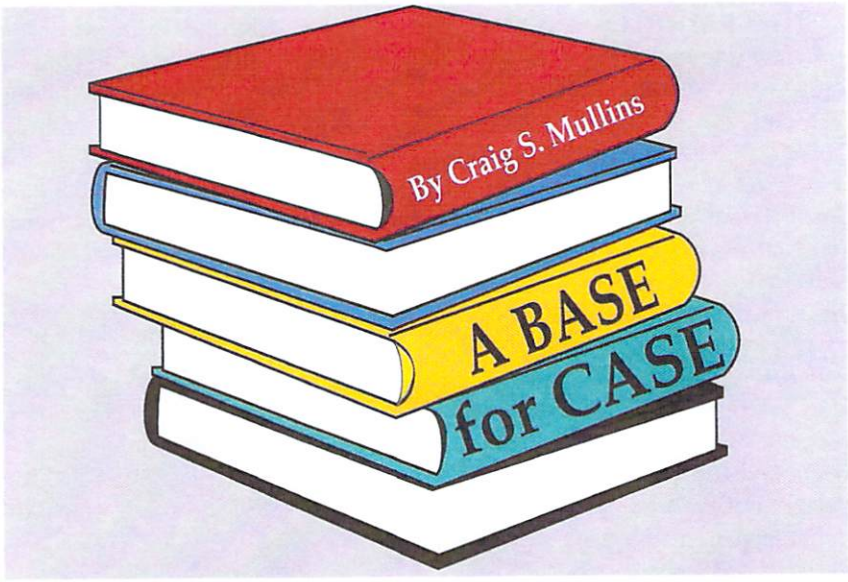

It seems like every new year brings with it a new buzz word in our industry. Some stay with us and others slowly fade into the past as a distant memory. The problem with buzz words is that very rarely is there a consensus as to an exact, precise definition for them. For example, ask 10 industry analysts what is meant by object-oriented (a current hot buzz word) and you will likely get 10 different opinions. Similar, maybe, but definitely not uniform.

Over the past few years we have been hearing quite a bit about CASE technology. CASE appears to be one of the buzz words that will stay with us. Yet few of us possess a firm understanding of the inherent capabilities and shortcomings of CASE. *What exactly is CASE? What can CASE accomplish? How can I effectively implement CASE at my company? Are programmers soon to become obsolete at the hands of CASE? Just how much of the marketing hype should we believe? Let's examine some of these questions.*

WHAT IS CASE?

It seems that almost every new product that is announced is touted as a CASE tool. Are the products that make these claims actually CASE tools? Simply labeling virtually everything as a CASE tool shrouds CASE in a veil of mystery. It is my opinion that this is counter-productive.

What is CASE, then? Quite simply, Computer Assisted Software Engineering. But this broad definition of CASE taken from its acronym is not very helpful. Doesn't the ISPF editor assist in your software engineering tasks? But who among us would call it a CASE tool? Evidently more rigor is needed to define what is truly meant by CASE.

Code generators (like APS from Intersolv or CSP from IBM), reverse and forward engineering products (like the Bachman Product Set) and full function data dictionaries are good examples of products that fulfill some of the promise of CASE. They are by no means a full implementation of all that is implied by CASE, but they are a good start. Beware of software such as program editors, programming languages and COBOL restructuring tools that are marketed as CASE tools. Don't get me wrong—they are very useful tools. But they aren't CASE tools.

The true power of CASE will be brought to bear by the integration of multiple "CASE" tools into a seamless software engineering environment. Some industry pundits separate CASE tools into two categories:

- Upper CASE tools which provide the user with the appropriate mechanisms to analyze, design and model software; and
- Lower CASE tools which enable models to be transformed into workable code.

Both of these are useful functions. However, the true power of CASE with respect to software engineering is the combination of these two categories into an integrated toolset.

Take special note of the term "software engineering" in the previous paragraph. This term is used on purpose, in place of application development or programming. It implies that an engineering discipline is involved. We must make software engineering the equivalent of, say, electrical engineering; providing a methodical blueprint for software development from specification through implementation and beyond. An integrated CASE environment can be the vehicle that will empower MIS with an engineering discipline by enforcing a rigorous, methodical development lifecycle through automation.

THE FUTURE OF CASE

CASE, in the coming years, will allow programmers to benefit from

computers in the same way that most other business functions have benefitted. Increased automation of the SDLC (Software Development Life Cycle) will be necessary to allow programmers to reap the benefits of computer assistance in their day-to-day jobs. This is the future of CASE. Will this eliminate programmers? One must keep in mind that people are still required to handle the business functions that were previously automated by computers. The only jobs that were completely eliminated were those that were repetitive and required no mental activity to complete. Does anyone mean to suggest that the role of a system designer, system analyst or senior programmer can be defined as requiring no mental activity?

In my opinion, no CASE tools currently exist that reliably automate a complete system development lifecycle. As the CASE industry has grown, some of the more robust tools have expanded to the point of covering a very wide spectrum of the SDLC, but the job is not complete, yet. When integrated tools covering the entire SDLC are available, DP will still not become obsolete. DP professionals are intimate with the workings of computers and know the best way to tackle a system development problem. The task of programming may ostensibly become obsolete but the very same people who used to program will be needed for systems analysis and design, and to understand the proper utilization of the CASE tools that produce the code. The nature of the beast may be modified, but the beast will not be eliminated. Simply stated, CASE will not enable business areas to develop every system as productively or as efficiently as a professional DP staff. Business areas should be involved in their day-to-day business, not in the development of computer systems. Why would a business analyst want to use a CASE tool? Would you use a CAD tool to design and architect your house without knowing anything about design and architecture?

INTEGRATING CASE TOOLS

Without a doubt, the most important aspect of CASE is integration. However, this fact is widely ignored. Some DP shops live with a situation where different CASE tools are used for different portions of the SDLC. With different vendors supplying a wide variety of different CASE tools, a situation develops whereby data needs to be passed back and forth between different products. Each tool has its own internal storage format, though, and the data is not easily shareable. Time and effort is wasted when this integration must be implemented by programmers. It can negate the time that is saved by using the CASE tool in the first place.

Take, for example, four of the most successful CASE tools in the IBM environment: Bachman, Knowledgeware (ADW), Intersolv (Excelerator and APS) and Texas Instruments (IEF). The tools that these vendors supply each support the software development lifecycle. Each tool has different strengths and weaknesses. If you wish to use different tools for different phases of the SDLC (to exploit the strengths of a particular tool for a particular task) you will be in for a difficult time. There is no integrated method for system development using multiple tools. Each tool stores data in a different format. The only form of integration that is available is an import/export facility for each product. This is inadequate because it does not address the integrity of the models across platforms. The "integration" must still be manually controlled in order to ensure integrity. This is one example of the integration problems that surround CASE.

When integration problems are solved, only then will the true promise of CASE emerge. For awhile, many felt that this problem would be solved by the IBM Repository. But time has shown that this product will never be successful (at least in its current incarnation). Standardization can only be viewed as a good development for the future of CASE. Obviously, some sort of standard repository and storage is desperately needed. But there are hurdles still to be jumped here as well. When will a general purpose repository be generally usable by the majority of the industry? Will it be the IBM Repository? Should it be a mainframe-based tool or will it (must it) follow a client/server framework? How will it perform?

A SOLID BASE

If computers can assist in lending rigor to the process of system analysis, design and development then this should be strived for. This is the true essence of CASE. However, we should not race ahead of technology. Today, a computer simply cannot develop application systems without human involvement. CASE tools can ease the system development process, but they cannot replace it.

The underlying theme of this diatribe is that a proper BASE is necessary before applying CASE. Development (and construction) processes of every type, from the construction of a building to the development of an application system, require a strong foundation, or BASE, in order to be successful. BASE, in the context of system development, can be thought of as Brain Assisted Software Engineering. Without BASE, the implementation and usage of CASE tools within your company will be unsuccessful. Let's not lose sight of the basics in the whirlwind of hype and exaggeration that surrounds CASE today.

Craig S. Mullins is a member of the Technical Advisory Group at PLATINUM technology, inc. He has more than seven years of experience in all facets of data base systems development, including developing and teaching DB2 classes, systems analysis and design, data base and system administration and data analysis.