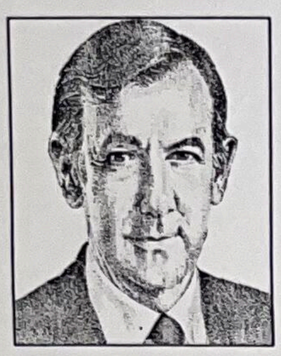
APPLIED INTELLIGENCE

Diagramming Methods Bring Precision to CASE Tools



JAMES MARTIN

In this, the second of a six-part series on integrated computer-aided software engineering, or I-CASE, technology, James Martin discusses the CASE tool function of converting diagram specifications directly into operational code.

The evolution of diagramming has picked up staggering speed in the last few years. Not long ago, systems analysts drew their diagrams with pencils and erasers.

Hand-drawn diagrams often grew very large, straggling across white boards or pasted onto large sheets of paper. And a design frequently consisted of multiple binders of nested data-flow diagrams and structure charts. These binders were usually fraught with inconsistencies and omissions. What's more, the diagrams were sloppy and the diagramming technique ill-conceived.

CASE tools bring to diagramming a method for enforced precision. A good CASE tool uses diagram types that are precise and computer-checkable.

Among the diagram types used by CASE tools implementing an information-engineering methodology are decomposition diagrams, dependency diagrams, data-flow diagrams, action diagrams (for specification of procedures), data-analysis diagrams, data-structure diagrams, entity-relationship diagrams, data-navigation diagrams, decision trees and tables, state-transition diagrams and dialogue-design diagrams.

CASE tools that support a software-engineering methodology use a subset of these diagram types: chief data-flow diagrams, decomposition diagrams and entity-relationship diagrams.

Large, complex diagrams can be handled by means of zooming, nesting and windowing, among other techniques. The computer quickly catches errors and inconsistencies even in very large sets of diagrams.

Today, business, government and the military need highly complex and integrated computer applications. The size and complexity of these applications are too great for any hope of accuracy in diagramming without aid of a computer.

Remember, it's the meaning represented by the diagram that's valuable. A good CASE tool stores the meaning of the diagram, not the diagram itself, in a computer-processible form. The tool helps build up a design, a data model or another deliverable segment of the development process in such a way that it can be validated and then used in a later development stage.

Diagrams and their manipulation by computer are forms of thought processing. The analyst, designer, programmer, user and executive need a family of diagram types to assist in clear thinking. Although there are a host of diagram types, a minimum number of icons should have to be learned, and their meanings should be as obvious as possible.

The diagrams must be sufficiently complete and rigorous to serve as a basis for code generation and for automatic conversion of one type of diagram into another.

The diagrams of the early "structured revolution" aren't good enough for this. In this earlier technology, the analyst and designer had to use human intelligence to bridge gaps between one type of diagram and another, and they often made mistakes in doing so. I-CASE tools need a complete, rigorous set of diagramming standards.

With appropriate diagramming techniques, it's much easier to describe complex activities and procedures in diagrams than in text. A picture can be worth much more than a thousand words. Computerized diagrams do not allow the sloppiness and woolly thinking common in textual specifications.

Engineers of all types use formal diagrams that are precise in meaning—mechanical drawings, architects' drawings, circuit diagrams, microelectronics designs, and so on. Software engineering and information engineering also need formal diagrams with standardized diagramming constructs.

As in other branches of engineering, the diagrams used by integrated CASE products become the documentation for systems (along with the additional information collected in a centralized encyclopedia when the diagrams are drawn). When changes are made to systems, the diagrams are changed on the screen, and the code is regenerated. The design documentation is generated automatically and thus does not slip out of date as changes are made.

Philosophers have often said that what we are capable of thinking depends on the language we use for thinking. The diagrams we draw of complex processes are a form of language. With computers, we may want to create processes more complex than those we would perform manually. Appropriate diagrams help us to visualize and invent those processes.

For someone developing a system design or program, the diagrams used are aids to clear ideas. A poor choice of diagramming technique can inhibit thinking. A good choice can speed work and improve the quality of the results.

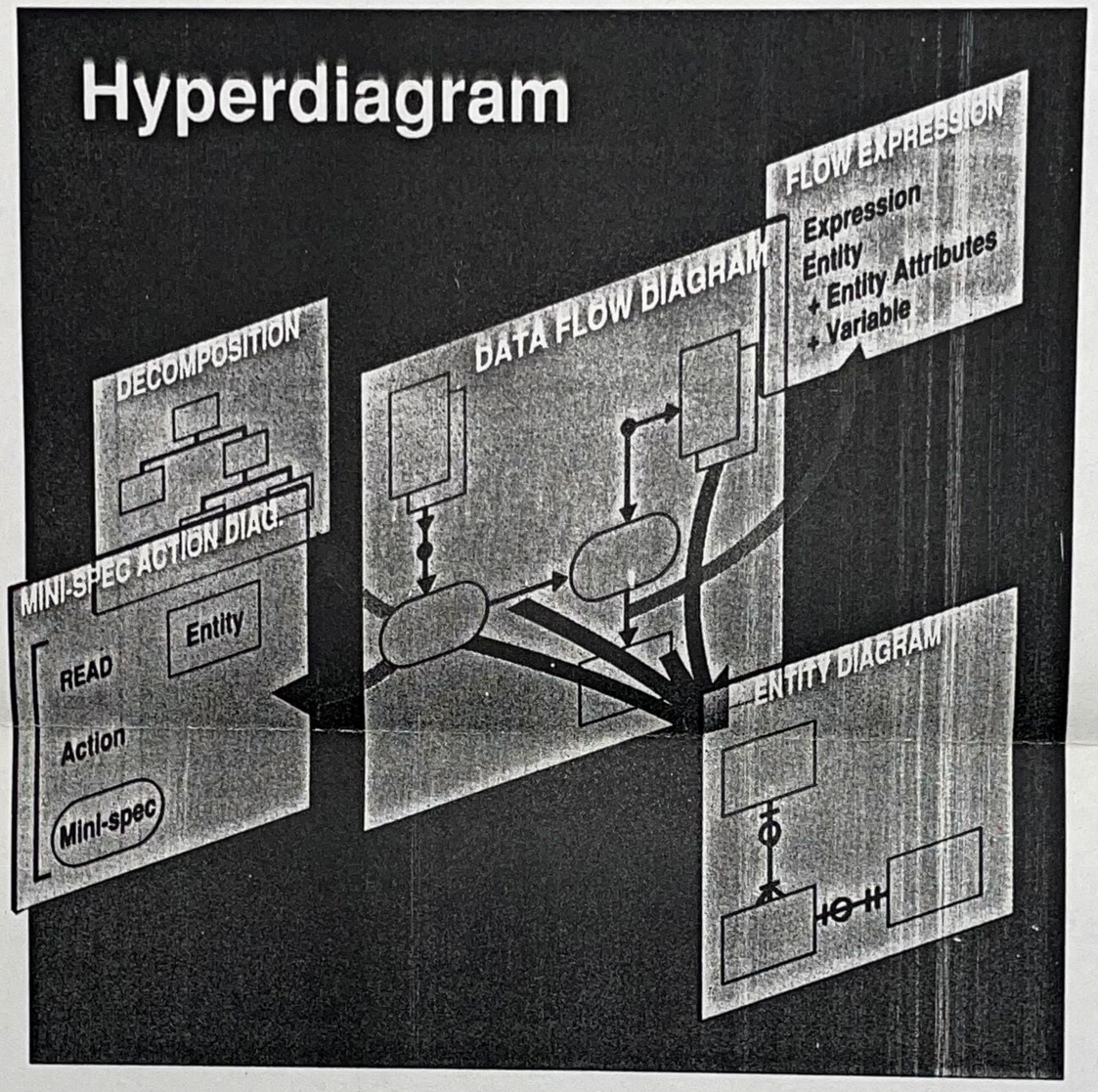
When several people work on a system or program, the diagrams serve as an essential communication tool. A formal diagramming technique is needed to enable the developers to interchange ideas and make their separate components fit together with precision.

When systems are modified, clear diagrams are an essential aid to maintenance. They make it possible for a new team to understand how the program works and then to design changes. When a change is made, it often affects other parts of the program.

Clear diagrams of the program structure enable maintenance programmers to understand the consequences of the changes they make. When debugging, clear diagrams are also highly valuable tools for understanding how the programs ought to work and for tracking down what might be wrong.

Diagramming, then, is a language, essential both for clear thinking and for human communication. An enterprise needs standards for its information-systems diagrams, just as it has standards for engineering drawings. A diagram and its associated information in a CASE tool can be very different from a diagram on paper. Paper constrains the diagram to two dimensions.

With a computer, many different representations of the design can be linked together logically. For example, the same block may appear on both a data-flow diagram and a decomposition diagram. Data access on an action diagram must relate to information specified on an entity-relationship diagram or in a data model. The inputs and outputs to a procedure represented by an action must be the same as those on the corresponding data-flow diagram.



David Hannum

Diagrams are aids to clear ideas. A poor choice can inhibit thinking. A good choice can speed work and improve the quality of the results.

Linking Diagrams Together

The terms "hyperdiagram" or "hyperchart" describe a representation of plans, models or designs in which many two-dimensional representations are logically linked together. A simple hyperdiagram is a diagram in which the details of objects may be displayed in windows. A more complex hyperdiagram uses many types of two-dimensional diagrams. A block or a line may be displayed in a window as text, as a fill-in-the-blanks form, an action diagram, a matrix or a different type of diagram.

The figure shows a family of screen windows that are part of one hyperdiagram. The hyperdiagram can be explored by pointing to objects or associations and displaying details of them. An I-CASE tool kit gives the implementor the facilities to explore or to build the hyperdiagram. The tool should enforce consistency within the hyperdiagram.

The set of screens in the figure illustrates how diagrams are linked to form hyperdiagrams. Each screen is part of a logically consistent structure. Because the hyperdiagram contains links between different types of representations and enforces consistency among these representations, it's a major advance over paper-oriented methods of analysis and design.

Next week, I'll discuss the very heart of an I-CASE tool: an encyclopedia or repository. ■

The James Martin Productivity Series, an information service updated quarterly, is available through High Productivity Software Inc., of Marblehead, Mass. (800) 242-1240. For information on seminars, contact 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.