SOFTWARE ENGINEERING EDUCATION IN THE ASSOCIATE-DEGREE-LEVEL VOCATIONAL/TECHNICAL COMPUTER SCIENCE PROGRAM*

by Elmer Raydean Richmond
Computer Science Department
Tarrant County Junior College
Fort Worth, Texas 76119

ABSTRACT: The need for software engineers has grown with the increased use of software and computers in our society. Presently, formal education of software engineers is being conducted at the graduate degree level. There are pressures to extend this education to other academic levels; in fact, many baccalaureate degree programs today include one or more software engineering courses. Difficulties confronting software engineering education at the associatedegree level include: limited educational and experience backgrounds of the students; faculty whose own education did not include software engineering; and severe time and content constraints imposed on such two-year programs. The author concludes that the two-year associate-degree-level vocational/technical computer science program should continue to focus on producing graduates with sound programming and problem-solving skills. Once these graduates gain entry-level employment, they can then build upon these fundamental skills, with experience and further education, toward a software engineering career. Software engineering education in the two-year vocational/ technical program, therefore, should be restricted to that which directly enhances the development of these basic skills; specific recommendations are offered for consideration.

KEYWORDS Software Engineering, Education, Associate-Degree, Two-Year College, Vocational/ Technical, Computer Science

INTRODUCTION

The demand for software engineers has paralleled the increasing use of software and computers in our society. To meet this demand, both in industry and in the educational community, software engineering and software engineering education have grown markedly since the inception of the discipline in the late 1960s. As this discipline matures, educational programs are being developed and implemented. The role of the two-year vocational/ technical computer science program in this educational process also needs to be considered. This paper examines that role, identifies potential problems, and offers recommendations to bound the expectation of software engineering education from such programs.

BACKGROUND

Software Engineering

At the heart of any definition of software engineering (SE) is a recognition of software as a product to which the same care and discipline must be applied during its construction, operation, and modification as would be given a more

* *This article first appeared in the SIGCSE Bulletin, Volume 21, Number 4, December 1989.*

tangible physical product. Rodjak's (1) definition, then, summarizes and characterizes SE as:

> ***"an interdisciplinary specialty that uses technical, managerial, and communicative skills to produce high quality software products that are on schedule, within budget, and meet user requirements."***

Software Engineering Education

Fairley (2) outlines the ideal scenario for software engineering education (SEE) as one which would include: an undergraduate degree; one to two years of programmer-level experience; a masters-level software engineering degree; one to two years professional-level apprenticeship; and for some, a doctoral degree. In his article, Fairley classifies the programmer-level work experience as imperative. Without that experience, Fairley argues, the undergraduate has no basis for understanding the importance and relevance of the necessary knowledge and skills of the software engineer.

Freeman (3) recently reaffirmed the 'essential elements of SEE,' first identified in 1976, as a set of content areas that should underlie any curriculum, specifically, the areas of computer science, management science, communications, problem-solving, and design methodology. Freeman also notes that emphasis should be given to the integration of management and technical issues into SEE.

Today, SEE is being formally conducted through masters-level programs at a small number of institutions, including Texas Christian University in Fort Worth, Texas (4). Additionally, as reported by Mynatt and Leventhal (5), many baccalaureate-level computer science programs include one or more software engineering courses.

The United States Department of Defense established a Software Engineering Institute (SEI) at Carnegie Mellon University in Pittsburgh, Pennsylvania. The SEI (6) has been given a specific role in education, that of influencing SE curricula development across the educational spectrum. SEI educational activity to date has concentrated on a graduate curriculum project, because, as Gibbs reports in (6), "... the best education remains a solid major in computer science followed by graduate professional education...."

The SEI does recognize that industry and government pressures for undergraduate degrees will increase.

> ***At the heart of any definition of software engineering (SE) is a recognition of software as a product to which the same care and discipline must be applied during its construction, operation, and modification as would be given a more tangible physical product.***

Further, the large number of undergraduate students suggests the greatest impact SEI may have on SEE as a whole is through such programs.

An undergraduate SEE project has been established within the SEI; specific materials produced by that project include: recommendations for a senior-level one-semester project-based SE course; a 10,000-line Ada system for software maintenance projects; and a workshop and pilot studies of the use of Ada in freshman courses. Also of interest should be the results of a SEI-sponsored workshop in the summer of 1989 at which the undergraduate SE curriculum was the topic of discussion.

Finally, industry-based professional groups such as the Data Processing Management Association (DPMA) and the Association for Computing Machinery (ACM) are also involved in SEE; their principal contributions are curricular recommendations such as (7), (8), and (9).

TWO-YEAR ASSOCIATE OF APPLIED SCIENCE (AAS) DEGREE PROGRAMS

Program Contents

In their recommendations (9), the ACM identifies two critical areas to be addressed within the two-year program. One emphasizes in-depth programming knowledge, techniques, and skills supported by training in problem-solving and logical analysis; a second deals with the environment of the programmer during the analysis, design, implementation, and operation phases of an application project. The ACM recognizes that software engineering will impact the role of the programmer but does not specifically address that impact in their recommendations.

Two of the DPMA associate-level recommendations (8) are directed at qualifying graduates for entry level positions as programmers or in jobs requiring integrated use of microcomputers. A course to introduce prototyping and fourth generation tools is also included in these recommendations.

Both the ACM and DPMA curricula models include a capstone project-oriented systems development course. Similarly, both acknowledge the importance of courses which provide a basic understanding of business and industry; courses such as Introduction to Business, Accounting, Economics, Introduction to Engineering, etc., must be a part of the AAS program.

An Example Program

The Texas Higher Education Coordinating Board (10) describes AAS programs as generally technical or paraprofessional in nature, and designed to prepare the graduate for immediate employment and/or career advancement. In Texas, the Board stipulates the length of such programs be 60 to 72 semester credit hours and contain a general education core of 15 semester hours.

These general education requirements, in turn, are mandated by

the regional accrediting institution, the Southern Association of Colleges and Schools (11). As a specific example of the AAS degree, the Associate in Applied Science in Computer Science Degree, Business Applications Option, at Tarrant County Junior College (TCJC) in Fort Worth, Texas (12) is outlined in Table 1.

TABLE1: Associate in Applied Science in Computer Science
Business Applications Option
Turrant County Junior College in Fort Worth, Texas, 1988-89

GENERAL EDUCATION	HOURS
English Composition I	3
College Algebra for the Social & Management Sciences	3
United States Government	3
Business & Professional Communications	3
Interpersonal Communications	3
BUSINESS	
Principles of Accounting I	3
Principles of Accounting II	3
Approved Electives	9
COMPUTER SCIENCE	
Fundamentals of Programming	4
BASIC (or Pascal) Programming	4
Assembly Language Programming	4
COBOL Programming I	4
COBOL Programming II	4
Systems Analysis & Design	3
Systems Implementation	4
Operating Systems	4
Approved Electives	10
TOTAL HOURS	71

DIFFICULTIES FACING TWO-YEAR PROGRAMS AND SOFTWARE ENGINEERING EDUCATION

Program Requirements and Time Constraints

Referring to the TCJC computer science degree program option presented in Table 1, it is clear the addition of any SEE course would require deletion of some other (so not to violate the Texas Higher Education Coordinating Board guidelines). The decision as to which course should be eliminated would not be an easy one to make.

First, the 15-hour general education requirement derives from state and regional accrediting agencies. Next, the 15 hours devoted to business background courses do not appear to be excessive. Finally, the computer science content of the degree compares favorably to the guidelines available (e.g., the DPMA recommendations); the total number of credit hours, then, does not seem unreasonable.

Student Preparation

The problem of student preparation in terms of prior education and experience is not inconsequential. As noted earlier, programmer-level experience is of paramount importance to the development of a software engineer. Yet, most often, students attend the junior colleges to gain the requisite language skills so they may then enter, or move into, entry-level programming positions. Thus, the emphasis in AAS degree programs is on programming and problem-solving, still the skills which make such graduates employable.

Even at the four-year Management Information Systems (MIS) degree level, a recent survey by Seeborg and Ma (13) of MIS graduates reported those graduates ranked COBOL courses as most useful. Also, a concern reported by Mynatt and Leventhal in their survey of undergraduate software engineering courses derives from the difficult reading levels of computer science journals and software engineering textbooks; certainly in the AAS program this problem is equally severe.

A final concern in the area of student preparation, is the difficulty posed by the project class. The limited experiences and educational backgrounds of the two-year students are constraining factors. These factors, when combined with the need to expose these students to the software development process as a whole, suggest the project class in a two-year program is very difficult to carry out, and less likely to be able to deal with large-scale projects than those in baccalaureate-level programs. Thus, the opportunities for "programming-in-the-large" activities (large systems developed by groups) at the two-year level are severely limited, at best, and are not very likely to be successful.

Faculty and Other Resources

Mynatt and Leventhal's survey (5) identifies staffing as a problem related to the relative newness of the SE discipline. Gibbs (6) also agrees that most teaching faculty do not have a background in SE. At the junior and community college level, the competition for individuals with SE backgrounds and education is exacerbated by salary differentials and by the more modest levels of computer resources and support.

One advantage the two-year technical/vocational programs may enjoy, however, relates to acquiring software engineers for adjunct faculty positions. In the vocational/technical AAS program, emphasis is placed on experience, vis a vis academic credentials, as a principal qualification of the faculty.

For example, in the state of Texas (14), faculty qualifications for an AAS vocational/technical program may be summarized as follows: a combination of education and experience; technical competence in the teaching field; and at least an associate degree (with higher degrees preferred).

Nonetheless, staffing issues will limit the near-term ability of two-year vocational/ technical colleges to integrate SEE. Time and significant resources will be required to educate the present faculty of these institutions. In terms of hardware and software, resource acquisition is an ever-present challenge to the educational institution. The two-year program faces the same difficulties identified by Mynatt and Leventhal, primarily due to serious competition for limited funds.

GUIDELINES FOR TWO-YEAR VOCATIONAL/TECHNICAL AAS PROGRAMS

The "programming-in-the-small" (individual) knowledge and experiences provided by existing AAS programming language courses are a necessary foundation on which the software engineer builds. What the AAS program should provide its graduates then, first and foremost, is the same facility with programming which is of paramount and fundamental importance to the software engineer.

As noted by Carver (15), Werth (16), Calhoun (17), and Tam (18), there exists a set of software development skills which are important no matter the size of the software product. Emphasis on structured development, structured programming, software quality, and complete documentation should be consistently applied across the curriculum.

"Programming-in-the-large", the software life cycle, and alternative design methodologies should be addressed by the inclusion of a software engineering overview in the two-year program. The AAS program simply must allow room for this introduction. A familiarity with the entire software life cycle is a necessary component of the AAS graduate's education; without it, the graduate's view of his or her role in an entry-level position will be distorted.

The one-or two-semester capstone course(s) typically exposes the student to the traditional life cycle phases and activities of analysis, design, implementation, testing, and maintenance. It should remain the course in which the student is exposed to the software development process; therefore, it is appropriate to include a SE overview among the topics presented. Moreover, it is the appropriate forum in which to present, illustrate, and experiment with the alternate design methods.

Many articles have appeared to propose modification and integration of the new "CASE" tools into such courses (see (19)-(23) for specific examples); however, one must be aware of the danger presented there to make these tools the focus of the course rather than the underlying concepts themselves. Perrone's (24) definition of computer-aided software engineering (CASE) states, "CASE generally refers to the automation of any software engineering task." CASE tools at the two-year college level present both positive and negative opportunities.

It should be the principal goal of the AAS program to build sound programming skills. Therefore, those CASE tools available which enhance the programming phase (through integrated programming environments, language-sensitive editors, interactive debuggers, etc.) should be introduced. Tools such as Microsoft's QuickBASIC (25), Borland's TurboPascal (26) and TurboC (27) must become a part of the AAS graduate's repertoire; they are the productivity tools of today's programming environment.

It is also a goal of the AAS program to expose the student to software development and the "programming-in-the-large" nature of most real systems. Therefore, it is tempting to make tools such as Index Technology's EXCELERATOR (28), Ashton-Tate's dBASE III+ (29) and Bytel's GENIFER (30), Oracle's ORACLE (31), or Microrim's R:Base System V (32) the center of these capstone courses.

The pitfalls associated with using these tools are described in many papers, including (22) and (33). Basically, the "trap" results from the fact that these tools are so powerful and broad, they may consume enormous amounts of preparation and class time at the expense of the underlying concepts. The use of any of these tools demands much from the instructor and the students. Any tool use, then, in the systems class must be such that it enhances the presentation of underlying software development processes and methodologies.

SUMMARY AND RECOMMENDATIONS

Software engineering has been defined and software engineering education (SEE) activities reviewed. The two-year AAS degree has been characterized and examined. Difficulties presented by the desired integration of SEE into the two-year associate-level degree vocational/technical computer science degree program have been considered. Proposed guidelines for the extent to which SEE can be, and should be, integrated into the two-year program have been discussed; following are specific recommendations offered to implement those guidelines.

1. An overview of software engineering must be integrated into the AAS program, with the traditional systems course as the most logical place for inclusion. The 1987 SEI interim report (34) recommends a top-down approach to SEE, and includes an overview which could be adopted. The integration of a SE overview into a systems analysis and design course has also been addressed by Steward (35); his book provides other implementation ideas, as would the curriculum modules and detailed course descriptions available from the SEI (6), (34).

2. The use of tools in the capstone

course(s) of the two-year program must be carefully controlled. First, most AAS graduates will gain employment in entry-level positions which relate directly to programming skills, not systems analysis skills. Second, the emphasis must remain on presentation of the underlying concepts, the structure by which and within which software solutions to problems are developed.

3. "Programming-in-the-large" activities are generally beyond the scope of the two-year program. Nonetheless, communications activities similar to those which characterize SE can be, and should be, integrated throughout the curriculum by the requirement for oral and written reports, and by the use of activities such as code inspections and structured walkthroughs.

4. The two-year program must concentrate on development of programming and problem-solving skills. The existing language courses build the programming expertise base upon which SE rests, so they are critically important. Those CASE tools which specifically pertain to the programming activities must become a part of the AAS graduate's tool set. Moreover, as the software industry moves toward screen-oriented systems, object-oriented solutions and languages, and so forth, it is these capabilities which the AAS program must adopt and integrate so that their program products, graduates prepared for entry-level positions, remain viable.

CONCLUSION

The author concludes that the principal contribution of the community and junior college vocational/technical computer science degree programs remains the production of graduates with fundamentally sound programming and problem-solving skills with which to enter the workplace, and upon which software engineering as a discipline rests. Software engineering education content of the two-year program, then, must be restricted to that which directly supports the development and production of such graduates.

REFERENCES

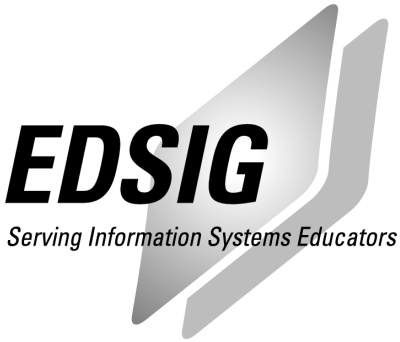
1. Rodjak, D. J., unpublished lecture notes for SODE 6113, Modern Software Requirements and Design Techniques. Fort Worth, TX: Texas Christian University, Spring Semester, 1986.
2. Fairley, R. E., "Software Engineering Education: An Idealized Scenario," in Software Engineering Education: The Educational Needs of the Software Community. New York: Springer-Verlag, 1987, pp 52-60.
3. Freeman, P., "Essential Elements of Software Engineering Education Revisited," in Software Engineering Education: The Educational Needs of the Software Community. New York: Springer-Verlag, 1987, pp 61-74.
4. Texas Christian University, Texas Christian University Bulletin: Graduate Studies 1987-88/1988-89, Vol. 81, No. 2. Fort Worth, TX: Texas Christian University, April 1987.
5. Mynatt, B. and Leventhal, L., "Profile of Undergraduate Software Engineering Courses: Results from a Survey," SIGCSE Bulletin, Vol. 19, No. 1, February 1987, pp 523-528.
6. Gibbs, N. E., "The SEI Education Program: The Challenge of Teaching Future Software Engineers," Communications of the ACM, Vol. 32, No. 5, May 1989, pp 594-605.
7. Data Processing Management Association, CIS'86 The DPMA Model Curriculum for Undergraduate Computer Information Systems. Park Ridge, IL: DPMA, 1986.
8. Data Processing Management Association, The Data Processing Management Association (DPMA) Academic Curriculum for Associate-Degree-Level Studies in Computer Information Systems. Park Ridge, IL: DPMA, 1985.
9. Little, J. C. et alia (Eds), "Recommendations and Guidelines for an Associate Level Degree Program in Computer Programming," in ACM Curricula Recommendations for Related Computer Science Programs in Vocational-Technical Schools, Community and Junior Colleges, and Health Computing, Volume III. New York: ACM, 1983, pp 207-229.
10. Texas Higher Education Coordinating Board, TECHNICAL AND VOCATIONAL PROGRAM GUIDELINES: A Manual of Guidelines and Procedures For State Funded Technical and Vocational Programs in Texas Public Post-secondary Institutions, Effective January 1, 1989. Austin, TX: Texas Higher Education Coordinating Board, 1988.
11. Southern Association of Colleges and Schools, CRITERIA FOR ACCREDITATION: Commission on Colleges, Approved by the College Delegate Assembly December, 1984-Atlanta, Georgia. Atlanta, GA: Southern Association of Colleges and Schools, 1987.
12. Tarrant County Junior College. Tarrant County Junior College 1988-1989 Catalog, Vol. XXI, No. 5. Fort Worth, TX: Tarrant County Junior College, May 1988.
13. Seeborg, I. S. and Ma, C. S., "MIS Program Meets Reality: A Survey of Alumni from an Undergraduate Program, Interface, Vol. 10, No. 4, Winter 1988/89, pp. 51-60.

14. Texas Higher Education Coordinating Board, QUALIFICATIONS OF TECHNICAL AND VOCATIONAL PERSONNEL: A Manual of Guidelines, Procedures, and Personnel Qualifications For State-Funded Technical and Vocational Programs in Texas Public Postsecondary Institutions, Effective January 1, 1989. Austin, TX: Texas Higher Education Coordinating Board, 1988.
15. Carver, D. L., "Recommendations for Software Engineering Education," SIGCSE Bulletin, Vol. 19, No. 1, February 1987, pp 228-232.
16. Werth, L. H., "Integrating Software Engineering into an Intermediate Programming Class," SIGCSE Bulletin, Vol. 20, No. 1, February 1988, pp 54-58.
17. Calhoun, J., "Distribution of Software Engineering Concepts Beyond the Software Engineering Course," SIGCSE Bulletin, Vol. 19, No. 1, February 1987, pp 233-237.
18. Tam, W. C., "A Multilevel Approach to Undergraduate Software Engineering Education," SIGCSE Bulletin, Vol. 17, No. 1, March 1985, pp 332-334.
19. Fritz, J. M., "A Pragmatic Approach to Systems Analysis and Design," SIGCSE Bulletin, Vol. 19, No. 1, February 1987, pp 127-131.
20. Poole, B. J. and Callihan, H. D., "Systems Analysis and Design: An Orphan Course About to Find a Home," SIGCSE Bulletin, Vol. 20, No. 2, June 1988, pp 54-57,64.
21. Chrisman, C. and Beccue, B., "Updating Systems Development Courses to Incorporate Fourth Generation Tools," SIGCSE Bulletin, Vol. 17, No. 1, March 1985, pp 109-113.
22. Amadio, W. J., "Systems Courses in the 1990s: The Promise of CASE and 4GLs," Interface, Vol. 10, No. 3, Fall 88, pp 14-18.
23. Bullard, C. L. et al., "Anatomy of a Software Engineering Project," SIGCSE Bulletin, Vol. 20, No. 1, February 1988, pp 129-133.
24. Perrone, G., "Low-cost CASE: tomorrow's promise emerging today," in New Product Reviews, Computer, Vol. 20, No. 11, November 1987, pp 104-110.
25. QuickBASIC, Microsoft Corporation, Redmond, Washington, 1987.
26. TurboPascal, Borland International, Scotts Valley, California, 1987.
27. TurboC, Borland International, Scotts Valley, California, 1988.
28. Excelerator, Index Technology Corporation, Cambridge, Massachusetts, 1987.
29. dBASE III+, Ashton-Tate, Torrance, California, 1986.
30. GENIFER, Bytel Corporation, Berkeley, California, 1986.
31. ORACLE, Oracle Corporation, Belmont, California, 1987.
32. R:Base System V, Microrim, Incorporated, Redmond, Washington, 1986.
33. Wilcox, R. E., "The Use of CASE Software in a Course in Systems Analysis and Design," Interface, Vol. 10, No. 3, Fall 88.
34. Ford, G., Gibbs, N., and Tomayko, J., Software Engineering Education: An Interim Report from the Software Engineering Institute. Technical Report CMU/SEI-87-TR-8. Pittsburgh, PA: Carnegie Mellon University, May 1987.
35. Steward, D. V., Software Engineering with Systems Analysis and Design. Monterey, CA: Brooks/ Cole, 1987.

AUTHOR'S BIOGRAPHY

Raydean Richmond is an Associate Professor of Computer Science at Tarrant County Junior College, South Campus, in Fort Worth, Texas. Raydean joined the College in 1982, and has also served as Department Chair since 1984. He holds BSIE and MSIE degrees from the University of Pittsburgh, Pittsburgh, Pennsylvania, and is a 1989 recipient of the Master of Software Design and Development degree awarded by Texas Christian University, Fort Worth, Texas.

Information Systems & Computing
Academic Professionals



STATEMENT OF PEER REVIEW INTEGRITY

All papers published in the Journal of Information Systems Education have undergone rigorous peer review. This includes an initial editor screening and double-blind refereeing by three or more expert referees.

ISSN 1055-3096