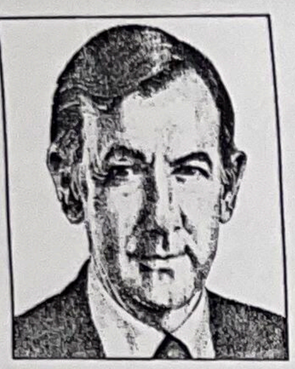
APPLIED INTELLIGENCE

Design Projects Require Tools with Multiple Functions



JAMES MARTIN

Although last week's column focused on the sweeping productivity benefits of integrated CASE tools, these tools fall short of providing a range of auxiliary or secondary functions. Therefore, users seeking project-managment assistance for design projects or support for alternative operating environments need to evaluate the non-integrated tool class that delivers many of these specialized capabilities.

To reiterate, the following functions are critical to obtain maximum productivity from a CASE tool: a graphics-oriented interface; support for the entire life-cycle process; desktop generation of code for 100 percent of an application; generation of code in COBOL, C, Ada, etc.; and support for industry standards such as the Structured Query Language (SQL) and IBM's AD/Cycle repository standard, which defines design information.

While the best-selling I-CASE tools support most of these primary requirements for high-end productivity, they lack the palette of extra functions that some users deem critical for the full application-development life cycle.

As shown in the figure, these additional functions include the following:

- generation of applications for the Unix environment;
- support for multiple analysts working on a LAN;
- support for real-time applications;
- project-management functions;
- configuration and version control;
- requirements traceability, or software used to generate reports tracing the linkage between original requirements specifications and the resulting source code and documentation;
- software backplane, or the ability to mix products from multiple vendors;
- compatibility with industry-standard architectures such as IBM's Systems Application Architecture and Digital Equipment's Network Application Support;
- support for cooperative-processing and client/server applications;
- portability of applications among multiple platforms;
- forward and reverse engineering; and
- embedded methodology aids.

A critical attribute of the non-integrated CASE category is support for the Unix environment—a trait not supported in any I-CASE tool.

With the worldwide pace quickening toward Unix, tools need to be capable of generating a full application in C language for a Unix target environment.

Most I-CASE tools also are unable to adequately support a work-group environment, which allows multiple analysts to share and consolidate design specifications across a LAN.

Similarly, most I-CASE tools provide poor support for real-time applications and for an integrated project support environment (IPSE).

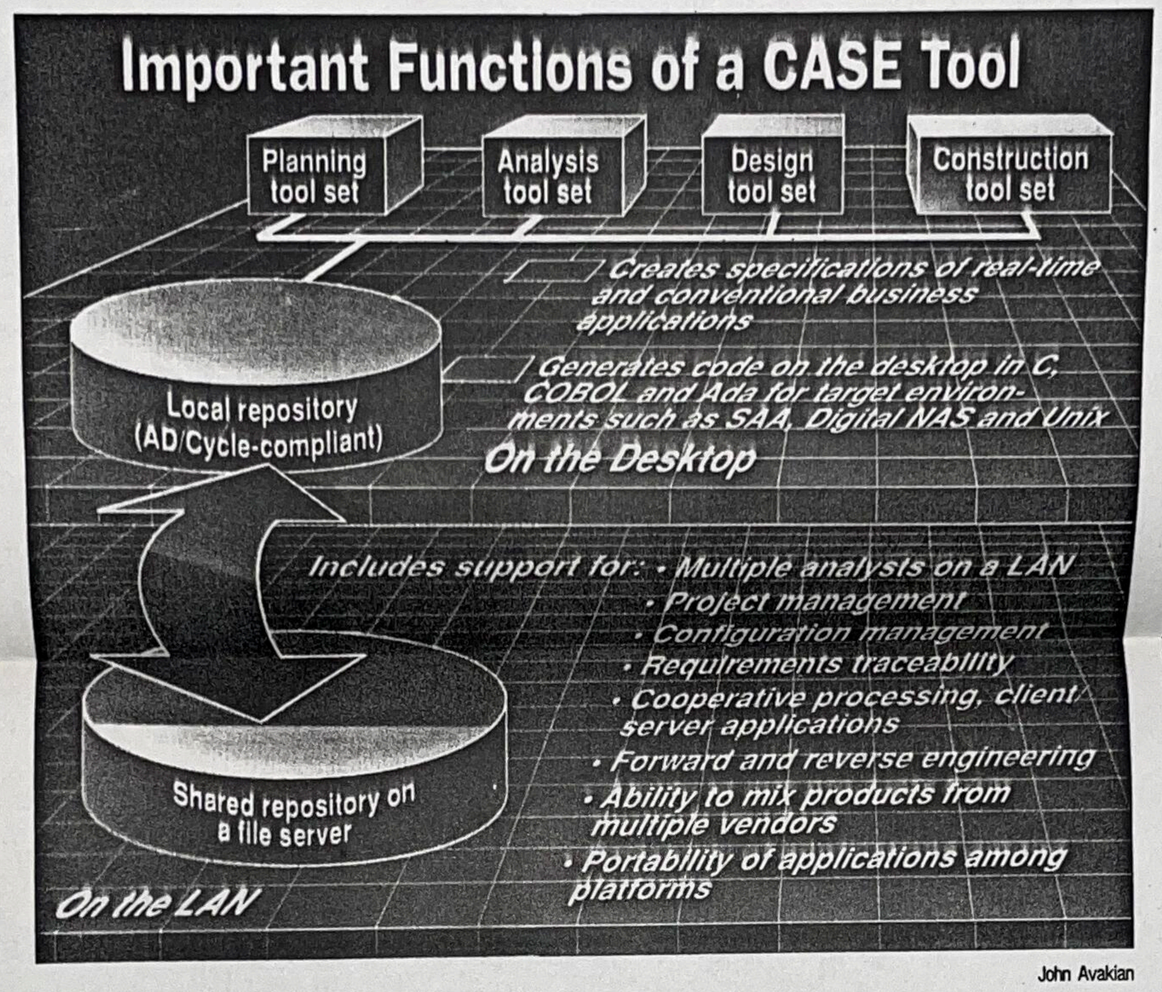
To be competitive, CASE tools will have to support requirements such as a software-backplane environment that will enable buyers to select "best of breed" CASE products for each phase of the life cycle.

CASE tools will also be required to handle cooperative-processing and client/server applications that are compliant with the services offered by IBM's SAA or Digital's NAS environments. In turn, these applications should be portable across a variety of hardware and software platforms.

Reverse engineering and re-engineering functions also can be critical to the success of a CASE tool. Reverse engineering is an analyst-assisted process that extracts high-level specifications from existing source code. Re-engineering merely scans the source code to resolve inconsistencies, while restructuring process or data definitions.

Embedded methodology aids, which lead the user through the development process step-by-step, are also a coveted extra feature in a CASE tool. To this end, embedded computer-based training tools will offer this training within a window in the CASE tool.

Next week, I will discuss the questions that should be asked when assessing the functionality of a CASE tool for your organization. ■



John Avakian

The concepts embodied in this article are described in the CASE volume in The James Martin Report Series. For more information on this volume, call (617) 639-1958. For information on seminars, contact (in the United States and Canada) Technology Transfer Institute, 741 10th St., Santa Monica, Calif. 90402 (213) 394-8305. In Europe, contact Savant, 2 New St., Carnforth, Lancs., LA5 9BX United Kingdom (0524) 734 505.

FourGen \ *Code Generator Revised*

Continued from Page 55

ware Inc. in Edmonds, Wash.

"This new code generator can't be compared with the old version because of the number of new features that we've added," Kloes said.

The new screen painter, designed for both novice and expert developers, builds data-entry screens simply by selecting the appropriate fields from an Informix database dictionary, Kloes explained. Once the screen design is complete, FourGen-Screen automatically produces the necessary code, he added.

The new version also delivers added flexibility in allowing developers to customize applications with special features, according to Michael Gonzales, president of Databank Information Systems Inc., a software-development contractor in El Paso, Texas. Databank is using the new version of FourGen-Screen to develop FourGen's in-house order-entry and inventory-control application, Gonzales said.

The new version lets users easily navigate through the system to gather the information needed to build applications, Gonzales said.

"You can position the data fields wherever you care to, and if you are unsure of the name of a specific table or column, you can hit a control key to list all available data," Gonzales explained.

Users can attach free-form notes to any data field—a feature that was lacking in the original FourGen code generator. "Attaching a note to a field usually requires a lot of extra coding," Gonzales said. The new version lets users attach notes to a field or a column without changing any of the underlying code, he added.

FourGen-Screen will run with a variety of Informix-4GL versions for Unix, as well as Unix operating systems from The Santa Cruz Operation Inc., AT&T, Digital Equipment Corp. and NCR Corp. Pricing starts at $2,495.

FourGen Software Inc. can be reached at (206) 776-5088. ■

Borland \ *Code Reusability Is Key*

Continued from Page 55

Projects that lend themselves to OOP languages are medium-sized to large applications, graphical user interfaces and integrated applications, according to David Intersimone, director of developer relations for the Scotts Vally, Calif., company. Device drivers or applications that have critical dependencies on efficiency and size are not good candidates for OOP, he said.

Competent C programmers can expect to learn C++ syntax in about four weeks and be up and running in about three to six months, Intersimone claimed.

Some attendees disagreed about the time frames. "I thought [Borland] was overly optimistic," said Paul Johnson, vice president of research and development for Great American Software Inc., a software developer in Nashua, N.H.

Even Borland officials acknowledge that the learning curve involved in switching to an object-oriented style can mean a significant investment in development time—initially, personnel hours could be magnified by as much as 40 percent, they said. Nonetheless, seminar attendees were enthusiastic about the long-range productivity payoffs—mainly, minimal redundant coding and fewer programmers required for a project.

"The primary benefit will be reusability of code, because I can't afford the number of programmers it takes now to keep up with our work and [stay competitive]," said David Steinberg, a senior software engineer for Analogic Inc., an electronics maufacturer in Wakefield, Mass.

Though interest was high, most seminar attendees were just evaluating C++ or using it for the first time. One factor that hinders its acceptance, attendees said, is the lack of debuggers, browsers and a full development environment for the language.

"I would like to use C++ in the next six months because there are a lot of clear advantages, but if the tools aren't there it won't do us much good," said Siemens' Bishop. ■