Computer-Aided Software Engineering (CASE)

BUTLER COX

P,E,P

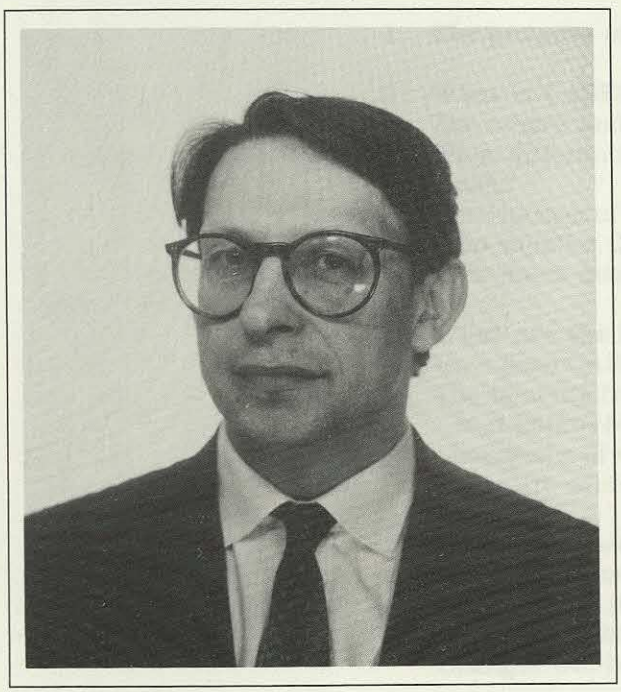
Position Paper 2, June 1987

BUTLER COX

P,E,P

Computer-Aided Software Engineering (CASE)

Position Paper 2, June 1987
by Simon Forge



Simon Forge is a Senior Consultant with Butler Cox, working out of both the Paris and London offices. Since joining Butler Cox in 1986, he has conducted research for a Butler Cox Foundation report on system development methods.

Before joining Butler Cox in 1986 he was an independent consultant, working for such consultancy groups as Arthur D Little and Ewbank Preece as well as software houses such as Steria in France.

Previously he was a Principal Consultant with Ewbank Preece, specialising in the convergence of computing with telecommunications. His consultancy work included:

— Project supervision on behalf of a third-world client of a Japanese contractor for software and hardware for an airport radar system, including contract negotiations.

— Software audit of a distributed system for power-station operation.

He has also acted as project manager for the design of ship automation systems as well as for the development software and hardware for meteorological data processing.

A graduate in automatic control from Sussex University, Simon Forge has MSc and PhD degrees in control engineering and digital signal processing. He is a Chartered Engineer, and a member of the Institution of Electrical Engineers. He is the author of various papers, the most recent being on software engineering for distributed processing.

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

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

BUTLER COX

P,E,P

Computer-Aided Software Engineering (CASE)

Position Paper 2, June 1987

Contents

Chapter 1
The purpose of this paper

Over the past few years a new term has been added to the jargon of system development — 'CASE' — an abbreviation for computer-aided software engineering. It is used to describe a certain kind of software development aid, analogous to the computer-aided design and engineering aids used in other branches of technology. CASE originated from the initiative of the US Department of Defense to support the writing of software for complex real-time systems, using the ADA language.

More recently, however, suppliers of proprietary CASE products have started to promote their wares to those responsible for conventional data processing applications development. A number of users have started to make use of such products. At the same time, there is an ever increasing range of other proprietary system development aids (methods, tools, workbenches, ...) appearing in the marketplace. Each supplier claims their own product is *the* answer to the system builders' problems. There is some confusion amongst both potential users and the suppliers of the respective roles and virtues of these products. It is not obvious how CASE relates to them. Is it really something different or is it just another name for a workbench or advanced system building tool?

Because of its origins, CASE may appear to be less relevant in the normal commercial systems environment than the other products developed to suit that environment. But our view is that CASE does warrant serious consideration and is likely to become a valuable aid in the future.

The purpose of this report is to present a snapshot of the state-of-the-art in the use of CASE and to attempt to answer such questions as:

1. What is CASE? How does it differ from the other system development aids currently being used?

2. What are the benefits from using CASE?

3. Who is likely to find CASE most useful and how should they choose which CASE product to use?

4. How is CASE likely to develop in future?

The paper draws on the author's research and personal experience in software development and management.

Chapter 2
What is CASE?

The first question we attempt to answer in this paper is, "What is CASE?" It is the application of a particular kind of computer-based tool to the system development process. It is normally implemented by buying a proprietary system consisting of a set of software tools and, sometimes, the hardware on which to run them.

CASE systems are not the only such aids that may be applied by the system builder. Below, we briefly review their origin and particular nature which distinguishes them from other tools. We illustrate their various facilities by reference to four typical CASE products. Finally, we give estimates of the costs that may be incurred by using CASE.

THE ORIGINS AND DEVELOPMENT OF CASE SYSTEMS

Since the 1970s there have been many attempts to make software development more consistent, reliable, and manageable. Various system development methods based on the concept of the system life cycle have appeared. They consist of standard procedures to be followed, techniques to be used, and standard documentation. The methods define *what* has to be done and the tools generally define *how* to do it. They primarily help in improving the quality of the resulting software and make the planning and control of the development process more manageable and less dependent on the personal skills of the staff.

Various development tools have also been invented to support the methods. They help the analysts and programmers execute the various activities and use the techniques forming the method. They are essential to the successful use of the methods and are the primary source of productivity gains. Unsupported methods can fail because of the clerical drudgery they can impose if the routine administrative procedures (such as documentation) are not automated.

CASE systems originated as a collection of tools used to support one kind of system development — realtime systems in the defence environment. Besides supporting the development process tasks directly, they also embody the concept of assembling software from common building blocks in a similar way to that used for constructing different electronic assemblies from a range of standard components. Their purpose is to make the software development process more reliable and repeatable.

CASE systems have developed over time typically by bundling together tools that were originally designed to be used individually and to support only one phase of the system building process. More recently, integrated toolkits have emerged in which the interfaces between the separate elements are fully automated.

CASE SYSTEMS HAVE WIDENED IN SCOPE

The scope of CASE tools has widened and is still continuing to do so. They are no longer confined to realtime or embedded-system development. They can now be used for large-scale administrative applications development.

Their scope is extending in the following ways:

- From being used for realtime, dedicated microprocessor systems running under monitors or specialised operating systems to being used for commercial data processing applications using Cobol.
- From offering support to only the programming phase, to supporting requirements' analysis, system specification and design, and maintenance.
- From treating data as a subsidiary concern to treating data and databases as the primary concern.

WHAT DIFFERENTIATES CASE FROM OTHER SYSTEM DEVELOPMENT AIDS?

The two key characteristics that differentiate CASE from other previous assortments of development tools are:

- The integration of the tools to serve the whole development life cycle in a consistent and homo-

genous way. A common user interface is provided for the developer.

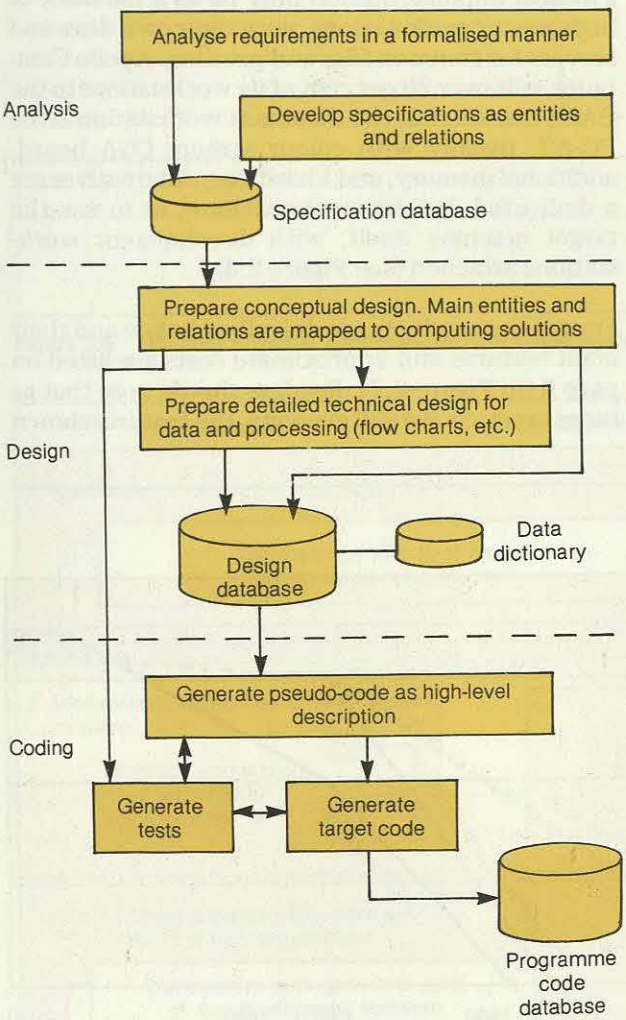
— The decoupling of the analysis and design of the system from the use of a specific target programming language.

The main factors that CASE systems incorporate to achieve these characteristics are:

— Integration of the various development phases so that handover of information from one to the next is performed automatically, without re-entry of data, or any human intervention to set up interfaces or translate formats. All the tools should be completely compatible.

— Use of a central database (or 'encyclopaedia') of all the design specifications to unite the tools and phases (see Figure 2.1). This central database may house its own data dictionary as well as the design specifications and program descriptions (see Figure 2.2). It is used to validate and check throughout development, acting at each phase as a repository for the design advances and for validation. Program tests may be generated from it.

Figure 2.1 CASE systems use a central database to unite the various tools



The CASE design encyclopaedia is the collation of databases from each of the three levels

Figure 2.2 The dictionary or encyclopaedia of system structure and specifications is the hub of CASE systems

	Typical encyclopaedia contents	
Element	*Definition*	*Examples*
Object	Function Person	Service Data Processing Information storage
Attribute	Information provided for each object	Name, type, definition, last update, comments
Relation	Relations between objects	Data domains containing objects Processing using an entity

— Reliance on graphics to produce a consistent user interface for all tools right through the life cycle.

— Support for database access. This varies considerably by product but the trend is to provide data-structure and access support for several types of database.

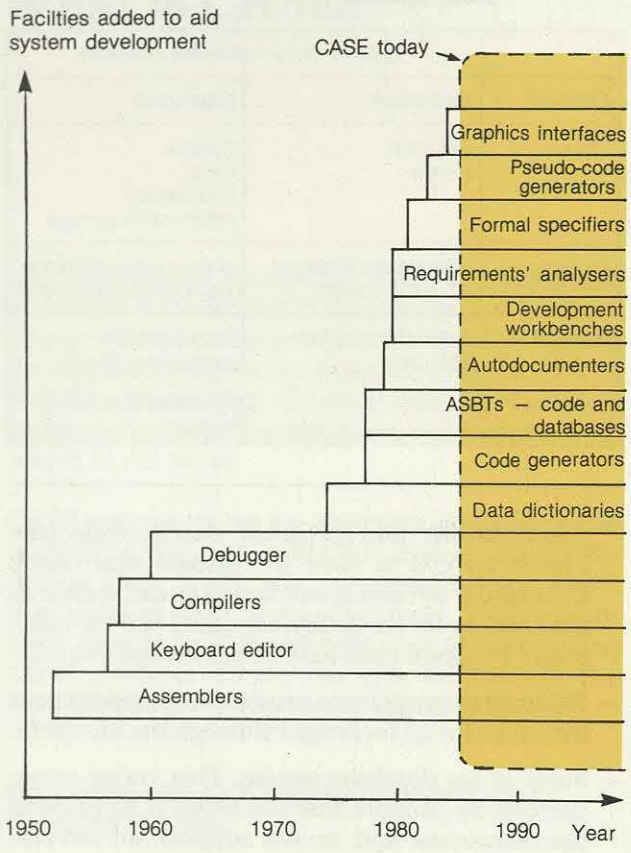
— Decomposition of complex system structures using object-orientated analyses — entities, relations, objects are the building blocks at all levels of the creative process.

— Use of intermediate levels of abstraction to describe the system. This can be at:

- The design specification stage, at an abstract conceptual level, close to the definition of business requirements.
- The program-design stage, using a pseudo-code above the normal high-level programming language. This allows freedom to generate code in any target language or environment. These tools for formalising design in a procedural way, like a high-level program, can be used for 'reverse engineering' — the ability to generate specifications from code. This is valuable in maintenance, for understanding and documenting changes. Existing code, not created on the CASE system may even be 'reverse engineered' in some systems.

The evolution from the earliest programming aids to the facilities offered by CASE systems today is shown in Figure 2.3 (overleaf). But CASE products are still evolving. Figure 2.4 (overleaf) shows how

Figure 2.3 The evolution of CASE by the integration of tools via a design database and a common graphics interface



their current state of development compares with discrete development tools and programmers' workbenches.

DESCRIPTIONS OF FOUR TYPICAL CASE PRODUCTS

CASE products vary enormously in scope, price, and application area. We concentrate on those intended for the information systems department of a large corporation.

CASE SYSTEMS CAN BE BASED ON WORKSTATIONS ALONE OR ON A DEVELOPMENT MACHINE

The CASE system is made up of a number of software modules residing in a workstation, a development machine, or both. They provide the presentation display in the workstation, with a windows manager, and modules for each phase based around the design encyclopaedia. Figure 2.5 (opposite) shows an ideal configuration for a CASE system.

Physical implementation may be as a network of high-power workstations, with their own discs and servers for common files and printing. Apollo Computer sells over 20 per cent of its workstations to the CASE market, but the dominant workstation is the PC-AT, usually with colour screen, CGA board, additional memory, and a hard disc. Alternatives are a dedicated development machine, or to use the target machine itself, with development workstations attached (see Figure 2.6).

Four major products are described below and their main features and approximate costs are listed on page 6 (in Figure 2.7). Readers should note that as these products are still evolving, the features shown

Figure 2.4 The relative maturity of system development tools

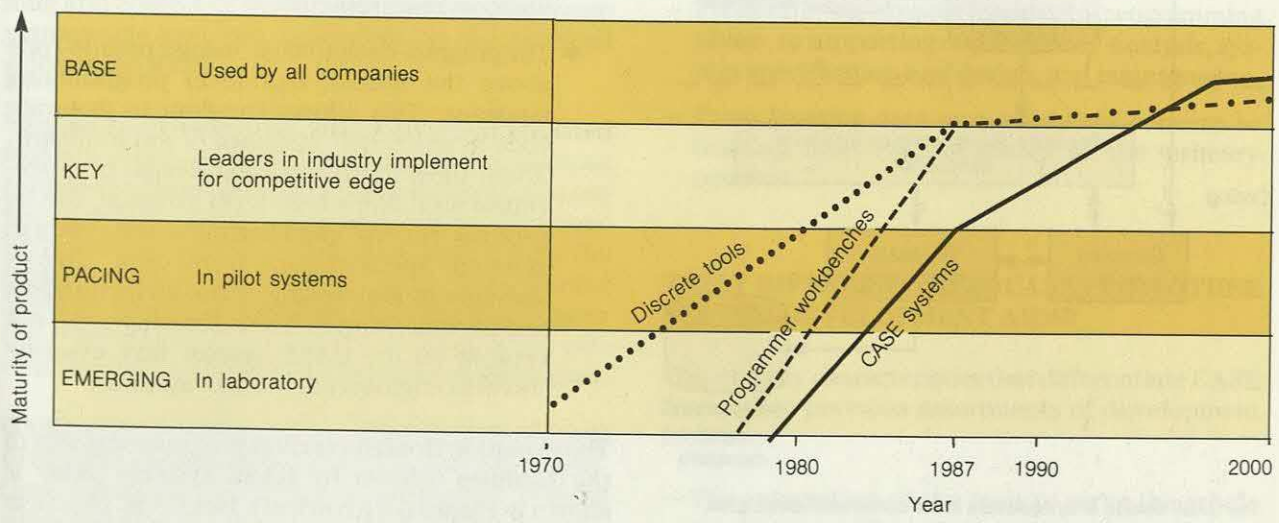


Figure 2.5 The ideal software configuration for a CASE system

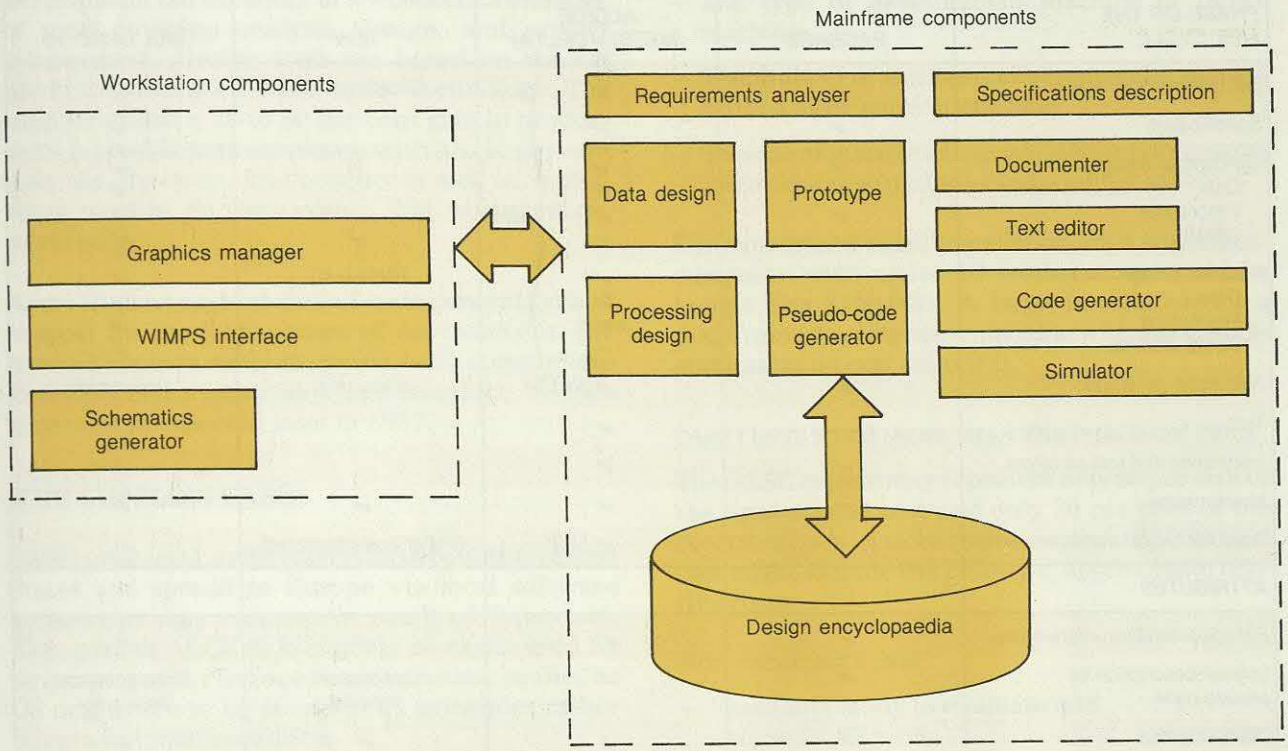


Figure 2.6 Possible physical configurations for CASE systems

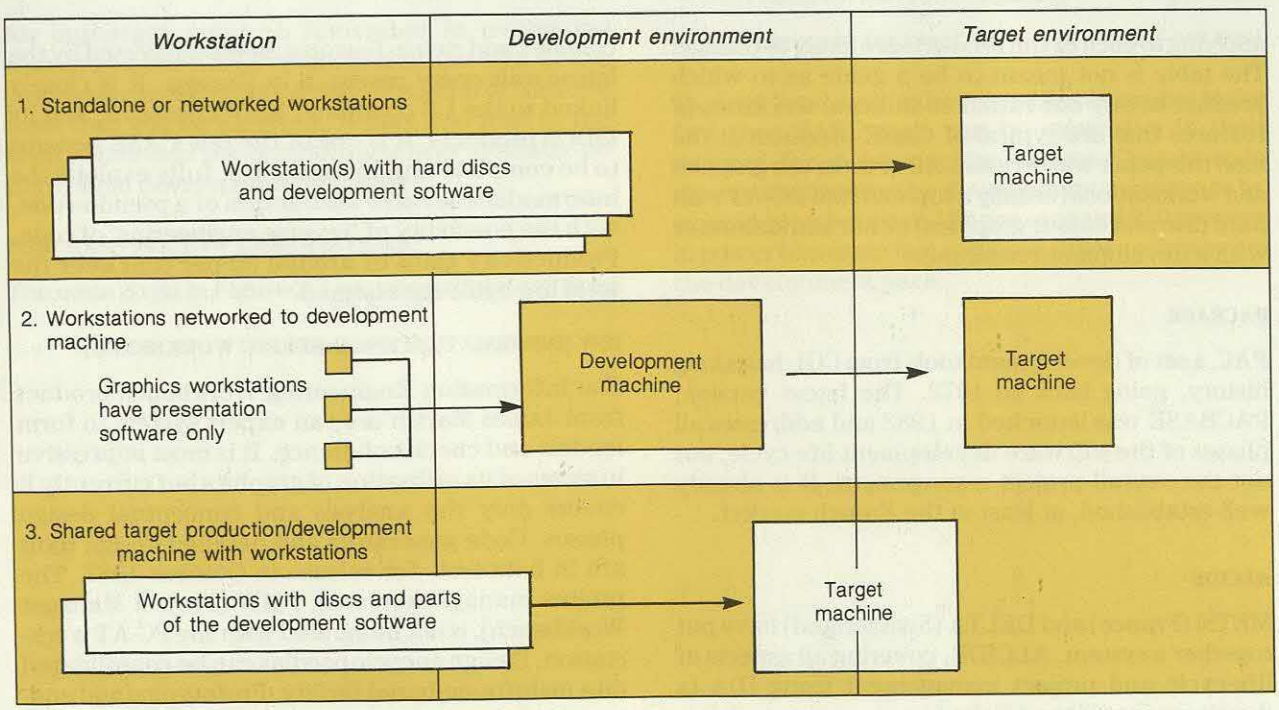


Figure 2.7 Four typical CASE products that cover all or a significant part of the life cycle

PHASE OF THE LIFE CYCLE	*PACBASE*	*ALCIDE (METSI + DELTA)*	*IEW*	*MULTIPRO V3*
Requirements analysis	✓	✓	— (Meeting support)	✓
Specification	✓	✓		✓
Conceptual design	✓	✓	✓	✓
Data	✓	✓	✓	✓
Processing	✓	✓ IDA	✓	✓
DBMS	✓	✓	✓	✓
Detailed design	✓	✓	(Beta-test)	✓
Data		✓		✓
Processing		weak		✓
DBMS access		✓		✓
Auto-code generator	✓	✓ DELTA	(Beta-test)	(late 1987)
Test		✓		
Integration and test on target	✓	✓		—
Maintenance	✓	✓	✓	✓
Supports project management		✓ MCP	(PMW, not integrated)	✓
ATTRIBUTES				
Encyclopaedia/structure-base	✓	✓	✓	✓
Logical description or pseudo-code	✓	✓	Conceptual models	Specifications base only
Cobol-oriented	✓	✓	✓	✓
Method-oriented	MERISE, Yourdon	MERISE, others	SDMS, MERISE, IEM, AXIAL, others	MERISE, SADT, EXPERT, others
Configuration	Target or development mainframe	Standalone development plus PC-AT workstation	Standalone PC-AT workstations; optional central consolidation	Standalone development machine on networked PC-ATs
Approximate cost of typical configuration	Small $150k Large $450k	Large (26 workstations) $400k, entry-level $200k	Workstation $12k Central facilitity $100k	$40k plus $15k per workstation

Key: ✓ = phase the product supports or attribute it has.

applying to each of the products are likely to change. The table is not meant to be a guide as to which product to buy but rather to indicate the kinds of features that are typical of CASE products at the time this paper was written. All of them use graphics and workstations (usually a top-end IBM PC-AT with hard disc and colour graphics) either standalone or with a development machine.

PACBASE

PAC, a set of development tools from CGI, has a long history, going back to 1972. The latest version, PACBASE was launched in 1983 and addresses all phases of the software development life cycle, but not the overall project management. It is already well-established, at least in the French market.

ALCIDE

METSI (France) and DELTA (Switzerland) have put together a system, ALCIDE, covering all aspects of life-cycle and project management using IDA (a development aid) and Delta (a code generator). It is complex and rich in features, being supported by the latest university research in Europe. It is closely linked to the US company, META Systems, and its ISDOS products. It is one of the few CASE systems to be complete and integrated. It fully exploits the intermediate level of abstraction of a pseudo-code, with the possibility of 'reverse engineering' of code. Productivity gains of around 50 per cent over the total life cycle are claimed.

IEW (INFORMATION ENGINEERING WORKBENCH)

The Information Engineering Workbench product from James Martin uses an expert system to form models and check coherence. It is most impressive in terms of its utilisation of graphics but currently it covers only the analysis and conceptual design phases. Code generators and detailed design tools are in beta-test, for release in October 1987. The project management tool, PMW (Project Manager Workbench), is not integrated with the PC-AT workstation. Design encyclopaedias can be consolidated on a mainframe-based facility. Prototyping and end-user interfacing are well supported.

MULTIPRO V3

A set of distributed PC-ATs provide MULTIPRO's development environment of a workbench and a set of tools covering analysis, design, and project management. Design tools are based on the US product Excelerator from Index Technology. The supplier claims a 20 to 30 per cent gain in productivity is possible in these phases with 8 to 10 per cent over the life cycle. The product is well accepted, being used in 50 sites with 1,500 workstations, worldwide.

Apart from its current lack of code generation and support for the later phases of development, the product appears well integrated, with comprehensive tools and a consistent user interface. A code generator is expected later in 1987.

WHO SUPPLIES CASE?

CASE products have evolved largely in the United States and spread to Europe via local software houses that may enhance or resell a US product. IDA, part of ALCIDE is entirely re-engineered by developments in European universities, so that its US origins are to be found in its principles rather than in its implementation.

The market is currently made up of several small companies that specialise in CASE, with a few larger software service companies that have developed a set of 'traditional' products, created in the early seventies, into an integrated environment.

HOW MUCH DOES CASE COST?

An important point to remember in evaluating whether to use CASE or not is that the cost of the proprietary product forms only a fraction of the total costs that will be incurred. It may also lead to the need to increase the computing power available on the system development machine.

PURCHASE PRICE OF CASE SYSTEMS

The costs given in Figure 2.7 are approximate. Prices vary due to differences in configuration:

- The type of development machine or target machine.
- The number of workstations connected; there is often a price per workstation.
- The size of installation; most CASE environments are modular so options can be added at will.

For example, a small installation on a superminicomputer with under 10 workstations could be bought for $150,000. A large installation on a mainframe development machine with more workstations could cost $500,000.

CASE COSTS MUCH MORE THAN THE PURCHASE PRICE

The CASE system may represent only 50 per cent of the total capital cost and only 30 per cent of the recurring costs. A more realistic estimate of the total cost might include the following approximate relative costs.

Non-recurring items:

— Feasibility study to evaluate and choose CASE tools	10 per cent
— Installation of tools, including manpower	20 per cent
— Training	20 per cent
— Purchase of CASE tools	50 per cent

Recurring items, per annum:

— Overhead on machine capacity	70 per cent
— Maintenance contract	30 per cent

The overhead item in the recurring costs is due to the fact that CASE tools can be expensive in their demand for more machine power. Certain code generators require the development machine to be expanded by between 150 per cent and 250 per cent in power to sustain fast response times and maintain the development pace.

Chapter 3

What are the benefits from CASE?

As we have shown in Chapter 2, the costs of implementing CASE are substantial. It is therefore now appropriate to try and evaluate the benefits. We also need to consider the limitations of current CASE products in achieving these benefits.

HOW DOES CASE PERFORM IN PRACTICE?

We have already outlined in general terms the benefits that CASE (and other development aids) are intended to achieve. Apart from improved productivity, measured in lines of code (or function points) per man-day, these other benefits are difficult to quantify, particularly in financial terms. We therefore feel it helpful to first refer to the actual experience of four user organisations to describe the impact of CASE. Below, we outline the recent experience of four different European organisations, each using a different CASE system. They are:

- A transport authority that has decided to integrate IDA, a development tool, with DELTA, a code generator, as the programming aid in the form of the product ALCIDE referred to above.
- A pharmaceutical and chemical company that has implemented various generations of PAC over the last 15 years.
- A manufacturer of electronic equipment that uses the Information Engineering Workbench (IEW).
- A major bank that uses MULTIPRO.

EXPERIENCE WITH ALCIDE IN A TRANSPORT AUTHORITY

A major transport authority with several different computing environments decided to integrate IDA, covering the development phase, plus a code generator, DELTA, as the programming aid. From these, the integrated product, ALCIDE, referred to in the previous chapter, has been produced.

Major problems promoting investment in CASE were:

- A heavy maintenance load which new programmers had difficulty in serving.
- The need to change computers every three to five years, with software 15 years old.
- The fact that users were demanding more application power.
- Further new system demands as more power became available.

To solve these problems tools were sought which would:

- Allow design by the users.
- Allow implementation by programmers.
- Provide documentation aids.
- Support maintenance of existing software as well as new developments.

No fourth-generation language is yet used, though a query language for the specifications database is used. The Cobol code generator was chosen as various target machines, from DEC MicroVAX to IBM, and Bull DPS7/8, HP, Philips, and Olivetti, can all be served from the same pseudo-code source. IDA was chosen because its specification language offers high-level design. The benefits are:

- It can be decentralised.
- It provides standardised documents.
- It offers experimental validation.

The database design workbench can be used for several database types — Codasyl, sequential indexed, and relational. ALCIDE's CASE environment follows MCP for project management and MERISE for the development method. Formal specifications, with object-relation structures are used to provide models for simulation and prototyping. Graphics output, now via DEC MicroVAX, is to be ported to a PC-based workstation, under MS-Windows. Experience so far is limited but the major costs of assembling and implementing a new product are considered worthwhile in view of the reduction in maintenance anticipated:

- Reversion from application to pseudo-code for new and existing programmes.

- Complete documentation.
- Generation of standard documentation for existing applications by reversion to pseudo-code.
- Three levels of cross-reference to validate from maintained code up to specifications.

Training courses of two weeks each on IDA and DELTA are needed for staff.

The project is only in the pilot phase now, with major effort having gone into formation of the CASE platform on a BULL DPS 8/70 with terminals, and on standalone MicroVAX workstations.

ALCIDE will be used for:

- Development of large projects involving Codasyl databases and Cobol for commercial data processing.
- Design of other large systems and microcomputer projects.
- Applications that require automatic code generation, without a design study.

ALCIDE has brought rigour in design and architecture of systems plus automation of documentation. Productivity gains, especially in the maintenance phase, are yet to be measured; maintenance is the area in which spectacular economies are hoped for.

The main difficulties have been with the programmer interface on the terminals. (The graphics workstations are better.) Moreover, the environment is incomplete. Tools to design screen forms and other implementation tools are still to be added.

Future development will be based on small test projects. In 1988 they expect to reformulate the development and maintenance strategy following the outcome of the pilot studies.

Technical developments will include:

- Re-engineering the graphics interfaces on the workstations.
- Providing interfaces to fourth-generation languages, and relational DBMS.
- Providing models and simulated animations.

EXPERIENCE WITH PACBASE IN A PHARMACEUTICAL COMPANY

A major pharmaceutical and chemical company has been using various generations of PAC products since 1972. Operations are based on IBM 3081s and IBM 3083s at four sites with 420 terminals, 250 video terminals, and 70 PCs, in various European countries. A dictionary with 18,000 references has been built up, and there are 3,500 files and 5,500 programs of 1.4 million PAC lines. A study in 1983 of the computing department efficiency concluded that the predicted increase from 2 million to 6.2 million lines of Cobol code would require an increase from 30 up to 78 programming staff in development and maintenance. Installation of PACBASE in 1983 has allowed the department to increase productivity three-fold to 163 lines per manday. The programming team has decreased to 28 people but has maintained the existing code and developed 4.2 million lines of Cobol.

Although tools are only used in programming, principally in rapid code generation, actual payback time of the PACBASE tools was 16 months. The company was in a strong position to exploit new tools as its basic programs were already in PAC form and a data dictionary existed. PAC-TP and PAC-Batch had been in use since 1980.

The major advantages are seen as ease of training, documentation aids, standardisation of procedures, and ease of use, as well as improved productivity. The toolset has some problems however. It is not designed for end users, only professionals. The performance and ease of use of PAC-code are not at the level of fourth-generation languages such as Focus, now being tried. Moreover, the data dictionary structure that comes with the kit does not have the rigour of a modern DBMS. This makes the task of purging and update of the data dictionary even harder. This is seen as a significant problem.

Ease of training is also seen as a particular advantage — a programmer can be trained on the toolset in 25 days. A programmer for the PAC environment who has a good secondary education and two years of higher education can become proficient within a few months of leaving college.

Costs of installation are in the order of:

Capital:	
Feasibility study	$6,000
Purchase PACBASE software	$150,000
Installation	$90,000
Total capital cost	$246,000
Recurring costs:	
Overhead on machine time, and backups	$50,000 pa
Maintenance	$23,000 pa
Total recurring cost	$73,000 pa

The above example covers program generation. The next step is to attack analysis and design and the company hopes installation of PAC-Design will replace the paper-and-pencil approach still used. This

will lead to tools that designers and users can use together.

EXPERIENCE WITH IEW IN AN ELECTRONICS MANUFACTURER

A manufacturer of electronic controls and equipment with over $1 billion turnover and 14,000 staff, has computing centred on IBM 3083, 3380, and 4381 machines, plus decentralised DEC VAXs and IBM 4341s, with 900 screens and 200 PCs in 65 sites in Europe. The information systems department has decided to use James Martin's Information Engineering Workbench. The tools are in initial testing phase now and are incomplete; only analysis and design are covered. IEM — a project management tool — and a code generator are now in beta-test, and is due for release later in the year. The tools support IEW methods. AXIAL (from IBM) had previously been tried and abandoned.

Tools are currently used at the interface with end users. The methodology used calls for heavy involvement from the system users. This is considered well worthwhile: if a user is not prepared to devote resources, the project is abandoned. The approach imposes structure and discipline on design and analysis, forcing end-user validation. Sub-groups of five to ten people are formed under two effective project leaders, one from the computing department, one from the user area. Tensions within this structure are inevitable but lines of responsibility into both departments have been found necessary. Up to ten sessions each of two to three days are necessary. Data structures are designed and added to a master encyclopaedia. Definitions of internal and external data flows, processing, and entity decomposition are carried out in these sessions. Heavy reliance is placed on the colour graphics PC-workstations. With these, a user can easily formulate and change proposed structures during the definition dialogue. Users find it easier to articulate needs and change prototype schematics. In this situation, the analyst becomes a consultant. Insistence on fixed-length meetings is aimed at achieving consensus between user and development-consultant.

The problems encountered include:

- Getting users to accept the need to structure data first.
- Tools are available for only the analysis and design phases.
- Events are not yet treated as objects.
- Output documents take a long time to formulate and print on the configuration used.

So far, four encyclopaedias have been built up for major projects involving 860 data flows, and 270 entities with 1,190 attributes.

The next step is to add a code generator.

EXPERIENCE WITH MULTIPRO IN A LARGE BANK

A major European bank, created from the fusion of two separate banks, found itself with many types of equipment, and a wealth of methods. Over 1,000 terminal users supported on IBM 3081, Unix, and other computers were demanding new applications. A large legacy of old programs had been bequeathed from the two previous banks. These systems needed integration or replacement; there was no documentation, nor dependable staff support as body-shop programmers were the norm.

The first step was to install a development method and to provide support for it. Two products, MERISE and EXPERT were chosen. A CASE system to formalise design, and that would support the method, force consistency and standardisation, and provide documentation aids, was sought via a series of discussions with suppliers. MULTIPRO was chosen as it was new, but not too complex. It could easily be assimilated by a development team which was already overloaded.

MULTIPRO is now used to support a range of tools. Its WIMPS ('windows, icons, mouse and menu pull-down system') interface is proving to be an excellent productivity aid. The tools are used in analysis and high-level design. Physical design aids and code-generators are still to be added.

A formal approach to installation was adopted:

- Awareness was created in top management.
- A team of four programmer/analysts was formed to undertake pilot trials on a standalone workstation for several products and configurations.
- Results were evaluated and a final choice made.
- An investment plan was presented. This consisted of a three-year budget of over $1 million to be spent on:
  - Workstation software and a network.
  - A workbench machine for MULTIPRO.
  - MULTIPRO software for the workbench.
  - A data dictionary.
  - PC-AT hardware for 100 developers.
  - Support for the MERISE method.

Additional installation costs include staff time, estimated at $400,000 a year over three years. It is too soon to evaluate the return on this investment.

THE BENEFITS FROM CASE ARE MAINLY IN HIGHER PRODUCTIVITY AND IMPROVED USER SATISFACTION

While the above case histories throw some light on the benefits to be expected from CASE, hard practical experience is limited by the novelty of using such products in a commercial environment. We therefore now draw some conclusions on the benefits that CASE should deliver, at least in theory.

Productivity and user satisfaction are the two important parameters in measuring the information system departments performance. CASE can improve both these parameters.

PRODUCTIVITY IS INCREASED IN A VARIETY OF WAYS

Higher productivity may be assessed as direct savings in analyst, programmer, and end-user time. However there are a large number of less tangible benefits that contribute to, or are additional to, better productivity.

Total savings

Suppliers' claims for total savings vary enormously. All suppliers say each installation is unique and generalisations cannot be made, as products have to be adapted to the user company and its methods. However, general claims of 10 per cent to 50 per cent direct savings over the whole development life cycle are common. One of our case-history users notes an impressive 60 per cent saving but this is in a carefully controlled, centralised environment with a long history of consistently using the predecessors of the CASE system.

Direct savings

The total savings quoted above come from a number of contributions that speed up development:

- In requirements analysis, the interaction with users can be accelerated by graphical aids.
- In system specification, a library of design specifications can validate later phases and be used for maintenance to understand the functional flow. Notions of entities, relations, and objects are used to build up a functional definition, and are well understood by end users.
- In system design, CASE graphically supports rapid generation of high-level structures from specifications for the data design and processing design, with clear indications of database requirements and validation checks.
- In prototyping, examples for end-user validation can be introduced at several phases to demonstrate the programmer's or analyst's understanding of the users' needs in a form users can more easily relate to.
- In detailed system design, the high-level concepts are translated into physical designs with a wealth of checks and rules, matching the target environment to the design specifications. A high-level pseudo-code description may be generated.
- At any stage, a file and documentation, describing the relations and composition of the software, may be automatically generated in a formalised manner.
- In coding, auto-code generators may translate the pseudo-code description to produce Cobol for the target environment, with savings in time and errors.
- In maintenance, CASE has its greatest benefits. Documentation, standardised structures, and high-level descriptions aid comprehension of old code. Validation of code against specifications checks for conflicts in the evolution of the system. The most advanced systems offer 'reverse engineering' — the ability to take old code, whether generated on the system or not, and reverse the process to provide a high-level specification from which new code can be generated for a new target machine, or to control extensions without introducing conflicts.

Less tangible benefits

Less tangible benefits include:

- Having a central resource holding all the company's programs in a high-level representation.
- Standardisation of the development approach, with a uniform human interface in all the development phases, so that transfer of development or staff is made much easier.
- Training time and end-user interaction time are reduced by the graphics and mouse-based interfaces.
- The end-user involvement with systems development and the systems department is improved as support for such interaction can be specifically included. Validation of system design by prototyping is an important technique here.

BETTER USER SATISFACTION COMES FROM THE IMPROVED USER INTERFACE

Few suppliers claim definite benefits from the improved user interface in CASE systems, but improved user satisfaction due to CASE appears to rest on:

- Faster development.
- Lower maintenance charges.
- Validation and tangibility of output during the design process, especially where prototyping can be used.

— Raising software production from the technical detail level to the level of business entities and relations, so that the gulf between users and the systems department is narrowed.

LIMITATIONS OF CASE

CASE systems are not a complete answer to the productivity problem. They are limited, in that:

— Their flexibility and versatility is restricted. They are currently oriented to centralised software production on larger projects. For small projects they can be cumbersome and expensive.

— They are still principally aimed at a single programming language, Cobol.

— They are not intended for complex multiple-machine architectures.

— The current CASE systems described in Chapter 2 are not intended for realtime systems.

— Their portability may be restricted. CASE systems are aimed at particular development environments, usually IBM, with DEC falling a poor second. On the target side though, some products specifically exploit the higher level abstraction to produce code for a range of target machines (for example ALCIDE).

— The cost of an entry-level system may be prohibitive.

— A particular management style and company structure are needed, where centralised operation and methods are in place.

— Training is needed to exploit CASE fully. In certain organisations, a complete rethink of the creative process is necessary, as it can limit design options.

— For very high performance systems, such as high-volume transaction processing, it may be necessary to go outside the languages and tools of the CASE system to a specialist environment.

— Database interaction may be limited. Most CASE systems offer outputs and interfaces to the major IBM databases but few do to all sequential indexed, relational, and Codasyl databases.

THERE ARE PROBLEMS IN IMPLEMENTING CASE

The problems experienced in implementing CASE occur in three areas:

— Technical.

— Company structure and systems department organisation.

— Training and acceptance.

Users appear more perturbed by the first than the other two problem areas.

TECHNICAL PROBLEMS

Technical problems include:

— The run-time power of a central development system may need to be increased much more than expected (perhaps by 250 to 300 per cent).

— Using a large CASE system on a development machine not demonstrated by the supplier may lead to unpleasant surprises in performance.

— Document output can be slow if the preparation and print facilities are inadequate.

— Interfacing CASE systems to existing data dictionaries may slow performance.

OTHER PROBLEMS

Training has proved a problem where there is strict delineation between end users, analysts, and programmers. Programmers, analysts, and end users tend to become more closely involved with each other when using CASE.

Acceptance may be difficult. CASE will mean very significant changes in system development procedures. Convincing all the dp staff to use the new approach and the tools may not be easy. Some CASE users express a hope that 80 to 90 per cent of the department would endorse the methods quickly, but the rest would take time to convert to the new methods.

Depending on the methods, various levels of commitment are needed from end users. Responsibility for specification errors shifts from the information systems department alone, to the information systems department plus the end user when using CASE systems. As end users become aware of this, they may resist it. One CASE system manager decided that if the end user will not commit a project manager during the requirements analysis phase, then he will not develop the application.

A managerial and technical problem is that of security. Storing the references, definitions, and designs of every piece of software and data in one location poses significant problems of access control and physical backup.

Chapter 4
How to choose and use CASE

Having reviewed the costs, benefits, and limitations of CASE, we now address the questions of whether, and how, to make the best use of CASE. Clearly, the answer to these questions depends on the nature of the organisation and its system development needs and problems. In the following sections we give some guidance on:

- Who should use CASE and how the decision on whether to use CASE may be made.
- How to choose the most appropriate product if it is decided to use CASE.
- How to manage the implementation of CASE.
- What the implications of using CASE may entail.

WHO SHOULD USE CASE AND HOW TO DECIDE

CASE systems are most appropriate for users that:

- Have a major backlog of applications and a heavy maintenance load.
- Have a computing load that is increasing.
- Use professional staff for centralised software development.
- Are prepared to invest large amounts in both money and skilled staff time.
- Have a strong vision of the future and are able to identify development and maintenance as a business problem that must be turned into an opportunity.
- Have a leader and top management strongly behind the need to change.

Those who expect that their business will need a 100 to 200 per cent increase in the volume of applications over the next five years should consider CASE seriously. Critical questions in deciding whether to use CASE include:

- Is it really needed? What is the justification?
- What budget is available? Entry-level CASE systems require at least $100,000.
- Can the existing development environment support CASE, or is a new one needed?
- Will the organisation accept a centralised repository for all the company's specifications and pseudo-code?
- Is it prepared to regenerate or coordinate existing code to be coherent with the new code?
- How can existing systems and code be maintained during the transition?
- Are the resources available to install CASE and train staff?
- Are the 'politics' favourable to a major change in the development process with more end-user involvement?
- Is the management style of the company in line with this form of development?
- What is the application portfolio? If it is a mixture of realtime and commercial dp, then the former may not be well accommodated. Packages and existing system building tools will have to be treated with care.

The key question is: "Can you afford to wait?" The CASE systems market is not mature but in the next two years major advances are expected. Deferring any decision for one to two years may be prudent, but initial planning could start now if you have serious quality, backlog, and user-dissatisfaction problems, as these are symptoms of a potential crisis.

The alternatives to using CASE now are:

- Adopting an advanced system building tool, perhaps one that involves end users heavily, if support can be coordinated.
- Integrating your own set of tools and methods, if the staff resources are available.
- Preserving the status quo, and waiting until CASE is more mature, some time in the 1990s.

CHOOSING THE PRODUCT TO SUIT YOUR NEEDS

Products currently available are limited and often only partially cover the diverse needs of the infor-

mation systems department. No proprietary method will cover all a company's needs. CASE products which are particularly suited to attacking the most serious development problems should be selected.

In selecting a product, questions to be considered include:

- What productivity increase can the supplier demonstrate?
- Will it support existing code, documentation, specifications and operating systems, as well as new ones?
- What is the net effect on development costs and on the life-cycle costs of software?
- What methods does the CASE system support?
- How will it support integration of applications from different areas of the business in the future?
- Which databases does it interface to now, and will it interface to in the future?
- What is the position on any advanced system building tool already in use?
- How much end-user involvement does it allow, or call for, and how much scope do you want to give end users?
- On what development environments does the CASE system run satisfactorily.
- What packages will it interface with, already in use or planned?
- How does it fit with your mix of computing (batch, TP, and so on)?
- How easy is it to use for analyst/programmers and end users, and what training is necessary for each (at what cost)?
- What security measures for access and backup are provided?
- What impact will it have on your existing skills and the development process? Can they be re-used?
- Does installation require long and expensive supplier support?
- Does it have a future? CASE systems may have to be written off over 5 to 10 years, so that what is in beta-test and what is promised is important.
- Is your culture centralised and carefully controlled or do you need pro-active support? The product should fit the culture.
- Who is the supplier and what are their survival chances? A CASE system should be chosen as much on the status of the supplier as on that of the product. Eventually all the company's existing and new systems (and so all the dp-related business) will depend on that system.
- How widely is it used?
- What interfaces are available to other products?
- Does it require a standalone development system or could it be used on an existing production machine and what load would that impose?

IN MANAGING IMPLEMENTATION, TWO STAGES ARE NECESSARY

Implementing a CASE system at your development centre implies fundamental change in the information systems department in organisational as well as technical ways. One way to smooth such a significant change is:

- To plan it, top-down.
- To implement it, bottom-up, turning plans into action in a piece by piece approach.

In planning, careful preparation is necessary:

- Prepare a vision statement which defines the goals and is easily assimilated by top management, with clear savings, benefits, expenditure, and payoff.
- Present this, and ensure top management is prepared to invest the resources and that the manager for the information systems department is completely in favour of the expenditure. Vital resources will have to be diverted to the CASE system installation.
- Prepare a budget schedule for a task force to implement a pilot project, with three or four people of average ability in it. Time and cost of implementation varies with the formalisation of the company structure.
- Enlist an evangelist in senior systems management, if possible the manager of the department, to promote the concept with user department managers.
- For implementation, review the processing power of the development system to ensure it is adequate and procure new capacity if necessary.
- Allow a timescale of at least three to six months for planning re-organisation, procedures, training, and installation and 12 to 36 months for complete conversion of all the systems development staff.
- Make use of the supplier's post-sales support if necessary.
- Set up a formal training scheme, initially for the task force and subsequently for the rest of the department.

- Use the installed CASE system on a pilot project that is not critical but does require serious effort (not a toy). Some users are restricting the use of CASE to only large, new projects (over 10 man-years).
- Evaluate the results and use them in planning how to progress the implementation.
- Begin re-organising the department along the lines dictated by the CASE system operation, once it has proved its worth, with standards and procedures that fall in line both with your own and with the system's standards and procedures.

CHANGES WILL BE NECESSARY TO EXPLOIT CASE FULLY

CASE systems bring a new style of working to the department that is best supported by:

- Use of system development methods.
- New organisation structures.
- Generalised roles for staff.
- New relationships with end users.

METHODS ARE NEEDED

A first change in the development approach could be to install a system development method, if none is already in use. CASE systems are designed to support a method; without one the creative process becomes chaotic.

ORGANISATION AND ROLES WILL CHANGE

The organisation of the department will have to be reviewed as CASE tools tend to blur the division between analysis and programming. At the same time, they reduce the need for detailed knowledge of the programming environment that was previously required. The distinct roles of analyst and programmer may need integration into one role. Some specification and analysis tools can be exploited by end users directly, so this should be allowed for. The trend to advance the responsibility of end users in the life cycle has to be carefully watched and exploited. A new team structure, of a smaller number of development staff, plus end users and a coordinator, for end-user interfacing, has been found to work.

In summary:

- Integrate analysts and programmers into small teams, each perhaps with its own office.
- Make end users part of the development team for certain phases.
- If existing applications have to be maintained outside the CASE environment, or if there is a demand for high-performance systems that cannot be produced within it, concentrate the specialist expertise required in separate teams. Try to introduce documentation that is compatible with the new systems.

One of the boons of CASE systems is that their users are shielded from some of the detail. This has reduced the training necessary for development staff. As we explained earlier, one user has been able to train programmers straight from a two year college course into full team members in under four months, a process that would normally take more than twelve months.

THE CASE APPROACH TO DESIGN INVOLVES PROTOTYPING

Two major impacts are on end-user commitment and documentation. CASE systems provide the opportunity for prototyping. This can improve system quality because of the iterative process of end-user validation which may also replace the documentation (depending on the method). Detailed written specifications may not be so necessary.

End-user commitment is increased by this validation 'process'. Moreover, CASE reduces the workload of documentation because it:

- Automates the 'mechanical' documentation, particularly structured specifications.
- Restricts the documentation written by staff to end-user manuals. These may be written by end users in the team. Detailed specification documentation can be generated, if necessary, from the database.

CASE SYSTEMS IMPOSE CONSTRAINTS

As the computer processing load in the interactive, conceptual, and analytic phases will be high, computer power provided for development needs to be adequate to cater for this added load.

All CASE systems have their own procedures and styles of working. In implementing a CASE system, some compromise between the department's normal mode of operation and that imposed by the CASE system will have to be accepted.

CASE SYSTEMS DELIVER DATABASES AS MUCH AS PROCESSING APPLICATIONS

CASE aids the link between users and databases, following the increasing trend towards providing management information as much as automation of routine operations. CASE may be used for development of databases plus end-user retrieval and analysis tools.

THE TECHNOLOGY IS STILL MATURING AND COMPANY REQUIREMENTS WILL CHANGE

It is reasonable to expect a far higher degree of automation plus price/performance improvements in the next five years, so development and expansion of the CASE system should be allowed for. Expect CASE systems to evolve.

Chapter 5
The future of CASE

Finally, we speculate on the possible future of CASE systems and the way in which their use may impact system development.

CASE PRODUCTS WILL EVOLVE AND BECOME LESS EXPENSIVE

We can expect CASE systems to evolve in several ways:

- The range of system development supported will extend from Cobol into fast TP and commercial realtime systems.
- Graphics (with WIMPS) will become the dominant human interface.
- Most CASE systems today are incomplete in some phases and aspects of the system development life cycle. We can expect these to be progressively filled over the next 10 years, to provide more of a 'flexible database/program creation system' than a 'software factory'. Moreover, the consistency of the human interface may extend from the CASE system to packages and the end-user application interfaces themselves. Also one incomplete area that many suppliers are intending to fill is that of automatic code-generation.
- The bias toward a centralised architecture may change. A federated structure, more suitable for end-user computing, is likely to emerge. Coordination of distributed design databases and data dictionaries will be provided.
- Current systems tend to concentrate on the phases after systems planning. An extension upwards into 'enterprise analysis' — driven by the business needs and information flow of the enterprise — is something that could improve the match between systems and the business and company structure.

For any given capability, we expect the price of CASE products to fall significantly. They are still embryonic products with suppliers trying to recover the high costs of development. Once their use becomes more widespread, their price will fall. But as we have pointed out, there are still many improvements needed so that the prices of the leading products, offering better functionality, are likely to remain fairly high for some time.

CASE RESEARCH IN THE LABORATORY

In attempting to look ahead, it may seem over ambitious (or optimistic) to expect too much of CASE.

There are, however, several publicly supported initiatives in Europe to develop the advanced facilities to which we have referred. For example:

- The Esprit project AMICE, with 20 companies involved, is exploring enterprise analysis for CIM systems, but with more general applications expected to emerge.
- The Eureka Software Factory (ESF), a 10-year programme to automate development at a professional level.
- The Eureka Advanced Software Technology project (EAST) a six-year drive to produce a range of integrated tools.
- The standard tools environment or workbench produced by the Esprit programme — Portable Common Tools Environment (PCTE). This is now being developed into a commercial product by Emeraude Gie, which has ambitions for it to become a standard platform for CASE systems.

We also expect that expert systems will have a role to play in future CASE products. IEW already exploits expert systems in a limited way for coherence checking and validating the phases. Expert systems could also assist in optimised code generation on human interfaces to accommodate naive users and advance with them. Expert systems themselves, with their emphasis on leveraging scarce expertise, prototyping, and animation are already seen by some as the future of end-user system development. A vision of the far future is end users being able to prepare optimised assembler code for a realtime system, from their business plans, the

technical expertise needed to analyse and program being encapsulated in an expert system.

CASE CAN HELP THE ROLE OF THE INFORMATION SYSTEMS DEPARTMENT TO CHANGE

The above advances will make end-user computing easier. This will reinforce a long-term trend towards the information systems department becoming more of a consultant, planner, and counsellor, and less of a developer, implementor, and maintainer.

Chapter 6
Conclusion

In this paper we have reviewed the current status of CASE and shown that a number of large European companies are starting to make use of it to ease the problems of commercial systems development. The products currently available are still immature, but in the right circumstances can lead to worthwhile benefits. As they evolve, they should offer a better match to their potential user's needs and offer advantages to a much wider range of system builders.

BUTLER COX

P,E,P

Butler Cox

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

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

The services provided include:

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

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

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

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

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

P E P

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

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

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

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

Position Papers
Four PEP position papers are produced each year. They focus on specific aspects of system development productivity and offer practical advice based on recent research and experience.

Forum Meetings
Each quarterly PEP forum meeting focuses on the issues highlighted in the previous PEP paper, and permits deep consideration of the topic. They enable participants to exchange experience and views with managers from other subscriber organisations.

Topics for 1987

Each year PEP will focus on four topics directly relating to improving systems development and productivity. The topics will be selected to reflect the concerns of the subscribers while maintaining a balance between management and technical issues.

The topics selected for 1987 are:

- Managing user involvement in systems development.
- Using tools to improve productivity.
- Planning and managing projects effectively.
- Using methods to improve productivity.

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

Benelux
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-Cedex 1, France
☎ (161) 48.20.61.64, Fax (161) 48.20.72.58

Germany (FR)
Butler Cox Deutschland Ltd.
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) 486 1760 Fax (212) 319 6368

Australia
Mr J Cooper
Consultants (Computer and Financial) plc Australia
Level 5, 303 Pitt Street, Sydney 2000, Australia
☎ (02) 2870400, Fax (02) 2870450

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

The Nordic Region
Statskonsult AB
Stortorget 9, S-21122 Malmo, Sweden
☎ (040) 1030 40, Telex 12754 SINTABS

Spain
Mr Sidney M Perera
Rosalía de Castro, 84-2°D, 20835 Madrid, Spain
☎ (91) 723 0995