CASE AND ITS CONTRIBUTION TO QUALITY

Alan Frazer

__CASE And The Software Crisis__ Originally computer systems were used to improve __efficiency__ by replacing tasks which previously had required significant paper-based filing systems and significant clerical effort - hence the term data processing. Soon efficiency systems were complimented with __effectiveness__ systems where the computer was used to provide key information to assist management with on-going decision making. Gradually these information systems began to address the needs of higher levels of management within the organisation resulting in the decision support system and the executive information system.

Today efficiency and effectiveness systems are being complimented by systems for __competitive advantage__ where the system is used to gain the edge on a competitor by providing a new service, drastically reducing product costs or even creating a totally new market altogether. Initially these started in sectors where information was a critical commodity such as financial services and travel but has now begun to penetrate organisations of all shapes and sizes irrespective of their business sectors.

Computer systems which function according to their intended users' requirements are very difficult to create. Even when they meet users' requirements typically they will have consumed much more effort to create than was originally intended and will have been delivered to the user much later than was originally promised. These problems are not limited just to the user of the software as the software developer is finding that the systems they have built are taking much more effort to maintain than was originally intended.

An accepted statistic is that for every six developers only about one and a half of them are writing new systems with the rest maintaining previously written systems. This causes further problems to the user as the developer's ability to deliver new systems diminishes as their maintenance burden grows. Hence the term applications backlog - the extent of the systems required by users which developers are unable to supply due to insufficient resources.

Much effort has been expended over the last forty years into improving the way we develop software systems. The focus of this has been on improving our software project management approach, techniques to help us capture requirements better and mechanisms to reduce the amount of software code which is written. The latter has concentrated on providing code reduction through high-level languages which make a given line of code carry out more processing and automation where software code is generated automatically from design information. Since about 1985 these advances in software development technology have been collectively referred to as CASE.

Most software commentators would agree that if the "software crisis" is to be abated it will be via the careful and selective application of appropriate CASE technology.

__The CASE Acronym__ Before we can introduce and define CASE we must first define what the acronym means. There are a number of different interpretations of the acronym and these include: "Computer Automated Software Engineering", "Computer Aided Systems Engineering" and "Computer Aided Software Engineering".

Alan Frazer is Consultancy Manager with VISION Software Engineering
VISION Software Engineering 2nd Floor Ulster Bank House Shaftesbury Square Belfast BT2 7DL
Tel: (0232)-313385 Fax: (0232)-313513

The first is inappropriate as the software engineering process will not be automated for many years, perhaps never. The second is favoured by the Government CCTA as it implies an element of hardware involvement. However, it is the final definition which is most widely accepted (it is also the definition used by the Commission of European Communities) and it is the one I will use here.

<u>**Definition of CASE**</u> The original definition of CASE was a narrow one and covered only a particular type of software tool designed to assist the software developer to follow techniques used in the analysis and design stages of the development. However, a much wider definition of CASE is accepted nowadays and CASE can be described as "any software tool used to assist a software engineer with any aspect of the software development process including the management and planning activities.".

Thus CASE includes:

Tools to DO the job (e.g. Analysis, Design, Development , Testing)
Tools to MANAGE & CONTROL the job (e.g. Project Management, Configuration Management)
Any Underlying METHODS & TECHNIQUES (e.g. SSADM, DFD, ERD)

and CASE excludes:

General purpose software (e.g. Word Processors, Spreadsheets
Applications software (e.g. Payroll Systems, Stock Control Systems)
"Primary software" (e.g. Compilers)

<u>**Usage and Importance**</u> CASE is used by software houses, computer systems vendors, vendors of non-IT systems with embedded IT systems (e.g. ABS in cars or function selection in cash dispensers) and computer systems users. The CASE market itself is not large (ECU 400 million in Western Europe in 1991 as compared to ECU 12 billion spent in Western Europe in 1991 [1]). However, CASE has a high profile and it seems clear that it will be key to bringing the software industry out of adolescence and assisting with the resolution of many of the industry's problems. The market is not large, but CASE is central to many issues in the industry. Some of the reasons for this are discussed below.

<u>**Benefits of CASE**</u> The benefits of CASE can be classified into two main areas - productivity gains and quality gains. Productivity gains are largely achieved by making the development process more efficient and the most obvious area where this has happened is in the coding area where code generators or 4th Generation Languages (4GLs) claim to improve productivity of development (and maintenance if correctly used) by up to a factor of ten. Quality gains are achieved by ensuring a rigorous and consistent approach to the development process and by assisting with product assessment at every stage of the development lifecycle. Figure 1 gives an indication of productivity and quality improvements achievable through correct usage of CASE

Figure 1 CASE Based Improvements in Quality and Productivity.

MEASURE OF	MEASURED IN	CASE IMPROVEMENTS
1 - Development Productivity	FP Per Months Effort	From 10 to 40 FP/M [2]
2 - Maintenance Scope	FP Per Developer	From 500 to 1500 FP [3]
3 - Internal (Process) Quality	Defects per FP	Down to < 4 [3]
4 - External (Product) Quality	User Discovered Defects	From 25% to 5% [3]
5 - External (Product) Quality	User Satisfaction Surveys	90% Excellent [3]

FP = Function Point(a measure of size)

<u>CASE and Quality</u> Five years ago productivity was of paramount importance in software production, however, now the spotlight is firmly on quality. The IT industry is currently the third largest in the world and is expected to be number one early next century. Thirty five percent of the EC's 4th Framework Research funding of 3,900 billion ECU is devoted to information and communication systems. Given this and the EC's acknowledgement that "I.T. competencies underpin all industry", the current interest in quality becomes understandable. Furthermore the knowledge that software errors can be enormously expensive and even fatal serves to increase the demand for quality.

<u>CASE and Product Quality</u> Software production is becoming an ever more complex activity and CASE tools and techniques are indispensable in delivering a quality product. Listed below are just some of the areas where CASE helps.

Testing is an established discipline in the software lifecycle and essentially involves reviewing the deliverable or exercising the code to check for errors or conformity to requirements. Tools can assist with managing the test data, monitoring results from successive versions and even identifying how much of the application has been checked with the tests employed. Techniques such as regression testing are also in widespread use.

Configuration management ensures that items in the development project are monitored, updated and released properly and that more than one developer is not updating a design object or code fragment at any one time.

Metrics provide information on code complexity, errors per lines of code etc. and these can be used to monitor pre-set site standards. CASE tools are essential for gathering, analysing and storing the metrics data.

<u>CASE and Process Quality</u> Much effort is currently focused on the software development process following the acceptance of the maxim that "a quality process will produce a quality product" and many initiatives are currently underway to certify or to assess and improve the software development process. The most often used slide in software conferences over the last two years has been the overview of the Software Process Maturity Model as produced by Watts Humphrey at the Software Engineering Institute in Carnegie Mellon University in the United States. This provides an assessment and improvement framework widely accepted throughout Europe and the States. In addition many development organisations are seeking BS 5750 and ISO9001 certification and the DTI has a TickIT initiative underway to help IT organisations achieve the BS5750 kite mark.

CASE can help improve the development process by ensuring consistency of approach (all staff could be encouraged to use the same diagramming notations or coding standards), by introducing new techniques (such as Object Orientation which offers a much improved re-use capability and therefore better use of tried and tested software modules) and by assisting with project management and control(by use of numerous CASE tool types such as estimating tools, risk analysis tools, project management tools etc.).

<u>Getting Ready for CASE</u> It should be noted that the introduction of CASE involves cultural and process/organisational issues and cannot be undertaken lightly. However the latest version of Software Process Maturity (i.e. the Capability Maturity Model) can be used to assess the 'gaps' between CASE plans and organisational abilities.

Conclusions CASE is essential if the software industry is to meet the demands made of it in the coming years and its use will be mandatory in producing stable and efficient systems.

Although the initial driving force behind CASE was productivity, the quest for quality will become the primary factor behind acceptance of CASE. Quality with productivity will be demanded and qualitivity will become the new buzzword.

Quality will drive the adoption of new CASE tools and techniques, but conversely without CASE quality will not be achievable.

References

[1] Source: OVUM
[2] The James Martin Productivity Series Volume 1 - High Productivity Technology
[3] Applied Software Measurement - Assuring Productivity and Quality
by Capers Jones published by McGraw Hill 1991