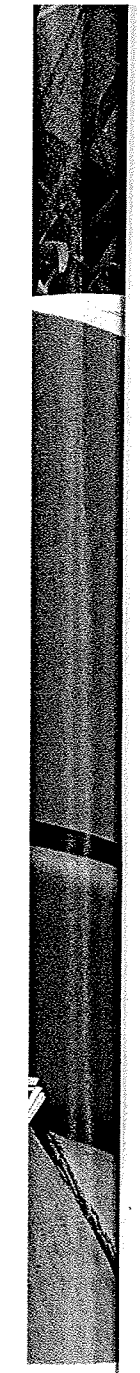

If you're in the market for a 4GL, you'd better be in the mood to answer 10 basic questions.

SHOPPING FOR A FOURTH GENERATION LANGUAGE

by George Schussel

Fourth generation languages are like a good set of tools—when you need them they come in handy. Today's 4GL software provides significantly different solutions for improving productivity compared with third generation COBOL and FORTRAN. The semantic and syntactical differences among 4GLs are also far greater than the differences between FORTRAN and COBOL. And although 4GLs are similar in some ways—most interface to one or more database management systems, provide query facilities and report writers, and support screen painting and some kind of procedural language—proprietary versions vary greatly in the types of applications and environments they are best suited for.

Some 4GLs, such as Natural from Software AG (Reston, Va.), Mantis from Cincom (Cincinnati), and ADS/O from Cullinet (Westwood, Mass.), are designed for the professional programmer who builds complex systems. Others, such as Focus from Information Builders (New York), Ramis II from Martin Marietta Data Systems (Greenbelt, Md.), and Nomad2 from D&B Computing Services (Wilton, Conn.), are geared more to the information center user. Still other 4GL products, such as Pacbase from CGI Systems (Pearl River, N.Y.), Telon from Pansophic Systems (Oak Brook, Ill.), and APS from Sage Systems (Rockville, Md.), generate COBOL and database calls. While these COBOL systems generators are also aimed at the professional programmer, they are usually complete life-cycle management tools with modules for project planning and control, librarian functions, and documentation.

There are 10 basic questions buyers should examine in evaluating whether a particular 4GL product is meant for their dp environments and applications. The first question is, *do you like the syntax?* Actually, a broad range of syntax is available for the different 4GLs. Some products like Focus, Natural, and Mantis, which are very language-oriented, have instructions that are written in code or English language-like statements. Support for both procedural and nonprocedural constructs also exists. People who have experience in programming will be very comfortable with language-based formats.

In contrast, other systems do much of their work through function generation. Ideal from Applied Data Research (Princeton, N.J.), Application Factory from Cortex Corp. (Wellesley, Mass.), Ally from Foundation Systems (American Fork, Utah), and ADS/O tend to fall into this category. Much of their functionality comes through interactive sessions and dialogs that are menu driven. These sessions are normally used to specify report formats, screens, and data definitions. In addition, menu driven front ends have recently been added to some language-oriented products, such as Focus and Ramis II.

Potential 4GL purchasers should then ask themselves the second question, *how procedural is the language?* If the language is procedural, as is the case with assembler or third generation languages like COBOL and FORTRAN, then the user, through constructs such as IF THEN ELSE, GO TO, and DO, tells the computer exactly how to take each step. If, on the other hand, the language is nonprocedural, then the user tells the computer *what* to do, not *how* to do it. Typical programming in a nonprocedural language supports various constructs such as the following: SELECT DATA FROM FILE, PRINT REPORT, and ORDER BY EMPLOYEE NUMBER.

While procedural languages are needed to build sophisticated, corporate dp systems, they can interfere with productivity in decision support and information center applications. Languages like Focus are therefore more effective in nonprocedural applications, while languages like Ideal and Natural, which have procedural features, offer enough detail control to solve almost any problem that can be handled by COBOL.

RUN-TIME VERSION NEEDED?

Next, 4GL buyers should ask, *is a run-time version of the system required?* The best-known 4GLs are proprietary languages like Focus, Natural, Mantis, ADS/O, and Sperry's Mapper, which combine some interpretive and some compiled features. All of these languages require a run-time version of the vendor's software.

This is a fine solution if you are using a particular brand of hardware for both building and running your applications. But what if you want to build in one environment and migrate to another or even several other environments? In this case, you need to look at 4GL environments that do not require run-time modules. Since most COBOL systems generation technologies—such as Gamma by Tarkenton Software (Atlanta), Transform/IMS from Transform Logic Corp. (Scottsdale, Ariz.), and Pacbase—generate structured COBOL code and native DBMS calls, they do not require the vendor's system at run time.

Before deciding on any 4GL, you have to determine who the user is going to be. So, question number four is, *who is the targeted user?* Some 4GLs are designed for people with a programming background. These products generally have more procedurally oriented languages. They also usually have performance features like multithreading capability, which allows transaction processing applications to be built that support up to several hundred simultaneous terminal users.

Most IBM software systems do not support the active dictionary concept.

Other 4GLs offer a friendlier interface that is easier for nontechnical types to master. These high-productivity, less procedural products are frequently used in information retrieval applications.

After the user is identified, you should ask, *what kinds of programmer support tools are provided and are they highly interactive?* Some 4GL products allow interpretive execution and debugging upon completion of each line of code or at the completion of each step of function generation. Others require you to complete programming tasks and then go into separate modules for execution. When an error is found in the execution module, the task must be terminated while the development module is brought back up and changes made. While IBM's CSP offers debugging trace facilities, it also requires the developer to change modules.

The sixth question you should explore is, *how diverse are the applications that can be coded?* Some 4GLs such as Natural and Ideal can be used for a broad range of applications. Almost any application that can be written in COBOL can be written in these languages—small or large databases, small or large numbers of transactions, batch or on-line systems, and logic that runs the gamut from very procedural to not so procedural.

Other 4GLs, like ADS/O, can be used for a slightly narrower range of applications. The building of batch logic, for instance, requires a separate system. And still other easy-to-use 4GLs, such as Focus, Ramis II, and Nomad2, provide more suitable performance for building departmental, as opposed to corporate, computing systems.

The next question that crops up in evaluating 4GLs is, *what support is there for an active dictionary?* An active integrated data dictionary/directory significantly enhances control of a fourth generation software environment. The active dictionary is a software component that stores data definitions, edits, and serves as the integrating/control unit for the entire software environment. Generally, the software technologies aimed at the professional programmer level provide active dictionary products.

Cullinet, Cincom, Applied Data Research, and Software AG are some of the companies that strongly support the active dictionary concept. IBM software systems do not support this active dictionary feature. Neither DB2 nor CSP, for example, support an active dictionary. Most 4GLs that are tailored to the information center support a passive dictionary. These products require the programmer to take procedural steps to invoke the dictionary.

WHICH SUPPORT SYSTEM?

That brings us to question number eight—*is your 4GL a language or a life-cycle support system?* Of course, 4GLs provide languages and facilities to build applications. A number of 4GL wares have even been enhanced to support such tasks as system and database design, table management, librarian functions, project management, and technical and user documentation.

Generally, the more support the system provides, the more suitable it is for adoption as a complete technology by the dp department. If, however, your dp shop already has adequate systems development support and only needs to get certain types of programs operational more quickly, then a full documentation and life-cycle support environment may simply provide too much extra baggage and unnecessary overhead.

You should also ask the related question, *how much education and investment are needed?* It normally takes more education and investment to make a complete life-cycle support system productive than it does to get products with less functionality up to snuff. For example, Pacbase, which provides a great deal of support for development, requires a major initial commitment. Mantis, on the other hand, can be installed and used quickly.

Finally, 4GL system shoppers should ask, *what type of integration is supported?* Usually 4GL products are either tightly coupled to a single DBMS or loosely linked with several of these systems. Ideal and ADS/O are examples of 4GLs that are tightly coupled to one DBMS. Both languages are designed for operation with their own proprietary data dictionary and DBMS engine. Used in this format, these languages are perfectly suitable for various updating and retrieval processing applications. Loosely coupled languages like Focus and Nomad2, which offer interfaces to a number of DBMS packages from different vendors, sometimes only support retrieval for reporting and querying. ◉

George Schussel is president of Digital Consulting Associates Inc., an Andover, Mass., consulting firm that specializes in DBMS and 4GL systems.