A Perspective on the Application of Fourth Generation Languages in Computer Software Development

By Dr. George Schussel

In this perspective Dr. Schussel presents his views on current developments in 4GLs.

One of the complaints that I hear often about 4GLs is that you can't define them. There are so many different products which are called fourth generation languages, that we really don't know what they are.

I heard one speaker say that a 4GL is any product whose vendor thinks it improves productivity.

There is a certain amount of truth in that. I think the critics who claim that you can't define it are really talking to the proposition that there are many categories of product beneath the 4GL umbrella.

Before reviewing some of the characteristics of 4GLs, I should start by identifying their heritage.

The first generation of software goes back to the 1950's and the procedural machine-level languages. Next came the second generation which encompassed assemblers and which typically had low levels of detailed computer code.

The third generation was compiler-level languages which were introduced in the 1960's and which include COBOL, Fortran and later PL-1. They allowed the programmer to make statements which the compiler would then expand to the appropriate level of instructions to the machine. These compiler languages were subsequently augmented with data base management systems and other auxiliary tools.

This generation is now giving way to the fourth generation of software: collectively, fourth generation languages, relational data bases and other modern tools of software development.

Unlike their predecessors, fourth generation languages are non-procedural. With these languages the developer can tell the computer what is wanted without necessarily telling the computer how to do it. They also include relational data bases and often feature multiple data base management systems, each suited to a specific task.

Decision support systems are another class of software that is an important part of the fourth generation. These include spreadsheets and capabilities that allow end-users to interact directly with a computer and to carry on simple programming functions such as modeling and simulations.

Why do we need non-procedural application development languages? The simple reason is that there is not enough productivity using third generation tools.

COBOL, for example, emerged in 1959. It is well on its way to being 30 years old. Look at how far the rest of computer technology has come since then. Clearly it is time for a new software technology.

This is the key driving force behind the movement to fourth generation technology.

Some people might suggest that we do not need another new language. After all, they will say, we have developed structured methodologies and we now have techniques to make COBOL very efficient.

It is true that structured technologies have clearly offered improvements. Unfortunately, improvements have been modest because these techniques still require considerable human interaction with the computer. The software

Dr. George Schussel is one of the world's leading lecturers on data base management systems, 4GLs and fifth generation application development. He is president and founder of Digital Consulting, Inc., Andover, Massachusetts, a prominent high technology education and management consulting firm.

"With these languages the developer can tell the computer what is wanted without necessarily telling the computer how to do it."

developer is using multiple, inconsistent tools and is forced to use procedural, labor-intensive languages.

Attempts to make COBOL more efficient can produce results, but the base is still an old technology that is not integrated with other productivity tools.

Fourth Generation Tools

Fourth generation languages, on the other hand, are designed specifically to deliver improved productivity. They actually consist of a number of development tools that reduce the time required to write an effective software application. At the center of this development universe is an integrated data dictionary.

This dictionary is used to store and process data definitions, and data interrelationships and becomes an automatic reference guide throughout the development process.

Around this dictionary are capabilities designed for specific functions and to reduce the workload of the developer. A representative sample would include:

- Application program developers utilizing an interactive workbench or workstation environment.
- On-line transaction processing services with transaction processing monitor facilities.
- Report writers to enable the preparation of complex reports without relying on procedural languages.
- A query facility to provide ease-of-access to all information and enable end-users to extract information on their own.
- A data base management system or systems to actually manage data and to divorce data management from the necessary housekeeping that exists in all programs.
- A fourth generation language or a series of languages to provide high-level non-procedural capabilities in the application building process.

As can be expected, there are different categories of 4GLs.

"As can be expected, there are different categories of 4GLs."

Five Categories of Fourth Generation Languages

At present 4GLs can be identified into five different categories:

Query and reporting packages — These are typically used in the data processing department and are not suitable for updating files, but are very suitable for retrieving information from existing data bases. An example is CCA's Imagine.

Programming-Oriented Languages — These are designed for someone who knows how to program and they serve as a replacement for COBOL. They are designed for data processing professionals and are used for building transaction and logic-intensive applications. Examples are Cognos' PowerHouse or Cortex Corporation's Application Factory.

Information Center 4GLs — These are human-like languages and are designed for non-data processing people. They are very effective for building personal use systems and smaller applications, but they have serious efficiency problems if used for large transaction-based systems. An example is Information Builder's FOCUS.

COBOL System Generators — These generate COBOL but do not generate direct object code that can be executed on a computer. They have certain advantages because their COBOL code can be compiled and moved to any object computer. Examples are CGI's PACBASE or Pansophic's TELON.

Decision support systems — These range from simple spreadsheets to complex multi-dimensional array-based data base management systems which have simulation languages built into them and allow modeling and "what if" simulations. An example is Execucom's IFPS.

All five are fourth generation languages, but they are all different in the functions that they are used to perform.

Management Issues

Equally as important in the discussion of 4GLs is management of the technology. If an organization installs fourth generation software tools but still manages the use of computers as it did in the 60's and 70's then that organization is not taking true advantage of fourth generation technologies.

The growing involvement of end-users in the use of computers has dramatically changed the way computer resources must be managed. This has led to the development of information centers located in the work area of non-data processing computer users, such as accountants, marketing staff and others. They interact directly with their organization's computer system to produce their own reports, to conduct their own modeling and so forth.

End-user involvement is an important ingredient of fourth generation computing and it happens on two levels.

First, prototyping has become a new management technique. With prototyping application development is done in a non-traditional way that involves considerable interaction with the end-user. In the development process an initial prototype is prepared for user review. The user then provides a critique and both developer and user work in an iterative loop to build the application.

Second, aspects of fourth generation languages allow the end-user to develop programs or to retrieve information from the computer system without the aid of a programmer.

Changing Role of the DP Department

The capabilities of the fourth generation are causing a change in the role of the data processing department. At one time the department held total responsibility for computing. That responsibility is now shared. However, the data processing department has to be the force driving the whole organization into strategic uses of data processing and the proper use of these new software tools and technologies.

There is quite a variety of products now available. They are distinguishable by some common characteristics, particularly their syntax and the degree of computer expertise that is required to use them.

Syntax refers to the complexity of the language used to interact with the computer. The more procedural the language level, the more the 4GL will resemble traditional programming languages. The higher the function level, the more the user will be able to interact with the system on a dialogue basis rather than through computer programming.

Some 4GLs are designed exclusively for use by data processing professionals. Others are appropriate for end-users who have limited technical knowledge.

All of the 4GL products currently on the market can be rated according to their syntax orientation and the requisite level of the user's computer skills.

For example, if we select one product, such as Application Factory, which is a good product for generating application systems, we would find that it has a very function-oriented syntax and that it is designed for use by data processing professionals.

Evaluation Considerations

Evaluating and selecting a 4GL necessarily starts with determining the required functions.

There is no one best fourth generation language. There are different kinds of products suited to different kinds of tasks.

In evaluating 4GLs it may be helpful to ask a number of questions:

- Am I buying a language?
- Am I buying something to write code in or is it something more than a language?
- Do I have an existing application or am I starting anew?
- Do I need a complete documentation system?
- Is my primary need performance or ease-of-use?

"There is no one best fourth generation language. There are different kinds of products suited to different kinds of tasks."

If you buy a 4GL that is just a language, you may be missing some functions. The advantage, however, is quick installation. Usually programmers can be productive with it in a matter of days, while many of the broader, more functional products require an indepth installation and training period. A particular point in the selection process concerns the type of applications already being used. The capabilities required for modification of existing applications may dictate a different 4GL than that needed for new system development.

Other considerations are the 4GLs' relationships with their data dictionaries and data base management systems. All 4GLs should have an integrated data dictionary function. With some products the function is tightly-coupled; with others it is loosely-coupled. In detailed program-oriented procedural applications, tightly-coupled, active data dictionaries tend to be more useful.

The degree of integration between the 4GL and its data base management system ranges from very tightly-coupled data base interfaces, which means that they connect only with certain systems, to other products which can be interfaced to a wide variety of systems, giving greater selection potential.

"...4GLs should have an integrated data dictionary function."

On the question of performance versus ease-of-use, there is usually a trade-off.

We came to this conclusion by looking at many different products that are in the marketplace. During our evaluation we found products that have high performance but are more complex and other products that offered greater ease-of-use but lower performance.

Trends Suggest New Capabilities

One thing is certain, the technology continues to change. Some of the developments we can expect in the next five years could include the following:

- Business graphics will become standard 4GL output.
- Decision support systems will be integrated with the traditional data base management systems.
- Expert systems will be applied to solve design problems.
- The microcomputer will become a dominant form of computing in business and government and we will have not only a micro-to-mainframe link but also have the ability to share the application load with part of the workload running on the mainframe and another on microcomputers.
- Artificial intelligence will become an important part of fourth generation software as it migrates to the next generation. Then we will be able to talk to the computer in a natural language rather than having to learn a detailed computer language.
- Functional software, such as accounting, human resource or manufacturing packages will be integrated along with data base management systems and fourth generation languages. In the future, broad integrated lines of software will be available.
- Distributed data bases will exist in which the software will automatically put the data where it is most efficient to the overall computer system.
- Continuous fault-tolerant processing will include both hardware and software.
- Syntax that is easier-to-use and easier-to-learn will also be available.