Software Technology People

Elliot J. Chikofsky, Index Technology

More than a decade after its introduction, CASE is emerging as a real-world technology whose promises are being fulfilled.

Computer-aided software engineering (CASE) has recently emerged as a widespread application of software-engineering techniques and computer technology to the current practice of information-systems development. For many of us in the field, this constitutes the fulfillment of 12 years or more of effort introducing tools and addressing the difficult issues of technology transfer.

Methods that can improve the practice of software and system development, including structured analysis and structured design, have been available for more than a decade, but they have not fundamentally altered the daily practice of systems organizations. In his conference presentations, Ed Yourdon estimates that, while 90 percent of software professionals are familiar with structured methods and 50 percent of their organizations have tried them at some time, only 10 percent actively use them. For many organizations, the labor-intensive nature of such methods outweighed the improvement they produced.

CASE changes the equation. Relying on readily available desktop computer technology — perhaps supported by local-area networks, central servers, or mainframe hosts — CASE environments make the general application of formal system-development methods practical and economical. They support the developer's work throughout the life cycle and give the project team analytic tools to ensure consistency, completeness, and conformance to standards. As Jeff Whitten of Purdue University noted at the May 1987 international workshop on CASE, "In the past, the tug of war between efforts to produce high-quality systems and efforts to improve productivity in systems development meant that one of these had to give. But CASE offers the best of both worlds, with less trade-off."

The CASE field

While the acronym "CASE" had appeared earlier, its acceptance as the identifier of the field was directly related to a Sept. 24, 1986, Wall Street Journal article. Now, the term "CASE" has achieved buzzword status in the marketplace. It raises images of CAD/CAM for software, which only partially captures its diversity. It has become a rallying cry for anyone with a new or repackaged software tool to sell. To differentiate the types of tools out there, some observers have already begun to classify each offering by its role in the life cycle, creating terms like "upper CASE" (the front end of the life cycle) and "lower CASE" (the back end).

What, then, is CASE? There is, as yet, no clearly accepted definition that satisfies the wide range of components and technologies under the CASE umbrella. CASE is primarily a production-oriented integration technology to meaningfully improve software and systems development. It brings together the fruits of several diverse streams of development methods and tool research into effective and commercially viable environments.

History. In many respects, CASE has been around in one form or another for many years. In January 1977, a landmark issue of IEEE Transactions on Software Engineering identified several approaches and pioneering research prototypes for improving requirements analysis and the front end of the system life cycle.
By 1981, the objective of producing viable software-development environments was firmly established, as exemplified by Tony Wasserman's tutorial. Software tool fairs became (and still are) such a regular part of technical meetings that, in 1983 and again in 1985, the ACM and the IEEE held Softfair conferences dedicated to software-development tools, techniques, and alternatives. Also in 1985, the April issue of Computer was dedicated to requirements-engineering environments, updating the 1977 Transactions on Software Engineering material.

At the same time, the IBM PC was becoming accepted as a usable platform for software development and was acquiring sufficient graphics capabilities to be a viable design workstation. The availability of a cost-effective delivery vehicle for commercial software-development environments triggered the broad introduction and widespread growth of CASE technology. This has been further supported by the rapidly decreasing price/performance ratio of 32-bit workstations.

Potential. With its orientation toward making practical and sound improvement in the actual practice of systems development, CASE has the potential to dramatically affect the software industry for years to come. Carma McClure has projected that "by the 1990s, CASE tools and individual workstations will be as common to software development as programming languages and compilers have been for the last three decades." She also contends that "CASE will soon replace 4GLs [fourth-generation languages] as the most powerful applications-development methodology."

Will this come to pass? It depends on how well the field continues to integrate the diverse technologies on which it depends.

Key areas

This issue of IEEE Software provides an overview of key areas in the CASE field. The articles were selected to present the broad picture of software-engineering domains that CASE brings together. Many of the articles were derived from the First International Workshop on CASE, which was held in May 1987 to provide a focus for advancing CASE technology as its scope and direction coalesces. To continue this work, CASE 88, the second international workshop, will be held this July in Cambridge, Mass., in cooperation with the Computer Society.

In this issue, we examine:

- Integration. CASE integrates multiple tools and facilities into a consistent environment. This is intended to provide a tailorables framework into which new tools and methods can be placed.

In this issue, Burt Rubenstein and I provide a structural look at CASE environments in the article "CASE: Reliability Engineering for Information Systems,"

while Bill Cureton's "The Future of Unix in the CASE Renaissance" argues that Unix is the essential link in building CASE environments that both the commercial and scientific communities can use.

- Database. The key component of a CASE environment is the integrated central repository of information. The application of advances in database technology, such as entity-relationship and object-oriented databases, is critical to CASE's evolution.

In "Hypertext and CASE," James Bigelow explores how the now-popular medium of hypertext applies to CASE architecture and the central repository. Ed Acly's "Looking Beyond CASE" considers the entity-relationship-based ANSI IRDS model. "The Metaview System for Many Specification Environments" by Paul Sorenson, Paul Tremblay, and Andrew McAllister illustrates the generalization of the repository concept through metamodeling approaches to generate customized CASE environments.

- Standards. The interconnection of tools from different vendors, particularly covering different phases of the life cycle, is already recognized as a key strategy in the CASE marketplace. As the technology matures, proposals for integration and exchange standards are beginning to emerge. The IEEE, for example, has recently formed a task force on professional tools to examine this area. In fact, there are now many independent standards efforts affecting CASE that a special coordination meeting between standards groups will be held at the July CASE 88 workshop.

The Cureton and Acly articles both examine the desirability of applying stan-
standards to CASE — Cureton looks at the evolving Unix standard while Acly examines the recently approved ANSI IRDS dictionary standard.

- Evolution. The capabilities of CASE environments must continue to advance, particularly for the power user, if the goal of life-cycle-wide automated development is to be achieved. The current state of CASE tools is but the first step in this evolution.

In "Second-Generation CASE Tools: A Challenge to Vendors," Charles Martin identifies some opportunities and requirements for the next advancement of CASE technology.

- Expert systems. CASE is ripe for the meaningful application of expert-systems technology. There are now many research projects looking at the creation of automated programming and design assistants.

In "ASPIS: A Knowledge-Based CASE Environment," Paolo Puncello and colleagues examine one such project under the auspices of the European Community's ESPRIT program that has applied AI approaches to the development of an analysis assistant. In "Creating a Software-Engineering Knowledge Base," Andrew Symonds identifies some of the issues in using knowledge-engineering technology with CASE and the application of these combined technologies to operating-system development.

- Assurance. The presence of a consistent description of the target system in the CASE repository affords new opportunities for analysis, testing, and reliability determination. The combination of simulation and prototyping capabilities with CASE environments offers the prospect of assuring system operation and performance before construction.

Luqi and Mohammad Ketabchi describe an approach to rapid prototyping with reusable components in "A Computer-Aided Prototyping System."

Beyond technology

What is missing from the areas cited above? People. Software-development productivity will not just have a technological solution. Organizations introducing CASE technology need to recognize the people-management issues involved. Well-planned training and consulting are very important to the successful introduction of, and continuing benefit from, CASE. While beyond the scope of this issue of IEEE Software, the area of management strategies for CASE is an important partner to the software technology.

References


Address questions about this issue to Chikofsky at Index Technology, 1 Main St., Cambridge, MA 02142; CSnet chikofsky@vaxe.coe.northeastern.edu.

---

ProMod Is... The Complete CASE Tool Set.

- Structured Analysis • PDL Editor
- Real Time Structured Design Aid • Ada Code Frame Generator
- Full Design Documentation • Module Design Language

ProMod. From requirements analysis to program code, ProMod gives you the complete tool set for your software project.

Transformation logic ensures that consistency is carried forward from each project phase to the next. Traceability options enable you to easily demonstrate the connection of code modules to requirements, while automatically generated documentation records current design information in each phase.

That's why ProMod's complete software lifecycle CASE modules are the best choice for your software design.

For more information on why ProMod is the complete, automatic integrated CASE system for your PC and VAX systems, call our toll-free numbers now.

Outside California: 1-800-255-2689 • 1-800-255-4310

In California: 1-800-255-2689 • 1-800-255-4310

PROMOD

23685 Bircher Drive, Lake Forest, CA 92630

Reader Service Number 3