Management Summary

BUTLER COX FOUNDATION

Computer-Aided Software Engineering (CASE)

BUTLER COX FOUNDATION

Computer-Aided Software Engineering (CASE)

Management Summary
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.

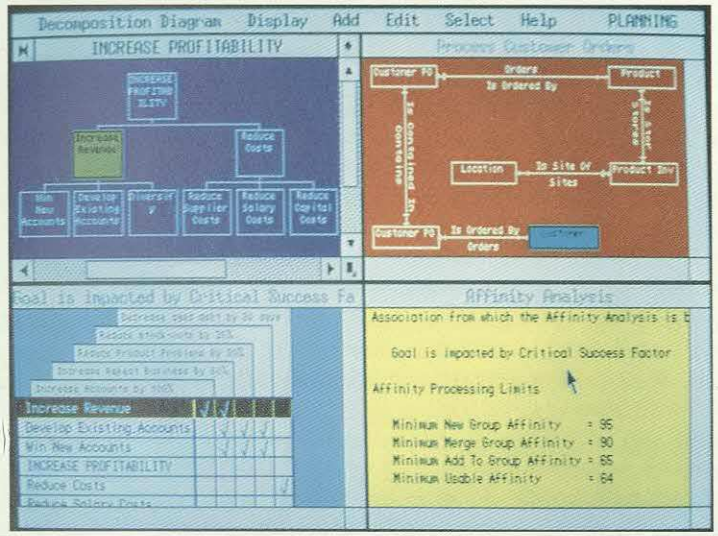
Computer-Aided Software Engineering (CASE)

Management Summary

This document summarises the main management messages from Foundation Report 67, published in December 1988. The full report is available to members of the Butler Cox Foundation.

Considerable media attention has been paid to computer-aided software engineering (CASE), which is 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 reality, the state of the art today is much more modest. Nevertheless, the CASE tools now available can provide substantial benefits, provided they are introduced and used carefully and their limitations are recognised. In particular, the tools should be chosen to support specific development techniques. It is vital to use the right combination of techniques and tools. Inappropriate techniques, or inappropriate tools to support the techniques, will only make the problems of systems development worse.

Figure 1 Typical graphical display produced by a CASE tool



At present, software engineering corresponds largely with structured analysis and design techniques, and CASE tools are the development tools used to automate those techniques. (Figure 1 shows a typical graphical display produced by an analysis/design CASE tool.) Increasingly, however, the term is being used to describe any tool that supports any stage of the software-development life cycle.

Use CASE tools to improve software quality and development productivity

Early experience with CASE tools shows that they can improve both software quality and systems development productivity.

The tools improve software quality in two ways: they ensure that the application systems match the business needs of their users and they improve the technical quality of the systems by reducing the number of software errors. Systems developed with CASE tools meet the needs of the business better because the tools and techniques encourage development staff to place greater emphasis on the analysis and design stages of the software life cycle.

The reduction in software errors brought about by CASE tools 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 — up to 1,000 times higher, according to a recent survey in the United States. CASE tools reduce the likelihood of technical errors, both at the early stages of the life cycle by providing automated support for the structured techniques used during the analysis and design stages, and at the programming stage. They also provide facilities for automatically checking the consistency of the successively more detailed versions of systems designs produced by methods based on structured techniques. Prior to the availability of such facilities, development staff had to record manually the complex interrelationships and dependencies generated by the methods. The inevitable result was that mistakes were made and the technical quality of the resulting application system was reduced.

CASE tools also improve the productivity of development staff at each stage of the life cycle. The greatest improvements at the analysis stage will be achieved where CASE tools are used to automate development methods that have previously been

implemented manually. Improvements of between 10 and 30 per cent are common, and result from the on-screen graphical representation of systems designs (which eliminates the need to draw the designs on paper), and the ability to transfer information automatically from the analysis stage to the design stage. These facilities also reduce 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 at the programming stage by using automatic code generators.

The greatest productivity improvements arise at the implementation and maintenance stages, however, as a direct result of the improved quality of systems. 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. In addition, CASE tools automatically generate complete and consistent software documentation, thus making it easier to maintain systems after they have been implemented.

However, it is not possible to obtain the full benefits from CASE tools unless they are used in conjunction with the method, or development technique, that they were designed to support. CASE tools are not a substitute for structured techniques or development methods; they are a necessary support for them.

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. It is widely recognised that present-day tools have substantial limitations.

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 limited coverage of tools means that user organisations need to integrate different products 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. 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.

Many new 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.

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 without CASE tools. CASE tools will offer significant help in the maintenance of existing systems only when they support reverse engineering — the process whereby systems 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 that have the intelligence to extract the underlying systems design.

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 this stage because their use will continue to require knowledge and experience of structured development techniques. Users will, however, become more involved in the development process. Indeed, some of the most significant benefits of

CASE tools will arise when users work with developers at the analysis and design stages of the life cycle.

Apply CASE tools carefully to realise their promise

Because the implementation of CASE tools will require a significant commitment both of funds and of effort, systems directors must apply them where the greatest benefits can be derived. The cost of implementing CASE tools can be considerable — in excess of $40,000 per developer once the total costs of workstations, software, training, and support from the method supplier are taken into account (see Figure 2).

Select CASE tools to suit the method, not vice versa

Because CASE tools have to be used in conjunction with a development method — usually based on structured techniques — care must be taken to ensure that the tools selected support the particular method. Not every tool supports the techniques 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. (In fact, this is the best way of ensuring the ideal combination of methods and tools, because they can be evaluated together.) The process of selecting CASE tools therefore depends on the methods that are already in use and how effectively they have been implemented. (An earlier Foundation Report, No 57 — *Using System Development Methods* — provided advice about selecting and using methods.)

Figure 2 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)

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 CASE 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 are based on 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 and support available from the tool supplier, rather than by the facilities provided by the tool.

Ensure that tools support the areas of concern

Different types of CASE tool 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. There are three categories of CASE tool — those that cover a single stage of the life cycle, those that cover two or more consecutive stages, and integrated development environments designed to cover all stages. The cost and strategic impact of the tools grows with increasing life-cycle coverage. A typical cross-section of products that fall into each of the three categories is shown in Figure 3, overleaf.

Tools covering a single stage usually (but not exclusively) support the back end of the life cycle, typically the programming and implementation stages. 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. Integrated development environments are designed 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.

Choose CASE tools with an eye to the future

CASE tools are still developing rapidly, and there will be considerable developments during the next few years. (The likely future developments are illustrated in Figure 4, overleaf.) It is necessary to

Figure 3 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.

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 changes will result from the introduction of integrated (or I-CASE) tools, which will be used to develop systems and manage information about the complete applications portfolio. Migrating to such tools will not be straightforward, and thought should be given to ways of protecting the initial investments made in CASE tools.

Three factors are 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 systems designs. 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 uncertainty about standards is likely to continue for several years, creating problems both for suppliers and for user organisations that wish to integrate discrete CASE tools.

Introduce CASE tools in a 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.

Figure 4 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

The application chosen should be one that will provide real business benefits, and should therefore be sufficiently important to the business to ensure that the user department is fully committed to using the CASE tools successfully. Satisfied users will be powerful allies in extending the use of the tools throughout the organisation. It is also important to select an application that is typical of the bulk of the mainstream development work done by the systems department.

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 structured techniques supported by the tools. Doing this will ensure that the pilot application provides a good indication of how well the tools will perform once they have been fully implemented.

One of the purposes of implementing CASE tools is to speed up the systems development process and there is a temptation to set deadlines for the pilot project to prove that this does, in fact, happen. However, considerable slack should be built into the timescale 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 an increase in productivity at the pilot stage. If tight deadlines are set, there is a risk that they will be missed and the result will be a demotivated project team and discredited CASE tools.

Plan 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 structure of the systems department will have to change as a result of implementing the tools. The impact of these changes can be reduced by planning ahead.

Changes in the skills mix

Using CASE tools increases the emphasis on analysis and design skills, and reduces the emphasis on programming skills. (The effect of using CASE tools on the level of effort required at each stage of the life cycle is shown in Figure 5.) 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.

A major consequence of the changing skills profile is a need to retrain existing programming staff in analysis and design skills, in addition to the training required to use new development methods and CASE tools. Some programmers, however, will be unsuitable for retraining, and others will be unwilling to retrain as analysts. Much of their resistance can be overcome by pointing out the ease with which analysis and design can be carried out by using CASE tools, and the increased professionalism that results from the rigorous use of the methods they support.

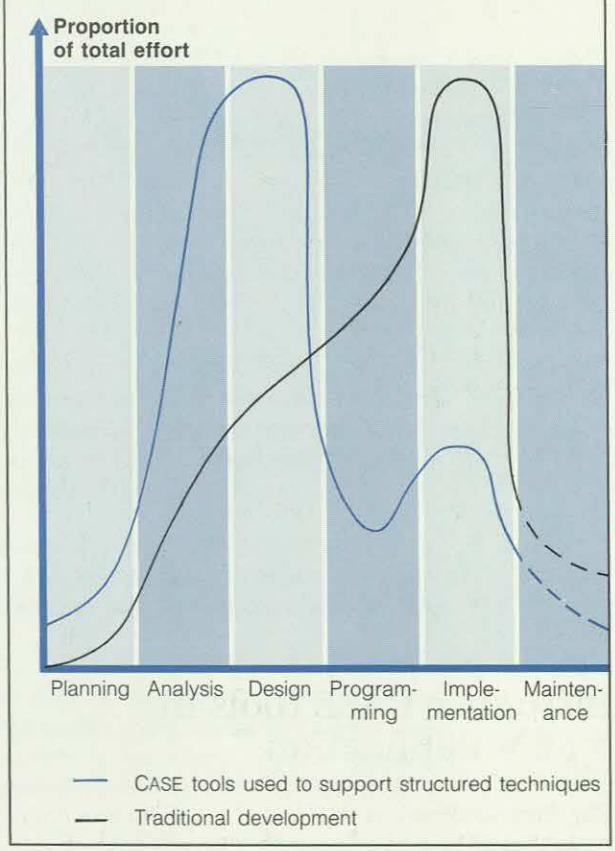
The increasing use of analyst/programmers is one example of the use of CASE tools breaking down the traditional boundaries between different systems development roles. The trend is away from employing staff with specialist technical skills, to staff with business skills and skills in several development 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 should include staff who worked on the pilot project.

Greater 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, although the expected increase of user involvement in developing mainstream applications has not, by and large, occurred. Our

Figure 5 **Use of CASE tools changes the level of effort required at each stage of the software life cycle**



research shows, however, that in many organisations, the implementation of CASE tools *has* resulted in increased user involvement.

There is no doubt that the greater user involvement at the analysis and design stages made possible by CASE tools results in software that matches the users' requirements better. 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.

Once a method and the CASE tools to support it have been implemented, the organisation will be committed to using them for years to come. The investment in training and in setting up the procedures to use the method effectively, together with the investment in the tools themselves, will make it very difficult to change them. The decision to choose a particular combination of methods and tools is therefore one of the most strategically important decisions a systems director has to make because the effects of an inappropriate choice will be evident for a long time. The full report contains detailed advice about how to make the right decision.

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 one of the services provided by Butler Cox. It provides the executives responsible for information systems in large organisations with a continuous analysis of major developments in the technology and its application.

The Foundation publishes six Research Reports each year together with a series of special Position Papers. The programme of activities includes a wide range of meetings that provide Foundation members with a regular opportunity to exchange experiences and views with their counterparts in other large organisations.

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

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

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

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

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

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

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

Italy
SISDO
20123 Milano, Via Caradosso 7, Italy
☎ (02) 498 4651, Telex 350309, Fax (02) 481 8842

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

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