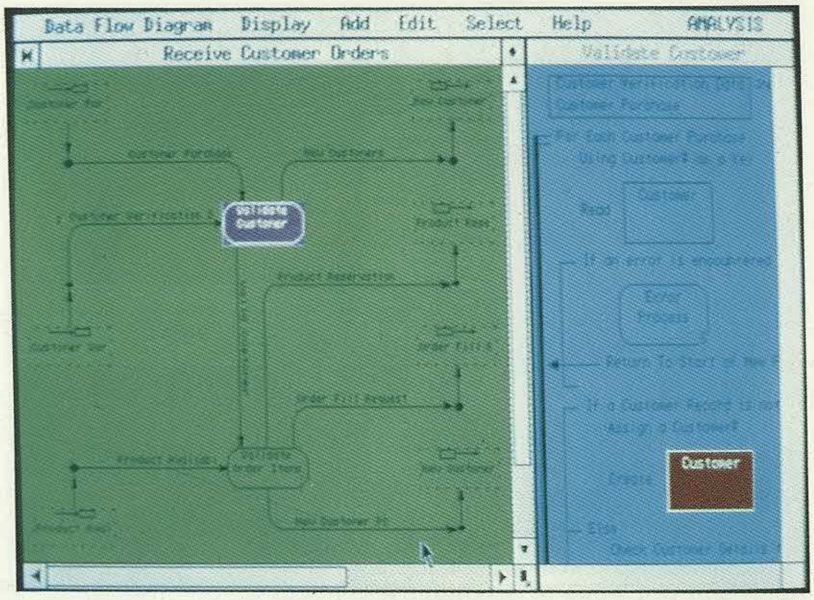
Minimum New Group Affinity = 95
Minimum Merge Group Affinity = 90
Minimum Add To Group Affinity = 65
Minimum Usable Affinity = 64

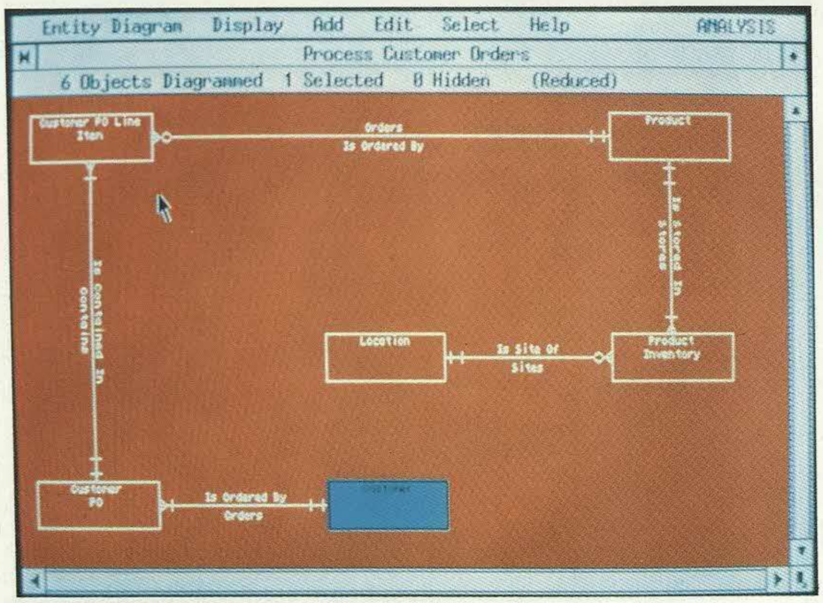


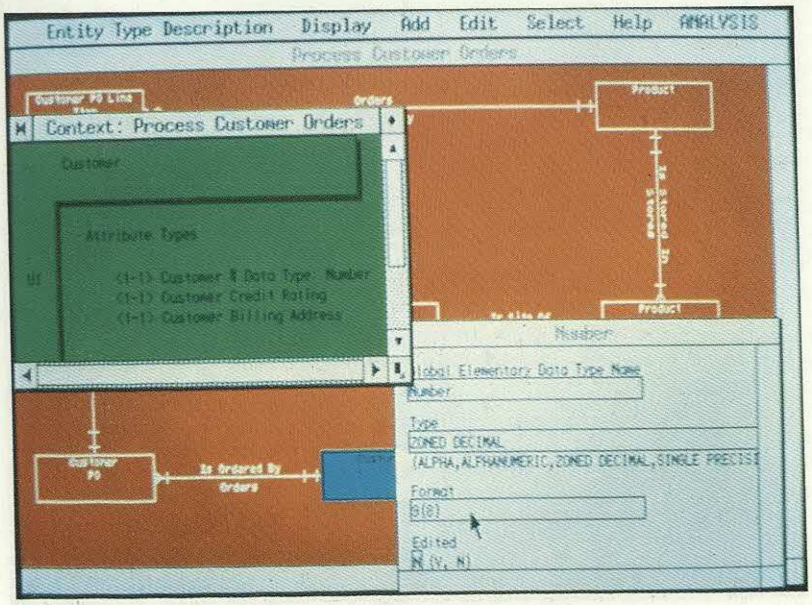


(Source: Arthur Young's IEW)

Examples of graphical displays produced by analysis and design CASE tools

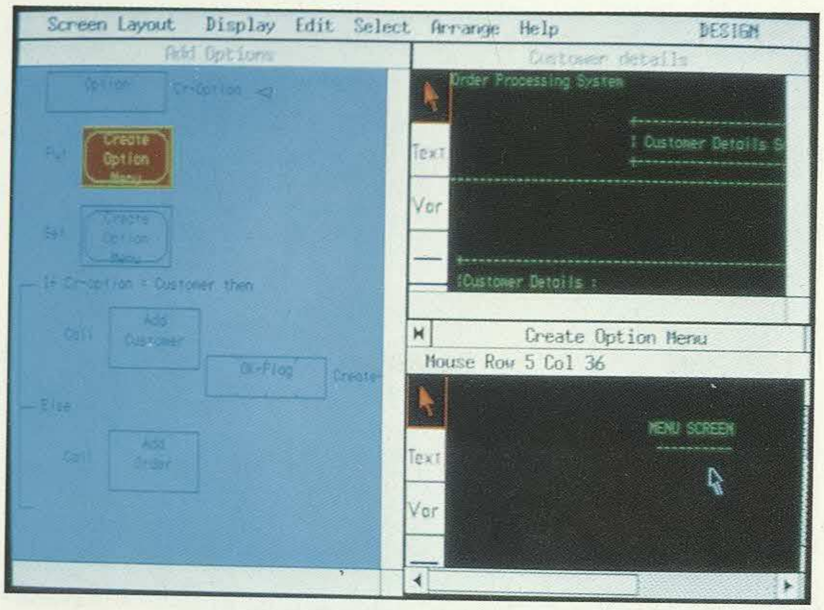


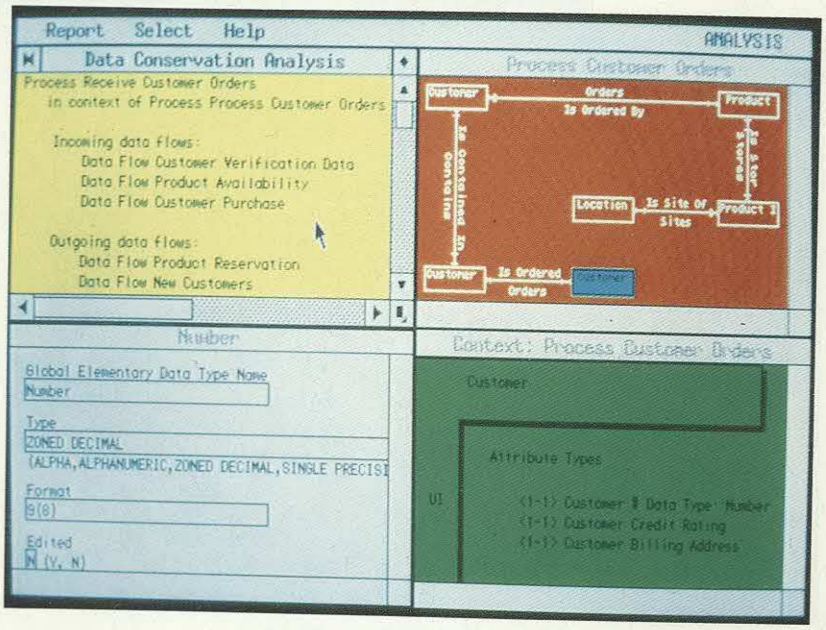



Entity Type Description Display Add Edit Select Help ANALYSIS
Process Customer Orders
Context: Process Customer Orders
Customer
Attribute Types
(1-1) Customer # Data Type: Number
(1-1) Customer Credit Rating
(1-1) Customer Billing Address
Global Elementary Data Type Name
Number
Type
ZONED DECIMAL
(ALPHA,ALPHANUMERIC,ZONED DECIMAL,SINGLE PRECISI
Format
9(8)
Edited
N (Y, N)

(Source: Arthur Young's IEW)

Examples of graphical displays produced by analysis and design CASE tools

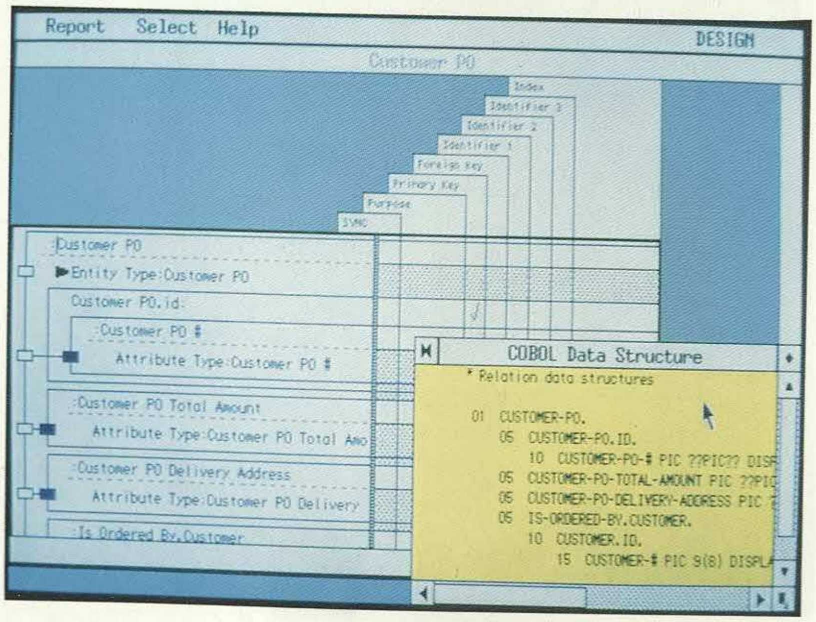




```
Process Receive Customer Orders
   in context of Process Process Customer Orders

  Incoming data flows:
     Data Flow Customer Verification Data
     Data Flow Product Availability
     Data Flow Customer Purchase

  Outgoing data flows:
     Data Flow Product Reservation
     Data Flow New Customers
```

```
Global Elementary Data Type Name
Number
Type
ZONED DECIMAL
(ALPHA,ALPHANUMERIC,ZONED DECIMAL,SINGLE PRECISI
Format
9(8)
Edited
N (Y, N)
```

```
Customer

    Attribute Types

UI      (1-1) Customer # Data Type: Number
        (1-1) Customer Credit Rating
        (1-1) Customer Billing Address
```



```
:Customer PO
  Entity Type:Customer PO
  Customer PO.id:
    :Customer PO #
      Attribute Type:Customer PO #
  :Customer PO Total Amount
    Attribute Type:Customer PO Total Amo
  :Customer PO Delivery Address
    Attribute Type:Customer PO Delivery
  :Is Ordered By.Customer
```

```
* Relation data structures

01  CUSTOMER-PO.
    05  CUSTOMER-PO.ID.
        10  CUSTOMER-PO-# PIC ??PIC?? DISP
    05  CUSTOMER-PO-TOTAL-AMOUNT PIC ??PIC
    05  CUSTOMER-PO-DELIVERY-ADDRESS PIC ?
    05  IS-ORDERED-BY.CUSTOMER.
        10  CUSTOMER.ID.
            15  CUSTOMER-# PIC 9(8) DISPLA
```

(Source: Arthur Young's IEW)

Chapter 2

CASE tools promise significant benefits

Early experience of using CASE tools suggests that they can improve both software quality and systems development productivity. In this chapter, we describe how CASE tools can provide these benefits. One organisation that has recognised the potential impact of CASE tools on productivity and software quality is NMB, a Dutch banking institution, whose experience is described in Figure 2.1.

SOFTWARE QUALITY CAN BE IMPROVED

There is now a substantial body of experience to show that using CASE tools to automate structured

Figure 2.1 CASE tools can improve development productivity and software quality

NMB

NMB, a leading Dutch merchant and retail banking institution, has achieved significant improvements in software quality and development productivity through the introduction of CASE tools. The bank has four commercial divisions: international financial services, domestic merchant banking, domestic retail banking, and stocks. It employs nearly 12,000 people and has assets of around Dfl80 billion ($40 billion). The central information services function, which provides computing services to all the divisions, consists of 350 people, 145 of whom are involved in systems development. The divisions employ a further 80 people who are dedicated to business systems planning and information analysis.

In 1980, the board of NMB, one of whose members had a specific interest in IT, realised that a consistent, professional approach to systems development was crucial to the future of both information systems and the business. As a consequence, NMB standardised on the IEF Information Engineering Facility, available from James Martin Associates, although the method was adapted to meet the bank's specific needs. The five standard phases of the Information Engineering method (information strategy planning, business-area analysis, business-systems design, technical design, and programming and implementation) were implemented, together with a specially constructed maintenance and support phase.

In 1985, the Maestro IPSE from Philips and Softlab was introduced to provide automated support for the development method. The implementation of Maestro was part of a strategy to improve software quality and development productivity. The strategy was based on achieving two goals:

- To use, throughout the organisation, a single, consistent development method that covers all phases of the software life cycle.
- To develop a fully integrated set of tools that can be used to implement, manage, and control the use of the method.

NMB realises that these are long-term goals and expects that it will take between seven and ten years to achieve them, and that they will require an investment of more than 80 work-years, in addition to the capital investment in the tools themselves. The board of NMB is fully committed to achieving the goals, a factor that makes this level of expenditure feasible. Maestro was chosen as the means for achieving the long-term strategy because NMB believes it provides an open framework that can be tailored and complemented with tools that closely match its own development environment.

The initial intention was to use Maestro as a development-library support tool, and as a means of providing communication and project-management facilities for project teams. However, NMB soon realised that, used in conjunction with tools for each development phase, Maestro had the potential to act as a framework to cover the entire software life cycle. The development environment now consists of 220 workstations, supported by eight networked Philips P7000 computers.

It has been necessary to make shorter-term tactical decisions as well. One of these is to use the IEW (Information Engineering Workbench) product, available from Knowledgeware and Arthur Young, for business-area analysis and business-systems design. NMB realises that it will have to replace this tool in order to achieve its long-term goals, but has decided to use it until tools become available within the Maestro framework to cover these development phases.

NMB is convinced that implementing CASE tools in this way has led to significant improvements in development productivity and in software quality. Since 1985, development productivity has improved by about 30 per cent, with most of the gains coming from automatic code generation, and from easier maintenance because of the better documentation that is now produced. The bank is now aiming for further productivity improvements of between 5 and 10 per cent a year. It also believes that the quality of its systems has been increased significantly because of consistent working practices, improved project management and documentation, and better communication between project-team members through the use of Maestro's electronic mail facilities. No attempt has been made to quantify the quality improvements, however.

NMB also believes that implementing the Information Engineering method, and the CASE tools to support it, has changed the way the systems function is staffed and organised. In particular, the increased emphasis on analysis and design has changed the skills requirements. No programmers have been recruited for more than a year; NMB now recruits and trains only analysts and designers. Overall, however, the number of systems development staff has not increased, although prior to introducing the CASE tools, the numbers had increased steadily for several years.

So far, the implementation of Information Engineering, Maestro, and the associated CASE tools has required 10 work-years of effort a year since 1984. NMB expects that this level of investment will continue for a further four years.

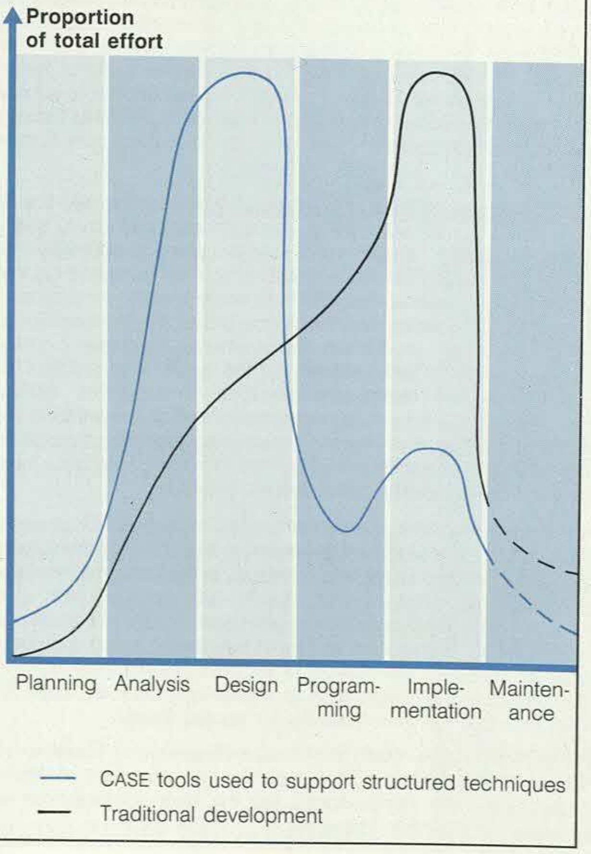
design techniques improves the quality of the applications software that is developed. Structured techniques improve software quality in two ways: they ensure that the application systems are a better fit with the business needs of their users, and they improve the technical quality of the systems by reducing the number of software errors.

BETTER FIT OF SYSTEMS TO THE BUSINESS

There are well-documented examples of systems being developed and never used because they fail to meet the business needs of the users for whom they were developed. CASE helps to overcome this difficulty because it encourages development staff to place a greater emphasis on the analysis and design stages of the software life cycle. In particular, CASE tools provide online interactive graphics facilities to support these stages. The effect of using CASE tools on the level of effort required at each stage of the life cycle is shown in Figure 2.2.

Bill Presley, Systems Development Manager at Glaxo Pharmaceuticals, a division of the multinational manufacturer of pharmaceutical products, confirmed that greater emphasis is now being placed on analysis and design. He believes that implementing Arthur Young's IEW has increased the proportion of the development effort devoted to analysis and design from 30 to 60 per cent. Using IEW, Glaxo Pharmaceuticals developed a computer-integrated manufacturing system on budget and to time, and with a noticeable reduction in the number of changes requested by users, and a corresponding increase in user satisfaction.

Figure 2.2 Use of CASE tools changes the level of effort required at each stage of the software life cycle



The interactive graphics facilities provided by CASE tools used at the analysis and design stages encourage a closer working relationship between developers and users, and allow users to be involved directly in the systems development process. One example of this was described to us by Lee Hawkins, software support manager at The Abbey National Building Society, a leading UK financial-services organisation. He has found that CASE tools provide a common language between developers and users, leading eventually to users participating directly in the development of the data and activity models required by structured techniques. This has reduced the misunderstandings that have often occurred in the past between developers and users, because the CASE tools encourage users to be specific. In addition, users find it easier to understand the design of systems because they no longer have to work through voluminous paper-based documentation.

The Department of Public Works of the Dutch Rijkswaterstaat, which is responsible for dykes, canals, and roads, reported a similar experience. This organisation has implemented the Excelerator analyst/designer workbench from Index Technology Corporation, and has found that the major benefit of this tool is that it reduces the effort at the design stage and results in higher-quality software. These benefits are achieved because Excelerator allows the software design to be presented to users early in the life cycle in a simple and understandable way.

BETTER TECHNICAL QUALITY

As well as improving the fit of systems to business needs, CASE tools help to improve the technical quality of software. This is an important benefit because the cost of correcting errors detected at later stages of the life cycle is much higher than correcting them at an earlier stage. A recent survey conducted by Barry Boehm in the United States found that errors in specifications cost 1,000 times more to correct after implementation than during analysis.

CASE tools help to ease this situation because they reduce the likelihood of technical errors both at the early stages of the life cycle (during analysis and design), and at the programming stage. By providing automated support for the structured techniques used during the analysis and design stages, CASE tools reduce the likelihood of mistakes being made. They also provide facilities for automatically checking the consistency both of related data-structure and activity diagrams, and of the more detailed diagrams used at the later stages of the life cycle. Prior to the availability of such facilities, development staff had to record manually the complex interrelationships and dependencies generated by structured techniques. The inevitable result was that mistakes were made and the technical quality of the resulting application system was reduced.

CASE tools also help to improve the technical quality at the programming stage. Several Foundation members confirmed that the use of CASE tools results in code that contains fewer errors than code written by average programmers.

DEVELOPMENT PRODUCTIVITY CAN BE IMPROVED

CASE tools can improve the productivity of development staff at each stage of the life cycle, although the most important benefits arise at the maintenance stage as a direct result of the improved quality of systems. Figure 2.3 shows the main types of benefit at each stage and indicates which may be measured and how.

At the analysis stage, it is possible both to improve productivity and to reduce the elapsed time. The greatest improvements at this stage will be achieved where CASE tools are used to automate development methods that have previously been implemented manually. Savings of between 10

Figure 2.3 CASE tools provide productivity and quality benefits at each stage of the software cycle

	Productivity		Quality	
Life-cycle stage	Nature of benefits	Measure	Nature of benefits	Measure
Planning			Combination of methods and tools can be used to support strategic systems planning (although this is rarely done in practice).	Long-term assessment of fit of systems with the business.
Analysis	Graphical support for modelling, leading to productivity improvements of up to 50 per cent.	Work-days to produce similar analysis with and without CASE.	Much more emphasis on this stage. Interactive modelling will improve communications with users.	Assessment of user satisfaction. Reduction of change requests at implementation and maintenance stages.
Design	Graphical support and transfer of data from analysis stage may reduce staffing levels by up to 50 per cent.	As above.	Consistency checking between analysis and design stages will reduce design errors.	As above.
Programming	Better logical and physical design, data transfer, and code generation, leading to potential productivity improvements of up to 100 per cent.	Number of lines of code per day.	Code generators likely to reduce coding errors. Transfer of data from design stage will reduce interpretation errors.	Number of errors detected during testing. Reduced effort in implementation.
Implementation	Potential reduction in effort and timescale due to greater emphasis on analysis/design.	Length of time from end of programming stage to final acceptance. Level of effort involved.	Better quality of analysis and design will have an impact on implementation.	
Maintenance and enhancement	Lower overall maintenance effort.	Ratio of maintenance to new development.	Re-usability of code and designs will improve quality of enhancements.	Effort required to maintain software.

and 30 per cent can usually be achieved. The experience of a manufacturing company demonstrates that CASE tools can produce productivity gains in both the analysis and design stages (see Figure 2.4). The reduction of 20 per cent achieved by this company in the time required between analysis and implementation is representative of the average gains achievable at these stages.

CASE tools can also improve productivity at the logical and physical software design stages. The improvements result from the on-screen graphical representation of data and activity diagrams, and from the ability to transfer information automatically from the analysis stage to the design stage. Finalising the design of a business application usually involves many iterations before the users approve the design. Using structured design techniques requires the data and activity diagrams to be amended at each iteration, which, done manually, is a time-consuming and error-prone process. The graphical and automated documentation facilities provided by CASE tools reduce the effort involved in producing and modifying systems designs. These facilities also improve the interaction between developers and users, thus reducing both the effort and the elapsed time required between starting the analysis stage and completing the logical design of a system.

Major productivity gains can also be achieved by using CASE tools (particularly automatic code generation) at the programming stage. A major French-owned multinational company involved in the chemical and pharmaceuticals industries found that, through the use of code generators, its code generation rate rose from 117 to 260 lines per person per day over a five-year period. (The industry average in France is 60 lines per person per day.)

Productivity improvements in the implementation and maintenance stages arise from the higher-quality designs produced when CASE tools are used at the analysis and design stages. The increased emphasis on analysis and design, and the improved working relationship between developers and users, help to reduce the number of software changes required during the implementation stage. Thus, CASE tools reduce the need to enhance applications soon after they are implemented in order to meet user requirements that were missed or misinterpreted at the analysis stage.

The improved technical quality obtained from using CASE tools also reduces the number of software errors and, hence, the amount of maintenance required after implementation. For example, Télémécanique, the French manufacturing group, has found that the code generated by the Gamma code generator (from Knowledgeware) is more efficient and error-free than the average manually produced code. CASE tools also automatically generate complete and consistent software documentation, thus making it easier to maintain systems after they have been implemented. Improved quality, greater consistency, better documentation, and fewer program changes mean that both the level and the difficulty of maintenance are reduced, thereby reducing the proportion of development effort required for maintenance activities.

The improvements in productivity arising from the better-quality software developed with CASE tools are difficult to quantify, however. There are three potential measures. The first is to measure the reduction in the number of changes requested by users during the development and implementation stages. The second, which is relevant to technical quality, is to measure the number of program errors discovered during the implementation stage. The third, which is concerned with overall quality, is to measure the reduction in maintenance work carried out by the development function. The first two measures apply to the development stages of the life cycle; the third becomes apparent only after applications have been running for some time.

Many Foundation members believe that using CASE tools at the analysis stage substantially reduces the number of changes requested by users both during development and after implementation. Few of them, however, have quantitative evidence to support this belief. One company that did provide data was the manufacturing company quoted in Figure 2.4: it found that the proportion of development effort devoted to maintenance work decreased from an estimated 70 per cent to 45 per cent. Another was a French multinational company in the chemical industry that found that, through its use of CGI Informatique's Pacbase for detailed design and implementations, the proportion of its effort devoted to maintenance remained constant at around 33 per cent, although its applications portfolio increased from 2 million to 6.2 million lines of code.

BENEFITS DEPEND ON THE USE OF METHODS

Many CASE tools have been designed to support the structured analysis and design techniques that form the basis of several proprietary development methods. It is theoretically possible to implement these methods without CASE tools, but, in practice, the volume of manually created paperwork that is required makes it impossible to use them

Figure 2.4 CASE tools can improve productivity at both the analysis and design stages

A MANUFACTURING COMPANY

Between 1984 and 1985, this manufacturing company (which wishes to be anonymous) examined the feasibility of implementing structured systems development methods. The programming department was already using Jackson structured programming. The conclusion reached was that, owing to the manual effort involved in introducing methods, some sort of support tool was required if methods were to be implemented successfully. As a result, the company installed the Maestro IPSE from Philips and Softlab. Initially, a 24-terminal system was used for team management, for preparing textual specifications, and for CASE development. Two further systems have now been installed. In addition, LBMS's Auto-Mate Plus tool has been used to support the LSDM method (also from LBMS) used by the company.

This company believes that it has been successful in implementing development methods only because it has had the tools to support the methods. Indeed, the implementation of LBMS's Auto-Mate workbenches has actually promoted the use of LSDM. Furthermore, it believes that the partnership of CASE tools and development methods has improved both productivity and quality.

Productivity at the maintenance stage has increased because of the improved quality of the systems now being maintained. Without methods and tools, the company estimates that it would be using 70 per cent of its development resources on maintenance activities instead of the 45 per cent actually used today. The methods and tools also deliver more functionality and allow more applications to be developed by fewer development staff. Productivity has also been improved because the CASE tools facilitate better interaction between developers and users and speed up the process of changing specifications. This has resulted in a 20 per cent reduction in the elapsed time from the feasibility stage to implementation. Quality has also improved because of the improved interaction between developers and users. Some users have become much more involved in the development process and have become more supportive of the systems department. This has contributed to users feeling that they 'own' the projects and systems, and has made the task of the systems department much easier.

successfully. The result is that the productivity, and morale, of development staff is reduced. As a consequence, the use of structured techniques has been discredited in many organisations; the benefits of consistency that result from using these techniques have been hidden by the difficulty of implementing them.

Conversely, it is not possible to obtain the full benefits from CASE tools unless they are used in conjunction with the method they were designed to support. The rigour imposed when such a method is implemented in conjunction with a CASE tool leads to the improvements in quality identified earlier. In turn, the improved quality leads to major gains in productivity during the programming, implementation, and maintenance stages of the life cycle.

In addition, the full life-cycle support promised by I-CASE tools can be achieved only if integrated development techniques are used consistently throughout the whole life cycle. This helps to explain why full I-CASE tools are not yet available, because, as we explained in Report 57, there is as yet no integrated set of development techniques that covers the whole of the life cycle. The major problem area concerns the difficulty of automatically translating logical designs into physical designs.

The implication is that many CASE tools must be selected to support particular proprietary development methods. Others, however, particularly analyst/designer workbenches, can be configured to support different methods. The decision to choose a particular combination of methods and tools is possibly the most difficult and strategically important decision that a systems director will have to make. Once a method and tools to support it have been implemented, the organisation will be committed to using the method for many years to come. The investment in training and in setting up the procedures to use the method effectively will make it very difficult, and expensive, to change to a new method. Thus, the effects of an inappropriate combination of methods and tools will be evident for a long time. The first stage in making the right decision is to decide where the methods and tools can be used with most effect.

Chapter 3

Deciding when and where to use CASE tools

The first task in selecting the types of application for which CASE tools are appropriate is to identify the stages of the life cycle where most problems occur and where CASE tools could help to improve productivity and quality. It is also essential to ensure that the existing development methods are being used effectively; there is no point in using CASE tools if they are not. The limitations of CASE tools must also be considered. There are some development situations where CASE tools are not the most appropriate answer.

CLARIFY THE OBJECTIVES FOR USING CASE TOOLS

Applying CASE tools in an indiscriminate way may bring some isolated benefits, but it will probably not provide value for money. Clarifying the objectives before setting out to implement CASE tools will help the organisation to ensure that the areas of greatest concern to the business are addressed, and that realistic expectations are established for the benefits.

ADDRESSING PROBLEMS OF MAJOR CONCERN

Most systems managers are well aware that systems development continues to present major problems. Few, however, can accurately describe what all of the problems are, let alone define how they arise. They know that there is a growing gap between the rate at which systems can be developed and the demand from users for new systems. They also know that user requirements are very often not met by the systems that are developed. Before the problems can be resolved, it is necessary not only to identify what the problems are but also where and how they arise.

Many software experts insist that the problems are caused by the low productivity of development staff, and that the solution is to concentrate on developing applications more quickly. Without doubt, tools that automate and speed up the production of designs and code will increase development capacity. The real problem, however, results from an inability to interpret and meet user needs and to deliver error-free code.

It is well known that a significant proportion of all business software produced by systems departments is either significantly reworked during development or never used. The development manager of Lend Lease, an Australian property and financial-services group, told us that difficulties experienced by the company in identifying and analysing user requirements (before the introduction of CASE tools) meant that the first six months of live operation were little more than a final systems development stage. The main cause of this type of problem is that mistakes are made at the requirements-analysis stage. A problem that at first sight appears to be a programming problem can, in fact, be caused by poor analysis and design.

Thus, the first step in planning to implement CASE tools should be to examine each stage of the life cycle to identify where the causes of the most difficult problems are. In carrying out this analysis, it is important to remember that the root cause of a problem may lie at a much earlier stage in the life cycle. For example, an excessive proportion of effort spent on maintenance may be caused by mistakes originally made at the analysis and design stages. The life-cycle stages that are the root causes of the problems should be identified, and CASE tools should then be chosen to support these stages.

CASE tools can be applied with benefit at most stages of the software life cycle. In general, however, the greater the number of stages covered by a CASE tool, the higher the cost of implementing the tool and the longer the payback period. For a development department with 50 staff, implementation costs, including the cost of hardware, software, and initial training, are typically about $20,000 per head for workstation tools covering only the analysis and design stages, and about $35,000 per head for an integrated development environment that includes a mainframe-based development dictionary, that covers several life-cycle stages, and that can be used for different projects. The cost can be much higher, however. We know of one organisation that has implemented standalone analysis and design tools for 30 development staff at a cost of $46,500 per head. (More details can be found on page 29.)

These high levels of investment will mean that many organisations will wish to phase the introduction of CASE tools progressively to cover a larger number of life-cycle stages. Identifying those stages in which the main problems occur, and placing them in order of priority, will make it possible to plan a phased implementation of CASE tools. Prospective tools can then be evaluated against the objectives set for each stage of the life cycle.

IDENTIFYING OPPORTUNITIES FOR IMPROVING PRODUCTIVITY AND QUALITY

Organisations considering the implementation of CASE tools find themselves in many different situations. The choice of tool, and the method and timescale for implementing it, all depend on the situation into which the tool is introduced. The first step should therefore be to evaluate the maturity of the development environment (in terms of the use of structured techniques and development standards, for example), the productivity currently being achieved on development projects, and the quality of existing systems.

Evaluate the maturity of the development environment

The existing level of maturity of the systems development function is a major influence on how best to implement CASE tools. In particular, the current development environment determines many of the constraints that will delay the introduction of CASE tools, and it has a direct bearing on the level of training that will be required.

The maturity of the development environment applies both to the level of experience and expertise in using development techniques such as entity modelling and data analysis, and to the use of other types of enabling software such as relational database management systems and fourth-generation languages. Other factors that will influence the way in which CASE tools are implemented are the complexity and size of applications, the size of the development department, and the range of development skills (assembler programming, and so on) in use.

Our research indicates that, if CASE tools are introduced into an environment where development techniques are not already practised, it takes up to a year for development staff to become fully proficient in using the tools, compared with a matter of weeks when techniques are already used. Most of the training requirement arises from the need to teach development staff how to use the structured techniques. Lend Lease, for example, expects new recruits to take a year to become proficient, with most of the training effort being devoted to teaching techniques. On the other hand, development staff at Lloyd's of London, a major UK insurance underwriting group, required relatively little training when this organisation implemented CASE tools. Colin Talbot, Manager, Office Systems, Development Department, told us that this was because a method (LSDM from LBMS plc) was already being used by the development teams. The use of this method has also reduced the time required for new recruits to become fully productive because Lloyd's now favours people who are already familiar with the LSDM method.

The maturity of the development environment also has an impact on the amount of preparatory work needed to reorganise the systems department for the implementation of CASE tools. If the department has either multiple standards and methods of working, with little control, or no formal working standards at all, it should implement either a single agreed set of standards or new standards. Failure to do this will not make the implementation of CASE tools impossible; it will merely mean foregoing an opportunity to rationalise working practices in order to achieve the maximum benefit from implementing such tools.

Assess existing productivity and systems quality

Most automation aids for systems development are typically sold on their potential to deliver productivity gains. CASE tools, however, are also expected to improve the quality of the software developed by using them. To assess the benefits of using CASE tools, it is therefore necessary to compare 'before' and 'after' levels of development productivity and software quality. In turn, this means that organisations contemplating the introduction of CASE tools should quantify their present levels of productivity and software quality by introducing some kind of measurement programme, to .provide a basis for subsequent comparisons.

Establishing such a programme is not easy. There is much disagreement about what measures are appropriate, and how to make them, and no universally agreed standard measure yet exists. The number of lines of code produced by a programmer in a given period provides an indication of the efficiency but not the effectiveness of software, which includes aspects of quality. Another commonly used measure is based on counting function points, although the usefulness of this technique depends on the type of application being developed. One function point may represent a large number of lines of code in a database application, but only a few in an application that uses complex algorithms. We believe that QSM's Productivity Index (PI) used in Butler Cox's Productivity

Enhancement Programme is a better way of measuring development productivity. No method of measurement is perfect, however, and measuring lines of code or function points is better than nothing.

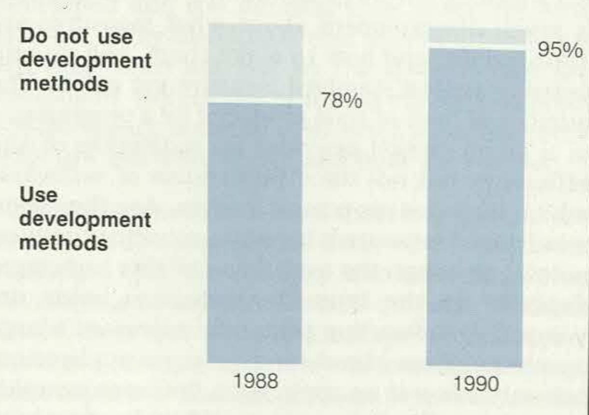
Less formal methods of comparison can be useful where formal measures are impractical. For comparing productivity, for example, two potential measures are the effort and elapsed time required to develop various project types.

Comparing quality is more difficult than comparing productivity because the comparison often depends on intuitive, subjective measures. Many managers feel that an implementation 'has gone well' or 'has been a disaster', based on their relationship with the users of the software. This type of intuitive judgement may convince some managers, but a quantitative measure, no matter how approximate, would be better. One possible way of quantifying software quality is to count the number of queries, change requests, and complaints made by users during the implementation stage. Another is to measure the proportion of overall development effort devoted to maintenance and minor enhancements.

Evaluate the effectiveness of existing methods

To be successful, many CASE tools must be used in conjunction with a development method that is based on structured techniques. Figure 3.1 shows that many of the Foundation members who responded to our survey have already implemented such methods and that most of the remainder plan to do so by 1990. It is essential, however, that the methods are being used effectively. Unless they are, it will not be possible to use CASE tools effectively. The effectiveness of development methods can be assessed against three criteria, each of which is equally important.

Figure 3.1 Use of development methods by Foundation members will approach 100 per cent by 1990



(Source: Survey of Foundation members)

First, the methods used must meet the requirements of the systems department. This can be judged by assessing whether, together, the methods support all stages of the software life cycle and all the development processes being used. As we pointed out in Report 57, no single method will support every stage of the life cycle, nor every development process. Some are suited to traditional large-systems development where project management is all-important; others are more applicable to the rapid, iterative style of development that is becoming more prevalent today.

Second, the methods should be used extensively throughout the systems development function. If they are used by only a few project teams or for some types of applications, they may not be as effective as they could be if they were used more widely. The widespread acceptance and use of methods may, for example, be inhibited because development staff perceive them as requiring unacceptable levels of manual intervention. Until the methods are used effectively, it will not be possible to make the most effective use of CASE tools.

The third criterion is that the methods should provide tangible benefits in terms of software quality. If they do, their credibility with managers will be established. If they do not, a great deal of effort will be needed to bridge the credibility gap.

The more effective the development methods are in terms of these criteria, the easier it will be to implement CASE tools and use them effectively. However, even if the methods are shown to be not particularly effective, it is invariably better to introduce CASE tools to support the existing methods. The only exception would be where the credibility gap is very wide indeed. Introducing a new method at the same time as CASE tools would be the most expensive element of implementing CASE.

RECOGNISE THE LIMITATIONS OF CASE TOOLS

CASE tools are still relatively immature, and it will be some time before they have developed to the stage where they can achieve their full potential. To date, relatively few major operational applications have been implemented using CASE tools, and it is widely recognised that present-day tools have substantial limitations. The effects of these limitations can, however, be minimised if they are recognised and managed.

UNDERSTANDING THE NATURE OF THE LIMITATIONS

The CASE tools available today have three important limitations: they do not support the complete software life cycle, they provide little support for the maintenance of existing systems originally developed with non-CASE tools, and they are not suitable for direct use by business users.

CASE tools provide limited life-cycle support

Most CASE tools available today cover either just the front-end analysis and design life-cycle stages, or just the programming stage. A few, such as Texas Instruments' IEF Information Engineering Facility, have begun to address the complete life cycle, but they do not yet fully support all the stages. The types of CASE tools available today, and the extent to which they support individual stages of the life cycle, are described in Chapter 4.

The limited coverage of existing CASE tools means that user organisations need to integrate tools from different suppliers if they are to create a CASE environment that covers the whole of the life cycle. However, several suppliers are working to link their products to those from other suppliers, the aim being to create a combination of products that covers several life-cycle stages. One example is Arthur Young's interface between its IEW product and Knowledgeware's Gamma code generator. Users of CASE products should be aware, however, that even when an interface between two products has been announced, it is often not as comprehensive as the suppliers' literature may indicate. For example, at the time of writing, the interface between Index Technology's Excelerator analysis and design workbench, and Pansophic Systems' Telon code generator transfers only screen layouts and report layouts. Details about processing logic and rules still have to be transferred manually.

Many other interfaces between CASE tools will be announced in the near future, but the type of information that can be transferred, and the consistency of the user interface, will both be restricted. Sometimes, user organisations may have to create the required interfaces themselves, but this could lead to problems of internal support for the interfaces and to difficulties with the suppliers when software problems have to be resolved.

Two contrasting examples of how organisations are tackling this problem are provided by Lend Lease in Australia and a British company. Lend Lease currently uses IEW for analysis and design, and the Corvision application generator from Cortex, a US supplier of software-productivity tools, for construction. Lend Lease wants to establish a link so that information can be transferred from the analysis and design stages to the programming stage. Knowledgeware and Cortex have now reached formal agreement on establishing a link between their products, avoiding the need for Lend Lease to do so. At the other end of the scale, we know of one British company that is considering building a link between Softlab's Maestro and Oracle's database management system. This is an example of linking two products that are used for different purposes rather than for different stages of the life cycle.

CASE tools offer little help for maintaining existing systems

Although CASE tools can be used to maintain software developed by the tools themselves (by re-using or modifying existing designs), they provide little support for the maintenance of existing systems developed originally with non-CASE tools. CASE tools will offer significant help in the maintenance of existing systems only when they support reverse engineering — the process whereby system designs are extracted automatically from existing programs and are used as the basis for enhancing and maintaining them. This requires tools that are able to read existing code and data structures, and have the intelligence to extract the underlying systems design.

Current support for reverse engineering is mainly in the form of products that process existing unstructured code and convert it to structured code. Recent products from Bachman Associates have started to take a more data-oriented approach, allowing some existing physical IDMS designs to be re-engineered into logical designs ready for 'forward engineering' into DB2 databases. These products are still at an early stage of development, however, and, at present, they are available only in the United States.

At present, reverse-engineering products are limited to restructuring code and data. We have not been able to identify any products that are able to extract a full systems design from existing software. Without doubt, there is a need for such tools. Until they exist, CASE tools will be of limited value in maintaining, for example, the estimated 77 billion lines of Cobol code in IBM-based systems alone. We believe that products able to carry out true reverse engineering are unlikely to be available until the mid-1990s.

CASE tools are unsuitable for business users

Although CASE tools can help to involve business users in the development process, they have not yet been developed to the stage where they can be used directly by them. Indeed, CASE tools may never reach the stage where they can be used exclusively by users, because their use will continue to require knowledge and experience of structured development techniques. In the foreseeable future, expertise in structured techniques will remain firmly with professional development staff. Users will, however, become more involved in the development process. In particular, the interactive graphics facilities available with CASE tools will mean that users will become familiar with the diagramming concepts that are an integral part of structured techniques. Indeed, as we have seen, some of the most significant benefits of CASE tools will arise from the ability of developers and users to work together at the analysis and design stages of the life cycle.

For this process to work, however, users need to take an active role in the development of systems. The benefit of such involvement is illustrated by the experience of Lloyd's of London. This organisation believes that systems will serve the needs of the business better if users understand, interpret, and help to modify the graphical representation of the system designs. The graphical facilities of the CASE tools in use at Lloyd's allow users to do these things.

AVOIDING THE LIMITATIONS OF CASE TOOLS

The limitations of CASE tools can be avoided by using other types of tool where appropriate, and by restricting the use of CASE tools to new developments and major enhancements.

Use non-CASE tools where appropriate

Regardless of the claims made for CASE tools, they are not the answer to all systems development problems and should not be seen as such. Other types of tool will be required as well to cope with the many different aspects of development, both within and outside the systems department. Thus, tools such as third- and fourth-generation languages and other prototyping tools will still be required after the implementation of CASE tools.

Each type of tool should be used for the job for which it is best suited. For example, applications developed by business users, and by the iterative development process (which requires rapid iterations through the design stage and, hence, relies on prototyping), both need fourth-generation languages and other prototyping tools such as screen-layout and report generators. Figure 3.2 shows how CASE and other types of development tool can be used to support different development processes.

The development processes shown in Figure 3.2 were described in detail in Foundation Report 57, *Using System Development Methods*, and are summarised below:

— The conventional development process has been used for many years for the development of commercial and business applications. Typically, the work is subdivided into well-defined steps or phases, with the workflow being controlled and monitored by formal project-management techniques. More recently, proprietary systems development methods have been used to standardise the tasks carried out during one or more of the life-cycle stages.

— The iterative development process is more appropriate for applications where the users' requirements are less easy to specify and where the scale of the application is small enough to allow a prototype to be built and revised quickly either by using advanced system-building tools such as Natural, Focus, Mapper, or Linc, or by using CASE tools that provide screen-painting and dialogue-definition facilities, such as IEW and IEF Information Engineering Facility.

— The small-systems development process is appropriate for new small systems and for small enhancements to existing systems. Typically, small systems take less than nine months' elapsed time to develop, and require no more than about two to three years of effort. The small-systems development process covers the same range of applications as the conventional development process.

— End-user development refers to applications developed entirely by users. These applications provide limited functionality and are usually very small systems designed to meet an

Figure 3.2 Different development tools are appropriate for different development processes

Development process	Development tool				
	CASE tools	Fourth-generation languages	Application generators	Application packages	High-level languages
Conventional	★★	★	★★	★	★★
Iterative	★★	★★	X	X	★
Small-systems	★★	★★	★★	★★	★★
End-user	X	★★	X	★	X

★★ Good support
★ Reasonable support
X Not suitable

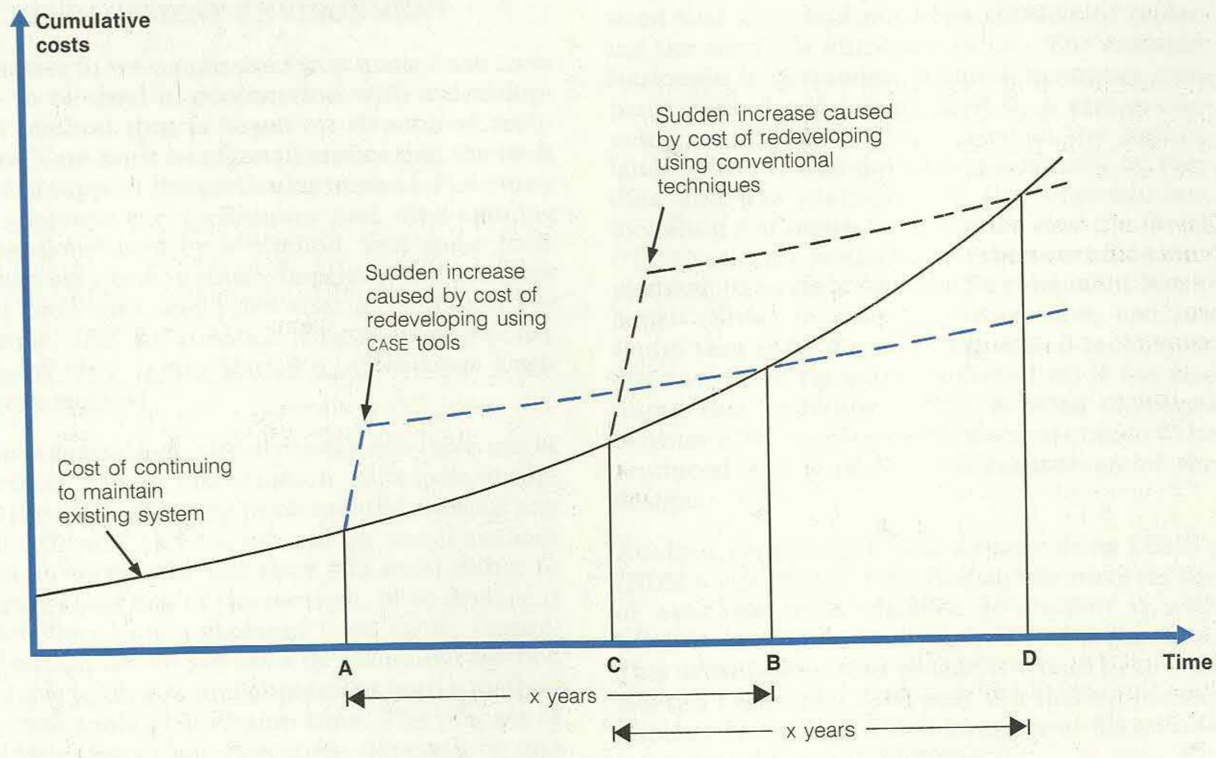
individual or departmental requirement. Professional development staff would normally not be involved in this type of development.

Restrict CASE tools to new development or major changes

Because existing CASE tools are of little benefit for maintaining or making minor enhancements to existing software, their use should be restricted to new development or to making major changes to existing systems. As we explained earlier, CASE tools will be useful in maintaining existing systems only if existing software can be reverse-engineered to make it conform with the development method that the CASE tools are supporting. Tools for automatic reverse engineering are still at a very early stage of development. A few organisations have attempted to carry out this process manually, but the cost can be enormous, requiring many work-years of effort, and the payback is not proven. However, some organisations are convinced that such an investment is necessary before the full benefits can be obtained from CASE tools. One German insurance company with an annual systems budget of DM140 million ($85 million) is planning to spend up to DM25 million ($15 million) over five years on just such an exercise.

However, the availability of CASE tools does affect the decision about when to replace existing software. In some instances, it will be more cost-effective to redesign and reconstruct a system than to maintain the existing software. The increases in development productivity brought about by the use of CASE tools, and the easier maintenance of the software developed mean that it is economic to replace existing applications sooner (see Figure 3.3). Furthermore, some systems managers who have implemented CASE tools claim that existing applications have to be rewritten sooner than originally planned in order to bring as much of the work as possible into the CASE environment. Unless this is done, the development department will have to retain the expertise and skills needed to maintain software developed originally with non-CASE tools. In turn, this will slow down the full

Figure 3.3 Use of CASE tools can bring forward the date when it is cost-effective to redevelop existing systems



A = Date at which decision is made to redevelop using CASE tools
B = Date at which cumulative costs of redeveloping using CASE tools are less than continuing to maintain existing system
C = Date at which decision is made to redevelop using conventional techniques
D = Date at which cumulative costs of redeveloping using conventional techniques are less than continuing to maintain existing system

In each case, the time required to recoup the redevelopment costs is the same. Thus, if this time equals the maximum period in which the redevelopment costs can be written off, existing systems can be redeveloped earlier by using CASE tools.

implementation of CASE tools throughout the organisation.

Having decided where CASE tools are likely to provide the most benefits, and identified the situations where the tools should and should not be used, the next stage is to select the most appropriate CASE-tool products.

Chapter 4

Selecting appropriate CASE tools

The most important consideration when selecting CASE tools is to ensure that the tools support the development methods being used. The first step in the selection process is therefore to review the development environment in terms of the methods being used. It is also necessary to identify the stages of the software life cycle that the tools are to support and to choose the CASE tools accordingly. Some tools support just one or a few of the stages. In this chapter, we show how consideration of these factors can be used to create a set of formal selection criteria, which are then used to select the most relevant CASE-tool products.

SELECT TOOLS THAT MATCH THE DEVELOPMENT ENVIRONMENT

In Chapter 3, we emphasised that many CASE tools have to be used in conjunction with a development method that is based on structured techniques. Care must be taken to ensure that the tools selected support the particular method. Not every tool supports the techniques and diagramming conventions used by a method, and some tools support only one method, imposing the development processes and rules that it contains. For example, IEF Information Engineering Facility supports only James Martin's Information Engineering method.

Some organisations already make effective use of a method, and, in this situation, CASE tools to support the method can be implemented without any great difficulty. Other organisations' use of methods is not so successful and they will need either to improve their use of the method, or to replace it before they can implement CASE tools. Organisations that do not yet use a development method will have to choose and implement both a method and CASE tools at the same time. The process of selecting CASE tools therefore depends on the methods that are already in use and how effective their use is.

WHERE METHODS ARE USED EFFECTIVELY

Where development methods are already used effectively, tools should be chosen to support the methods already in place. Although it may seem obvious to do this, many systems departments are often tempted to select the latest tool with the newest features, regardless of whether it is suitable for use with the existing methods. Where tools are being selected to support existing methods, the cost of implementing the tools is less than in other situations, because it is not necessary to select and introduce appropriate methods at the same time. In particular, it will not be necessary to provide the training, and the support from the method's supplier, that is required to introduce and use structured development techniques.

Most of the Foundation members we interviewed who had successfully implemented tools emphasised that they had not even considered replacing the methods currently in use. For example, Nationale Nederlanden, a Dutch insurance company, was already using SDM II, a systems development method widely used in the Netherlands. (SDM II was developed originally by Pandata and was sponsored by five organisations, including Nationale Nederlanden and the Dutch PTT.) Nationale Nederlanden then decided to implement Pandata's Systems Development Workbench (SDW) to support the method, and has found that SDM II and its associated techniques are now more rigorously adhered to. It has also found that software quality is being improved because SDW enables better documentation to be produced and improves the consistency of the designs.

Another organisation was already using LBMS's LSDM method, and, following an extensive review of available tools, decided to support it with LBMS's Auto-Mate Plus analysis and design tool. This organisation told us that it would have considered replacing LSDM only if a tool supporting another method provided an order-of-magnitude improvement in productivity.

The need to select tools that support the methods being used is strongest when the method covers the analysis and design stages. In this situation, there may be a CASE tool that has been designed explicitly to support the method being used. Sometimes, however, it is possible to select a CASE tool that can be used with a variety of methods — Arthur Young's IEW and Index Technology's Excelerator analysis and design workbench product are two examples.

Another option is to use a tool that can be configured to support different diagramming conventions, development standards, and methods. An example is the Virtual Software Factory product developed by Systematica, a small UK-based supplier of software-development tools.

WHERE THE USE OF METHODS IS INEFFECTIVE

Our survey of Foundation members shows that nearly 80 per cent of them already use methods based on some form of structured technique. However, several of the interviewees for the research for this report told us that these methods are often not used effectively — a view that corresponds with the findings of Butler Cox's consulting work in this area. An organisation that finds itself with methods in place that are not being used effectively will have either to improve its use of those methods or to replace them completely before implementing CASE tools.

The difficulty that can be encountered if CASE tools are introduced to support a poorly used method is illustrated by the experience of an organisation that had implemented a well-known development method without thinking through the full implications. Subsequently, the CASE tool designed to support this method was also introduced. However, neither the method nor the tool has produced the expected benefits — there has been no improvement in development productivity or in the quality of systems. This situation has occurred because the method and tool have not been fully accepted by the user and systems communities, and are not used for all development work. In addition, this organisation uses a code generator but, since this is not interfaced to the CASE tool, its prototyping capabilities are used instead of a design stage. This leads to design inconsistencies and poor-quality systems. The experience of this organisation shows that if the method and the tools to support it are not fully effective and universally applied, the potential benefits of using CASE tools will not be realised.

Improve or replace the development method

If a method is introduced as part of the implementation of CASE tools, the costs associated with the method will form the bulk of the total costs — of the order of 60 to 70 per cent according to our research. The decision to replace existing methods should therefore not be taken lightly. It will always be less costly and less disruptive to improve the method in place than to replace it completely.

There are many reasons why existing development methods may be used ineffectively. They include difficulty with implementing the methods, insufficient or incomplete coverage of the software life cycle, and a mismatch between the methods and the development style of the organisation. Only after every possible way of reducing the mismatch between the existing methods and the needs of the development department has been examined, should replacing the methods be considered.

Difficulty in implementing a method should not be taken as a reason to replace it, unless its credibility has suffered too much for it to be re-established. Implementing CASE tools to support the method will often go a long way towards solving the problem. One company that was already using the Information Engineering method and its associated techniques, and had formal development standards in place, decided to implement IEF Information Engineering Facility because of its code-generation capabilities. The introduction of IEF increased the interest in methods and techniques among development staff, and, as a result, the effectiveness of the method improved considerably.

Incomplete coverage of the life cycle by an existing method may require the use of complementary methods to cover more of the stages, or the extension of the existing method to support the missing stages. This approach may appear fragmented, but, if the methods are consistent, there is no reason why it should not work. (The use of complementary methods to cover several stages of the life cycle was discussed in detail in Foundation Report 57.)

Implement CASE tools to support the method

Regardless of whether an organisation decides to improve its existing methods or to replace them completely, CASE tools need to be chosen carefully to support the methods that will be used. If the method is being improved or enhanced, tools that support the existing method can be chosen. There is no need to wait until the improvements to the method have been implemented. Selecting and implementing tools concurrently with the process of improving the method will allow developers to become familiar with the tools and to use their facilities within the framework of the method. Where existing methods are replaced completely, the new method and the tools to support it should be selected together, using the procedures described below.

WHEN DEVELOPMENT METHODS ARE NOT USED

The situation where development methods are not used at present is the most difficult and costly in which to implement CASE tools. It will be difficult to convince development staff of the need to introduce methods because it will be necessary to persuade them that the horror stories they might have heard about the manual effort of implementing structured techniques are not relevant when CASE tools are used. It will also be necessary to convince them that the discipline and standard working practices imposed by using structured techniques bring worthwhile benefits. Senior business and systems managers will also need to be convinced that investment in a development method is worthwhile, particularly the costs involved in teaching development staff and users how to use structured techniques successfully.

Despite the difficulties, implementing CASE tools in conjunction with introducing a method provides the best opportunity for successfully introducing such tools. It is also the best way of ensuring the ideal combination of methods and tools, because they can be evaluated together.

The most important selection criterion, however, is to ensure that the method chosen matches the style of the organisation. If an inappropriate method is chosen, it will be impossible to use it successfully, regardless of how good the CASE tools to support it are. Once the use of methods has been discredited, it may be impossible to start again. (Report 57 contains detailed advice about how to select a development method that matches the style of the organisation.)

Even though the selection of the tool is subordinate to the selection of the method, the existence of tools to support a method is a powerful incentive to choose the method. Systems managers who have successfully implemented methods and tools tell us that they would not recommend introducing a method that cannot be supported by tools. It is unlikely, however, that there will be a clear choice between a method that is supported by tools and one that is not. In practice, most development methods make use of a limited range of structured techniques and diagramming conventions, and some CASE tools have been designed to support the techniques rather than a specific method. Thus, the choice of tool will often be determined by the level of technical assistance, training, and support available from the tool supplier, rather than by the facilities provided by the tool.

SELECT CASE TOOLS THAT SUPPORT AREAS OF CONCERN

Different generations of CASE tool and, to a certain extent, different tools within each generation, support different stages of the development life cycle. Once an organisation has defined the areas of greatest concern and, hence, the stages of the life cycle that need to be addressed, it can select the appropriate tools. To do this, the systems manager needs to know what tools exist to support the different stages of the life cycle.

Some commentators categorise CASE tools according to the life-cycle stage they support — analyst/designer workbenches or programming-support environments, for example. We believe, however, that it is more useful to categorise CASE tools according to the *range* of stages they support, because the cost and strategic impact of the tools grows with increasing life-cycle coverage. We define three categories of CASE tool: tools covering a single stage of the life cycle, tools that cover two or more consecutive stages, and integrated development environments designed to cover all stages. A typical cross-section of products that fall into each of these three categories is shown in Figure 4.1 overleaf.

Tools covering a single stage usually (but not exclusively) support the back end of the life cycle, typically the programming and implementation stages. A typical example is Program Definition Facility (PDF) from Michael Jackson Systems, used for designing and generating programs. Another is Gamma, a code generator from Knowledgeware Inc.

Tools covering several stages of the life cycle typically support the front-end analysis and design stages. Most of the better known CASE products fall into this category. A typical example is the ADT Yourdon Analyst/Designer Toolkit, from Yourdon International. This product covers the analysis and design stages of the life cycle.

Integrated development environments are the initial manifestation of I-CASE. They attempt to cover all stages from planning through to implementation. In practice, none of the products available today successfully achieves full coverage of all the stages, or provides complete integration between life-cycle stages. The best-known and most widely used product of this type is IEF Information Engineering Facility from Texas Instruments and James Martin Associates. Another more recent development is the CASE* series of products from Oracle. We believe that these will go a long way towards providing a true integrated development environment.

USE FORMAL CRITERIA AS THE BASIS FOR SELECTION

The criteria used to select the specific tools that best meet an organisation's requirements will be based on the factors discussed earlier in this

Figure 4.1 Examples of products in the three categories of CASE tools

Tools covering one life-cycle stage	Tools covering two or more consecutive stages	Integrated development environment covering most life-cycle stages
Telon (Pansophic Systems Inc); code generation.	Corvision (Cortex Corp); detailed design and programming.	IEF Information Engineering Facility (Texas Instruments/James Martin Associates).
Netron CAP Development Center; (Netron Inc) code generation.	ADT Yourdon Analyst/Designer Toolkit (Yourdon International Ltd); analysis and design.	FOUNDATION Integrated Environment for Software Engineering (Arthur Andersen & Co Management Consultants).
PDF (Michael Jackson Systems Ltd); program-design.	Managerview (Manager Software Products, Inc); analysis and design.	CASE* (Oracle Corp).
VAX Cobol Generator (Digital Equipment Corp); code generation.	ProKit Workbench† (McDonnell-Douglas Information Systems Group); planning, analysis, and design.	IEW (Knowledgeware Inc/Arthur Young Information Engineering Services).
	Auto-Mate Plus (LBMS plc; also marketed by Cullinet Software Inc as IDMS/Architect); analysis and design.	Maestro (Softlab Inc and Philips Business Systems Ltd).
	Excelerator (Index Technology Corp); analysis and design.	

† Framework only. Provides project database and management facilities to support other tools.

chapter — the existing development environment and the stages of the life cycle where the greatest problems occur. The first objective of establishing selection criteria is to narrow the field to a shortlist that matches the stated objectives and the structure of the systems department. The second is to select tools from the shortlist that meet the technical and commercial criteria important to the organisation.

DEFINE THE SELECTION CRITERIA

The first, and most important, group of criteria are those used to narrow the field of tools that need to be considered in detail. These criteria are derived from the style of the organisation, the stated objectives of the systems department and the type of development method currently being used. The most appropriate category and generation of tool can then be selected against these criteria. Other criteria form the basis for selecting specific tools. This second group of criteria are used to evaluate the technical merit of the tools and the commercial and technical performance of their suppliers.

Identify appropriate categories and generation of tools

The most sophisticated CASE tools with the greatest life-cycle coverage are not necessarily the best tools for a particular organisation. Implementing a complex integrated development environment supported by a mainframe-based development dictionary is an expensive exercise, and is appropriate only for large organisations experienced in using structured development techniques. Tools covering either one or a few stages are less costly and require less effort to implement, and may therefore be more appropriate for less sophisticated or smaller organisations. One of the most important parts of the selection process is establishing the most appropriate category of tool for a particular organisation.

Selecting the appropriate generation of tool is also an important part of the process. Just as programming languages from earlier generations are still available, there are different generations of CASE tools on the market, each providing a different level of functionality and different cost/benefits. (The characteristics of each generation, together with representative products, were shown in Figure 1.1 on page 3.)

The size of an organisation largely dictates the most appropriate category and generation of tool. In general, the benefits from using tools increase with the size and complexity of the applications portfolio, but even small organisations with only a few development staff can benefit from the ability to speed up the production of designs and documentation that results from using an analysis or design tool.

With the exception of realtime applications (for which there are specialised development tools), the type of business application that will be developed is not a particularly significant factor in selecting appropriate CASE tools. The underlying analysis and design concepts will be the same regardless of the type of system being developed. The size of application, however, will affect the level of benefits that can be gained.

Define detailed evaluation criteria

The next step in the selection process is to define the criteria that will be used to perform a detailed evaluation of the shortlisted products. Apart from cost, the criteria relate to two areas — the technical merit of the product and the commercial viability of the supplier.

The technical criteria include the ability of the tool:

- To display graphically the data and activity diagrams required by structured methods.
- To meet the requirements of the systems department in terms of response times and shared use of workstations and development dictionaries.
- To provide a high level of consistency checking between the diagrams produced at different stages of the development process.
- To generate code automatically from the output of the design stage.
- To provide a high level of software reliability.
- To provide links to other CASE tools, databases, and data dictionaries.
- To be used in the organisation's existing hardware environment.

The supplier criteria are concerned with the financial and market position of the supplier, the availability of post-sales support, including international support, the supplier's commitment to the product, and, where appropriate, the relationship of the supplier with the product's developer. Typical detailed technical and supplier criteria used by Butler Cox's consultancy practice are shown in Figure 4.2.

Figure 4.2 Product and supplier criteria for selecting CASE tools

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 multi-method 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

When considering the supplier criteria, it is important to recognise that the market for CASE tools is relatively immature and that the dominant suppliers and products are only now emerging. Without doubt, the structure of the CASE-tool supply industry will change considerably during the next few years.

EVALUATE AVAILABLE PRODUCTS AGAINST SELECTION CRITERIA

The final stage in the selection process is to use the selection criteria defined above to identify the specific CASE tools to be implemented. At first sight, it may appear that there is a wide range of effective CASE tools from which to choose. The reality is different. At a recent CASE conference and exhibition in the United States, tens, rather than hundreds, of suppliers were represented, and many of the products were, in fact, only marginally associated with CASE. Furthermore, many of the suppliers were small companies and lacked the resources either to develop mainstream CASE products or to support them in an increasingly competitive market.

Assessing how well a particular product or its supplier meets a particular criterion can, however, be fraught with difficulties. There are few, if any, detailed product reviews and the assessors will usually have to rely on information provided by the suppliers. The best course of action is to discuss the products and suppliers with existing users. Suppliers will generally be willing to provide the names of customers who can be approached for this purpose.

Using the selection criteria will ensure that the CASE tools that best meet the needs of the business are chosen. However, it is at the end of this stage that the most important task begins: the implementation of the chosen tools. We describe how to do this in the next chapter.

Chapter 5

Managing the implementation of CASE tools

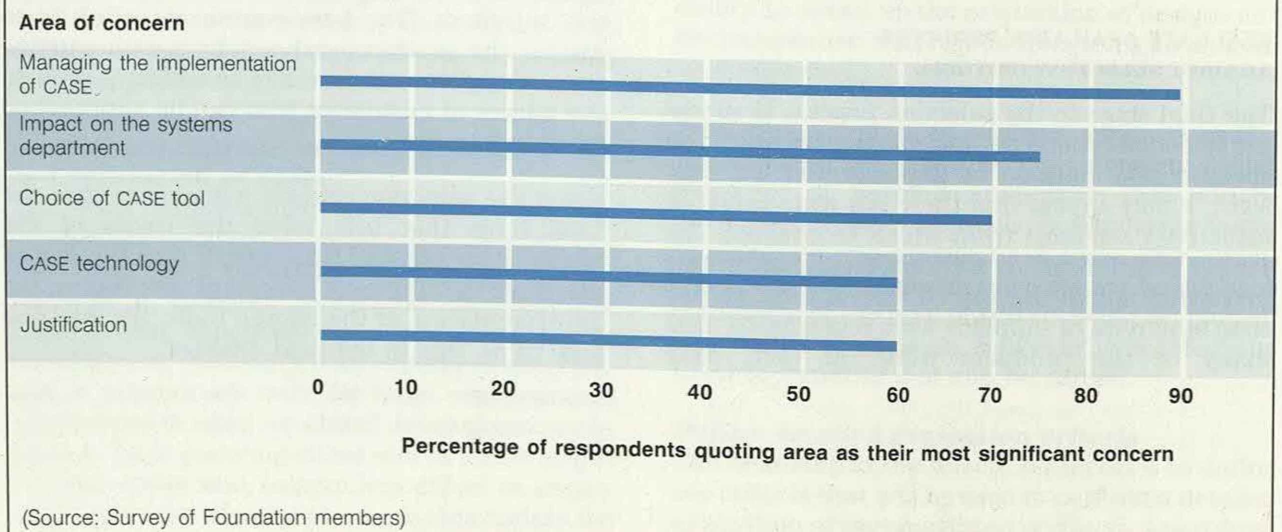
The long-term success of CASE tools depends on how well their implementation and subsequent use is managed. If their initial implementation is not managed properly, their credibility will be reduced, staff will not be motivated to use them, and neither systems management nor user management will feel committed to use them in the future. The importance of managing the implementation of CASE tools is certainly recognised by Foundation members. As Figure 5.1 shows, nearly all of those who responded to the questionnaire believe that this is the most important and difficult aspect of introducing CASE tools.

There are several actions required to ensure the successful implementation of CASE tools. The first is to gain the commitment of both system and user management before the process of selecting and implementing CASE tools begins. This can be achieved through education and through presentations about, and demonstrations of, CASE tools. The aim at this stage is to present realistic estimates of the benefits of CASE tools and their potential impact on the process of developing software. It is also important to highlight the benefits that improved software quality and development productivity will bring to the business.

A similar education programme is required for the development staff who will use the tools. This time, the aim is to motivate them to want to use the tools. There will also be a need for extensive training of systems staff, not only in using the tools, but also in using the structured techniques that the tools will be used to support. In particular, training in analysis skills will be required because development methods based on structured techniques place a much greater emphasis on the analysis and design stages.

It is also important to create, at an early stage, an enthusiastic team of independent staff committed to the success of CASE tools. Once they have been trained in using the tools, they should use them to develop a pilot application. The pilot application is an important part of the implementation of CASE tools. It is part of the learning process, and will provide valuable lessons for the future. It will, for example, provide a useful indicator of the productivity and quality gains possible from using CASE tools — provided, of course, that these are measured accurately and can be compared with equivalent measurements for applications developed without the help of CASE tools. It is important, however, to select a suitable pilot

Figure 5.1 Managing the implementation of CASE tools is the most significant concern



(Source: Survey of Foundation members)

application. The application should be important to the business but it should not be an extremely urgent or critical application that has to be developed in a very short time. Although CASE tools will provide productivity benefits, it is unrealistic to expect these to be achieved fully with the pilot application.

The implementation process does not end with the pilot application, however. It is then necessary to extend the use of CASE tools throughout the systems development department, ensuring that each additional use is justified in its own right.

Finally, the CASE implementation team must prepare for the future. CASE tools will evolve rapidly over the next few years, and it is important to plan, from the outset, to move to later generations of tools as they emerge.

GAIN SUPPORT THROUGHOUT THE ORGANISATION

It is generally accepted that the successful implementation of a new information technology depends both on the support of senior business managers and acceptance by the systems and user communities. The level of senior-management support required depends on the cost of purchasing and implementing the CASE tools. Third-generation tools covering several life-cycle stages will be the most expensive, and will have the greatest strategic impact. They therefore require the highest level of senior-management involvement and, hence, also require the greatest amount of effort to convince senior managers that they need to be involved in implementing the tools.

Senior managers need to be convinced that the introduction of CASE tools is a business issue rather than a technical one. They should therefore be made aware of the benefits that will accrue to the business from the investment in CASE tools. Managers in the systems department should also be involved in the decision-making process because they will have to implement the decisions. Serious implementation problems can arise if the commitment of these managers to introducing CASE tools is not gained in advance. For example, a leading German insurance organisation met significant resistance from the project managers and group leaders, who felt that they derived little direct benefit from the CASE tools that were implemented. It was not possible to involve these staff in the initial decision to introduce CASE tools but, nevertheless, they were expected to take on extra supervisory and control tasks as a result. To counter the antagonism of these staff, this organisation has now initiated a full training programme and has arranged for full consultancy support to be available. So far, however, these moves have met with only limited success.

SYSTEMS DEVELOPMENT STAFF

The introduction of automation is usually most strongly resisted by those whose work is to be automated, and systems development staff are no exception. They need to be convinced that CASE tools will bring benefits to the systems function and to the business as a whole. In particular, they need to be made aware of the advantages of using CASE tools in terms of automating and supporting the more difficult aspects of systems development, facilitating the use of systems development methods, and promoting an image of greater professionalism. However, they should also be aware of the significant amount of training and retraining that will be required, the need to acquire new skills, and the changes in working practices that will be brought about by the use of CASE tools.

Inevitably, there will be resistance to the changes. For example, the Westpac Banking Corporation, a leading Australian financial institution, believes that, although the implementation of CASE tools has enabled it to recruit new employees, one in five of the existing staff were opposed to their introduction. Failure to deal with the resistance sensibly will result in lower staff morale and loss of credibility for the CASE tools.

The best way of gaining the support of systems staff is to set up a small team of key staff who will act as the centre of expertise for CASE tools, and who will form the project team for the pilot application. Ideally, these people should already be experienced in the use of structured development techniques. They must also be willing to promote the use of CASE tools among their colleagues. The Westpac Banking Corporation believes that its success in introducing CASE tools was partly due to its choice of a small number of systems staff who were willing to initiate change, to accommodate new ideas, and to champion the use of CASE tools (see Figure 5.2 overleaf).

THE USER COMMUNITY

Because the implementation of CASE tools improves development timescales and software quality, it will have a direct impact on user departments. The emphasis in gaining the support of the user community should therefore be on promoting the benefits that CASE tools will bring in terms of a closer fit between applications and user needs. This will not be the only effect of CASE on users, however. User departments should also be made aware that the use of CASE tools means that it is easier for them to be more closely involved in the development process. Indeed, user involvement

Figure 5.2 Small units of expert staff help to ensure successful implementation of CASE tools

WESTPAC BANKING CORPORATION

The Westpac Banking Corporation is an Australian banking and financial services organisation, with assets of A$70 billion ($56 billion) and an annual information systems budget of around A$150 million ($120 million). It has 2,700 information systems staff, 600 of whom are involved in systems development. Westpac first introduced CASE tools early in 1986. Fifty per cent of the systems development staff are now using CASE tools, supported by one workstation per two to three developers. Both productivity and software quality have improved.

Westpac has a long history of trying to implement systems development methods. Structured analysis and design techniques were introduced in the 1970s to develop systems, but the experiment failed because the high level of manual effort required to keep the documentation up to date negated any productivity benefits. When Westpac tackled the problem again in the mid-1980s, it therefore had to start from scratch with new methods supported by CASE tools.

The corporation adopted a step-by-step approach to CASE implementation. The new, in-house method was established first. Then, small information systems units were created, separate from existing departments, to exploit the new technology and to become centres of expertise and excellence. Only then did Westpac introduce CASE tools. Since early 1986, Westpac has been using Netron/CAP Development Center, a code generator from Netron Inc, and IEW, from Arthur Young. It uses IEW in the early stages of the life cycle for defining user requirements and for systems analysis, and Netron/CAP for generating Cobol source-code during the programming stage.

Westpac believes that its step-by-step approach, coupled with senior management promotion and commitment, middle management enthusiasm, and the use of a small number of exceptional personnel to provide technical leadership, has ensured the success of its CASE implementation. It measures this success in terms of improved productivity and software quality. The main productivity benefit has been in the programming stage, through the generation of re-usable Cobol, and in software maintenance, through improved documentation and more comprehensible code.

Westpac has also found that greater emphasis on analysis and design has produced definitions of user requirements that are more rigorous, flexible, and clear. Positive feedback from users, who approve of an approach that models systems before they are fully developed, has also been received. Users feel that they have better control because they can manage the definition of their own requirements.

There are three important features of Westpac's move to CASE tools. The first is that the corporation moved from a process-centred approach to software development, to a data-centred approach. Second, it placed greater emphasis on specifying user requirements. With the support of IEW, staff can now spend more time with more users, without any adverse impact on overall development time. Third, systems staff can document new systems requirements and existing systems in areas that were previously regarded as too difficult to tackle.

is vital to the successful implementation of CASE tools.

It is therefore equally important for there to be a small nucleus of user staff who are committed to introducing CASE tools and who can act as the focal point for promoting the use of CASE technology among the user community. These staff should have the respect of the rest of the user community and be able to influence their acceptance of CASE tools and the changes in working practices required by the tools. They should be involved right from the beginning, which means that user managers need to be persuaded to release key people to work on the selection and implementation teams.

The involvement of user representatives on the CASE implementation team will help to ensure that other team members are made aware of the business implications of their actions. Ideally, these representatives should work for the department for which the pilot application will be developed, so they can experience at first hand the benefits of using CASE tools.

BE REALISTIC ABOUT THE COSTS AND BENEFITS

In the same way that suppliers have over-sold the benefits of CASE tools, there is a temptation for systems directors to emphasise their benefits and play down their disadvantages. It is important to resist this temptation because raising expectations to an unrealistic level is bound to lead to disappointment at a later stage. It is important therefore to emphasise that the cost of implementing CASE tools can be high and to set realistic expectations for the benefits that can be achieved.

Many of the Foundation members we interviewed indicated that they had underestimated the costs and overestimated the benefits of CASE tools. A UK-based multinational oil company, for instance, found that the costs associated with gaining support for the introduction of CASE tools, and the costs required to train staff in how to use the method supported by the tools were both at least 50 per cent higher than expected, as was the amount of hardware that needed to be dedicated to the tools. Consultancy-support costs were also underestimated.

IMPLEMENTATION COSTS ARE LIKELY TO BE HIGH

Although the cost of implementing CASE tools depends on the category and generation of the tools chosen, it can be high in all but the simplest applications. The costs fall mainly into two categories: technology (hardware and software), and support (training and consultancy). It is a mistake, however, to reduce resources in either of the categories in order to reduce costs. Inadequate resources will inevitably lead to dissatisfaction among users and will slow down the rate at which the use of CASE tools can be extended after their initial implementation.

The high level of costs that can be required to implement CASE tools is illustrated by a manufacturing organisation that has implemented standalone analysis and design tools to support 30 development staff with one workstation per developer. The total implementation costs in this organisation were $1.39 million, or $46,500 per developer, of which 45 per cent was for training and consultancy support. An analysis of the implementation costs is shown in Figure 5.3. The costs would have been even higher if this organisation had chosen to link each development workstation to a mainframe-based development dictionary. In addition, the training and consultancy costs will, of course, continue after the initial implementation. Some organisations have found that they require consultancy support for two or three years.

Figure 5.3 Implementation costs for CASE tools can be substantial

Costs of implementing analysis and design tools for 30 development staff, with one workstation per developer.

Workstations	$170,000
Software	600,000
Training	500,000
Consultancy support	120,000
Total	$1,390,000 (or $46,500 per developer)

Technology costs

The initial technology costs will vary according to the generation of CASE tools chosen, and the stages of the life cycle covered by the tools. The later the generation, and the more extensive the coverage, the more expensive will be the hardware and software costs. In general, however, our research shows that the cost of software represents only about 20 to 35 per cent of the initial costs of implementing CASE tools in conjunction with a development method. Figure 5.4 gives a full breakdown of the costs of two representative implementations, one for an integrated development environment and the other for analysis and design tools. The figure also gives the continuing costs expected to occur during the first three years after the initial implementation.

The technology costs include the cost of hardware and supporting software. The hardware costs always include the cost of workstations for development staff (we recommend at least one workstation for every two staff). Hardware costs will also include increased use of central mainframe resources where a mainframe-based tool or development dictionary is used. Workstation costs vary greatly according to the type of equipment, from a few thousand dollars for an MS-DOS personal computer to tens of thousands for a powerful workstation with high-resolution graphics capabilities. Sometimes, the hardware costs will also include the cost of a network to provide multi-user

Figure 5.4 Typical costs for installing and implementing two types of CASE tool over a three-year period

Based on providing facilities for 30 development staff

Cost item	Analysis/design tool	Integrated development environment
Initial costs		
Hardware (workstations or PCs)	$150,000	$150,000
Workstation software	125,000	125,000
Mainframe software	—	250,000
Implementation support	25,000	50,000
Training (including training in the method)	400,000	500,000
Total initial costs	$700,000	$1,075,000
Recurring costs (per year)		
In-house technical-support group	$100,000	$100,000
Software maintenance	15,000	50,000
Hardware maintenance	15,000	15,000
Continuing training	25,000	25,000
Other (documentation, external meetings, . . .)	20,000	20,000
Total recurring costs (per year)	$175,000	$210,000
Total costs over 3 years	$1,225,000	$1,705,000
Total costs per developer over 3 years	$40,833	$56,833
Average cost per developer per year	$13,611	$18,944

working, particularly where the development workstations are not linked to a mainframe.

Supporting software costs will vary with the type of tool installed. For single-user, PC-based tools, no additional software will be necessary. At the other end of the scale, a full implementation of an integrated development environment using a mainframe-based development dictionary requires not only the CASE tools, but also a database management system to support them. Having to install DB2, which is necessary with some tools (notably IEF Information Engineering Facility), is very expensive and requires careful thought. The organisation's whole strategy for databases and database management systems, not just for CASE tools, needs to be thought through and agreed.

Support costs

The support costs associated with implementing CASE tools are likely to be more than the technology costs although this depends on the maturity of the current development environment and the experience of using structured techniques. For example, the training costs will be high if a new method has to be introduced as part of the implementation. One Foundation member told us that training 50 development staff to use a new method required each of them to attend a four-week course at a total cost of around $350,000. This cost excluded the cost of productive time lost during training and the learning period, which would more than double the training cost. People who have been trained in the use of a method can take up to a year to become fully productive.

In addition, to support the method during its early implementation, some consulting advice will probably be required from the suppliers of the method. The number of consultants involved will depend upon the size of the development department, but a ratio of one full-time consultant to 50 development staff is typical.

The introduction of a new method is also likely to lead to a need to reorganise the systems department and to change its procedures and practices. The personnel costs associated with these changes will depend on the extent to which the skills profile of the department has to be changed. The impact of these changes is discussed on page 31.

FULL BENEFITS CAN TAKE SEVERAL YEARS TO ACHIEVE

In Chapter 2, we described the benefits that can arise from using CASE tools. In practice, however, the benefits are often overestimated and the timescale required to achieve them is underestimated. Implementing CASE tools throughout the systems department can require a substantial investment in hardware, software, and education. Many organisations will authorise such investments only if they are likely to produce a payback in a relatively short time — perhaps as little as two years. The full benefits of CASE tools may take longer than this to realise, and it is a mistake to justify the investments on the basis of benefits that cannot realistically be achieved within the required payback period. It is therefore sensible to ensure that the implementation of CASE tools can achieve short-term benefits, even if this means delaying some of the longer-term benefits.

Short-term benefits are easiest to achieve with CASE tools that cover one or a few life-cycle stages, because these types of tools can be implemented reasonably quickly. In addition, the costs of using them build up progressively. Integrated development environments, however, require a high initial investment, and where it is important to achieve short-term benefits, it may be necessary to install a less sophisticated tool. Implementing code-generating tools to support the programming stage will usually provide the most immediate improvement in development productivity and software quality. These tools very quickly increase the number of lines of code per programmer-day, and reduce the coding errors.

One company told us that it had been using the Focus product as a means of improving development productivity. It then decided to use the DB2 database management system for its mainstream applications, which meant that it could no longer use Focus for these applications (PL/1 was used instead). However, the use of Focus had increased productivity at the programming stage by a factor of five, so this company had to find an alternative way of improving productivity when it reverted to PL/1. It decided to install the Telon code generator from Pansophic Systems Inc, a US software supplier, to increase coding productivity. This was a tactical decision that delivered the required short-term benefits, while providing the time to find a product that fitted in better with the long-term strategy.

PREPARE FOR ORGANISATIONAL CHANGES

However good the match between an organisation and the CASE tools it chooses, it is likely that both the skills profile and the organisation of the systems department will have to change as a result of implementing the tools. The use of CASE tools also requires more involvement by users in the development process, and this will affect the way in which user departments are organised. The impact of these changes can be reduced by planning ahead.

CHANGES IN THE SKILLS MIX

One of the most profound changes that systems departments will have to cope with as they implement CASE tools is the increased emphasis on analysis and design skills, and a reduced emphasis on programming skills. (Figure 2.2 on page 10 showed how the use of CASE tools increases the proportion of effort required at the analysis and design stages and reduces the effort at the programming stage.) In addition, as the use of CASE tools increases, the proportion of development resources used for software maintenance will decrease, allowing more effort to be spent on developing new applications.

The trend towards a reduction in the number of programmers was very evident in a survey carried out as part of Butler Cox's Productivity Enhancement Programme (PEP). More than 600 development staff from seven organisations responded to this survey. Only 16 per cent were classified as programmers. Fifty-one per cent were classified as analyst/programmers, and 4 per cent as analysts. (The remainder were classified as systems development managers, project managers, or project leaders.)

A major consequence of the changing skills profile in the systems department is a need to retrain existing staff in analysis and design skills, in addition to the training required to use new development methods and CASE tools. There will, however, be some resistance because many programmers will be either unwilling or unsuitable to retrain as analysts. Much of the resistance can be overcome by pointing out the ease with which analysis and design can be carried out by using CASE tools, and the image of increased professionalism produced by the rigorous use of the methods they support. The team of staff promoting the use of CASE tools will also have a role to play in persuading their more conservative colleagues to adopt the working practices demanded by the methods and tools.

The increasing use of analyst/programmers is one example of the use of CASE tools breaking down the traditional boundaries between different systems development roles. In general, there will be a shift away from employing staff with specialist technical skills, to staff with business skills and skills in several development functions. These changes will lead to a much flatter organisational structure for the systems department, which will typically consist of business analysts and either analyst/programmers, or analysts and designers who are provided with limited programming-support staff.

In addition, there will be a need to create a specialist team to support the use of both the methods and CASE tools, and to provide advice about their use to the project teams. This team could be formed from the staff who work on the pilot project.

INCREASING USER INVOLVEMENT

In the past, several systems development techniques and tools have been heralded as the breakthrough that would allow users to be involved directly in the systems development process. Fourth-generation languages and data modelling are two examples. However, the expected increase in user involvement has not, by and large, occurred.

Initially, CASE tools were not seen as a means of increasing users' involvement in the development process, because analysis and design tools require a knowledge of structured techniques that are not familiar to users. Our own research shows, however, that in many organisations, the implementation of CASE tools has resulted in increased user involvement. For example, a major French manufacturing company, which uses Arthur Young's IEW, has found that the graphics capabilities of the tool have encouraged users to participate in the analysis stage, and to take more responsibility for quality and for meeting project deadlines.

The interactive, screen-based facilities provided by analysis and design tools encourages users to 'sit-in' at the analysis and design stages, and to take a more direct interest in the development of their applications. There is no doubt that greater involvement at these stages results in software that better matches the users' requirements. Some users will resist the need to be involved more, however, believing that software development is the responsibility of systems professionals. Organisations should make strenuous efforts to overcome this resistance, because the ultimate success of CASE tools in improving software quality depends on increased user involvement at the analysis and design stages.

The need for development staff to work more closely with users highlights the need for analysts and designers to have effective interpersonal communication skills. Sitting beside a user who is directly involved in the development process requires very different skills from those required to write a specification that is given to the user for approval.

START WITH AN APPROPRIATE PILOT APPLICATION

The first application that is developed using CASE tools should be a pilot project. The aim is to check that the chosen tools (and the methods they support) will work in the particular organisation, and to lay down the ground rules for extending the use of the tools throughout the systems department. The pilot application is not part of the process of selecting CASE tools. The application chosen should therefore be one that will provide real business benefits and that can be used to measure both the performance of the development team and the quality of the software produced. It should be sufficiently important to the business to ensure that the user department is fully committed to implementing it successfully. However, it should also be an application where a short delay in implementing it would not be disastrous for the business. Even so, the potential users of the pilot application should be made aware of the risks involved in using new methods and tools to develop it.

It is also important to select an application that is typical of the bulk of the mainstream development work done by the systems department. A primary aim of the pilot project is to begin to build up experience of using the CASE tools that can be transferred to other project teams and applications as the use of the tools is extended. For example, a database application should be selected if much of the new development work is database-oriented.

A successful pilot application will also ensure that the managers of the user department in question are enthusiastic about the use of CASE tools. They will then be powerful allies in extending the use of the tools throughout the organisation. The characteristics of a successful pilot project are listed in Figure 5.5.

MINIMISING THE RISKS

Using new tools for the first time always carries the risk that development could take longer and cost more than if traditional approaches were used. The deadlines for the pilot project should therefore be set to take account of this. The risks associated with the pilot application can also be minimised by ensuring that the pilot-team members are fully trained in using the tools (and the methods they support) before commencing the project.

Establishing realistic deadlines

One of the purposes of implementing CASE tools is to speed up the systems development process. Thus, there is a temptation to set the deadlines for the pilot project to prove that this does in fact happen. However, considerable slack should be built into the timescale for the pilot application because it is inevitable that unforeseen problems will occur as the CASE tools are used for the first time. It is usually unreasonable to expect to achieve the full productivity increases at the pilot stage. Developing the pilot application within existing timescales will normally be sufficient to judge the pilot use of CASE tools to be a success. If tight deadlines are set, there is a risk that they will be missed. The result will be a demotivated project team, and the CASE tools will be discredited in the eyes of user management.

Training the pilot team

The staff selected to provide the core expertise about CASE tools should be used for the pilot team. They will already be fully committed to the idea of using CASE tools, and the pilot project will provide them with practical experience of using the tools.

Before starting on the pilot, all the team members should be fully trained in the use of the tools and, if possible, experienced in using the methods supported by the tools. Doing this will help to shorten their learning curve and will ensure that the pilot application provides a good indication of how well the tools will perform once they have been fully implemented.

MEASURING PERFORMANCE

The lessons learnt from the pilot project will be maximised if the effort involved and the quality of the software produced is measured accurately. These measurements will provide valuable information for the future and a comparison, however broad, with the past. The ideal is to develop the same application with and without CASE tools and compare the results, but few, if any, organisations can afford this luxury. The alternative is to compare the measurements from the pilot project with those from past projects, or, if these do not exist, with measurements from applications developed at the same time as the pilot project.

These comparisons allow the impact of CASE tools on development productivity and on software quality to be assessed. However, the absence of productivity improvements does not necessarily imply that the pilot project has shown that the CASE tools will fail to deliver the predicted benefits. Productivity improvements usually appear

Figure 5.5 Successful pilot projects have common characteristics

An important business application
Realistic time constraints
User management committed to the use of CASE tools
Project team fully trained in both tools and methods, and, if possible, experienced in the use of the methods
Project planned effectively, productivity and quality measured, and results audited

only in the long term; the more immediate benefit of using CASE tools is improved software quality.

EXTEND THE USE OF CASE TOOLS

All of the decisions made and the actions taken up to the pilot-application stage are aimed at creating a foundation on which the use of CASE tools can be extended throughout the organisation. The implementation process does not end with a successful pilot application but continues until the CASE tools are used for all aspects of software development for which they are appropriate. The increasing use of the tools must, however, be continually justified. There will always be areas of software development where other tools will be more appropriate.

The data gathered in the original justification exercise and in the course of the pilot project will also be useful in justifying the wider use of CASE tools. The lessons learnt from the pilot project will also help to extend the use of the tools, by providing the basis for training development staff, for refining development methods so they meet the needs of the business better, and for improving working methods to make the best use of the tools.

The measures gathered from the pilot application must be made on a continuing basis, for all projects, regardless of whether CASE tools are used. Comparing development productivity at each stage of the life cycle will help to provide the quantitative information required to justify the further use of CASE tools. Measurable improvements in terms of reduced maintenance costs resulting from better analysis and design will appear only in the medium to long term and will thus be identified only by a long-term measurement programme.

However, the use of CASE tools is not the only factor contributing to improving quality and productivity. Project-management skills, individual technical skills and working practices, and development schedules all play a part, and they should also be monitored continuously.

PREPARE FOR THE FUTURE

CASE tools are still developing rapidly, and there will be considerable developments during the next few years. It is necessary, therefore, to consider the likely changes as CASE tools are initially implemented so that the transition to later generations of tools can be as smooth as possible. The most significant developments will occur in the field of I-CASE tools. Migrating to such tools will not be straightforward, and consideration should be given to ways of protecting the initial investments made in CASE tools.

MOVING TOWARDS I-CASE

The ultimate aim is to provide an integrated set of CASE tools that cover all stages of the software life cycle, beginning with the definition of business requirements, moving through the analysis and design stages, then to the automatic generation of programs, and finally facilitating the maintenance of operational systems. An associated goal is to provide CASE tools that can 'read' existing programs that were not originally developed with CASE tools, and automatically generate designs that conform to the method supported by the tools. It will be some years before integrated CASE tools with these abilities are available, but the emergence of integrated development environments and the growing number of partnerships between suppliers of different types of tools are beginning to result in products that display some of these characteristics. The likely future developments in CASE tools are illustrated in Figure 5.6.

The cost of developing a fully integrated set of CASE tools to cover every aspect of the development life cycle is enormous, however. ITT, one of the CASE-technology pioneers, has estimated that such a set of I-CASE tools would take five to six years to develop, at a cost of $85 million for the software alone. Texas Instruments, the developer of IEF Information Engineering Facility, is said to have spent $50 million on developing the product, and is spending up to $10 million per month on promoting and marketing it in an attempt to dominate the industry. Even the cost of developing and supporting a CASE tool that covers just one

Figure 5.6 There will be significant development in CASE tools

Likely developments	Timescale
Appearance of simple reverse-engineering tools that will create system designs from existing programs and data structures	1988/1989
Formal agreements reached and interfaces established between most analysis and design tools and code generators	1989/1990
Increasing availability of tools that can be customised to any language or method	1989/90
Use of expert systems to provide advice during the analysis stage	Early 1990
Availability of expert-system support for reverse-engineering	Early 1990
Development of a consensus on CASE tool standards	Early 1990
Use of expert systems to provide advice during the design stage	1990/1991

stage of the life cycle can run into millions of dollars. Because of the high costs involved, some commentators believe that no single company has either the resources or the inclination to develop a full set of I-CASE tools.

We believe that the large and increasing cost of product development, as well as increasing competition in the marketplace, will lead to a reduction in the number of suppliers able to supply a fully integrated set of CASE tools. Although, in the short term, there will be an increase in the number of CASE-tool suppliers, in the longer term companies providing complementary products will merge, and smaller specialist companies whose products complete the portfolio of more powerful competitors will be taken over. Those who retain their independence will survive in niche markets or will form partnerships with other suppliers.

I-CASE is beginning to emerge but progress will be slow

The concept of I-CASE is based on the use of a database and data dictionary that store information about the business, about data, and about the activities modelled by computer systems. I-CASE tools will ensure that these different types of information are integrated and used consistently throughout the software life cycle and throughout the applications portfolio. The logical and physical databases used by operational systems will be derived from the information held in the data dictionary.

Most I-CASE products are likely operate in development and operational environments that support mainstream database management systems, especially IBM's DB2. Several suppliers have already produced products in this category — one example is Texas Instruments' and James Martin's IEF Information Engineering Facility. This product provides a mainframe-based development dictionary that is, in fact, a DB2 application, and it generates SQL (Structured Query Language) statements that are compatible with DB2. Another example is Oracle's CASE* products, which use Oracle's own database management system, but which will eventually generate SQL statements that are DB2-compatible.

Progress towards integrated CASE tools is also being made as the result of partnerships between suppliers of different types of tool. For example, Knowledgeware and Arthur Young have bought Tarkenton Software's Gamma code generator to interface to the IEW workbench product, and Index Technology has agreed to interface its Excelerator analyst/designer workbench to Pansophic Systems' Telon code generator.

Three factors are, however, holding back the emergence of I-CASE tools. The first is the difficulty of producing code automatically from the output of the analysis and design stages. The best that has been achieved so far is to generate code from program-structure diagrams or activity diagrams. The second factor is the difficulty of creating reverse-engineering tools that can be used to bring existing software into the CASE environment. The hope is that artificial intelligence techniques can be used to analyse existing software and extract the underlying business processes. The third, and possibly most important, factor is the lack of commonly agreed standards for CASE tools. Without standards, it will be difficult, if not impossible, to integrate tools from different suppliers.

The two areas where standards matter most are in exchanging data between CASE tools, and in allowing tools designed to operate in one hardware and software environment to be used in a different environment. Some progress is being made in both of these areas. The ANSI and the ISO standards organisations have been working since the early 1980s to define a common data-dictionary standard known as Information Resource Dictionary Standard (IRDS). The original aim of this standard was to make it easier to transfer data between data dictionaries. It is to be extended, however, so that it can be used as a standard for the development dictionaries used with CASE tools.

Standards that will allow CASE tools to be used in different hardware and software environments are also being developed. In Europe, for example, the Portable Common Tool Environment (PCTE) has been proposed by a consortium of European companies, which developed the proposals as part of the European Community's Esprit research programme. PCTE is essentially a framework for software tools that defines a core standard interface for use by tool suppliers. This standard is likely to be adhered to more by European CASE-tool suppliers than by their counterparts in the United States, who will use their own standards, which are essentially incompatible with PCTE. PCTE has not, however, been formally adopted as a standard, and commercial tools conforming to it are not yet available. This uncertainty about standards is likely to continue for several years, creating problems both for suppliers and user organisations that wish to integrate discrete CASE tools.

Migration to I-CASE will not be straightforward

The current immaturity of CASE tools, the impending changes in the structure of the CASE-tool supply industry, and the increasingly rapid advances in the facilities that CASE tools provide, will inevitably mean that some organisations implementing CASE tools today will subsequently have to migrate to other products. Furthermore, the absence of standards means that many of today's products use proprietary design structures and project-data structures, making both integration and future migration difficult. Such tools are 'closed', in that they do not make it easy to transfer data to other tools and to other environments.

The problems of migrating to the next generation of CASE tools will therefore be formidable. In many respects, the problems are similar to those faced by organisations as they move from conventional hierarchical databases to relational databases. In general terms, the advice given in Foundation Report 64, *Managing the Evolution of Corporate Databases*, will be relevant. Undoubtedly, the suppliers of more advanced integrated CASE tools will provide some degree of automated support for migrating to the new CASE environment. Even so, the effort required will be significant.

PROTECTING THE INVESTMENT IN CASE TOOLS

With the situation changing so rapidly, organisations obviously need to take steps to protect their investment in CASE tools and the methods supported by the tools. Although the investment in the tools themselves will be significant, it will be small in comparison to the investment in the methods supported by the tools. Training staff in how to use a method, and establishing the working practices required by the method, requires considerable investments of time and money. It is important, therefore, to perceive the tools as being subordinate to the method, and to make a long-term commitment to using the method. Hence, CASE tools should be replaced only if the new tools support the same method.

The investment represented by data about the organisation's computer applications that is stored by the CASE tools also needs to be protected. The costs of creating this data will often represent tens of years of effort. When migrating to new tools, it may not be possible to transfer the data to the new tools. It is therefore important to assess how easy it will be to transfer data from the CASE tools currently being implemented. Although common data-structure standards do not yet exist, some tools are 'open', in that they provide facilities for at least the partial transfer of data to other environments.

A second way of protecting the investment in CASE tools, which is particularly relevant if an integrated development environment is being implemented, is to choose products that either use a mainstream database management system such as DB2, or that provide interfaces to it. This will make it possible to migrate to new tools while maintaining the analysis and design dictionaries in the same environment.

REPORT CONCLUSION

In this report, we have emphasised that CASE tools are not the solution to every systems development problem. To be effective, most CASE tools have to be used to support a development method that is based on structured techniques. The tools will therefore be only as effective as the methods and techniques are. Nevertheless, CASE tools can provide substantial benefits both in terms of improving software quality and increasing the productivity of development staff.

However, it is necessary to choose tools that address the life-cycle stages where the greatest problems occur. Different types of CASE tool support different stages of the life cycle. No CASE tool yet fully supports all stages, although a few products now partially cover the complete life cycle. Current CASE tools also have other limitations: they offer little help for maintaining existing systems developed originally without CASE tools, and they are unsuitable for use by business users.

The implementation of CASE tools needs to be managed carefully. The investment in hardware and software can be considerable — but the cost of training staff in how to use the methods supported by the tools is even more. It will also be necessary to change the organisational structure of the systems department. CASE tools encourage a greater emphasis on the design and analysis stages, and less emphasis on the programming stage.

Finally, it is necessary to take steps to protect the investment made in CASE tools and the methods they support. CASE tools are evolving rapidly and it will probably be necessary to migrate to a new generation of tools in the future. The tools implemented today should be chosen with this in mind. The most critical decision, however, is to select a development method, and then choose CASE tools to support the method.

BUTLER COX
FOUNDATION

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

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

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

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

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

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

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

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

Forthcoming reports
Mobile Communications
Software Strategy
Electronic Document Management
Human Resources for the Systems Function
Future Information Technologies
Managing Multivendor Systems

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

BUTLER COX FOUNDATION