## CASE CSFs

CASE is considered a strategy to reduce system development time, cut maintenance costs, and facilitate the greater standardization of work procedures and adherence to design discipline. However, it has been hard to implement, due to such obstacles as cost, resistance by system developers, and unacceptable learning curves. There may be a mismatch between the work methods that CASE tools support, and the actual activities of information system designers, so that CASE is not effective when it is superimposed upon current work systems.

A study was done to examine the critical success factors (CSF) in information systems analysis and design, and whether CASE tools support these CSFs. For requirements analysis, two factors were considered both important and difficult to achieve: the ability to involve the client in the development process, and the ability to set the scope of a project. For systems design, the ability to understand the client's business was found to be most important and difficult.

In detailed system design, the ability to establish effective communications between the designer and the user was considered most important. Most difficult was the ability to coordinate project activities so that tasks are completed within time and cost constraints.

Finally, for implementation, the ability to obtain customer acceptance of the final product was most important. Most difficult was the ability to manage the process of organizational change.

The study found that CASE users did not view CASE tools as supporting the achievement of highly ranked CSFs in systems development. The technical focus of CASE technology may explain the findings of this study.

CASE is not a panacea that can improve the productivity of the system development process by itself. Instead, as part of a total work system in which both technical and social objectives are addressed, CASE technology may have an impact on improving the quality of the system development process.

**Source:** "The impact of CASE: Can it achieve critical success factors?" by Mary Sumner and Terence Ryan, *The Journal of Systems Management*, June 1994, pp. 16-21.

ENGINEERING

# CASE: Not a Joke, Now a Threat

**By Joseph R. Schofield, Jr.**

*Many projects with new technologies doomed in the womb; True CASE tools a threat to masquerading software engineers*

Atrocities to avoid, helpful hints, horror stories, user experiences; such topics abound in industry journals, especially with relationship to the introduction of new technologies. "Implement it faster," "Minimize the learning curve (or is that curse?)," and "Improve today's work flow," are all familiar battle cries. And, of course, there's the issue of whom to ask when the technology isn't achieving all these myopic goals.

Despite the plethora of available training, consulting, and reported expertise, many projects undertaken with new technologies are never implemented. Instead, they are doomed in the womb. This dilemma has raised its head once again with computer-aided software engineering (CASE). Many CASE pilots have failed because the technology was exploited for reasons related to the glisten, glitter, or novelty of CASE; or to sheer desperation.

So this article is not another potpourri of tidbits on how to introduce a new technology into an organization. Rather, it's a revelation of the turmoil experienced after an apparently successful technological introduction.

### Three-stage Technology Life Cycle

Michael Treacy, formerly of MIT's Sloan School of Management, is credited with describing a three-stage life cycle for the introduction of new technology. Those three phases are the "joke," "threat," and "obvious" stages.

The "joke" stage occurs during the unveiling of something new. Words and phrases used to describe this phase include: 'new and coming technology,' 'not quite there yet,' and 'worth investigating in the future.'

The "threat" stage occurs as early success stories are reported, and other technologies are challenged. Once the technology is more widely accepted as less than heretical, its maturity fosters the "obvious" stage.

Industry is far from recognizing the "obviousness" of CASE. Yet it is quite apparent that CASE has entered into the "threat" stage in some organizations. Until now, however, I am unaware of any author who has described the impact of dealing with the "threat" of CASE.

Remember that CASE has been unsuccessful in a number of organizations; that is, the technology has not delivered automated solutions to business problems in a timely fashion. But let's shift our focus away from the cacophony of philosophical entanglements so often explored in this situation, and excavate some yet-to-be exonerated factors.

### All-too-typical Project

Imagine the all-too-typical project: less than optimal resources, unrealistic schedules, matrix project composition, team turnover, and creeping requirements. Since few of us relate to a typical project, let's add some "visibility" to this pilot.

Flavor it with a corporate migration toward quality and customer satisfaction, divided sponsorship agendas (a need to migrate to a new hardware environment versus the need to imitate an existing environment), an untested communications network, and insidious security problems. Finally, crown the list with a new technology: CASE. While this abbreviated list of ailments may appear formidable to the timid of heart, what follows is a synopsis of an actual CASE pilot.

### Successful but Threatening

Accelerated processing in the new system pleased the primary customer. Second

ary customers expressed jubilation with the expanded capabilities, calling them "terrific" and "wonderful." More customers used the system for more transactions than the system it replaced.

Management confidence in the technology resulted in the initiation of three additional projects. Oh sure, you may say, qualitative data may be an indicator of satisfaction, but what about measured results?

The application, originally developed in 22 person-years, was completed in less than 14. The first day of implementation, one transaction was executed at the equivalent of 1,000 times per hour during the first eight hours of implementation. Furthermore, after 27 months of daily on-line production usage, the application has never "crashed." Successful? Well, perhaps. Threatening? Definitely!

Despite years of attempted effort in applying engineering principles to software, the reliance on rigorous and structured tool-enforced methodologies threatens that lingering liberty also known as creativity. Manifestations of this threat take other forms: the theories that methodologies need to be loosely defined to fit the particular task; that CASE tools are like religion (implying divisive or cultic); that not every project needs a methodology; or that there's no one right way to engineer software.

True CASE tools are a threat to masquerading software engineers who avoid structured processes. Validation checks against CASE models in the analysis and design phases of software development can identify defective products much earlier and less expensively than otherwise, while enforcing unprecedented levels of integrity. Thus CASE engineers can articulate the life cycle processes used, while those who are threatened continue contemplating their drawing tool of choice.

CASE threatens the culture of an organization. Partners in the development process, formerly called "users," need to be able to define the application's business requirements. However, this task continues to challenge whatever technology is employed because partners can alter those requirements—often quite easily with CASE.

As needs arise later in the project, then, the mode of infinitely prototyping through requirements that takes place using CASE, calls for less commitment and reduced cerebral activity than system developers are accustomed to providing. In turn, this shifting may cause uncertainty regarding their job requirements and performance. In times of project turmoil and uncertainty, well-defined CASE processes also eliminate the project safety valves afforded by the ill-defined processes that were used previously.

CASE is certainly a threat to potential CASE engineers. Some, unable or unwilling to apply new techniques, fall as early victims. Others, who have experienced CASE, drop out, convinced that other threats will eventually overwhelm what should be the obvious benefits. Unrelenting doubts about the future use of CASE, at all levels in the organization, fuel frustration, and even change the hearts of former risk takers.

*Vendor promises of CASE benefits have heightened the "threat" of CASE.*

Vendor promises of CASE benefits, an effective, if self-serving marketing strategy, have heightened the "threat" of CASE. One vendor in particular was perceived to have "frozen" the CASE market in September of 1989, by promising a suite of tools intended to legitimize CASE. Those promises were, however, empty. The threat of what might next be promised, and later reneged on, limits the wide-scale acceptance and implementation of CASE in some organizations.

## CASE Characterization

The variety of perceptions associated with CASE today dictates clarification when the acronym is used. Therefore, I will provide a characterization of CASE that will be used for the remainder of this article.

First, the toolset is integrated. Thus it appears the same between phases, it has the "look and feel" of a single tool, and it is coordinated by a single encyclopedia. Integrated toolsets are not reliant on import and export facilities, conversions to other formats, or accounting for a multitude of versions and releases from a superset of suppliers.

Second, the toolset is based upon a methodology, providing enforcement and integrity to the models. Third, the toolset supports the entire system life cycle. It is not merely a standalone conceptual representation in the analysis and design phases, nor does it merely provide code generation toward the end of the life cycle, void of consistent foundational models. And the clear definitions provided can be a continuing threat to competing CASE suppliers.

A recovering survivor of a CASE pilot might well ask: "How justified is the threat perception?" One approach to answering this issue, to be taken here, is to explore what was expected of CASE implementation, versus what actually happened.

## Treachery of the Learning Curve

At Sandia National Laboratories, we were forewarned about the treachery of the learning curve—no imaginary obstacle. After two and one-half years of CASE use, not all of our CASE veterans understood every aspect of the tool. We acknowledged this by deliberate division of labor.

While the learning curve did not grow progressively difficult, it sustained itself as we journeyed from one phase of the pilot to the next. For most team members, it took six to nine months to achieve a comfort level with the toolset.

Avoiding CASE use leads to obsolescence, not a surprising finding, nor one unique to any new technology. However, CASE development using a consistent, single-user interface accelerates both the learning process, as well as relearning where necessary.

Training costs were expected to approach $4,000 per "seat." Courses were added to the curriculum mix, raising the total to about $5,000 per seat, a reconcilable difference. The resulting successful completion of the project without on-site or intensive ongoing consulting support still mystifies the supplier, who would have preferred to partner with the development team.

## Justifying the Funding

Earlier, a productivity improvement of eight person-years was mentioned, which did not consider the impact of increased functionality, nor the more complex environment. A cost comparison of projected versus actual labor rates demonstrates that the savings on people costs alone would have justified the funding, not only to procure the CASE toolset, but two additional sets as well. In viewing the cost savings data from another perspective, we found that the initial CASE project saved three times the cost of the product in labor costs alone!

If there are no corresponding productivity metrics from the original project, comparisons are mere conjecture. The calculated function point per person-month (FPPPM) metric of 13.7 for this pilot thus seems rather dismal without accompanying detail, which will not be discussed here.

Instead, recent numbers released by Capers Jones in the January 1993 issue of *Software Magazine* seem to shed a more gracious light on the FPPPM metric. More importantly, two subsequent, albeit smaller and more well-contained CASE initiatives, have yielded FPPPM exceeding 150.

However, reports of high productivity are so threatening that management often demands proof. A potential benefit of this disbelief might be implementation of aggressive new metric programs.

**The Market**

If one observes the alternatively expanding and shrinking number of suppliers in the CASE market, it appears that the threat of lost support could be an additional impediment to long-term commitment to a particular tool or vendor. Without consulting tarot cards or other predictive methods, it seems that immunity to failure cannot be guaranteed for any hardware or software supplier today. And obvious considerations such as the size and history of a supplier are, at best, only rules of thumb.

However, the loss of a CASE supplier does not imply that the purchasing company cannot continue to use the toolset. A predominating premise preceding our investment in CASE was that the tool maintain models—*never* that the software engineer not maintain code—the CASE tool generates the code from the models.

By applying this principle, we ensure consistency throughout the model, and disallow any alteration of the generated code. However, the decision to maintain models threatens those accustomed to changing the code under impending urgency at 3:00 a.m.

Given the ongoing threats of introducing new technologies and, in particular, CASE, it is no simple matter to identify projects that will be successful. Since CASE threatens existing methodologies and techniques, neither qualitative nor quantitative prognoses or assessments are conclusive. However, Treacy's three-phase technology progression, which predicts a transition from the "threat" to the "obvious" stage, provides immeasurable reassurance against hysteria.

*This work, performed at Sandia National Laboratories, is supported by the United States Department of Energy under contract number DE-AC04-76DP00789.*

*Joseph Schofield, Jr., CQA, is a computer systems specialist at Sandia National Laboratories in Albuquerque, NM. In the computing field since 1969, he has a masters degree in MIS from the University of Arizona, and specializes in software metrics and software productivity. He's spoken on these topics at many conferences, including CASE World, Guide, and Share, and teaches graduate classes in the field of computing at the College of Santa Fe.*

HOLOGY

# "Reward Not-Yet-Success In Order To Succeed"

**By Charles B. Maclean**

*How R&D people motivate themselves when factors out of their control deep-six a product that they gave their blood, sweat, and tears to produce*

Albatross companies still call it failure. They punish, ostracize, demote, isolate, chastise publicly, or fire those who "fail."

But at InFocus Systems in Wilsonville, Oregon, manufacturers of bright image LCD projection systems, they're committed to "creating an environment where outstanding people do extraordinary things."

Unlike most companies, their strategic plan includes the value of rewarding "informed risk taking"—separate from the short-term result. How do they do that?

Daniele Joudene, manager of training and development at InFocus, says, "We model, look for, teach, and reward these behaviors:

- Embracing change,
- Challenging paradigms,
- Listening to all ideas and viewpoints,
- Learning from successes and mistakes, and
- Encouraging breakthrough, visionary thinking."

But does it work, or is it just good eyewash on paper? The people of InFocus Systems introduce six to eight totally new products to the marketplace each year. Their average revenue per employe has grown from $277K per employee in 1991 to $450K per employee in 1994, at the same time the number of employees has doubled. Their profit margin has grown from minus 8.8 percent in 1989, to plus 13.7 percent for the period ending July 1994.

Tom Peters, author of *Liberation Management*, issues a wake-up in saying, "Life is pretty simple: You do some stuff. Most fails. Some works. You do more of what works. If it works big, others quickly copy it. Then you do something else. The trick is in the doing something else. You must take pot shots at today's star before you are mimicked. Today's radiantly blooming flowers are tomorrow's mulch."

It's no secret that worldwide competition has shortened the mind-to-marketplace cycle time. It means that best efforts may produce a great product that never makes it to market, because a competitor's cycle was shorter by six weeks, and the window of opportunity in the marketplace was lost. How do research and development people motivate themselves when factors out of their control deep-six a product that they gave their blood, sweat, and tears to produce?

At one software company, an "acknowledgment deprived" research and development team was working under a stressful deadline (oops, make that lifeline). They decided not to wait for top management to recognize their best efforts against great odds, especially because the project outcome was uncertain. They decided to acknowledge each other and motivate themselves.

With $250 from petty cash, they filled a file drawer with the stuff of instant rewards. The "stuff" included $10 lunch certificates, chocolate, coffee mugs, trashy novels, and cartoon books. Any team member who sees another team member going the extra mile can give an instant reward (one of the principles of encouraging thoughtful risk taking).

All they do is say, "I appreciate your _____(fill in the behavior being appreciated). Walk to the file drawer with me. Now pick something."

They merely write down in the log who was being recognized, and for what. From time to time at staff meetings the team re