Making CASE work

John Parkinson

Ernst & Young Management Consulting Services
Rolls House, 7 Rolls Buildings, Fetter Lane
London EC4A 1NH, UK

Abstract

Much of the attraction of Computer Aided Software Engineering (CASE) tools lies in the sophistication of the technology and the claims made for the tools to solve major problems of quality and productivity in system development. The number of tools available has grown from a handful in the early 1980s to over 1000 today. Yet despite the promises, users have been slow to adopt the tools and benefits have been equally slow to appear. So what has gone wrong? Do the tools actually work, or are users failing to appreciate how to get the best out of them? What can we learn from the experience of those who have successfully introduced CASE? This paper looks at how far these claims for CASE are justified, and examines what organisations have to do to get the best, or indeed anything out of their use of CASE tools.

Introduction

Tools that broadly match the definition of CASE have been available in the UK since c. 1984, and in the USA for about 18 months longer. Individual software productivity tools, particularly code generators and 4GLs have of course been available much longer than this, but were not offered on the same basis as current CASE tools. We will therefore take 1984 as the baseline for reviewing experiences so far with the use of CASE, giving us a 5 year period and at least two generations of tools to consider. The adoption process, from initial review to widespread, or at least relatively common use (or, in the case of failure, virtual abandonment) has typically taken between 18 and 24 months, and some organisations have repeated the process with several tools, or tried out more than one simultaneously.

Thus, those organisations that started with the first generation of tools in 1983 or 1984 and stuck with the same tool throughout, now have between two and three years of experience. However, there are relatively few examples of anyone using CASE extensively for this length of time. Most current tools did not become commercially available until 1986 or later, and users of these products have barely reached the stable use phase.

User Experiences

User experience of CASE tools has been extremely mixed, ranging from a high level of successful use through to a series of costly failures. Looked at as a whole, there are a number of common experiences that point at reasons why CASE has been slow to be adopted:

- **Confusion**. What is a CASE tool anyway? Once the CASE bandwagon got rolling in 1986, everything from a full function applications generation environment to a C compiler with window-based editor on a PC became a CASE tool. It's no wonder those IS departments who started looking at CASE were (and probably still are) confused

- **Over-sold capability**. Early tools were limited in the functions and features they offered and in the size and type of project they could support. The exact limits weren't known to either the vendors or the buyers, so users kept running into "edge case" problems with the products. They didn't know if the problem was with the tool or with their inexperience. Vendors didn't have sufficient experience with field support to easily identify the source of errors, and users often made slow progress, with frequent stops and starts, blind alleys and detours

- **Changes in working methods**. Adopting CASE tools changes more than just the look of the deliverables from a project. IS departments were slow to adopt the necessary new methods and management approaches, and did not therefore realise some of the expected benefits from the tools. Of course, the vendors did not emphasise this aspect of CASE, so users were as often as not expecting to be able to install the tools and keep everything else the same. It didn't work out like that, however, and may copies of tools ended up being used as documentation aids

- **Too little initial support.** Vendors as well as users were new to the CASE market, and did not have answers to all (in some cases any) of the necessary "how do I do this with your tools" questions. Once in the hands of "real" users, the tools got used for things that their developers had never intended. Often the tools actually couldn't help with some of the problems, but vendor support couldn't recognize that this was the position and blamed user incompetence. Tools were often put aside or abandoned altogether by their users in the interests of fixing a crisis

- **Learning curve effects** were underestimated. Many of the early CASE tools had very easy to learn graphics interfaces, and there was a common mis-conception that, having learnt to "use the controls", analysts and designers would be able to make productive use of the tools. In fact, the exact opposite was often true. Users could draw diagrams and collect information easily enough, but they had little idea what should go into the diagrams or what information should be collected. A lot of time was wasted learning effective ways to use the tools, and while this learning was going on, mistakes were made. Because no one was very experienced with CASE, not all these mistakes were recognised until they had had an impact on the project being used to evaluate the tools. In many such instances, the difficulties were remembered long after the easy to use interface had been forgotten

- **Poor match to audience**. Given the supposed importance of CASE, it's amazing how often the initial evaluation and selection process was carried out by relatively junior and inexperienced staff. Tools were often selected on features that were irrelevant to their proposed use. By the time decision makers and IS management got involved, the project was already underway with an appropriate tool. This was very frustrating for vendors who tried to sell the tools on the basis of what they believed the current benefits to be, yet found them being bought on the basis of future features. Quite a lot of the early tools were actually difficult to learn and use, because they were technicians tools, built by programmers and not suited for use by Business Analysts

- **The Critical Mass effect**. Everyone involved in the introduction of CASE should have been aware that there were certain to be critical mass effects with use of the tools. These effects actually manifested themselves in two ways.

- Until a tool covers a large proportion of all the activities required during a project, using it is actually counter-productive, even if it's very useful for the things that it does do. The coordination and communication overheads required to keep tool-based and non-tool-based activities in step outweigh the local advantages provided by using the tool

- Until a sufficient body of experience in the practical use of the tools develops, an IS department can't leverage new projects with the results of previous tool-based work, and much time is consumed in trying to fit existing designs into the new approach. This is where retrofit tools would be invaluable, if only they existed. A second flavour of this effect concerns the process of handing over the tool-based products of one team for use by another. Because the recipients have not seen the models developed, they tend to distrust them, or at best to feel that they did not understand them fully. This lack of confidence meant that they either proceed much more slowly than expected, or, worse, re-do all the modelling work

By and large, CASE vendors did not help the adoption process. Many did not build their CASE tools any better than their customers had traditionally built applications systems, and early tools were far from bug-free or reliable. Users of such unreliable tools quickly discovered that, for all its limitations, paper-based information is at least accessible. A corrupted design database on a CASE tool may not be. Added to this, early tools did not address some significant problems of practical use, particularly the issue of multiple users on a single project, and of shared access to a central repository of information. There were good technical reasons for this but many users were reluctant to get started on large projects with single user tools and the vendors waited too long to come up with a credible approach.

Nevertheless, there have been successes with CASE, some significant, many more modest but worthwhile. It should come as no surprise that the most successful cases tend to be in organisations that were already well organised and able to understand the requirements for a major change in the way they worked. CASE was not seen as an instant cure for all systems development problems, more as an essential component of an overall improved approach to the management of the development life cycle. Successful CASE users also tend to assemble a small portfolio of tools, recognizing that there is more than one problem to solve, and that no one tool addresses all requirements. They then persevered with their selection, discarding a tool only when it was clear that it did not work or could not meet users' requirements.

Does CASE really work ?

One of the most common questions asked about CASE is "does it really improve productivity and quality - and if so, by how much?". Vendors usually sidestep this question, with some justification, for two good reasons:

- There is little real performance data on the use of CASE, since few organisations have completed enough real projects, as opposed to trials and pilots, to establish controls for interpreting information on use

- Few CASE users (or any other type of IS department) have accurate or comprehensive performance data from their previous tools and methods, and most did not bother to establish baselines against which to measure the impact of introducing CASE

Despite the fact that everyone professes interest in productivity, there are really very few well established metrics, and those that do exist assume stable operating environments, not subject to multiple changes. This makes comparisons difficult, especially when measuring the combined effects of learning new tools and methods of working. For what it's worth, my own experience with several clients, tools and methods indicates that during planning and analysis, much better quality is achieved with about the same IS resources in slightly greater elapsed time. This is in part due to the increased involvement of users, and the extended time for achieving agreement and approval for requirements models. A further, albeit unquantified gain comes from the increased confidence that end users have when they are involved in and understand the results of the specification process.

A distinct and significant gain can be achieved during the Design and Construction phases, but this is primarily dependant on the nature of the implementation environment selected for the application. Where code or application generation tools have been used, the gain has been large. Where conventional hand coding is used, the gain is still apparent, because the coding and testing cycle is less affected by design changes, but is much less.

No one yet has much (any) hard data on the impact on reduced maintenance effort, although some (good) anecdotal information is appearing. The jury is still out on this one.

Getting Started with CASE

Buying and trying out a CASE tool can be either very cheap or very expensive. Single phase workbenches for analysis cost only a few hundred pounds, and can be learned in a day or so of "hands on" effort. If hardware and human resources are costed in, "trying out" CASE on this basis still costs only £20,000 to £25,000 - an insignificant amount for a major IS department and in most organisations less than half the annual cost of an analyst. At the other extreme, a full I-CASE toolkit may cost upwards of £250,000 to buy and another £250,000 to install and implement.

In each situation, however a plan for the introduction and effective use of CASE tools of any sort is really an essential. Without it, much time and expense can be wasted for little or no return. This section looks at some of the key issues in adopting CASE, in particular:

- Getting effective commitment from IS and user management
- Setting reasonable expectations for what CASE can do
- Making the cost/benefit case
- Starting in the right place in the development life cycle
- Starting with a pilot project vs the "Big Bang" approach
- Choosing a tool or tool kit
- Picking a good first project

In the next sections, we will also look at some of the major practical problems with current tools, and at some of the factors that have proved critical to the successful adoption of CASE tools in organisations that have made them work effectively.

Getting Effective Commitment

One of the real dangers with CASE is that it will be used by IS managers as "something to keep the technicians happy" and that there will be no real commitment to adopting it. Given that CASE is just one element of a change in the way the IS department works, it's not surprising that many middle level managers are reluctant to look at it seriously. In particular, changes that require much more direct user contact and accountability are often strongly resisted. In working with a number of organisations to introduce CASE, I have found that most analysts, designers and programmers take to CASE fairly readily. Senior managers are also reasonably easy to convince. It's the levels in between who offer the strongest resistance, and who are often the ones who actually control the selection process.

Without effective commitment, however, CASE will be just another "evaluation exercise" and even the best CASE tools can't demonstrate their true potential in isolation. So ensuring that there is commitment at senior levels in the IS Management to carry through the implementation is essential. Equally essential is the communication of this commitment to all levels of the IS Organisation. Users will also be involved in the introduction process, and the impact on them may be just as great. It's also necessary, therefore, to keep them and their management informed of what's going on, how it will affect them, and what benefits can be expected.

This makes the adoption process essentially a corporate decision, with implications that go well outside the IS department, and hence there has to be corporate commitment as well. In most organisations, that's a lot to ask for, as senior general management is not used to having to decide on these sorts of issues. Nevertheless, the adoption process will go much more smoothly if everyone involved knows that the Chairman or Chief Executive is behind the project and actively watching progress. It also raises the stakes for the IS department. With this level of visibility, they will be under a lot of pressure to make the introduction successful.

Setting Everyone's Expectations

CASE is far from being the answer to all IS development problems and it's important for those directly involved to understand this, and to make the point clear to everyone else. In particular, CASE does not:

- Remove the need for understanding and analysis of users' business problems
- Do all the work of information gathering, model building and evaluating options - it just makes some of the activities involved less tedious and time consuming
- Speed up the overall process of Analysis or Design. The requirement to use a structured method correctly and to introduce rigour into the modelling process actually slows things down - that's the price of improved quality
- Make life easier for users during the requirements definition process. Users may be required to become more involved, not less, and will be expected to take on a more responsible role in agreeing and quality assuring deliverables

Many extravagant claims are made for CASE tools and in every instance where they were believed they've re-bounded on the implementors and delayed or destroyed real but less extravagant benefits. Setting realistic expectations may make the selling process more difficult, but in the end it's always easier to do well against an expectation that was realistic in the first place.

Making The Cost/Benefit Case

There is a long standing joke among management consultants that something that you really want but can't cost justify ends up as an essential part of corporate strategy. CASE tools are no exception. Few organisations have so far taken on CASE as a necessary strategic tool for future IS developments, although the number is increasing as the tools improve. That means that most IS departments will have to provide a credible cost justification for the tools they want. The costs are relatively easy to identify (although they vary considerably from tool to tool) but are usually understated by ignoring or reducing the estimates for training and learning curve effects. Costs associated with the integration of new tools with the existing environment are also generally forgotten. If you add everything up, it turns out that the cost of the tools themselves is a fairly trivial element in the overall equation.

Benefits assessments are divided between:

- Quantifiable, which usually means estimates of:

    - Improved staff productivity from analysts, designers and programmers

    - The benefits in terms of reduced maintenance from better quality systems

    - Reduced machine resources from better designed and integrated applications using common data bases

  Of course, the reliability of some of these estimates may be in doubt, but so long as the underlying assumptions are stated and seen to be reasonable, estimates can usually be derived for these categories of "benefit"

- Unquantifiable which covers:

    - Other aspects of quality, such as usability, common standards and reduced requirements for user training and support

    - Increased user confidence in key operational systems through enhanced reliability

    - Improved contribution of information systems to the support of corporate objectives

    - Improved responsiveness of the IS department to changing business needs

    - The ability to attract and retain better quality IS staff through an improved image as a "leading edge" department and the resulting decrease in disruption caused by staff turnover

  Some of these "Unquantifiable" areas can actually have significant quantifiable savings attached to them, (user training and reduced staff turnover are two obvious examples) but few IS departments have any baseline information from which to generate a reliable estimate

As with all cost/benefit calculations, the value of the exercise depends on how well the current situation is quantified and whether the proposed savings can be realised in cash (or at least budgetary) terms. In IS departments, the position is usually fairly grim on both counts. Very few departments keep detailed productivity records. Quite a lot don't even keep accurate resource utilisation records structured by application development and maintenance and recorded on the basis of individual applications. So the baselines for estimating savings are weak at best.

The answer in most instances is to use the accounting equivalent of a "thought experiment" - see what the answer has to be to justify the cost, and then show that, within very conservative margins, this will be easily achievable if only x% of the claims for CASE are correct. Provided x is low enough, this usually passes scrutiny.

Starting in the right place

Most CASE implementations start with an Analysis project. There's nothing really wrong with this - requirements specification is one area where CASE is supposed to make a big impact - and it's an area that most IS departments feel that they already know. Unfortunately, it's often a bad place to start from the point of view of a successful implementation because:

- Structured methods, which CASE tools need to have in place to realise their full benefits, are often absent, or if present, are used badly or not at all. Analysts don't understand how to use the tools effectively because they don't really understand the techniques that the tools support

- Many of the architectural issues raised by using CASE can't be resolved if there is no Strategic Information Systems Plan in place

- Scoping an Analysis project is very difficult without corporate models for organisation, data and functions, so pilot projects have to be adjusted as they go along

- Good pilot projects are not always available, so inappropriate projects or "make work" exercises get used instead

- There is a significantly reduced opportunity for added value from the first project if the models built with the CASE tools are limited to one business area and are not readily usable in others

- Training often gets skimped in the rush to get started, resulting in confusion and uncertainty in the project team and low morale during the project

- There is a severe danger of "the blind leading the blind" on the project. Recently trained or even untrained CASE users have to become instant experts, and they make mistakes in their use and understanding of the tools. These mistakes can easily become working practice because there's no one to tell them how to do things correctly or effectively

All these potential problems make analysis projects a less than optimum place to start the CASE adoption process.

The best place to start is with a proper Information Strategy Planning exercise on a major organisational division, so that the key corporate models (organisation, data, function) are in place and agreed and the essential architectures (data, applications, technology) are identified, planned and understood. Subsequent analysis projects can then work from a common and consistent baseline, with assigned and agreed priorities and high added value from the planning models. The information strategy planning process is also less dependant on structured methods skills (although it still requires the basic skills of data and function modelling to be available to the team) and planning teams usually consist of more experienced staff. The duration of the planning phase also allows time for sensible methods and tools training for analysts and designers who will be involved in later projects.

There seems to be an strong case for starting with an ISP project, so why do so few organisations actually start somewhere else ? I think there are a few specific reasons why this happens:

- The choice of CASE tools that support planning is limited. If your favourite vendor's tool doesn't provide support (yet), you won't want to start with a project that will show it in a bad light. There's a much bigger range of choice in Analysis, so it's safer to start there and move on to Design

- Decisions about "trying out" CASE are made in the IS department at a level below that which could initiate or propose an ISP project. CASE is often brought in without senior management's knowledge or consent, "just to get some experience" and this isn't possible if the ISP route is chosen. Starting with a small low risk analysis project also avoids facing the issue of convincing senior managers that CASE is a good idea

- Not everyone feels confident that an ISP is necessary or desirable, or the organisation already has, or claims to have, one, and doesn't want to repeat the exercise

If a SISP already exists, a good approach is to use the CASE tools to document it thoroughly, and to construct the relevant corporate models. This helps validate the contents of the plan and gives any subsequent Analysis projects a head start with tool-based data and function models.

Nevertheless, 8 out of 10 users start their use of CASE with an Analysis project.

Pilot Projects vs Big Bang

Conventional wisdom in IS departments and elsewhere recommends that you start something new with a pilot project. The rationale is that pilot projects can fail without doing serious damage to the business, so the risk is contained. They also involve only a small proportion of the IS department's resources, so they're relatively cheap to run and not too difficult to manage. There's not much published data on the success rate of this approach in terms of achieving rapid implementation of new technology, even when the projects themselves are successes, but from my own experience and from anecdotal material, it's not high. There are some relevant reasons for this:

- Because pilot projects don't really matter, they seldom actually fail. If everyone knows that this is "just a pilot" they tend to relax and get on with things in a way that you don't see on "real" projects. If they do hit serious problems, that doesn't really matter either - its assumed that the problem will get fixed before the tools get used "for real"

- No one believes that pilots actually prove anything except that a basic level of functionality is present and works. Pilots are used as "exclusion" tests not "inclusion tests", yet no one considers the logic of trying out a tool that is in such poor shape that it's going to fail on this basis. From this point of view, pilots are just extended demonstrations without the salesman present

- Pilots are seldom chosen so that the experience gained on them can be transferred to larger or more complex projects, with different staff involved. They're just not typical enough to be useful indicators of how the tools will work for other people in other situations

- The process takes too long to produce any relevant results. In particular, any significant benefits are likely to be delayed well beyond the end of the pilot project until sufficient additional use has been made of the tools

That's not to suggest that pilot projects should be abandoned altogether - just that the project used needs to be chosen carefully, and that the attitude of the organisation towards the trial needs to be carefully managed.

The alternative to the pilot project approach is to recognize the strategic role that CASE will play in IS developments, and mandate its use on all projects, without going through the pilot stage. This "Big Bang" approach certainly demonstrates commitment but does require that the tools chosen be suitable for the work to be done and the environment in which they will be used. From that point of view, this is a much more risky approach. On the other hand, given a good choice of tool, the benefits will be available that much sooner.

There's no doubt that this approach can work, but it imposes a considerable strain on the management of the organisation attempting it.

Choosing a Tool Set

How to choose which tool or tools to use is a very common question. Like most selection processes, there's no simple answer to making the right choice, but there are some practical guidelines. At the very least, an organisation planning to adopt CASE tools needs to know why it wants them and what it expects to use them for. If you think that this is stating the obvious, you're absolutely right, but my experience is that it needs stating, over and over again. Here are a few of the key items that should be considered, not in any order of priority:

- **Match to methodology.** If you don't actively use a structured analysis and design methodology, you're not going to get much benefit out of CASE tools. If you already have, and actively use, a methodology, and you're happy that it will fit in with automated support, select from the tools that fit the approach used by the method. Methodology is a key determinant for the success of CASE tools, and, on a more trivial level, determines the type and format of diagrams that users feel comfortable with. Cosmetic issues may seem trivial in the selection process, but they turn out to play a significant role in the ease and success of the initial adoption

- **Match to technology architecture.** Pick tools that are consistent with your technology architecture and infrastructure. You are going to have to support the hardware and software platforms used by the tools, and if it's a new one for your support staff, there will be costs associated with this. Don't expect to have all the necessary hardware and software available already. Virtually all CASE tools require some additional investment in suitable platform resources

- **Match to development and implementation environments.** Pick tools that fit in with your development environment. For example:

    - If most of what you do is maintenance, retrofit tools and re-engineering generally will be important - planning tools won't be, although you'll probably need them eventually

    - If you are oriented towards using packages, you don't have much need for code generation tools, but you might still want end user or information centre oriented tools for package customisation

    - If you are involved in real time developments make sure that the functional modelling facilities of the tools will support this requirement

  and so on.

Also pick tools that are a good match for your implementation environment. You don't want to spend a lot of time manually adapting automated designs to fit the target environment, although some manual tuning is almost always necessary. If you have more than one of these environments (as is increasingly the situation), expect to select a set of complimentary tools covering all the targets, although all the tools selected should be able to use a single repository

- **Match to organisational style and culture.** If you're a sophisticated IS department that already has a data administration or information resource management function, you'll be able to use more sophisticated tools than someone who has yet to install a DBMS. Pick tools that match the level of sophistication you can achieve, but remember that you will become more sophisticated much faster than the tools will, so don't go for something too basic that you will outgrow quickly. If your organisation is highly centralised with tight central control and strict design and documentation standards, pick a tool that can help to enforce these. If you're not so strictly controlled, or operate in a highly decentralised fashion, you'll need a tool that allows for more user flexibility, while still maintaining consistency between users' views

- **Match to implementation approach.** Select tools that fit in with they way you've planned to implement CASE. The tools will need to be strong in the areas that will be addressed first, because that's where the most mistakes will be made. Make sure that the selected tools cover all the areas you plan to address for the period of your implementation plan, not just the initial project

- **Match to budget.** If your budget is tight, select a tool that lets you start small and grow in easy increments, and that minimises all the hidden costs of adoption, like support, training and learning curve effects

- **Match to Vendors.** You're going to have to live with your tool choices for some time, so select vendors that you are comfortable working with and who can provide the type and amount of support that you're going to need. This includes satisfying yourself that the vendors will continue to be in business to provide the required support. If you're selecting more than one tool, make sure that the different vendors can and will work together to help to integrate your use of their products

The final choice is almost always a balancing act between conflicting objectives and compromises between what the buyer would like and what the tools can actually do. Using CASE changes many things related to the development life cycle and the choice of tools can make a big difference to the ease with which these changes are accomplished. So it's worth putting some effort into the selection process. Don't give the job to a couple of junior analysts to do in their spare time. Vendors are good at spotting who is serious about CASE tools and who is just "kicking tyres". Since they have a living to make, they tend to concentrate on those they believe to be the serious prospects. If you don't behave like one, you won't get treated like one.

Once you have identified a few tools in which you are really interested, arrange a product demonstration by the vendor, and go prepared to ask, and get answers to, the "make or break questions" that decide whether the tool is really a contender or not. Insist that the vendor makes technical expertise available to answer your questions, and does not just leave it to the salesman to "get back to you on that". If possible, prepare a short exercise that you want the tool to be put through, and let the vendor have a copy in advance, so that they know what to expect and can plan their time accordingly. Expect to spend at least a day on each product demonstration, but don't expect that this will prove all that much. The true test of a tool can only be on a project, and the purpose of a demonstration is to generate sufficient confidence to make setting up a trial project worthwhile.

Picking a Good First Project

If you are one of the 8 out of 10 installations for whom the preferred implementation route is a pilot project in Analysis, followed by Design, then there are some important issues to consider. Picking a good first project can make the difference between getting CASE going successfully and missing the opportunity. Some of the things to consider are:

- The project should be one that's going to be done anyway, not something invented for the purpose of the pilot. That means that it is a "real" project, the outcome of which matters to the IS department and to the business. Given that there is a risk in introducing CASE tools, it should not be a strategically vital development. It should, however be a non-trivial development that will be implemented on completion and for which the IS department has made delivery commitments

- The project should be one that's already been scoped and sized, so that some comparison can be made between what was expected without the use of CASE tools, and what actually happened

- The Project should be the right size, so that:

  - It's not too big, giving training and co-ordination problems to an inexperienced team, and not so small that the results will be dismissed as un-representative

  - It lasts long enough to get the team over the worst of the learning curve effects, but not so long that the benefits assessment and evaluation are delayed

  A good guide-line to use is a team of 3 - 5 people for 4 - 6 elapsed months, with an absolute upper limit of 9 months for the elapsed time. The team should include at least one user from the area under investigation. Larger projects can be considered, if adequate account is taken of the additional management input and co-ordination effort that will be required

- The project should be in an area that is already well understood by the business and the IS department, to minimise additional learning effects. Try to avoid areas that are changing for other reasons, or have serious organisational or managerial problems

- The area to be investigated should not be associated with the use of unfamiliar technology or software apart from the CASE tools. Once again, the idea is to minimise the number of variables that the project team has to cope with

- If possible, the project should provide opportunities to add value to the modelling work by re-using it in subsequent projects

Most medium and large scale installations will have a project that fits the majority of these criteria, although it may be a problem in smaller departments. Where no obvious candidate can be found, you should consider exactly why you are introducing CASE, and select a project that will best allow the performance of the products selected to be matched against these reasons.

And just to prove that all the above is really only a guide-line, I have also seen a successful installation (at a merchant bank) where they changed virtually everything in the IS department all at once (hardware vendor, operating system, DBMS, development language and departmental location), and still managed a successful implementation project in a critical business area.

Resources

What's needed for the first project is a team that will have a good chance of being successful, but not one that's so unlike normal project teams that everyone knows that the results aren't typical. It's also advantageous if the team members can be used to form the nucleus of future project teams and thus spread experience as rapidly as possible. In practice, it is very difficult to satisfy both of these criteria simultaneously. The main considerations are:

- The team members need to be enthusiastic about CASE, but not to the point of fanaticism. They must be prepared to be positive in the face of the difficulties that will inevitably arise, but also objective in their assessment of the contribution that the tools make to the success of the project. They should be people whose views and abilities are trusted and respected by their colleagues in the IS and user departments, so that they can act as ambassadors for the CASE approach once it has been proven

- Avoid using only "star" analysts or designers. The average CASE tool user will be someone with less than 2 years experience in their current job, and the tools must be useful to someone at this level. If they can only be used by real experts, they won't get used effectively by 80% or more of those who really need them

- At the same time, avoid using people who are very inexperienced and who will require constant supervision and help from other team members, who may themselves be uncertain as to what they should be doing. Also avoid those who are strongly against the CASE concept for whatever reason. They will spend their time on the project in trying to prove that CASE doesn't work, rather than finding out what it can actually do

- Get at least one user on the project team, and pick someone who is pro-technology. If the project creates a favourable impression with users and generates a lot of positive interest outside of the IS department, it will be difficult for IS staff not to accept CASE tools

- Pick a really good project manager. Project management skills are one of the main determinants of success for any development project, and ensuring that the project will be effectively managed is a key issue

Provided that the pilot CASE tools project has sufficient management commitment, the staff needed can usually be made available. Once the team has been selected, it's a good idea to train them in the use of the tools as a team, so that they get used to working together, and understand each other's strengths and weaknesses. This also makes life a lot easier for the project manager, who will know what to expect from each team member, and adjust the project roles and responsibilities accordingly.

It's also a good idea to budget for additional review and presentation time to allow non-team members to be briefed on progress during the course of the project. A monthly seminar, given by the project team to groups of colleagues, will stop the project being seen as "only for the privileged few". Using CASE tools should not be seen as difficult, and keeping everyone informed is one way to avoid this happening.

Timescales

All these considerations mean that the CASE tools implementation process can't easily be rushed through without risking longer term success. In the ideal approach, starting with an ISP project, the selection, preparation and first project will normally take between 9 and 15 months. A shorter elapsed time than this can be achieved if the choice of tools is relatively simple, but a minimum of 6 months is really the lower limit. Any longer and there is a danger that other factors, such as new tool developments or changing business requirements will invalidate the selected approach and tools.

Starting with an Analysis project gives a similar timescale. Three months to review and select the tools. A month for team training and project start-up and then four to six months for the project itself gives an elapsed time of 8 to 10 months. About a year seems to be typical. One of the major hold ups experienced by a lot of sites when starting up their first CASE project relates to ordering the correct hardware and software configuration for the selected tools. Why this should be I have no idea, it may be that the complexities of the PC world are simply unfamiliar to IS departments, or that the defences erected by many organisations to prevent end users buying PCs indiscriminately get in the way. Whatever the reasons, expect the process to be difficult, time consuming and error prone.

Getting Beyond Getting Started

All that the initial project really does is confirm that the CASE approach is potentially viable for an organisation, and that the tools selected work to an acceptable degree in the circumstances in which they were used. There is still the much larger problem of how a successful first project involving only half a dozen people can be transformed into widespread use of the tools in an IS department that may be hundreds strong. More than one organisation has put so much effort into its initial CASE projects that it had no energy or enthusiasm left to continue the implementation process once the first project proved successful. Yet CASE implementation is not complete until everyone who could be using a CASE tool is routinely doing so.

There are two main issues to consider once the first project has been completed successfully:

- How do we "finish" the application that has just been specified and/or designed?
- What should the second CASE-based project be?

If you wait until the first project is over before making these decisions, there will be an inevitable loss of momentum and enthusiasm and interest will be lost. You will also lose the opportunity of starting up subsequent projects before the end of the first. Because there are inevitably a lot of factors to be juggled, it's a sensible idea to start the process with a strategy for CASE tools implementation that covers more than just the initial project.

I hope that it is now clear why the right place to start the CASE implementation process is with an Information Strategy Planning project, even if it's only for the IS department itself. If you start with this, you and your organisation will already have done most of the work required to plan the development projects to be undertaken over a two to five year period, and you will understand the priorities, project dependencies and opportunities for added value from one development to another. Under these circumstances, it's easy to answer the "what should we do next" questions, and possible to plan ahead for the additional tools and facilities that will be required for later projects.

Despite all this logic, most installations still won't opt for this approach, but should still develop a strategy that will get them to the same end, albeit with more effort and risk. The strategy should cover:

- Extending the business models produced during the initial project, particularly the data model, to cover the whole business division or the complete organisation
- Identifying subsequent development projects and project phases, for which CASE will be used
- Planning the acquisition of any additional tools and training that will be required by these projects
- Continuing an education and training program for IS department and user management and staff

The plan should also address the overall migration issues concerned with converting existing systems into the CASE tools environment.

Most IS departments have the beginnings of an architecture for their use of technology. Without an SISP, however, there probably won't be equivalent architectures for data, organisation and applications. These architectures are essential for the planning and control of application development, whether or not you use CASE tools. An ISP project builds them explicitly. An Analysis project doesn't, but does create sufficient "seed corn" models, allowing a separate architecture development project to start once the business area models are available. The data and function models generated during Analysis become available fairly early on in the course of the project, especially at the level needed for architecture development work, so there is no need to wait for the end of the project before starting. In particular, the data model for a business area will usually contain most of the major organisation wide entities needed for the data architecture, and can form a starting point for a corporate data model. As the architectures for data, applications and technology develop, they need to be consolidated into a single corporate information model, under the aegis of the corporate data administration function. This is an iterative process, which should be started as early as possible during the development of each individual architecture, and which is greatly assisted by the use of CASE tools.

So, if the first CASE implementation project is not an ISP, the second should be an architecture development project, building on the business modelling work from the first and starting as soon as the models are stable. This will usually be about halfway through the planned elapsed time for the first project. Following the same approach as was used to prepare the initial project team, a selection and team training programme will be needed. Since CDA will take over responsibility for the architectures one they are developed, it's sensible to use some CDA staff on this project. The team should also contain some IS department analysts, who will be future "customers" of CDA and who will use the architecture on future projects. Their involvement prevents the worse aspects of "not invented here" being associated with the architectures and, particularly in the case of the data model, helps to stop the worst excesses of abstract data analysis from occurring.

It is actually possible to start the architecture project effectively at the same time as the first Analysis project and combine the modelling efforts. This approach provides the Analysis team with additional start-up resources and helps to get the models built quickly, but requires a larger initial training effort and introduces possible co-ordination problems.

Selecting subsequent projects is still going to be something of a problem. The architecture team will need longer to finish their development than the first project team will need to finish their requirements definition, so selection on the basis of the architectures will not be possible. The best approach is often to select a project that's a larger version of the first. There should be some value from the data models that will by now be available, but added value opportunities will probably be limited.

Education and Training

Good CASE tools look easy to use, in fact ease of use is one of their strong selling points. In terms of "operability" it's true that most people can learn the controls in a few hours and be comfortable and efficient with them in a few days. The leading tools are, however, extremely rich in functionality and highly integrated via their repositories. It takes much longer to learn how to use these facilities in an effective fashion. This learning process can't be effectively carried out without a significant amount of education and training:

- **Education**, because many analysts and designers are unfamiliar with the approaches required to make effective use of the tools, and need to be shown the assumptions that tool developers make about how their tools will be used

- **Training**, because there are new skills to be acquired, and these skills require both demonstration and practice

Much of the training needs to take place in order to ensure that analysts and designers understand structured techniques well enough to use the tools sensibly. This increases the training requirement substantially, but we have found it to be necessary in almost every organisation adopting CASE tools. Training in the basic techniques of data analysis is particularly necessary. In addition to basic skills acquisition, project teams may need to work together on exercises at the start of major projects. 3 - 5 day workshop classes based on prepared case studies can be used to re-enforce basic skills and establish understanding of the role the tools play in the project as a whole.

This is a lot of training for an IS department to commit to, and there will be a tendency to skip some of it, or to train just a few staff, and expect them to pass on their skills and knowledge to the rest of the department. This won't work. Training doesn't have to be done all at once, but it does all have to be done, and it's cheaper in the long run to buy high quality professional training than it is to rectify the problems caused by badly trained staff. CASE tools don't reduce the skill levels required for good analysis and design. All they do is make it easier for skilled staff to deploy their skills, and help to catch routine errors of detail. A badly analysed problem can be modelled perfectly on a CASE tool - but the solution designed from the model will not be the one required. So training may be expensive, but not training people is even more expensive, and can be disastrous.

What works and what doesn't

Like most new technologies, CASE promises many functions and facilities for users that are not fully developed or readily usable. This situation is not the fault of the tool developers alone, although they get blamed for any deficiencies or difficulties. Actually, tool developers have a major problem when trying to decide which functions and facilities to develop, and how to package them into sets that match the way users work. Only by trying out tool features on real projects is it possible to tell what will be required by users, and what approaches work well. When you add in the difficulties caused by variations in project management, user skills and all the other variables in real life development projects, it's not surprising that the developers don't get everything right first time.

It's important therefore to ensure that all those who are to be involved in CASE projects know what to expect, and are not led to believe that the tools will do things that they can't. This is important for both:

- **IS staff**, who may become frustrated if the tools don't meet their expectations and make their work slower or more difficult. If the IS staff don't believe in what they are trying to do, it's highly unlikely that the project will succeed

- **Users,** who will rapidly lose confidence in both the development process and the CASE technology. Since their co-operation is essential to a successful implementation, this loss of confidence must be avoided. Once it's happened it's much more difficult to re-establish.

Setting realistic expectations is therefore important, and needs to be done from the outset of a CASE implementation programme. Those involved need to understand that:

- CASE is expected to work effectively, but not necessarily all at once, or immediately

- Problems are to be expected, but resources and skills are available to deal with them

- There is management commitment to see the process through to a successful conclusion

Everyone involved must also understand that CASE tools are just that - tools - and their effect depends on how well they are used, not just on the fact that they are being used. The tools help developers to carry out some procedures more effectively, but they don't remove the need for those procedures.

Keeping it simple

The approach, therefore, should be to formulate a simple message about the use of CASE, and to set expectations relatively low, but to provide visible encouragement and support for initial projects. Do:

- Emphasise commitment to enhancing the development process and to furthering the development of the IS department's ability to respond to user needs

- Support the requirement to develop high standards of professional skill, and to enhance the effectiveness of those skills through the use of CASE

Don't:

- Stress the technology of CASE

- Make the tools seem easier to use than they really are

- Force the use of the technology in user contacts until the IS staff are comfortable with it

Since the initial CASE experience will surface problems and concerns, try to anticipate where they might occur, and be prepared to deal with them quickly and decisively.

Where and when do the benefits appear ?

Given the significant costs of introducing CASE tools, IS and business management are always keen to know when the benefits and payback will appear. Predicting and measuring benefits is difficult, in part because of the lack of information about current performance and quality in IS development. Nevertheless, the benefits question needs to be answered sooner or later, and there are a number of considerations that need to be taken into account when trying to predict where and when benefits will occur.

Learning to operate CASE tools is generally simple and quick. Learning to use them effectively is not. In addition, CASE tools focus on the need to carry out planning, analysis and design activities with much greater attention to detail, and with much more awareness of users requirements. Learning how to do this effectively is not easy for most IS departments, and while they are learning, they are not going to perform as efficiently as they usually do. They're also going to make mistakes, possibly serious ones, in the early projects, usually due to unfamiliarity with the techniques and methods that the tools help to automate. These mistakes will not always be easy to spot and, once spotted, to correct.

The upshot of all this is a steep and extended learning curve, during which it's likely that performance and productivity will be at best equivalent to that achieved without the tools. Only after this learning curve effect has been removed can IS departments be expected to show measurable improvements in productivity. Only a few CASE users have reached this point so far, and the evidence indicates that it will take 12 - 18 months to bring a significant proportion of an IS department's development staff up the learning curve.

So if there are no quick gains in productivity from the adoption of CASE, are there benefits to be gained from improvements in the quality of information systems developed using CASE tools ? Once again, the problems posed by lack of a sensible baseline against which to measure hampers the argument. Generally, however, achieving significantly higher quality can be expected to involve more detailed analysis and design procedures, coupled with additional project management and control activities. It also implies the expanded involvement of users in the development process, and in the review and approval of work products. All this implies that more time and resources will have to be put into system development, and many of these resources are scarce and expensive. That's why we don't do things better now. It's not that the will and the expertise aren't there, its just that there is never enough of them. So can CASE make enough of a difference on the productivity side that resources can be released to focus on quality ?

The evidence so far is that it can, although it barely achieves a real difference during the initial implementation period. The consistency and completeness checking facilities provided by the leading CASE tools do make a significant difference, but only once IS staff understand how to use them effectively. The code and application generation capabilities of Lower-CASE and some I-CASE tools are also major contributors to improved quality, but only if the designs they work from have been correctly developed as solutions to user requirements.

In essence, we have to come up with a number of criteria against which to evaluate the effect of CASE, and to translate the measures of this effect into benefits that IS or user management will agree can and will be achieved. CASE alone won't deliver benefits to anyone. Only if managers and users agree to make use of the potential that CASE offers to improve some aspects of the systems development process will benefits be achieved, and even then, only after the use of CASE has become firmly established.

Critical Success Factors

Finally, lets summarise the critical factors in successfully implementing CASE in an organisation. Not necessarily in this order of importance, all the following are required:

- **Management Commitment**. The first lesson has to be that the adoption of CASE tools is not something that can be undertaken by the IS department alone. There is a temptation, however, to begin experimentation with CASE as an IS only activity, and to let others in the organisation become involved once the experiment has proved to be successful. When they are finally allowed to participate, users and user management treat CASE as "just another attempt by the IS department to spend our money on their toys" (this is an actual quote from the chief executive of a large insurance company, asked to expand a small IS pilot project from 5 users of CASE to over 100).

  Organisations who have succeeded with CASE ensured that both IS management and senior user management knew what was going on right from the outset, and were prepared to support the implementation process publicly. This is usually a new and difficult role for IS departments, and generally they do it badly or not at all. If your IS staff can't or won't set up and run some such programme of management education, you'll need to get outside help to do so. If you omit this step, you'll find it much harder to get the use of CASE accepted. Once you have identified and agreed a sponsor, and set up a management education programme, it's time to take a look at the structure and management of the IS department, and start to get this into shape for effective control of the use of CASE.

- **Methodology**. No Methodology - no hope. In more than 4 years of involvement with CASE implementations I have yet to see an organisation successfully adopt CASE without already having a competent structured development methodology in widespread use. Quite a lot of organisations have tried, and have discovered a some stage in the implementation process that no one really understands what to do with the tools. They then have to stop or slow down their use of CASE while a suitable methodology is selected and introduced. Those organisations that never adopt and implement structured methods fail to implement CASE, or demote the tools to use as simple diagraming aids.

  Note that buying a structured methodology product is not the same as implementing the method. There is just as much shelfware in methodology as there is in software and three feet of dusty manuals on a shelf, or a £0.5 million training programme does not mean that the method is being used. The UK is particularly prone to this situation. Methodologies are adopted, staff are trained and the result ? At best, structured documentation, at worst nothing at all to show for the investment.

- **Training**. Training is expensive, takes key staff away from essential work and never really seems to teach staff what they need to be productive on their return. So it gets skimped or skipped altogether. Since the leading CASE tools are always sold as easy to use, why spend the time and resources on expensive training courses ? Just install the tools, let the chaps play with them for a while, and away you go! Sound familiar? It should. I must have heard variations on this theme dozens of times from IS managers, prepared to spend tens or hundreds of thousands of pounds on tools, but little or nothing on the training needed to make effective use of them.

  And training is needed. Development using CASE tools is sufficiently different from conventional development that IS staff can't usually work out how to use the tools to best effect unless someone shows them. They can then take the basic principles of CASE tool usage and adapt them to the needs of their own organisation. Without training, this process of learning and adapting to the use of CASE will take a long time and may never be achieved. Unless the learning process takes place, however, CASE tools will not achieve their potential.

- **Involving Users.** Most CASE tool developers and the majority of those adopting them have assumed that CASE is essentially for the support of professional IS developers. The tools generally support tasks and techniques that have been the preserve of IS analysts and designers, and offer little direct support for non-IS staff who may participate in projects. Modern structured methods, in contrast, stress the need to involve users, both as sources of information on business requirements and as active participants in project teams. If the benefits of user contribution are to be fully realised, they to must be able to understand the need for, and if necessary use, CASE tools to support their participation. In my experience, where users do receive encouragement and training, they take up the use of the tools just as fast, if not faster than professional development staff. It can be argued that their use is at a trivial level, since they do not really understand the techniques that are being automated by the tools, but that's not the users' perception. They see themselves as making a direct contribution, alongside and on a par with IS staff, and this strengthens their enthusiasm and support for the CASE-based development process. Provided that the extent of their contribution is effectively managed the overall performance of a mixed team of users and professionals supported by CASE tools can be significant. Just how far the principles of user involvement and direct use of the tools should be taken is still a matter of debate and of research efforts, both in the US and the UK. It is already evident, however, that users have a great deal to contribute to the implementation of CASE, and if they are prevented from participating, they will probably adopt a posture of active or passive opposition.

- **Selecting Appropriate Tools.** Just how easy it is to get IS staff and users started with CASE depends on which tools are selected, and what they are to be used for. In many ways, the selection of appropriate tools produces a dilemma:

  - The initial requirement is for tools that are easy to learn and can be productive quickly, addressing widely accepted problems and showing rapid payback. Such tools do exist, but are mostly limited in capability, and so reach the limits of their effective contribution relatively quickly

  - In the medium term, the requirement is for high functionality tools that cover most or all of the life cycle and support sophisticated users in complex problem solving environments

  The leading tools try to address this dilemma by providing a simple to use user interface, that hides, but does not eliminate much more sophisticated functions and facilities. Nevertheless, someone adopting CASE for the first time must decide between:

  - Starting with a simple tool, and planning to replace it at a later stage with something more capable and sophisticated, with the consequent disruption, increase in acquisition costs (possibly including new hardware and software platforms), retraining requirements and implementation overheads

  - Starting with simple use of a more complex and capable tool, and actively managing the process of bringing more functions and facilities into effective use as user experience grows. This solves the disruption and duplication problem, but has consequences for training (an incremental training programme is needed) and requires much more management involvement in the process. It can also be frustrating for those users who want to move ahead faster than their managers are willing to allow

Both approaches have been tried in practice, with mixed results. The evidence suggests that organisations adopting the first approach reach the limits of their initial tools capability faster than they expect, but have considerable problems getting approval for the migration to a second tool. On the other hand, organisations adopting the second approach seem to reach a plateau in their use of the tools functions that is ahead of that achieved with simple tools, but well below the full capability of the selected product. Getting things moving forward again seems to pose significant difficulties.

One factor is, however, common in successful implementations. More than one tool is involved. The introduction of standards for interworking among CASE tools, particularly a standardised repository, will tend to promote the merits of multiple specialised tools rather than single vendor I-CASE products and further extend the tool kit concept.

- **Picking the Right Approach.** It's very common to find organisations starting their involvement with case without any clear idea of where they intend to go with their use of CASE tools. Developing a sensible CASE strategy is often something that they do after the initial experimentation, perhaps even after the initial implementation attempt. It makes a lot more sense, right from the start, to plan for the extent of CASE usage and the impact of a CASE implementation programme. Developing a CASE strategy can also save considerable effort in the investigation and experimentation stages by eliminating tools that are a poor match to the selected approach. Adopting CASE is no different from any other type of major project. It needs a definition of scope, of the approach to be adopted and of the selection criteria to be used. It's always cheaper to do this first, rather than trying to retrofit scope and approach to an existing situation.

  Once a CASE strategy is in place, the IS department can concentrate on implementation tactics. This should involve developing a short and medium term implementation plan, not just planning the investigation and implementation stages. The implementation process is not a short term effort, and treating it as though it is will reduce the chances of success, or at least delay the process significantly.

- **Getting Help.** Getting all of these elements in the implementation process right, and keeping every key aspect progressing at a compatible rate is a major challenge to IS departments, already faced with significant pressures on budgets and staff resources. It makes sense, therefore, to supplement IS resources with external resources and expertise. Many organisations are reluctant to do this, however, for a variety of reasons:

  - It's expensive, and not necessarily value for money unless effective transfer of skills and experience take place
  - It is seen to reflect badly on the capabilities and competence of the IS department
  - Many of those offering help have no more experience with CASE than those they are trying to assist

  Nevertheless there are very few examples of successful implementations that did not make use of external help, especially during the first few critical CASE-based projects.

- **Expectations.** Even if you get everything else right, it's essential to remember that CASE tools are just tools, and won't do anything for you unless they're used effectively. Even if they are used properly, they still won't do everything. All those involved with the introduction of CASE need to be reminded of this on a regular basis.

This may seem like a long list of things to get right, but changing the fundamentals of the system development process is not likely to be simple or straightforward. It's important to remember as well that you don't have to get all these things correct in one go or at the same time. What's critical is that you have them as objectives, and create a CASE strategy that works towards all of them. Then CASE will be a success for you. Leave any of them out and it probably won't.