FOURTH AND FIFTH GENERATION PROGRAMMING LANGUAGES, VOL. I:

Integrated Software, Database Languages, and Expert Systems

by Dimitris N. Chorafas

FOURTH AND FIFTH GENERATION PROGRAMMING LANGUAGES, VOL. I:

Integrated Software, Database Languages, and Expert Systems

by Dimitris N. Chorafas

Library of Congress Cataloging-in-Publication Data
Chorafas, Dimitris N.
Fourth and fifth generation programming languages.

Includes index.
Contents: v. 1. Integrated software, database languages, and expert systems.
1.Programming languages (Electronic computers)
I. Title.
QA76.7.C485 1986 005.13'3 85-16666
ISBN 0-07-010864-1 (v. 1)

1234567890 DOC/DOC 898765

ISBN 0-07-010864-1

The editors for this book were Stephen G. Guty and Galen H. Fleck, the designer was M.R.P. Design, and the production supervisor was Thomas G. Kowalczyk. It was set in Century Schoolbook by McGraw-Hill Information Systems and Technology.

Printed and bound by R. R. Donnelley & Sons Company.

PREFACE

The choice of a programming language greatly impacts the way in which the application is made, the quality of the resulting software, the maintenance capability and its costs, and, quite importantly, system analyst and programmer productivity. The choice should result from a careful, well-documented study on the advanced software tools that current computer and communications environments can support. The aims should be:

1. To keep up with the best technology can offer
2. To benefit from the cost-effective tools currently available
3. To provide the company with a competitive edge over its competitors

Therefore, fourth-generation languages (4GLs) are a very worthwhile subject within the framework of the strategic objectives established by management. Their implementation is an urgent matter, and decisions are needed to orient the company on future commitments.

This book is primarily written for computer professionals who want to update their skills in terms of *new tools for application development.* In its 16 chapters they will find compelling arguments for developing and implementing computer software in novel, highly productive ways—and to fast timetables.

The book should also appeal to the more advanced computer users. They will find documented evidence that fourth-generation languages have moved computers and communications beyond the mystique stage to a comprehensive level of application. At the same time, fifth-

generation languages—more precisely, *expert systems*—have opened up opportunities unparalleled by any previous methods. Both the computer professionals and the end users find it easier to communicate with intelligent machines.

The cutting edge of technology suggests the wisdom of concentrating on fourth- and fifth-generation programming. This has its counterpart in hardware: Today nobody would project a computer with small- or medium-scale integration chips. All effort is being directed not only to large-scale integration but also increasingly to very large-scale integration.

Under no circumstances should the applications programming effort rest on 25-year-old so-called high-level languages. As a lot, they are obsolete, although some of them, notably Cobol, are more obsolete than others. Like obsolete chips, obsolete languages are noncompetitive and result in spoilage of resources. They have far-reaching detrimental effects because they adversely impact on human time, which computers are supposed to assist.

Fourth-generation languages can be classified in five major groups:

1. Programming extensions to the operating system through command interpreters
2. Database management and query
3. New programming languages such as graphics
4. Productivity-oriented tools, through precompilers
5. Spreadsheet systems and integrated software

Fourth-generation languages are not just end user facilities. They can serve both the end users and the computer specialists—each at a different level of sophistication. What is more, they are in full evolution. We can expect a significant 4GL announcement each 6 months or so.

As experience with natural-language programming and expert systems accumulates, the fourth generation will be followed by a fifth. The next generation of languages (5GL) can be expected by the end of this decade and will largely rest on breakthroughs in artificial intelligence.

* * *

As this book was originally designed, I had thought to include a comprehensive approach to Unix, the command interpreter shell, and the facilities supported by the programming language C, in which both Unix and the shell are written. Part of this thought was to cover the use

of Ingres as a programming language, including aggregation operations and application by forms.

In terms of sophistication and programming power, the command interpreter of an operating system is followed by the database management system language. The DBMS was originally conceived to manage data on run time—and it has been used that way for over twenty years. But now we have come to realize that, in a data-centered environment, it is also, if not primarily, a programming tool.

Finally I opted out of that approach and chose instead to write a separate volume on Unix, Ingres, and the IBM 4GL. This meant that more space, and greater attention, could be devoted to integrated software, a most fundamental subject for interactive workstations.

* * *

At the level of software development by professionals, 4GLs are found to be powerful system tools. Automation of analysis and programming functions, improved development time, lower cost, and higher software quality result in an innovative system product. That is explained in Chapter 1, which treats information systems as a top knowledge industry. It presents the challenge of remaining a knowledge worker, suggests how to keep on being computer-literate, and offers new perspectives for computer professionals.

To focus attention on the developments which are under way, Chapter 2 presents an overview of *computer technology: today and tomorrow.* Then attention is focused on *efficient program development* and *portability*, the subject of Chapter 3.

Fourth-generation programming is a concept. There are key points to be brought into perspective and images to be created. There is training to be done. Chapter 4 explains why. The easiest 4GL for the nonexpert is the spreadsheet, and slightly more complex is integrated software. Chapter 5 makes this evident and gives examples with Lotus 1-2-3.

Implementation-wise, one of the fastest-growing concepts is commodity software. Chapter 6 examines this issue. It is followed by the fundamental notions concerning interactive graphics (Chapter 7), the manager's own shorthand, decision means, and tool for corrective action.

PC-level operating systems have their own DBMSs for programming purposes. An example is dBase II. It is discussed, along with applications examples, in Chapter 8. Then, in two chapters, emphasis is placed on operating systems: from MS-DOS and CP/M to the competitive OS features embedded in Unix and Pick.

A *natural-level language* is best employed by end users, but, as Chapter 11 explains, we can find much in them in terms of implementation capabilities. They act as the front end and include a wealth of tools for software developers.

Expert systems, *knowledgebanks*, and the way to use the tip of the 5GL iceberg are the issues to which Chapters 12 and 13 are addressed. They open with an overview of artificial intelligence efforts and close with examples of interaction with expert systems. Chapter 14 expands on this notion and gives practical examples of expert system implementation.

Chapter 15 underscores an issue of growing importance: *software publishing* and the associated quality assurance requirements. The IEEE recommended standards for software developments are among the highlights.

Finally, the last chapter is devoted to case studies. The time has come to draw a very sharp line between the old images which dominated "EDP" for over three decades and the new concepts necessary to design, implement, and exploit information systems in the business world of tomorrow.

Let me close by expressing my thanks to everyone who contributed to making this book successful: from my colleagues for their collaboration, to Steve Ross for his advice, to the organizations I visited in my research for their insight, and to Eva Maria Binder for the drawings, typing, and index.

Valmer and Vitznau **DIMITRIS N. CHORAFAS**

CONTENTS

1

INFORMATION SYSTEMS AS A KNOWLEDGE INDUSTRY

Knowhow, Webster's *Ninth Collegiate* tells us, was first used as a term in 1838. That makes it about 150 years that our civilization has had the concept and has been coping with its implications. *Knowhow disappears when we don't use it—most particularly when we cling to one-dimensional perceptions* that arise from misunderstanding the competitive processes in industry. That's why we must not only acquaint ourselves with but *take the lead in applying the latest developments in our fields*, which now are the advances in computers and communications.

That is totally different from the philosophy that electronic data processing (EDP) professionals, and the second-generation DP/MIS people have followed for 30 or so years:

- You want a report? Submit a request form and we will think about it. Sometime next year, at the earliest.
- You need changes in the tabular presentation of hardcopy output? Forget it.
- You complain about mainframe's tons of paper snowing you under? Our laser printers are very efficient.

Such reactions, to which end users have been steadily subjected, have two origins. The first is that professional data processing people have been slow to adopt the new computer and communications technologies.

In the early to mid-1950s, corporations began to establish data processing departments based on the computer as the means of solving

their various data management problems. The implementations, however, created incredible bureaucracies, and all the decisions about who gets what information and when were made by staff hierarchies.

I recall that, in the mid-1950s, Robert Oppenheimer observed that the computer is an engine totally different from an electronic accounting machine—and should be used in novel ways to get maximal worth out of it. It took the user companies, and their computer professionals, three decades to understand what Oppenheimer meant. This was a conceptual failure.

The second origin of EDP reactions was inertia. As their numbers steadily grew (and the software/hardware resources they managed sprawled), computer professionals became burdened with acquired habits. Their judgment has often been warped by their wanting to preserve compatibility with software and hardware of the past.

As a result, data processing (DP) and management information systems (MIS) did not deliver on their promises. Instead of shouting hosannas for increased efficiency, end users muttered nasty remarks about people in computer glasshouses. Large stacks of ply paper gathered dust on desks. Distrust of data processing and its associated technology grew—to everyone's detriment.

Now is the time to change all that. Fourth-generation languages (4GLs) present an opportunity not to be ignored. The 4GLs will be followed by the fifth generation of intelligent systems. This should deeply interest all computer professionals, to whom this book is addressed.

THE CHALLENGE OF REMAINING A KNOWLEDGE WORKER

The two most evident impacts upon society in this century have been made by the *automobile* and the *computer*. Today over 90 percent of the population between the ages of 21 and 65 are auto-literate, but less than 5 percent of that same group are computer-literate. Cars and status no longer correlate; but cars and lifestyles are closely linked. The same thing will be true of computers, with impacts on both information systems professionals and end users.

As managers and other end users grow more knowledgeable about computers and communications, they resent receiving information regulated by someone else's perceptions of their needs. They like to make decisions about the type of information they want and the rate, content, format, and delivery of information into their hands. When they can't, they lose interest.

Today users are better able to identify their real problems because the microcomputer and its easy-to-use languages have arrived in their offices. Originally, the personal computer (PC) made the same promises that MIS made, but it embodied a very important difference: *the user's intimate involvement.* Early microcomputers needed no centralized DP departments, no intermediaries, no bureaucratic structures. The solution was user control of the software, the hardware, and the operation.

Computer-literate people do not have to understand microprocessor design, but they should have an appreciation of basic concepts and tools:

1. Managing their data
2. Systems versus applications software
3. Communications protocols
4. Input/output media
5. Menus, prompts, and help features

In a symposium at which I was a participant in January 1985 in Geneva, Bankers Trust, a leading money-center bank, identified as follows the goals of its office automation, information systems, decision support projects: "*Assure that managers and professionals can spend at least 68 percent of their time on direct, computer-assisted work.*" (Figure 1.1)

That is a first-class goal for any of us, but we must plan for it and document it in a valid, factual way. No longer should managers and other professional workers be merely receivers of information products. They should be creators, manipulators, and owners. Whether communicating with databases, mainframes, or workstations, whether processing words, calculating numbers or formulas, or retrieving and handling documents, systems must bring information to management to a degree of efficiency greater than ever before.

At the same time, just as microcomputer users experience a sense of personal accomplishment, so must computer professionals. Here fourth-generation languages can play a leading role. The reason is that the implementation of modern technology is marked by three interrelated trends:

1. Increasingly powerful tools at our disposal
2. Greater time and space separation between planning and execution
3. A steadily faster pace of development

The computer profession has greatly changed. In the mid-1980s there was little resemblance to what had been true 10 or even 5 years before.

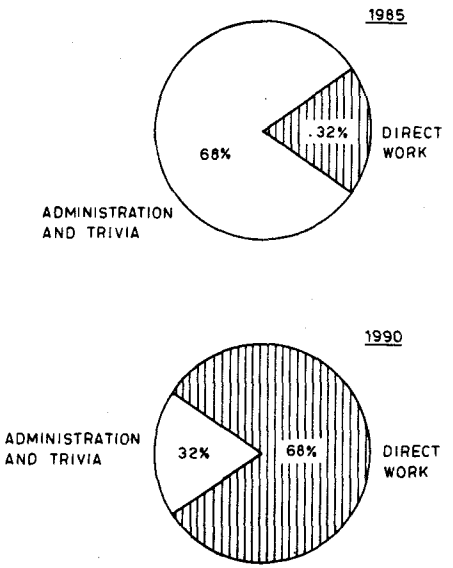

Figure 1.1 Professional time distribution goals of a bank's office automation, information systems, and decision support projects. Note the complete reversal.

This has not yet been widely recognized, but consciousness will increase as time goes by. As it is, when we think of computers, we often tend to conjure up the "hardware" and "software" of the past, forgetting about the complex technology that supports their use and interdependence. Now we should direct such usage to more profitable ends.

The new hardware emphasis will be on microelectronics and multiprocessor architectures. From this, developers hope to obtain at least a thousandfold increase in net computing power.

The orientation also will change. *The new software (and eventually hardware) components will focus on artificial intelligence (AI), particularly in expert systems (Esystems).* They will provide machines with quasi-human, intelligent capabilities. That includes natural-language understanding, vision, speech, and various kinds of automated reasoning.

On top of this technology base, new end-user-oriented applications are being developed. Their success will be in very different guise than originally envisioned: the power of computers and communications as a technological solution to a problem turns out to be only part of the answer.

The real answer is wide diffusion of knowhow. End users no longer accept the proposition that only a few insiders who deal with the

day-to-day operation of the machines can be intimately involved with the information process. This has been the dark side of technological solutions. *Now end users want participation.*

This participation is precisely what computer professionals can offer through fourth-generation languages (4GLs). As we will see in the following chapters, 4GLs offer the computer professionals the possibility of having, at the same time, a tool, a mission, and a message. To appreciate that, let's look at the environment.

The typical knowledge worker interacts with diverse systems and workstations, each having its own set of languages, procedures, and databases. The challenge is to:

1. Make office workstations an aggregate of functions while remaining flexible and expandable.
2. Identify the workstation concepts and applications that will be carried over into the next generation of workstations accepted in the office
3. Provide a radically new kind of flexibility and adaptiveness

Requirements analysis and software processes involve design problems and building methods. Computer professionals need a way to assist in the transition between current tools and the tools of the future. At the same time, one of their top challenges is adherence to strict development timetables. The new programming tools at our disposal can be a great help in both directions.

Good answers will be found not in generalizations but in specific choices for well-defined problems. If the first sound advice is *identify your problem*, the second is *choose the appropriate analysis and programming language.*

Figure 1.2 demonstrates a dual movement. One is away from "all professions' language" and toward specialization by type of problem. The other makes itself felt in greater sophistication in programming tools. Both trends are here to stay.

As Niclaus Wirth remarked in his Turing Award lecture, the difficulty of resolving many demands with a single language had emerged, and the goal itself had become questionable. Also, the size of the compiler had grown beyond the limits within which one could rest comfortably with the feeling of having a grasp, a mental understanding, of the whole program. Systems programming required an efficient compiler generating efficient code that operated without a fixed, hidden, and large so-called run-time package.

Fourth-generation languages are the means of transition in both the above-mentioned directions. In the majority, they are defined with clarity and their syntax is specified in a rigorous formalism. Such clear

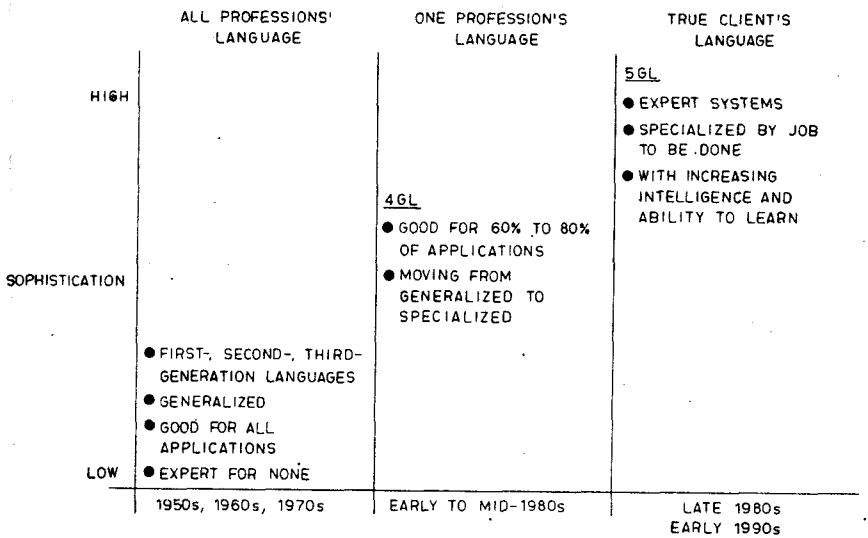

Figure 1.2 Movement away from all-professions' languages toward language specialization by type of problem, greater sophistication, and improved productivity in programming tools.

specification is a necessary (though not sufficient) condition for a reliable and effective implementation.

Keeping up with the latest developments characterizing the professional aspects of computers and communications is part and parcel of the challenge of remaining a knowledge worker. *The wave of change to which computer specialists must now adapt should be seen not as a problem, but as an opportunity.*

NEW PERSPECTIVES FOR COMPUTER PROFESSIONALS

When we speak of information systems as a knowledge industry, one of our key obligations is to education. There is a natural and growing interdependence between lifelong learning and professional results. Even the foundation on which the training of computer and communications scientists rests must be restructured. Back in the 1960s, thousands of computer specialists were working in industry, but they had been educated as mathematicians, physicists, and engineers. They learned computer science on their jobs—in effect, by apprenticeship.

Only in 1964 was the first advanced degree in computer science awarded. The roots should be looked for nearly two decades ago—a fact not appreciated because of lack of knowhow. In the late 1960s and early