A FOURTH GENERATION APPROACH TO THE INTRODUCTORY PROGRAMMING COURSE

by Dr. Fred K. Augustine, Jr.
and Dr. Theodore J. Surynt
Department of Accounting and
Information Systems
School of Business Administration
Stetson University
DeLand, FL 32720-3774

ABSTRACT: The rapid rate of change in the environment faced by CIS educators presents a challenge to maintain the currency and relevance of instructional methods and the CIS curriculum. Currently the way that computing power is utilized in organizations is undergoing a significant shift as a result of the end-user revolution and the advent of fourth generation languages (4GL). The focus of this analysis is on the use of fourth generation tools in the context of a CIS curriculum. Specifically, the application of a fourth generation language to the introductory programming course is examined. Fourth generation languages offer significant advantages over their third generation counterparts currently being used at the introductory programming level. These advantages include, but are not limited to, enhanced utilization of scarce instructional time and resources.

KEYWORDS: Fourth Generation Languages, CIS Curriculum, Programming Languages, End-User Computing.

In recent years, CIS educators have been confronted with an ever increasing array of tools available to them for use in a CIS curriculum. Indeed the curriculum itself has undergone a significant amount of change in terms of content and the breadth of topic areas included.[1,2] One constant element in CIS curriculums is the teaching of programming languages. The rationale for teaching programming as part of the curriculum goes beyond the requirement to teach CIS students to be programmers. Reasons often cited include the need to make students aware of programmer needs, time requirements, and fundamental logic used in developing systems.[3] This rationale remains valid but the emphasis is shifting with the increasing role of end-user computing in organizations.[4] Thus, the selection of a language to use in the introductory programming course is a critical one for any institution seeking to develop or implement a CIS curriculum.

SELECTING AN INTRODUCTORY PROGRAMMING LANGUAGE

The language selected for use in the introductory programming course must be one which satisfies several criteria. First, the language must be one which incorporates elements appropriate for the use of the structured programming approach. Next, it should be a language which is relatively easy to learn and will allow students to work on meaningful programming assignments. Finally it is vitally important to select a language which will be useful to graduates of a CIS program.[5]

THIRD GENERATION ALTERNATIVES

Currently, there are several schools of thought as to what constitutes an appropriate language for use in the introductory programming course. The DPMA '86 curriculum specifies the use of COBOL in the introductory and intermediate applications programming courses. The selection of COBOL is primarily based on the fact that it is still the most widely used language for business applications, particularly on larger systems.[6,7,5] COBOL, however, presents a significant set of problems to the beginning programmer.

First of all, the various physical structures within the language itself (Divisions, Sections, Paragraphs, etc.), while valuable in a production programming environment, are unnecessarily cumbersome and confusing to the new programmer. Another characteristic which makes COBOL the language of choice in the real world is its self-documenting feature. To the new programmer whose programs are neither long nor complex however, this "wordiness" of the language is a disadvantage. The fact that the vast majority of COBOL environments involve compilers rather than interpreters is also a disadvantage. The easy diagnostics and immediate feedback of an interpreter is far more desirable in a first language programming course. Finally, and perhaps most importantly, the interactive capabilities of the unenhanced COBOL language are not acceptable if student programs are to be at all meaningful. The ACCEPT and DISPLAY statements are simply no substitute for the advanced screen handling capabilities of a fourth generation environment.

...the need to tell the system not only what the programmer wants but also how to get it, is at the core of what is taught in the introductory programming course.

BASIC is the language of choice for small systems (primarily microcomputers) and is widely used in colleges and schools of business.[8,6] BASIC offers the advantages of widespread availability, especially on microcomputers, and was specifically designed to be a beginner's programming language. The major disadvantage of using BASIC to teach programming is the fact that it is a less structured language (particularly in its earlier versions) than other popular procedural programming languages such as COBOL or PASCAL.

PASCAL is a popular language used widely in computer science and other non-business programs.[9] PASCAL is a highly structured language which lends itself to teaching structured programming methods. It is, however, a more demanding language for the beginning programmer in terms of its syntax and requirements for structure.

A common characteristic of COBOL, BASIC, and PASCAL is the fact that they are all third generation, procedure oriented languages. Procedure oriented languages made programming possible for people who did not understand the intricacies of the computer but who were able to comprehend the logical requirements of the problems to be solved.[10] This type of language is oriented toward the procedures or algorithms that the computer is to follow and therefore requires the programmer to provide instructions to the computer in the correct form and sequence to accomplish desired tasks.[11] This requirement of third generation languages, i.e. the need to tell the system not only what the programmer wants but also how to get it, is at the core of what is taught in the introductory programming course.

FOURTH GENERATION LANGUAGES

The term fourth generation language (4GL) is by no means a new one [12], yet it continues to convey a variety of meanings. 4GL's have been described variously as non-procedural, end-user, and very-high-level languages. A non-procedural language is a problem oriented language in which the user specifies the results desired and the language produces the appropriate procedures or algorithms. 4GL's have been variously described as including numerous tools, for example: report writers, form generators, relational database management systems, and applications generators.[13,14] While each of these tools are generally described as being non-procedural, most of them have procedural components which can be used to enhance the capabilities of the non-procedural elements.

Hansen [10] proposes a set of principles which provides a broad working definition of a 4GL. This definition holds that all 4GL's include:

(1) Database structures and programming
(2) A centralized data dictionary
(3) A graded-skill interface
(4) Visual Programming
(5) An interactive, integrated, multi-function programming environment.

Given this working definition, it is clear that a 4GL goes beyond the traditional definition of a programming language. Indeed a 4GL may include, within its programming environment, a command or statement oriented language. These command oriented languages allow users to solve problems using multi-statement procedural methods (similar to a third generation language) or with single statement or visual oriented, non-procedural methods. It is this capability that is critical to the suitability of 4th Generation Tools for use in teaching programming.

It should be understood that any recommendation regarding the use of 4GL's in a programming class is relevant only to the introductory course. There is no doubt that students anticipating a career in programming would require an in-depth knowledge of production programming concepts. These concepts are commonly associated with third generation languages such as COBOL and would usually be taught in a separate intermediate or advanced level programming course.

THE CASE FOR A FOURTH GENERATION APPROACH

The growth of the use of 4GL's can be attributed to the significant advantages offered in the area of applications development. Among the benefits offered through the use of 4GL's are: more effective utilization of programmer resources; reduction of applications backlog; facilitates the implementation of end-user computing; and makes prototyping a feasible system analysis technique.[4,10,14,15] There are however, disadvantages to the use of 4GL's. Some of the more frequently cited are; applications produced using 4GL's tend to be less efficient in terms of computing resources; they are deficient in terms of documentation; and more planning and control is necessary due to increased end-user involvement in the development process.[10,11,15]

It is important to note that none of the disadvantages relate specifically to the use of 4th Generation Tools in the introductory programming course. Since, by definition, 4GL's are not completely non-procedural and incorporate "interactive, integrated, multi-functional programming environments," it would appear that their use in this context would be entirely appropriate.

AN EXAMPLE OF A 4TH GENERATION PROGRAMMING TOOL

All software tools which have been given the 4GL label are not appropriate for use in teaching programming. The fourth generation tool chosen should satisfy all of the criteria specified in the working definition of a 4GL. There are a number of commercial products available with significant educational discounts or student versions, which meet these criteria. Among these products are: dBase III and IV from Ashton-Tate; R:Base for DOS from Microrim, and Oracle from Oracle Corporation. For the purpose of discussion, and based on the authors' experience, a single fourth generation tool, dBase III Plus, has been chosen and the advantages of its use in the introductory programming course are described.

DBase III Plus is the best selling and most widely used of the microcomputer based database management systems.[8,6] It incorporates all of the features required to be classified as a 4GL. It includes: program, report and screen generators; visual programming; a database definition and manipulation language; and a powerful, multifaceted environment for applications development (including a procedural programming language).[10] The dBase programming language is made up of non-procedural commands, procedural programming statements, and structures which may be utilized in either the interactive/non-procedural or program/procedural mode. Many of these statements/commands are similar to language statements found in BASIC or PASCAL.

As programming languages go, the dBase language is relatively structured. The unconditional branch (GOTO statement) is not supported while structured programming elements such as DO WHILE and DO CASE are included. The dBase language also incorporates a large library of predefined functions. Overall, dBase provides most (if not all) of the elements and structures of most programming languages, at least at the level required for an introduction to programming. It has been noted that dBase lacks certain programming structures which are critical to the process of teaching a programming language, among these being the ability to use arrays.[16] Array processing however, is supported by dBase IV which is currently available in both commercial and student versions.

ADVANTAGES OF THE 4TH GENERATION APPROACH

Anyone who has been involved with an introductory computer programming course (either as a student or an instructor) has experienced the frustration that comes with expending large amounts of time and effort to produce programs which are relatively trivial. Given the objectives of the course and the tools available in a third generation environment this rather negative outcome was seen as being unavoidable. This is not true, however, in a fourth generation environment.

A fourth generation tool, such as dBase, incorporates command elements which reduce certain complex and repetitive programming tasks to single statements or commands. Many of these tasks are those associated with the organization and maintenance of data. Indeed, this is the main reason for the existence of database management software. As an example, consider the requirement to teach students to write programs which utilize direct/random access methods. The inclusion of random access processing in a computer program requires the programmer to possess both a conceptual knowledge of data organization and file maintenance methods and proficiency with a programming language which goes beyond the introductory level. Thus, students spend most of the introductory programming course working on programming assignments which utilize sequential access methods.

Using a 4GL such as dBase, students are able to incorporate direct access processing (such as the use and maintenance of index files) into their programs using a few simple statements. One statement (INDEX) builds an index file while another (SEEK or FIND) is used to perform a direct access search using the index. All maintenance functions for the index file are performed automatically by dBase as long as the index file is in use when changes are being made to the data file. Contrast with this, a sample program using BASIC [17] could use over 80 lines of executable code were necessary to accomplish the same tasks.

The above example points out the main advantage that a procedural programming language which is part of a 4GL has over a third generation language. The routine, repetitive, yet complex and difficult to program data maintenance tasks can be accomplished with very little programmer effort. This leaves the student free to concentrate on the logical requirements of the problem at hand and to work on meaningful and relatively sophisticated programming assignments early on in their education as programmers.

Another of the major benefits resulting from the use of a 4GL in the introductory course is that instructors can spend more time on such basics as programming logic, language syntax, and structured programming methods, without being forced to neglect other important topics. Topics such as data input, error control, and user interface design, which are often mentioned and quickly dismissed in introductory programming courses, can now be addressed in more detail both conceptually and by students applying these concepts in their programming assignments.

THE PLACE OF THE 4TH GENERATION APPROACH IN THE CIS CURRICULUM

An important question which needs to be addressed is: How does the fourth generation approach to introductory programming instruction fit with the rest of the CIS curriculum. The answer is that it provides an important transition between the introductory CIS course and subsequent courses in the curriculum. Consider the typical introductory CIS course. This course normally includes an introduction to computers with emphasis on computer requirements in organizations, history, hardware functions, programming, systems development, and computer operations. In addition, there is often a hands on or experiential component where students receive introductory level practice and instruction in one or more software environments.[2]

This course component was originally envisioned as being an introduction to a procedural programming language (most often BASIC). In recent years, the trend has been to substitute instruction on microcomputer based software packages with most programs opting for spreadsheets, database management systems, and occasionally word processing.[8,6,3,18] In a recent survey, Whitt indicates that nearly one half of the schools surveyed included experiential coverage of database topics (hands on experience with database management software) in the introductory CIS course.[19] This represents a significant increase (up from less than 23%) over a survey published three years earlier.[20] The trend in this data is clear.

More institutions are offering experiential instruction in database software in the introductory CIS course each year. It is therefore, increasingly likely, that students enrolled in an introductory programming course will have had some hands on experience with a 4th Generation tool (such as dBase) prior to entering that course. Thus, the use of a 4th generation tool to introduce students to programming becomes part of a natural progression which offers economies of time and resources to institutions employing this approach. Students will develop a more well defined and experientially based, conceptual foundation for subsequent CIS courses (databases, MIS, end-user computing).

THE 4TH GENERATION APPROACH AND END-USER COMPUTING

The proliferation of microcomputers in the work and educational environments has brought about a drastic shift in the way that computing power is utilized in organizations. Users have become more independent as they have gained increased access to and control over computing resources.[21] The term end-user computing has been used to describe this emerging environment. The importance of end-user computing is underscored by Day, et. al., who maintain that:

> this end-user style of computing represents the future of business computing... The dominance of end-user computing requires professionals who know how to support and train the users who will develop their own systems using databases in a networked environment.[22]

If graduates of CIS programs are to be prepared to successfully function in an environment dominated by end-user computing, it is essential that these students gain as much experience as possible using 4GL and DBMS products.[19] Currently CIS curricula are dominated by courses which are based on the more traditional centralized data processing model with coverage of end-user computing and 4GL's reserved primarily for elective courses. It is important that these tools receive greater exposure in the core courses of the CIS curriculum. The use of a 4GL in the introductory programming course represents a method of increasing this exposure while at the same time leaving the curriculum organization unchanged.

CONCLUSION

The environment faced by CIS educators is one which is characterized by a rapid rate of change. Those of us who are involved in CIS education and curriculum development must be in touch with this environment and be prepared to make the adjustments necessary to insure that our students graduate from programs which are relevant to the challenges they will face in the working world. As a result, instructors find themselves under increasing pressure to add new topics to the curriculum and to expand and refine the coverage of existing topics. The use of 4GL's to introduce students to computer programming is one mechanism to help the instructor meet these challenges.

The major advantage to be gained from this approach is the redistribution of time requirements in the introductory programming course. Using the fourth generation approach, more time is spent on the fundamentals of programming. Students are also able to work on meaningful programs more quickly, thus, decreasing frustration and increasing the level of interest. This approach also fits very well into the CIS curriculum in terms of the transition between the introductory CIS course and the more advanced courses. Finally, the fourth generation approach provides the student with experience based instruction on a fourth generation tool which will facilitate the integration of end-user computing into the CIS curriculum.

REFERENCES

1. Adams, D.R. and Athey, T.H., Editors, "DPMA Model

Curriculum for Undergraduate CIS Education," DPMA, September 1981.

2. CIS '86: The DPMA Model Curriculum for Undergraduate Computer Information Systems, Second Edition, DPMA, July, 1986.
3. Behling, R., "Course Content in the Introductory MIS Course: A Curriculum Survey," <u>Interface,</u> 11(1). 1989: 12-17.
4. Panko, Raymond R., <u>End-user Computing: Management, Applications and Technology.</u> John Wiley and Sons, 1988.
5. Hatch, R.A., Loster, A., and Marder, S. "Selection of Programming Languages for the Computer Information Systems Curriculum." <u>Interfaces</u>. 10(2). 1988:10-12.
6. Frand, J.L., McLean, E.R., and Britt, J.A. "Fourth Annual UCLA Survey of Business School Computer Usage." <u>Communications of the ACM</u>. 31(7). 1988:896-910.
7. Benham, H.C. "PC Based COBOL Instruction." <u>Proceedings of the Conference Sessions, Information Systems Educators Conference.</u> DPMA Education Foundation. 1988: 55-62.
8. Chacko, C. and Mercure, A., "Microcomputer Courses in U.S. and Canadian Business Schools: The State of the Art." Western Michigan University, February, 1989.
9. Athey, S. "A Comparison of Undergraduate Information Systems Programs and the DPMA Model." <u>Interface</u>. 10(4). 1989:68-73.
10. Hansen, G.W. <u>Database Processing with Fourth Generation Languages.</u> Southwestern Publishing Company. 1988.
11. Wojtkowski, W.G. and Wojtkowski, W. <u>Applications Software Programming with Fourth Generation Languages.</u> Boyd and Fraser Publishing. 1989.
12. Read, N.S. and Harmon, D.L. "Assuring MIS Success." <u>Datamation</u>. February, 1981: 109-120.
13. Snyder, C. "No Easy Answers: The 4GL/DBMS Dilemma." <u>Business Software Review</u>. 6(6). 1987:53-62.
14. Weitz, L. "Not Cannibalism, But Coexistence with CASE." <u>Software Magazine</u>. 8(12). 1988:43-59.
15. Necco, C.R., and Tsai, N.W. "Use of Fourth Generation Languages: Application Development and Documentation Problems." <u>Journal of Systems Management</u>. 39(8). 1988: 26-33.
16. Te'eni, D. "Using Lotus 1-2-3 and dBase III to Teach Programming in an Introductory MIS Course." <u>Interface</u>. 11(1). 1989:41-43.
17. Spear, B. <u>Basic Programming: Fundamentals and Applications.</u> Merill Publishing Company. 1987.
18. Klein, Ronald D. "The Business School Introductory Computer Course: A Survey." <u>Interface</u>. 10(2). 1988:26- 28.
19. Whitt, Jerry D. "4GL and DBMS Coverage in the Introductory MIS Course: A Survey." <u>Interface</u>. 10(1). 1988:10-14.
20. McCleod, R. "The Undergraduate MIS Course in AACSB Schools." <u>Journal of Management Information Systems</u>. 2(2). 1985:73-85.
21. Omar, M.H. "Productivity Tools in the MIS Environment." <u>National Productivity Review</u>. 7(4). 1988:346-352.
22. Day, J.C., McClanahan, A.H., and Perotti, J.L. "Incorporating End-user Computing into an MIS Curriculum." <u>Interface</u>. 11(1). 1989:50-53.

AUTHORS' BIOGRAPHIES

*Fred K. Augustine, Jr., Ph.D. (Florida State University) is Assistant Professor of Information Systems at Stetson University. He has written articles published in the **Journal of Systems Management** and the **Journal of Computer Information Systems**. His primary research interests are in the areas of computer simulation of dynamic systems and issues involved in the development and implementation of information systems for small businesses.*

*Theodore J. Surynt, Ph.D. (Georgia State University) is Associate Professor of Information Systems at Stetson University. His background includes eight years of professional experience with IBM as systems engineer and business planning analyst. He has written articles published in the **Journal of Business Communication, The Journal of Computer Information Systems, The Journal of Data Education,** and **SAM Advanced Management Review**. His main research concerns are the continuing evolution of the CIS curriculum and the behavioral and performance implications of group productivity technology.*

STATEMENT OF PEER REVIEW INTEGRITY

ISSN 1055-3096