The Pennsylvania State University

The Graduate School

Department of Computer Science

A Review and Analysis of "Curriculum '68"

A Thesis in

Computer Science

by

Gerald Lawrence Engel

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Education

August 1974

Date of Signature:

July 10, 1974

July 10, 1974

Signatories:

Bruce H. Barnes

Bruce H. Barnes, Associate
Professor of Computer Science
Chairman of Committee
Thesis Advisor

Rosemary Schraer

Rosemary Schraer, Acting Head,
Department of Computer Science

Table of Contents

List of Tables

List of Charts

Introduction

"'Curriculum '68', Recommendations for Academic Programs in Computer Science, A Report of the ACM Curriculum Committee on Computer Science" [58] has served a critical role in the development of computer science education, establishing the field as an academic discipline and providing a framework for the description of courses and programs.

Current textbooks in computer science contain statements such as: "Specifically the 'Curriculum '68' report of the ACM Committee explicitly recommends course B3, 'Introduction to Discrete Structures' which has strongly influenced this book not only in title but also in selection of topics"[1], "In fact the book covers all the topics (and more) listed for the course I5 (Compiler Construction) recommended by the ACM Curriculum Committee in the March 1968 issue of the Communications of the ACM"[2], and "the book is written to follow the course description developed by the Association for Computing Machinery (ACM), published in the March 1968 issue of the Communications of the ACM"[3]. In addition, at least one publisher has superimposed his titles in the

1. Franco P. Preparata and Raymond T. Yeh, Introduction to Discrete Structures (Addison-Wesley, Reading, Massachusetts, 1973), p. v.

2. David Gries, Compiler Construction for Digital Computers (John Wiley and Sons, New York, 1971), p. vii.

3. Richard C. Dorf, Introduction to Computers and Computer Science (Boyd and Fraser Publishing, San Francisco, 1972), pp. ix-x.

prerequisite chart found in "Curriculum '68" to establish to the user where the texts fit into the curriculum.[4]

References to "Curriculum '68" are not, however, limited to publishers. At a recent national meeting of computer professionals,[5] in announcing faculty openings, several institutions referenced courses to be taught by the course numbers of "Curriculum '68". The literature of computer science education also abounds with references to the report such as: "the Undergraduate program includes courses roughly equivalent to most courses in 'Curriculum '68'"[6], "Table 2 shows a comparison of the subject areas emphasized in the ACM 1968 Recommendations for Academic Programs in Computer Science"[7], and "The courses B3, I6, and I7 above have been remarkably stable perhaps because of the relative distance of their content from the frontiers of research. Perhaps some minor changes might be considered: course B3 could include some material on the first-order predicate calculus rather than just the propositional calculus, and course I7 still lacks a textbook which is teachable and has a good balance between intuition and rigor. Nevertheless, the 1968

4. See the 1973-4 McGraw Hill catalogue of books in computer science.

5. 1974 Computer Science Conference, Detroit, Michigan, February 11-14, 1974.

6. J. Tartar and J. P. Penny, "Undergraduate Education in Computing Science - Some Immediate Problems", SIGCSE Bulletin, 4, 1 (March 1972), 1.

7. George A. Mapp, "A Proposal for a B. S. in Information Systems", SIGCSE Bulletin, 5, 1 (February 1973), 92.

descriptions of these courses are still a very good approximation to current course offerings in a number of university and college departments of computer science".[8]

In this study, the development of "Curriculum '68", the impact of the recommendations, and subsequent developments in computer science education will be considered. From this study of "Curriculum '68" with a perspective of six years of implementation and experience, a sense of why the recommendations developed as they did will be determined, as well as an identification of the strengths and weaknesses of the document.

The study will have two contributions to the field of computer science education; the first in curriculum development, and the second in curriculum implementation.

Regarding curriculum development, "Curriculum '68" in its present form has been available for six years. Its origins can be traced back further. For some time requests for updates and revisions reflecting changes in technology, and experience in offering instruction in computer science have been made, but such revisions have not come about on any major scale. Specific requests for such updates have come from the IFIP World Conference on Computer Education 1970[9] and the attendees of the ACM Institute on Computer

8. P. C. Fischer, "Theory of Computing in Computer Science Education", Proceedings of the AFIPS 1972 SJCC (AFIPS Press, Montvale, New Jersey, 1972), 857.

9. Reported by W. F. Atchison at C^{3}S meeting 1971 FJCC.

Science Education.[10] This study will isolate those areas in need of revision and updating, and thus serve as the groundwork necessary for the revision of "Curriculum '68".

Regarding curriculum implementation, the study will supply a means of interpretation of "Curriculum '68", as well as a source of additional materials regarding curriculum work and implementation. In this way, the study will be useful to those institutions which are in the process of just beginning programs in computer science, or are anticipating beginning such programs.

The study uses "Curriculum '68" as its primary source document. The earlier report of the ACM Curriculum Committee on Computer Science [57] is also considered, as are reports related to the other curriculum activities within ACM. In addition the literature of computer science education appearing in professional journals, conference proceedings and informal publications of special interest groups are considered. Minutes of meetings of the ACM Curriculum Committee on Computer Science subsequent to the publication of the report and several unpublished reviews of curriculum implementations are also reviewed.

In chapter 1 the "Curriculum '68" report itself is reviewed. Primary emphasis is placed on the descriptive material such as the subject classification and the program description. Chapter 2 considers reviews and critiques

10. Reported at evaluation meeting of the Institute, August 1971.

of the report which appeared immediately following its publication. Chapter 3 considers the relationship of "Curriculum '68" to the earlier works of the ACM Curriculum Committee on Computer Science, and to curriculum work in computing carried on in the same time period by the COSINE Committee of the Commission on Engineering Education and the Committee on the Undergraduate Program in Mathematics of the Mathematical Association of America.

While the first three chapters are concerned with "Curriculum '68", the material leading up to it, and, work parallel to it, chapters 4, 5, and 6 look at implementation and subsequent work. In chapter 4 the relationship of existing programs to the recommendations are considered. Three studies involving what is being offered in computer science as well as three reports on the reactions of graduates of such programs to the programs are reviewed. In chapter 5 work in specific courses, both those recommended in "Curriculum '68" and other courses are considered. In chapter 6, subsequent work of the Curriculum Committee on Computer Science, its sub-committees, and other groups involved in curriculum work in computer science is presented.

Chapter 7, utilizing the information of the first six chapters, summarizes and assesses the impact of "Curriculum '68". Strengths and weaknesses of the report are indicated; ways in which subsequent work meets these weaknesses is shown; and appropriate areas for future work in curriculum development are identified.

Chapter 1 - Overview of "Curriculum '68"

> This report contains recommendations on academic programs in computer science which were developed by the ACM Curriculum Committee on Computer Science. A classification of the subject areas contained in computer science is presented and twenty-two courses in these areas are described. Prerequisites, catalog descriptions, detailed outlines, and annotated bibliographies for these courses are included. Specific recommendations which have evolved from the Committee's 1965 Preliminary Recommendations are given for undergraduate programs. Graduate programs in computer science are discussed and some recommendations are presented for the development of master's degree programs. Ways of developing guidelines for doctoral programs are discussed, but no specific recommendations are made. The importance of service courses, minors and continuing education in computer sciences is emphasized. Attention is given to the organization, staff requirements, computer resources and other facilities needed to implement computer science educational programs.[1]

This abstracts "Curriculum '68" [58]. In this study the impact of this major report of the Curriculum Committee on Computer Science (C^3S) of the Association for Computing Machinery (ACM) will be considered for its impact on computer science education. To do this the report itself must be reviewed and this will be done in this chapter. While the most attention has been given over the years of the life of the report, to the course descriptions and prerequisite structure of "Curriculum '68", the report encompasses a great deal more.

The major sections of the report are a subject classification of computer science, description of courses, undergraduate

1. Curriculum Committee on Computer Science, "Curriculum '68, Recommendations for Academic Programs in Computer Science", Communications of the ACM 11, 3 (March 1968), 151.

programs, master's degree programs, doctoral programs, service courses, minors and continuing education, and implementation. This is followed by the more detailed course outlines and bibliographies. It must be observed that the recommendations are limited in scope and objective and this is clearly stated at the outset:

> ...these recommendations are not directed to the training of computer operators, coders, and other service personnel. Training for such positions, as well as for many programming positions, can probably be supplied best by applied technology programs, vocational institutes, or junior colleges. It is also likely that the majority of application programmers in such areas as business data processing, scientific research, and engineering analysis will continue to be specialists educated in the related subject matter areas, although such students can undoubtedly profit by taking a number of computer science courses.[2]

The justification for the existence of computer science as an academic discipline was covered in the earlier reports of C3S [57]. "Curriculum '68" does not attempt to repeat this, but rather defines computer science in terms of three major subject areas:

Subject Areas of Computer Science[3]

I. Information Structures and Processes
   1) Data Structures
   2) Programming Languages
   3) Models of Computation

II. Information Processing Systems
   1) Computer Design and Organization
   2) Translators and Interpreters
   3) Computer and Operating Systems
   4) Special Purpose Systems

III. Methodologies
   1) Numerical Mathematics

2. Ibid., 154.

3. Ibid., 154-155.

2) Data Processing and File Management
3) Symbol Manipulation
4) Text Processing
5) Computer Graphics
6) Simulation
7) Information Retrieval
8) Artificial Intelligence
9) Process Control
10) Instructional Systems

It is further noted that there are a number of related areas essential to a balanced computer science program. Work in the specific areas of "mathematical sciences" and "physical and engineering sciences" warrented inclusion within the classification:

IV. Mathematical Sciences
1) Elementary Analysis
2) Linear Algebra
3) Differential Equations
4) Algebraic Structures
5) Theoretical Numerical Analysis
6) Methods of Applied Mathematics
7) Optimization Theory
8) Combinatorial Mathematics
9) Mathematical Logic
10) Number Theory
11) Probability and Statistics
12) Operations Analysis

V. Physical and Engineering Sciences
1) General Physics
2) Basic Electronics
3) Circuit Analysis and Design
4) Thermodynamics and Statistical Mechanics
5) Field Theory
6) Digital and Pulse Circuits
7) Coding and Information Theory
8) Communication and Control Theory
9) Quantum Mechanics

It is interesting to note that C^3S did not consider material dealing with things like sociological, economic, and educational implications of developments of computer science, feeling that these areas were more within the realm of interests of philosophy

and sociology departments who should develop such course work in cooperation with computer scientists.

To meet the educational needs of computer science twenty-two courses are specified, each carrying three semester hours credit. These courses are divided into basic (freshman-sophomore level), intermediate (junior-senior or beginning graduate level), and advanced:

The Courses of "Curriculum '68"[4]

Basic Courses
- B1) Introduction to Computing
- B2) Computers and Programming
- B3) Introduction to Discrete Structures
- B4) Numerical Calculus

Intermediate Courses
- I1) Data Structures
- I2) Programming Languages
- I3) Computer Organization
- I4) Systems Programming
- I5) Compiler Construction
- I6) Switching Theory
- I7) Sequential Machines
- I8) Numerical Analysis I
- I9) Numerical Analysis II

Advanced Courses
- A1) Formal Languages and Syntactic Analysis
- A2) Advanced Computer Organization
- A3) Analog and Hybrid Computing
- A4) System Simulation
- A5) Information Organization and Retrieval
- A6) Computer Graphics
- A7) Theory of Computability
- A8) Large-Scale Information Processing Systems
- A9) Artificial Intelligence and Heuristic Programming

In terms of the subject matter classification cited above, the courses can be listed by category as follows, recognizing, of course, that some of the courses cross

4. Ibid., 156-160.

classification boundaries:

Information Structures and Processes
- B1) Introduction to Computing
- B2) Computers and Programming
- B3) Introduction to Discrete Structures
- I1) Data Structures
- I2) Programming Languages
- I6) Switching Theory
- I7) Sequential Machines
- A1) Formal Languages and Syntactic Analysis
- A7) Theory of Computability

Information Processing Systems
- I3) Computer Organization
- I4) Systems Programming
- I5) Compiler Construction
- A2) Advanced Computer Organization
- A3) Analog and Hybrid Computing

Methodologies
- B4) Numerical Calculus
- I8) Numerical Analysis I
- I9) Numerical Analysis II
- A4) System Simulation
- A5) Information Organization and Retrieval
- A6) Computer Graphics
- A8) Large-Scale Information Processing Systems
- A9) Artificial Intelligence and Heuristic Programming

The special place of mathematics is recognized, with reference specifically to the reports of the Committee on the Undergraduate Program in Mathematics (CUPM) A General Curriculum in Mathematics for Colleges [40], Recommendations on the Undergraduate Mathematics Program for Engineers and Physicists [42], and A Curriculum in Applied Mathematics [41]. A parallel structure in mathematics within the structure of prerequisites for computer science is given in the report.

A detailed prerequisite structure is given, however, it is noted that other structures are possible, and that the given structure will likely change as the field advances. In the case of the advanced courses, the structure is

especially open to modification based on the orientation of the program.

Within the structure, of particular interest, is the collection of computer science and mathematics courses that are considered the "core" of computer science:

Core of Computer Science[5]
Computer Science Courses
- B1) Introduction to Computing
- B2) Computers and Programming
- B3) Introduction to Discrete Structures
- B4) Numerical Calculus
- I1) Data Structures
- I2) Programming Languages
- I3) Computer Organization
- I4) Systems Programming

Mathematics Courses (Referenced to CUPM)
- M1) Introductory Calculus
- M2) Mathematical Analysis I
- M3) Linear Algebra
- M2P) Probability

With the core specified, the report turns to the definition of an undergraduate program in computer science. In the specification of the program, which C^3S felt helped mark computer science as an established field of study, efforts were made to integrate basic concepts of computer science with professional techniques.

The committee observed that professionals were divided as to whether or not undergraduate programs should be started at the time of the writing of the report, and although recommending in favor of such program, a warning was sounded against allowing the glamor of the field from leading to the premature establishment of programs and thus lowering

5. Ibid., 157.

of standards. The program leading to the undergraduate degree was stated in terms of the various aspects of professional development; computer science course work, programming experience, mathematics course work, technical electives, and possible areas of specialization.

The computer science courses recommended for the undergraduate program were to establish the groundwork in the field. These courses, selected from the basic and intermediate list, concentrate on the divisions of information structures and processes, and information processing systems:

> The Undergraduate Program[6]
>
> The major in computer science should consist of at least 30 semester hours including the courses:
>
> B1) Introduction to Computing
> B2) Computers and Programming
> B3) Introduction to Discrete Structures
> B4) Numerical Calculus
> I1) Data Structures
> I2) Programming Languages
> I3) Computer Organization
> I4) Systems Programming
>
> and at least two of the courses:
>
> I5) Compiler Construction
> I6) Switching Theory
> I7) Sequential Machines
> I8) Numerical Analysis I
> I9) Numerical Analysis II

Programming experience was not regarded as the primary objective of the undergraduate program, however, it was anticipated that it would be an achievement of such a program, both through course work, and some form of "true-to-life" programming within the program.

---

6. Ibid., 161.

Mathematics was regarded as essential. In effect a strong minor was recommended, selected from courses from the CUPM Curriculum Recommendations [40]:

> Mathematics Course for the Computer Science Undergraduate Program[7]
>
> The supporting work in mathematics should consist of at least 18 hours including the courses:
>
> M1) Introductory Calculus
> M2) Mathematical Analysis I
> M2P) Probability
> M3) Linear Algebra
>
> and at least two of the courses:
>
> M4) Mathematical Analysis II
> M5) Advanced Multivariate Calculus
> M6) Algebraic Structures
> M7) Probability and Statistics

Students were encouraged to take technical electives in computer science or related fields. It was suggested that not more than three additional computer science courses be taken to avoid problems of overspecialization. The use of the technical electives was considered especially valuable when the courses were combined with the optional computer science courses and electives from outside areas to give degrees of specialization in certain areas:

> Examples of Course Selection for Areas of Specialization[8]
>
> Applied Systems Programming
>
> Optional courses
> I5) Compiler Construction
> I6) Switching Theory
>
> Electives from courses
> A2) Advanced Computer Organization
> A5) Information Organization and Retrieval
> A6) Computer Graphics
>
> Electives from areas
> IV.8) Combinatorial Mathematics

7. Ibid., 162.

8. Ibid.

IV.9) Mathematical Logic
IV.11) Probability and Statistics
IV.12) Operations Analysis

Computer Organization and Design

Optional courses
I6) Switching Theory
I7) Sequential Machines

Electives from courses
A2) Advanced Computer Organization
A8) Large-Scale Information Processing Systems
A4) System Simulation

Electives from areas
IV.3) Differential Equations
V.2) Basic Electronics
V.6) Digital and Pulse Circuits
V.7) Coding and Information Theory

Scientific Applications Programming

Optional courses
I8) Numerical Analysis I
I9) Numerical Analysis II

Electives from courses
A3) Analog and Hybrid Computing
A4) System Simulation
A5) Information Organization and Retrieval
A6) Computer Graphics

Electives from areas
IV.3) Differential Equations
IV.7) Optimization Theory
V.4) Thermodynamics and Statistical Mechanics
V.5) Field Theory

Data Processing Applications Programming

Optional courses
I5) Compiler Construction
I6) Switching Theory

Electives from courses
A4) System Simulation
A5) Information Organization and Retrieval
A8) Large-Scale Information Processing Systems

Electives from areas
IV.7) Optimization Theory
IV.11) Probability and Statistics
IV.12) Operation Analysis
V.7) Coding and Information Theory

The discussion of the undergraduate program concludes with a set of semester by semester sequences of courses, showing how workable programs can be fit into a four year sequence.

The master's degree program recommended by the committee contains the following statement of purpose:

> The proposed program embodies sufficient flexibility to fulfill the requirement of either an "academic" degree obtained in preparation of further graduate study or a terminal "professional" degree. Until clearer standards both for computer science research and the computing profession have emerged, it seems unwise to attempt to distinguish more definitely between these two aspects of the master's degree programs.[9]

The program description first considers the necessary undergraduate background:

> Undergraduate Preparation[10]
>
> The recommended preparation for graduate study in computer science consists of three parts as listed below. The course work which would provide this background is indicated in parentheses.
>
> a. Knowledge of computer science including algorithmic processes, programming, computer organization, discrete structures and numerical mathematics (courses B1, B2, B3, and B4 or I8).
> b. Knowledge of mathematics, including the calculus and linear algebra, and knowledge of probability and statistics. (courses M1, M2, M3, M4, M2P, M7 of CUPM)
> c. Additional knowledge of some field such as computer science, mathematics, electrical engineering, physical science, biological science, linguistics, library science, or management science which will contribute to the student's graduate study in computer science. (Four appropriate courses on an intermediate level.)

The program is described in general terms:

> The Master's Degree Program[11]
>
> The master's degree program in computer science should consist of at least nine courses. Normally at least two courses, each in a different subject

9. Ibid., 164.

10. Ibid., 163.

11. Ibid.

area, should be taken from each of the following subject divisions of computer science:
I. Information Structures and Processes
II. Information Processing Systems
III. Methodologies
Sufficient other courses in computer science or related areas should be taken to bring the student to the forefront of some area of computer science.

The program was then to have degrees of both breadth and depth; the breadth from the requirement for at least two courses in each of the three major subject divisions of computer science, and the depth from the selection of courses to bring the student to the state of the art in some area of computer science. The special place of mathematics was noted here as in the undergraduate program with the recommendation that the student who does not have a "strong" mathematical background take additional mathematics courses or computer science courses with a high mathematical content.

The following are examples showing how the requirements for the program can be pulled together to form an area of specialization. The numbers here refer to subject areas:

Possible areas of Specialization in the Master's Program[12]
Theoretical Computer Science
I.1) Data Structures
I.2) Programming Languages
I.3) Models of Computation
III.3) Symbol Manipulation
III.8) Artificial Intelligence
IV.8) Combinatorial Analysis
IV.9) Mathematical Logic
V.7) Coding and Information Theory
Applied Software
I.1) Data Structures
I.2) Programming Languages
II.1) Computer Design and Organization
II.2) Translators and Interpreters
II.3) Computer and Operating Systems

12. Ibid., 164.

III.3) Symbol Manipulation
III.6) Simulation
IV.9) Mathematical Logic

Applied Hardware
I.1) Data Structures
I.3) Model of Computation
II.1) Computer Design and Organization
II.3) Computer and Operating Systems
III.5) Computer Graphics
IV.7) Optimization Theory
IV.9) Mathematical Logic
V.6) Digital and Pulse Circuits
V.7) Coding and Information Theory

Numerical Mathematics
I.1) Data Structures
I.2) Programming Languages
II.1) Computer Design and Organization
II.3) Computer and Operating Systems
III.1) Numerical Mathematics
III.6) Simulation
IV.5) Theoretical Numerical Analysis
IV.6) Methods of Applied Mathematics
IV.9) Optimization Theory

Instrumentation
I.1) Data Structures
I.2) Programming Languages
II.1) Computer Design and Organization
II.4) Special Purpose Systems
III.6) Simulation
III.9) Process Control
IV.6) Methods of Applied Mathematics
IV.7) Optimization Theory
V.8) Communication and Control Theory

Information Systems
I.1) Data Structures
I.2) Programming Languages
II.1) Computer Design and Organization
II.3) Computer and Operating Systems
III.2) Data Processing and File Management
III.4) Text Processing
III.7) Information Retrieval
IV.7) Optimization Theory
IV.9) Mathematical Logic

Though a master's project or thesis was not specified as such in the report, the recommendation was given that some requirement be made to insure that the student gains experience in computer applications at the level of a major

project.

The doctoral program is not specified in any way in the report. The committee did consider ways in which guidelines could be developed, and acknowledged that many of the problems of doctoral programs were addressed at the Stony Brook Conference [74]. A series of articles on doctoral programs were, at the time, scheduled to appear in the Communications of the ACM, which would discuss, in depth, various aspects of such programs. Among the topics to be addressed in these articles were the following:[13]

1) Definition of subject areas, possibly in terms of an annotated bibliography.
2) Prerequisites for work in the area at the doctoral level.
3) Outlines of appropriate graduate courses in the area.
4) Examples of questions for qualifying exams in the area.
5) Indication of suitable thesis topics and promising directions for research.
6) Extent to which subject areas ought to be required of all doctoral students in computer science.

The matters of service courses, minors, and continuing education which are so necessary in any field are addressed in the report. The need for service courses is spelled out in terms of the recommendations of the Pierce Committee [150]. For the student, referred to in the Pierce Report, as being in the more quantitative fields, course B1 is recommended with course B2 or B4 serving as a second course. It was also observed that the interested student could easily pick a minor out of the other courses in the recommended curriculum.

13. Ibid., 165.

A slight realignment of course B1, to stress applications in text processing and non-numeric areas was recommended for other students. It was observed that care should be taken in the design of such a course to be sure that a student completing it could take additional work, like courses B2 and B3, if he so desires. It was also felt desirable by the committee to consider introducing a survey course with primary attention to the implications of computer technology, though such a course itself was not outlined.

It was the belief of the committee that the service course be offered by the computer science department, however, it was stressed that the department must, at all times, be sensitive to the needs of the various users.

The needs for programs in continuing education were addressed in the report, though no firm recommendations for courses in this aspect of computer science education were given:

> Finally, the need for continuing education in computer science must be recognized. Much of the course material discussed in this report did not exist 10-15 years ago and practically none of this material was available to students until the last few years. Anyone who graduated from college in the early 1960's and whose "major" field of study is related to computing is already out-of-date unless he has made a determined effort to continue his education. Those responsible for academic programs in computer science and those agencies which help to direct and support continuing education should be especially alert to these needs in this unusually dynamic and important field.[14]

The final section of the report deals with the problems of implementation of computer science programs. The committee

14. Ibid., 166.

looked for diffusion and chaos if computer science departments were not formed and instead courses in computing developed all over the campuses:

> The organizational problems for this new field are serious and their solution will inevitably require new budget commitments from a university. However, failure to come to grips with the problem will probably prove more costly in the long run: duplicated courses and programs of diluted quality may result, and a major upheaval may eventually be required for reorganization.[15]

Perhaps the most critical organizational problem in the development of computer sciences departments was acquisition of staff. The committee recognized that initially people would come into the programs with a variety of formal backgrounds in related areas like mathematics or electrical engineering, however, it was felt essential that this faculty consider itself composed of computer scientists. While joint appointments were felt desirable in some cases, it was considered important that a substantial part of the faculty be fully committed to computer science. It was observed that a critical size, probably 5 full time equivalents, is necessary for a reasonable coverage of the areas of computer science. It was also observed that due to the lack of teaching materials, assignments of the faculty should be such that time is available for the faculty to develop teaching materials.

The committee considered computer science as a laboratory science as far as physical facilities are concerned, and recommended that facilities be available for card preparation,

15. Ibid.

study of listings and so on. In addition to the usual library facilities needed for all disciplines, it was observed that facilities for holding and using research reports, manuals, and programs would be necessary. Reference was also made to computing facilities:

> Degree programs require regular access to at least a medium-size computer system of sufficient complexity in configuration to require the use of an operating system. The total cost of such systems are at least $20,000 per month. In terms of hours per month, the machine requirements of computer science degree programs will vary according to the number of students enrolled, the speed of the computer and the efficiency of its software, and the philosophy of the instructors. It is entirely possible that an undergraduate degree program might require as much as four hours of computing on a medium-sized computer per day.[16]

In addition to these considerations, adequate space and good turn around from the computer center were stressed.

It was observed that the study of systems programming would require special facilities and possibly separate equipment. In advanced programs, additional equipment was also anticipated for such things as graphics, numeric control of machines, process control, simulation, information retrieval and computer-assisted instruction.

The existence of a strong computer science program on campus, should enhance the programs and activities of the computer center, even though the two are separate entities. The report does stress that these activities should be closely associated:

---

16. Ibid., 167.

> It should be realized, however, that the basic philosophies of providing services and pursing academic ends differ to such an extent that conflicts for attention may occur. At one extreme, the research of a computer science faculty may so dominate the activities of the computer center that its service to the academic community deteriorates. At the other extreme, the routine service demands of a computer center may inhibit the faculty's ability to do their own research, or the service orientation of a center may cause the educational program to consist of mere training in techniques of only transient value. Considerable and constant care must be taken to maintain a balance between these extremes.[17]

Summary

"Curriculum '68" gives detailed specifications for twenty-two courses fundamental to the study of computer science, as well as putting these courses into a prerequisite structure that also involves mathematics.

While the questions of definition and justification of computer science are not addressed, the subject matter is classified into three major areas of computer science and two related areas. A core area is defined, and based on this core area the undergraduate program is detailed involving computer science courses, mathematics courses, and technical electives.

Graduate degree programs are mentioned but not detailed to the extent of the undergraduate program. The master's program is outlined in terms of areas while doctoral programs are only mentioned in general terms, with ideas put forth for further work.

17. Ibid., 168.

Additional topics receive some attention, these including service courses and minors, continuing education programs, program implementation problems, and facilities.

Chapter 2 - Critiques of "Curriculum '68"

Since the publication of "Curriculum '68", a great deal has been said about it, both pro and con, and its influence is seen in many descriptions of courses and programs. At the same time, however, suprisingly little has appeared in the literature which directly analyzes or critiques the recommendations. The period immediately preceeding, and immediately following the publication of "Curriculum '68" produced several reviews and critiques which give a good deal of insight into the role and nature of the document, and in many ways put into writing those things which have been subsequently said. In this chapter several of these documents will be considered.

In June 1968 "Curriculum '68" was independently reviewed by E. I. Organick [144], R. T. Gregory [92], and S. Rosen [155]. The reviews were uniformly positive toward the report with a typical statement being that of Organick:

> This outstanding report will serve as a landmark contribution of ACM and NSF to educators seeking to establish useful programs and courses in computer science at the college and university level.[1]

Rosen also stresses the value of the report to those establishing programs noting that the report will be of special value to those who have not been deeply involved in the early developments in computer science, but want to develop and offer a program.

1. E. I. Organick, "Review of Curriculum '68", Computing Reviews (Review Number 14,389) 9, 6 (June 1968), 303.

He too notes that "Curriculum '68" can form the basis for the curriculum even in the case where individual institutions might wish to make a number of changes in the actual program they offer.

Within all the reviews there are some expressions of concerns. Organick was concerned that the report might serve as too much of a blueprint for all computer science education:

> The only major reservation this reviewer would have is that the report could be regarded as too good...less-than-fully imaginative educators might be tempted to urge accreditation of B. S. programs based on proximity to the applied model.[2]

Rosen expresses another item as his major concern:

> Perhaps my strongest criticism of the work of the Curriculum Committee is that it seems to give insufficient emphasis to the interdisciplinary aspects of computer science.[3]

Other items were expressed which were of some concern to the reviewers. Organick was concerned over the fact that there was little experience by the members of the Curriculum Committee in the running of the programs they were recommending. He was also concerned with the interrelationship of calculus with the computer science core courses, and especially I2 - Programming Languages, I3 - Computer Organization, and I4 - Systems Programming.

Gregory's only specific concern was with the fact that the area of theoretical numerical analysis was grouped under

---

2. Ibid., 304.

3. S. Rosen, "Review of Curriculum '68", Computing Reviews (Review Number 14,391) 9, 6 (June 1968), 305.

the mathematical sciences as a "related" subject area rather than directly in computer science.

Rosen notes that in many ways the curriculum recommendations represents the point of view of those who have come into computer science from mathematics and that the program is highly oriented to mathematics. He notes that while the bibliographies for the courses are extensive they do fall into the trap of containing some irrelevant and poor quality material.

Rosen notes the significance and importance of the Data Structure area specified in "Curriculum '68", however, he did have some reservations as to the course as described:

> There is no question in my mind as to the importance of the topics listed as the subject matter of this course. Every computer scientist should know much of this material, but I do not see how that particular collection of material can be organized into a course for college undergraduates.[4]

In spite of his several objections, however, Rosen does conclude that the report is most valuable:

> Although this reviewer has a few reservations and criticisms, the report represents a most important and useful document, and the committee that produced the report deserves the thanks of all of us who are interested in curriculum problems in computer science.[5]

In 1967, the Association for Educational Data Systems (AEDS) held a conference on the Computers in American Education. Though this conference was primarily devoted to educational use of computers and not computer science education, there are references to the latter area. Sylvia Charp [35] describes

4. Ibid., 306

5. Ibid.

courses designed for secondary school students and addresses some of the problems of teacher training, while John Caffrey [34] expresses some commonly held ideas that spawned the development of computer science instruction:

> ...As the number of users, especially students, grows and grows (not just in mathematics and the hard sciences), the need for easier access via remote terminals and more effective operating systems, with short turnaround, grows insistently. One even begins to hear it said that in the humanities and liberal-arts program there is growing realization that the citizen of tomorrow had better know as much as he can about computers.
>
> Hence the demands for instruction in computer science grows apace. It is possible to find in college catalogues across the country a great variety of courses designed for special purposes; the engineer and the business administrator apparently need different kinds of courses in programming; there are courses covering the general functions of computers, the design of computers, computers in system analysis, and so on. It is hard to find a discipline in which the computer has not obtruded, even in art and music.[6]

The most relevant article to computer science education, of this collection is by Ottis W. Rechard [153]. Though written before the formal publication of "Curriculum '68", it was written with a knowledge of what would be in the document, and expresses many of the concerns and objections that were aired subsequent to publication.

Rechard traces the development of computer science education noting that at the time of writing there were at least thirty-three institutions offering some kind of program in the field. He notes that these programs vary a great

6. John Caffrey, "Computers in Higher Education", in D. D. Bushnell and D. W. Allen (eds.), The Computer in American Education (John Wiley and Sons, New York, 1967), p. 220.

deal, and that at the time of writing, it is difficult to identify a common basis for an undergraduate program:

> The diversity among these curricula is very great indeed, reflecting at this point little uniformity of opinion as to what represents adequate academic training in computer or information science. Beyond some basic instruction in algorithmic processes and a procedure-oriented language (usually FORTRAN or ALGOL), some numerical analysis and some systems programming, it is difficult to find a common element. For this and other reasons, the numerous attempts to establish undergraduate majors in computer science seems premature and misguided. At a time when the field is still struggling with problems of identity, it does not seem fair to the undergraduate student to encourage him in the belief that a degree in computer science will mean as much as a degree in one of the more established disciplines. After all, it is reasonable to expect that two students majoring in mathematics at different universities will each have been exposed to a rather large common body of knowledge. The same is true of physics, biology, and electrical engineering, but it is definitely not true of computer science.[7]

Rechard goes on to specifically raise questions regarding the report of C^3S:

> In an attempt to deal with this problem, the ACM Curriculum Committee has recently proposed a curriculum for an undergraduate major in computer science. While there is much to applaud in this effort, many of the recommendations are nevertheless puzzling. Why, for example, should a course in "combinatorics and graph theory" be listed as a basic course in computer science while courses in "linguistics" and "analog computers" are included in a list of supporting courses? The catalog descriptions of certain of the courses can cause some confusion. In addition, if one regards courses such as "numerical analysis", "mathematical optimization techniques" and "constructive logic" as belonging as well to mathematics as to computer science, the proposed curriculum seems really to represent a sound major in mathematics with options in computation, such as has

7. Ottis W. Rechard, "The Computer Sciences in Colleges and Universities", in D. D. Bushnell and D. W. Allen (eds.) The Computer in American Education (John Wiley and Sons, New York, 1967), pp. 157-158.

been described by Perlis in the Communications of the ACM.[8]

It is noted that much of the pressure to offer undergraduate programs in computer science had come from the fact that mathematics departments had been reluctant to consider offering applied programs.

Though Rechard was against the formation of undergraduate programs in computer science, he was not against graduate programs or the offering of undergraduate courses:

> This qualified view of undergraduate majors in computer and information science should not be interpreted as opposition to undergraduate courses in the subject. Perhaps the most satisfactory arrangement at the present time is a department of computer and information science which offers a master's and Ph. D. degree and teaches a number of undergraduate courses, so that students can acquire background in the subject ranging from simple experience with a procedure-oriented programming language to familiarity with compiler and operating system construction.[9]

In September 1968 a Conference was held, with the support of The National Science Foundation at Park City, Utah on Computers in Undergraduate Education [178]. The conference, chaired by William Viavant was concerned with the general issues of computers in undergraduate education, but in large measure, addressed itself to computer science education, and interpretation of "Curriculum '68":

> The real objective of the conference was to stir things up. A casual reading of the report of the ACM Curriculum Committee, "Curriculum '68" might give to

8. Ibid., p. 158.

9. Ibid.

> a naive college administrator a feeling that the problems have been studied and solved, and that a school need only follow the recommendation of the report. If this conference persuaded even a few participants that they must themselves contribute to the creative use of computers in education, that there are serious problems in such a rapidly changing field, but exciting opportunity as well, and that any formula or program currently specified, should be used only as a skeleton at best, then it has been of value.[10]

An interesting presentation was made at this conference by Sam Conte on the "History and Activities of the ACM Curriculum Committee" [45]. After discussing some of the history of the committee, he turns to criticisms. First the question of the composition of the committee itself is addressed:

> Various criticisms have been leveled at the membership on this committee. One is that most of the committee were mathematicians by training. It is indeed true that 8 of the 12 members are mathematicians, 2 are engineers, and 2 are physicists. The composition simply reflects that fact that, whatever the reason, most of the leaders in computer science at that time were indeed trained as mathematicians. A second criticism was that most of the members were of the "Computing Center Director" type whatever that might imply. At the time the committee was formed 9 of the 12 members were Computing Center Directors but of course, at that time academic programs in computer science were almost non-existent and this was the only avenue available to those who wanted to maintain an interest in computing at universities. It is interesting to note that only 2 of these same members are now primarily directors of Computing Centers. A third criticism was that the committee was dominated by numerical analysts. Although 4 of the committee members can be classified as numerical analysts, it seems to me that the report places remarkedly little emphasis on numerical analysis. Indeed, considering the tremendous inefficiency of committee structures in general and the divergent view points represented

10. William Viavant (ed.), Proceedings of the Park City Conference on Computers in Undergraduate Education (The University of Utah, Salt Lake City, 1969), p. 5.

> by the committee, it seems to me quite remarkable that we are able to come up with as well balanced a document as I think the 68 report represents.[11]

Criticisms directed to the report itself are noted involving such matters as the amount of emphasis placed on mathematics and the amount of emphasis placed on hardware. There were also criticisms raised of a "philosophical" nature:

> The committee was criticized for not taking positive and forceful positions on the following questions:
> 1) Is Computer Science a discipline in the traditional academic sense? Does it have the cohesion and intellectual depth normally associated with a discipline? And closely related, should there be an undergraduate program in computer science?
> 2) Is computer science a branch of mathematics, or a branch of engineering or is it something else?
> 3) Where should computer science be located and how should it be organized? Should computer science be developed within existing departments such as mateh-matics or engineering, and if so which? Should there be separate departments of Computer Science? Or is an Institute of Computer Science the best structure?[12]

Conte claims that each of these questions were addressed by the committee. The committee took a practical approach to question 1 in that there was a need for education and such courses and programs would come into existence. They felt that Computer Science was a blend of engineering, mathe-matics and other disciplines, as well, leading to a truly distinct field worthy of study. They felt that computer science should exist as an independent entity on campus.

---

11. Sam Conte, "History and Activities of the ACM Curriculum Committee", in William Viavant (ed.), Proceedings of the Park City Conference on Computers in Undergraduate Education (The University of Utah, Salt Lake City, 1969), pp. 39-40.

12. Ibid., p. 41.

Conte, and the committee, did not feel that the publication of "Curriculum '68" ended work in the development of curriculum in computer science:

> ...we feel that it would be a mistake for all schools to attempt to implement these recommendations. We do hope that they will provide guidelines for programs in existence and for programs in the formative stages. Because computer science is still in a rapidly changing state and because we feel that curriculum is an evolving process, the committee feels that a continuing body should be established to constantly examine the development of programs in the country. In addition to a continued examination of curricular matters there are a number of other aspects of computer science education which our committee did not have time to explore.[13]

Among the new problems to be addressed were programs for smaller colleges, junior colleges and technical schools; the relationship of computer science to other disciplines including both those things computer science might offer other departments in the university, and what the computer science students would need from other departments; program implementation problems and related cost; the development of graduate programs; and course content development. In addition it was anticipated that the curriculum committee would sponsor a program of visitation and consultation to further assist in the development of computer science programs.

Ever since the publication of "Curriculum '68" and the development of academic programs in computer science, there have been indications that industry has many misgivings about such academic programs. Though there has been much

13. Ibid., p. 43.

said on these matters, not a great deal appears in the literature. At the Park City Conference there was a session of industry representatives and the comments of John E. Hale [94] of Burroughs Corporation and Kay Magleby [125] of Hewlett-Packard Company appear in the Proceedings. The remarks of Hale in many ways reflects the stated views of many representatives of industry:

> It doesn't matter to me whether a person working for us has a degree in computer science or mathematics or chemistry or music, or English or whatever. As a matter of fact, there are some very good people in the systems programming area who don't have degrees at all. But I think there is an opportunity for the university or the college to provide some fundamental knowledge to those who are interested in going into the computer end of the business, the programming end. But the colleges by and large have not fulfilled this responsibility. I would like to have a person have a fundamental acquaintanceship with what a machine is. That, he can only obtain in my opinion, by actually working with it. That is, he must have had the opportunity to become intimate with the machine. Now it doesn't really matter which machine it is, or whether it's a poor or good machine, but he ought to be able to play with it. To sit at the console and do things so that he will understand how the machine really works....I have interviewed people with degrees in computer science who possessed varying degrees of competence. One, whom I will never forget came with a Master's degree in computer science from a large university, and I was shocked to find out that all he knew about computers was numerical analysis...he didn't have the slightest understanding of what a machine was...I don't care whether he knows the machine actually uses vacum tubes or screws or bolts, that to me is not important. But he must have an understanding of the functional operation of the system.[14]

Magleby in many ways reinforces these views however suggests that computer science education cannot and should not try

14. John E. Hale, "Remarks by Representatives of Computer Manufacturers", in William Viavant (ed.), Proceedings of the Park City Conference on Computers in Undergraduate Education (The University of Utah, Salt Lake City, 1969), p. 107.

to duplicate industry's on the job training. Studies outside of the area of computer science to round out an educational program are stressed.

Marvin Minsky [140] delivered an invited address at the Park City Conference. In his talk he stressed the basic importance of computer science, however, he cautioned against too fast a move into computer science, in particular if this implies a movement away from mathematics:

> Now, of course there are different needs of students. I can't help being interested in students who are going to be the next generation's theorists and contribute to the theory. For them I regret to say that the best curriculum for computer science is a course in pure mathematics, on the grounds that the mathematical theories have stood the test of time. We have found that they are powerful and useful in developing new theories in computer science, whereas computer science theories as of today are very superficial: they haven't ramified. Nobody has found out what is important and what isn't since the work of Chomski in the middle 50's and that theory is untested. So I think it is important to keep in mind that if you have a student who is obviously a theoretical type, he's going to be a scientist, you should encourage him to take all the mathematics he can: algebra, topology, things like that. They won't be directly useful but he will become a mathematician and a most powerful theorist in general. Five years from now perhaps we will have the computer science which will incorporate those parts of mathematics that are going to be relevant. Right now we can't tell what they are.[15]

In working sessions of the Park City Conference there was a workshop dealing with Curriculum and Programs chaired by Earl Schweppe [163]. In the report of the workshop there

15. Marvin Minsky, "Speculations about Man and Machines", in William Viavant (ed.), Proceedings of the Park City Conference on Computers in Undergraduate Education (The University of Utah, Salt Lake City, 1969), pp. 150-151.

was a strong recommendation that most colleges and universities should consider offering work in computer science approaching a twelve to eighteen hour minor at a minimum. There were also indications that roughly the core program of "Curriculum '68" would fill this need.

In the 1968 ACM Turing Lecture R. W. Hamming [102] devoted a major portion of his remarks to computer science education and "Curriculum '68". Hamming places somewhat more emphasis on programming work than does "Curriculum '68":

> Were I setting up a computer science program, I would give relatively more emphasis to laboratory work than does Curriculum '68, and in particular I would require every computer science major, undergraduate or graduate to take a laboratory course in which he designs, builds, debugs, and documents a reasonably sized program, perhaps a simulator or a simplified compiler for a particular machine. The results would be judged on style of programming, practical efficiency, freedom from bugs, and documentation. If any of these were too poor I would not let the candidate pass. In judging his work we need to distinguish clearly between superficial cleverness and genuine understanding. Cleverness was essential in the past, it is no longer sufficient.
>
> I would also require a strong minor in some field other than computer science and mathematics. Without real experience in using the computer to get useful results the computer science major is apt to know all about the marvelous tools except how to use it. Such a person is a mere technician, skilled in manipulating the tool but with little sense of how and when to use it for its basic purpose. I believe we should avoid turning out mere idiot servants--we have more than enough "computniks" now to last us a long time. What we need is professionals!
>
> The Curriculum '68 recognized this need for "true-to-life" programming by saying, "This might be arranged through summer employment, a cooperative work-study program, part-time employment in computer centers,

> special projects courses, or some other appropriate means." I am suggesting that the appropriate means is a stiff laboratory course under your own control, and that the above suggestions of the Committee are rarely going to be effective or satisfactory.[16]

Hamming considers the question of how much mathematics is necessary in a computer science curriculum. With some reluctance, recognizing that many practitioners in the field do not have strong backgrounds in mathematics and would in the future be excluded by such requirements, he concludes that with the present state of the art a good deal of mathematics is necessary in the curriculum. This leads naturally to the question of what mathematics. At least one course in numerical analysis seems necessary, in addition topics like abstract algebra, queing theory, statistics, probability, coding theory, and graph theory are appropriate. He notes that with the then current state of mathematics education, it would be somewhat difficult for a student to pick up such material without having to take a good deal more material which would not be particularly relevant.

Hamming notes the problems voiced in the areas of business applications and their relation to computer science instruction; this in turn has application to all the disciplines which use computing:

> One of the complaints regularly made of computer science curriculums is that they seem to almost totally ignor business applications and COBOL. I think that

16. R. W. Hamming, "One Man's View of Computer Science", Journal of the ACM 16, 1 (January 1970), pp. 5-6.

> it is not a question of how important the applications are, nor how widely a language like COBOL is used, that should determine whether or not it is taught in the computer science department; rather I think it depends on whether or not the business administration department can do a far better job than we can, and whether or not what is peculiar to business applications is fundamental to other aspects of computer science. And what I have indicated about business applications applies, I believe, to most other fields of application that can be taught in other departments. I strongly believe that with the limited resources we have, and will have for a long time to come, we should not attempt to teach applications of computers in computer science departments - rather, those applications should be taught in their natural environments by the appropriate departments.[17]

Hamming notes some of the concerns expressed by other representatives of industry regarding the usefulness of some university trained computer scientists:

> At present there is a flavor of "game-playing" about many courses in computer science. I hear repeatedly from friends who want to hire good software people that they have found the specialist in computer science is someone they do not want. Their experience is that graduates of our programs seem to be mainly interested in playing games, making fancy programs that really do not work, writing trick programs, etc. and are unable to discipline their own efforts so that what they say they will do gets done on time in practical form. If I had heard this complaint merely once from a friend who fancied that he was a hard-boiled engineer, then I would dismiss it; unfortunately I have heard it from a number of capable intelligent, understanding people. As I earlier said, since we have such a need for financial support for the current and future expansion of our facilities, we had better consider how we can avoid such remarks being made about our graduates in the coming years. Are we going to continue to turn out a product that is not wanted in many places? Or are we going to turn out responsible, effective people who meet the real needs of our society? I hope that the latter will be increasingly true; hence my emphasis on the practical aspects of computer science.[18]

17. Ibid., 7.

18. Ibid., 8.

Much of the blame for this situation is attributed to the fact that many of the faculty are "pure" mathematicians in their background, and hence teach their courses as "pure" mathematics. Additionally, this means that quite often courses are taught in the wrong way. This is especially well illustrated in the teaching of programming:

> To parody our current methods of teaching programming, we give beginners a grammar and a dictionary and tell them they are now great writers. We seldom, if ever, give them any serious training in style. Indeed I have watched for years for the appearance of a Manual of Style and/or Anthology of Good Programming and have as yet found none. Like writing, programming is a difficult and complex art. In both writing and programming, compactness is desirable, but in both you can easily be too compact. When you consider how we teach good writing - the exercises, the compositions, and the talks that the student gives and is graded on by the teacher during his training in English - it seems we have been very remiss in this matter of teaching style in programming. Unfortunately only few programmers who admit that there is something in what I have called "style" are willing to formulate their feelings and to give specific examples. As a result, few programmers write in flowing poetry; most write in halting prose.[19]

Hamming turns his attention to some of the problems of the course descriptions of "Curriculum '68" observing that in many ways the committee concerned itself too much with details of the courses, and not enough with underlying ideas.

He concludes by addressing an issue not covered in the curriculum report; that of professional ethics:

> Along these lines, let me briefly comment on the matter of professional standards. We have recently had a standard published and it seems to me to be a good one, but again I feel that I am justified in asking how this

19. Ibid., 9-10.

> is being incorporated into the training of our students, how they are to learn to behave that way. Certainly it is not sufficient to read it to the class each morning; both ethical and professional behavior are not effectively taught that way. There is plenty of evidence that other professions do manage to communicate to their students professional standards which, while not always followed by every member are certainly a lot better instilled than those we are presently providing for our students. Again, we need to examine how they do this kind of training and try to adopt their methods to our needs.[20]

He concludes that these kinds of issues can be best handled by the example of the faculty, but that it is important that it be taught.

Summary

Initial reaction to "Curriculum '68" was favorable. Representatives of academic institutions looked on the document with more regard than did those from industry.

Specific criticisms were for the most part not directed to the courses themselves nor toward the suggested programs, but rather to more philosophical matters. Among these concerns were such things as whether the report represented too much of a manual for program construction, the interdisciplinary nature of programs and whether or not these were adequately reflected, whether or not undergraduate programs were advisable, and whether or not there was too much emphasis on mathematics.

Representatives of industry expressed primary concern on whether or not enough practical programming experience

20. Ibid., 11.

was in the program. Related to this it was noted that methods of introducing such experience was not adequately covered. Additionally professionalism questions in computing were not addressed.

Chapter 3 - Relation to Previous and Simultaneous Work

"Curriculum '68" was an outgrowth of "An Undergraduate Program in Computer Science - Preliminary Recommendations, A Report from the ACM Curriculum Committee on Computer Science" [57]. This document can be traced back to a Panel at the 1963 Annual Conference of the ACM, reported in the April 1964 Communications of the ACM, dealing with Computer Science Curriculum. In approximately the same time period the COSINE Committee of the Commission on Engineering Education prepared course guidelines for Computer Science in Electrical Engineering [46], and in May 1964 the Committee on the Undergraduate Program in Mathematics prepared "Recommendations on the Undergraduate Mathematics Program for Work in Computing" [39]. In this chapter these recommendations will be reviewed, looking for relationships between them and "Curriculum '68".

First the 1965 recommendations of the ACM Curriculum Committee will be considered. Conte in comparing the 65 and 68 report notes the following:

> ...as compared with the 1965 report, the 1968 report accomplished the following:
> 1) Revised some of the 1965 recommendations, sharpened some of the course outlines and added some new courses
> 2) Attempted to produce some order out of the diverse aspects of computer science by introducing a classification scheme

3) Made some recommendations on graduate programs in computer science especially at the MS level.[1]

The 1965 recommendations were drawn up by a committee consisting of:

S. D. Conte
John W. Hamblen
Thomas A. Keenan
William B. Kehl
Silvio O. Navarro
Werner C. Rheinboldt
Earl J. Schweppe
David M. Young, Jr.
William F. Atchison

In the period following the publication of the 1965 report, the National Science Foundation funded the project, allowing for more frequent meetings, and Thomas E. Hull, Edward J. McCluskey, and William Viavant joined the committee.

The Curriculum Committee viewed "Curriculum '68" as a major refinement and extention of the 1965 work:

> Of the sixteen courses proposed in the earlier recommendations, eleven have survived in spirit if not in detail. Two of the other five have been split into two courses each, and the remaining three have been omitted since they belong more properly to other disciplines closely related to computer science. In addition seven new courses have been proposed of which course B3 on "discrete structures" and course I3 on "computer organization" are particularly notable.[2]

The breakdown of these courses is shown in Chart 3-1.

The 1965 report was the product of approximately three years of work, and at the time of its publication it was

1. Sam Conte, "History and Activities of the ACM Curriculum Committee," in William Viavant (ed.), Proceedings of the Park City Conference on Undergraduate Computer Education (The University of Utah, Salt Lake City, 1969), p. 43.

2. Curriculum Committee on Computer Science, "Curriculum '68, Recommendations for Academic Programs in Computer Science", Communications of the ACM 11, 3 (March 1968), 153.

Chart 3-1

Comparison of "Curriculum '68" and "Curriculum '65"

I. Courses Appearing in both Curricula

"Curriculum '65"		"Curriculum '68"	
1.	Introduction to Algorithmic Processes	B1.	Introduction to Computer Science
2.	Computer Organization and Programming	B2.	Computers and Programming
3.	Numerical Calculus	B4.	Numerical Calculus
4.	Information Structures	I1.	Data Structures
6.	Logic Design and Switching Theory	I6.	Switching Theory
7.	Numerical Analysis I	I8.	Numerical Analysis I
8.	Numerical Analysis II	I9.	Numerical Analysis II
9.	Computer and Programming Systems	I4.	Systems Programming
11.	Systems Simulation	A4.	System Simulation
15.	Formal Languages	A1.	Formal Languages
16.	Heuristic Programming	A9.	Artificial Intelligence and Heuristic Programming

II. Courses Split

5.	Algorithmic Languages and Compilers	I2.	Programming Languages
		I5.	Compiler Construction
14.	Introduction to Automata Theory	I7.	Sequential Machines
		A7.	Theory of Computability

III. Courses not Appearing in "Curriculum '68"

10. Combinatorics and Graph Theory
12. Mathematical Optimization Techniques
13. Constructive Logic

IV. Courses new in "Curriculum '68"

B3. Introduction to Discrete Structures
I3. Computer Organization
A2. Advanced Computer Organization
A3. Analog and Hybrid Computing
A5. Information Organization and Retrieval
A6. Computer Graphics
A8. Large-Scale Information Processing Systems

anticipated that the report would primarily be a basis from which more detailed and comprehensive recommendations could be developed. More effort was made throughout the 1965 work to justify the existence of computer science as a field of study:

> Although much change has been accomplished within existing programs, such as mathematics and electrical engineering, there is a sizable area of work which does not naturally fit into any existing field. Thus, it is now generally recognized that this area, most often called Computer Science, has become a distinct field of study. This development is reported by the Committee on Uses of Computers of the National Academy of Science, by several individual authors and in a report of the Committee on the Undergraduate Program in Mathematics (CUPM)....[3]

The hope of the committee in this report was that the report would be a coordinating force for the many efforts already going on in university education in computer science.

It was recognized that there was a definite demand for computer work in various academic programs. This, however, was not the primary purpose of this report:

> The Committee has chosen to direct its attention first to students whose primary interest is in computer science. The Committee solicits comment and criticism of the present report with the view of both strengthening the current recommendations and illuminating the relation between computer science and other fields of study.[4]

The relationship of this work to other groups with an interest in curriculum in computing, and especially CUPM,

---

3. Curriculum Committee on Computer Science, "An Undergraduate Program in Computer Science, Preliminary Recommendations", Communications of the ACM 8, 9 (September 1965), 543.

4. Ibid., 544.

was recognized:

> The report of CUPM entitled "Recommendations on the Undergraduate Mathematics Program for Work in Computing" is very well done. Members of C3S, some of whom had consulted with CUPM, were greatly pleased with the perception shown in this document and with the fact that the program suggested by CUPM to some degree paralleled their own thinking. That report, however, was for the mathematics major and we feel that additional work is needed for the computer science major, especially in view of the fact that so many schools are moving toward such a major.[5]

The curriculum was intended to meet the usual requirements of undergraduate programs; entrance to graduate programs in computer science, direct entry into the profession, or entrance to graduate study in other fields. Thus the program consisted of a small number of required courses:

1) Introduction to Algorithmic Processes
2) Computer Organization and Programming
3) Numerical Calculus
4) Information Structures
5) Algorithmic Languages and Compilers

A series of "highly recommended electives":

6) Logic Design and Switching Theory
7) Numerical Analysis I
8) Numerical Analysis II
9) Computer and Programming Systems

and finally a list of "other electives":

10) Combinatorics and Graph Theory
11) Systems Simulation
12) Mathematical Optimization Techniques
13) Constructive Logic
14) Introduction to Automata Theory
15) Formal Languages
16) Heuristic Programming

Mathematics and other related courses were also mentioned

5. Ibid.

within this context of "required", "highly recommended electives", and "other electives".

An extensive justification of Computer Science as a discipline as well as an explanation of its purpose is given:

> Computer science is not simply concerned with the design of computing devices - nor is it just the art of numerical calculations, as important as these topics are. Computer science encompasses many specialized areas, all developing with extreme rapidity. It is natural that one should associate some small part, to which he has been exposed, with the subject as a whole. Even among those specializing in computer science, few (if any) are able to keep pace with the flood of innovation throughout the field.[6]

So much then for what computer science isn't; the question then is what computer science is?

> Computer science is concerned with information in much the same sense that physics is concerned with energy; it is devoted to the representation, storage, manipulation and presentation of information in an environment permitting automatic information systems. As physics uses energy transforming devices, computer science uses information transforming devices. Some forms of information have been more thoroughly studied and are better understood than others; nevertheless, all forms of information - numeric, alphabetic, pictorial, verbal, tactile, olefactory, results of experimental measurements, etc. - are of interest in computer science.
>
> Mathematics too is concerned with information and its structure and this tends to confuse those not well versed in both mathematics and computer science. The mathematician is interested in discovering the syntactic relation between elements based on a set of axioms which may have no physical reality. The computer scientist is interested in discovering the pragmatic means by which information can be transformed to model and analyze the information transformations in the real world. The pragmatic aspect of this interest leads to inquiry into effective ways to represent information,

6. Ibid.

> effective algorithms to transform information, effective languages with which to express algorithms, effective means to monitor the process and to display the transformed information, and effective ways to accomplish these at reasonable cost.[7]

The fact that academic programs had developed in a number of institutions further points to the fact that computer science is a discipline in its own right.

In the description of courses, there were hard decisions to be made as to whether a course was assigned to computer science or was assigned as a supporting course from another discipline. The decision was made on the basis of the relevancy of the material in question to the entire computer science program. In the case of this report, a breakdown as to the amount of time for each topic is not given. It was also noted that although references are given for the courses outlined, texts were not available for some of the courses.

The courses described in this report, as with "Curriculum '68", were in terms of three semester hour courses:

> In applying these recommendations to any given college, consideration should clearly be given to the possibility that certain courses may already be available by modification of existing courses. Double listing of courses is a technique that has sometimes been used with apparent success.
>
> We cannot be specific with regard to faculty organization for this curriculum simply because of the diversity of the American College structure. In some schools an independent department of computer science

7. Ibid.

> may be appropriate; in others such a department of computer science may not. We concur with CUPM that those responsible for computer science curricula should be closely linked to the computer scientists engaged in providing computational facilities. Although we further agree that most of the computer science courses described in this report will not be taught within existing mathematics departments, there is considerable benefit to be gained from mutual cooperation. In any case either new departments will need to be established, or existing mathematics (or other) departments will need to recognize the participation of computer scientists in the faculty development plan.[8]

The committee noted that several courses would well serve students in other academic areas; among these are course 1 (Introduction to Algorithmic Processes) for science and engineering students; there is enough flexibility to meet most of the requirements for the CUPM major in mathematics with emphasis in computing; engineering students can use course 6 (logic design and switching theory) and course 10 (combinatorics and graph theory); and the courses in numerical calculus and numerical analysis (courses 3, 7, and 8) are of value to a number of science related fields. The question of the needs of the computer science student from other departments is mentioned:

> We have not investigated the question of whether some modification of standard courses in mathematics, physics, etc., would be desirable from the viewpoint of a computer science major. Some intuitively feel that this may be the case but admit that the question needs careful consideration.[9]

The committee concludes the formal portion of the report

8. Ibid., 545.

9. Ibid.

with their anticipation of future work, and their feeling of the limitations of the present work:

> The work of C^3S is in its early stages. Readers of this report are encouraged to communicate their ideas, experiences and criticisms to the Committee. Not only will the present recommendations need periodic review but many suggestions concerning further computer-oriented curricula are sought. Study of the computer science needs of students in the non-physical sciences is appropriate. It has been suggested that the educational needs of those who will plan and design the computing and communication equipment of the future should be given special consideration. More specific thought will be given to the education of those wishing to prepare themselves for work in information retrieval, management science, the life sciences, and the behavioral sciences. Finally, the present recommendations do not deal with graduate computer science. A committee to study graduate curriculum is in the process of formation.[10]

Detailed descriptions of the courses in the curriculum follow. The way in which the courses fit together into the curriculum is shown in Chart 3-2.

The 1965 Curriculum Report was preceded by a series of papers appearing in the April 1964 issue of the Communications of the ACM dealing with course programs in computer science. These papers had first been presented at the 1963 ACM Annual Conference in Denver by a panel consisting of A. J. Perlis [146], Bruce W. Arden [6], George F. Forsythe [78], Robert R. Korfhage [119], Saul Gorn [86], and David E. Muller [141]. In the printed version, critiques were added to each of the papers, and a paper by Thomas A. Keenan [117] attempting a definition of computer science and a paper by William F. Atchison and John W. Hamblen [18] indicating the current

10. Ibid.

Chart 3-2

Structure of the Curriculum[11]

I. Required Courses
   A. Basic Computer Science
      1. Introduction to Algorithmic Processes
      2. Computer Organization and Programming
      3. Information Structures
   B. Theory Courses
      5. Algorithmic Languages and Compilers
   C. Numerical Algorithms
      4. Numerical Calculus (or Course 7)
   D. Supporting Courses
      Beginning Analysis (12 Credits)
      Linear Algebra (3 Credits)

II. Highly Recommended Electives
   A. Basic Computer Science
      6. Logic Design and Switching Theory
   B. Numerical Algorithms
      7. Numerical Analysis I
      8. Numerical Analysis II
   C. Supporting Courses
      Algebraic Structures
      Statistical Methods
      Differential Equations
      Advanced Calculus
      Physics (6 Credits)

III. Other Electives
   A. Basic Computer Science
      10. Combinatorics and Graph Theory
   B. Theory Courses
      13. Constructive Logic
      14. Introduction to Automata Theory
      15. Formal Languages
   C. Computer Models and Applications
      11. Systems Simulation
      12. Mathematical Optimization
      16. Heuristic Programming
   D. Supporting Courses
      Analog Computers
      Electronics
      Probability and Statistics
      Linguistics
      Philosophy
      Philosophy of Science

11. Ibid., 546.

state of computer science were added.

In Keenan's [117] definition of computer science, he identifies four basic topics of interest to the computer scientist:

> 1) Organization and interaction of equipment constituting an information processing system
> 2) Development of software systems with which to control and communicate with equipment
> 3) Derivation and study of procedures and basic theories for the specification of processes
> 4) Application of systems, software, procedures and theories of computer science to other disciplines[12]

He then goes on to give examples of the kind of work that goes on under each of these topics.

Within this definition of the field, Keenan turns to the question of education in computer science. Here he identifies five areas:

> 1) General Education - to insure that the public is properly aware of computers and what they can do
> 2) Training of Programmers
> 3) Orientation of Scientists
> 4) Education of Computer Specialists
> 5) Development of Computer Scientists[13]

Keenan then notes that each of these areas carries its own problems of emphasis and standards, and he relates the material in the set of papers to these levels of education:

> The courses described in this issue neither encompass the full domain of material available in computer science, nor signify the end product of the present curriculum evolution. The descriptions give but a snapshot of current thought on desirable material and methods of presentation in selected areas. Some of

12. Thomas A. Keenan, "Computers and Education", Communications of the ACM 7, 4 (April 1964), 206.

13. Ibid., 207-208.

> the courses (Perlis, Arden) are appropriate as introductions for computer specialists or for the orientation of students of other sciences; others (Muller, Korfhage, and Forsythe) described courses for computer specialists - i.e. undergraduate computer scientists - which may also merit consideration as electives for other disciplines; Gorn describes material largely of his own development which is placed at the graduate level in computer science.[14]

Perlis [146] describes a one semester first course in programming. The course which covered: (1) structure of algorithms; (2) structure of languages; (3) structure of machines; (4) structure of programs; and (5) structure of data, did not concentrate on the details of a language or a machine, leaving the mastery of this material to the problems and exercises. Perlis then goes on to outline a curriculum for an option in computation in an undergraduate mathematics major. Effectively this option adds to a traditional mathematics major course in Boolean Algebra and Switching Theory, Numerical Calculus, Computer Systems, Constructive Logic, and Advanced Theory of Computation.

T. M. Gallie in a brief critique of the paper finds no fault with either the course or the program described, noting that both require enthusiastic and knowledgeable professionals to be offered successfully. He indicates that few such individuals currently exist and expresses the hope that the universities will set out to prepare such people.

14. Ibid., 209.

Arden [6] presents a course designed as an introduction to computing for engineering and science students. Recognizing that such students will only have an opportunity to take one course in computing, this course attempts to select significant topics from computer systems, numerical analysis, and programming. As with Perlis's course, a great deal of emphasis is placed on exercises which supply many of the details not covered in the formal classroom instruction.

Forsythe [78] presents a curriculum in Numerical Analysis consisting of a one quarter freshman-sophomore course, a three quarter senior course and he includes brief mention of a three quarter graduate course. He too stresses the importance of programming exercises:

> All of these courses should involve good exercises for the student, both in analysis and in use of an automatic computer. Several exercises should require the marriage of good analysis to imaginative programming. This means that their formulation must depend critically on the experience of the students in analysis and programming, and on the power of the computer to be used and the sophistication of the available languages. Devising good problems is very difficult, and yet it is the central part of preparing these courses. In my experience, the computer problems have usually been too many and too easy. They tended to test the student's facility at syntactical debugging more than his power in devising effective algorithms.[15]

David Young in a critique of the paper suggests a bit more mathematical background for the work in numerical analysis, and an integration of the work in numerical methods and programming.

---

15. George E. Forsythe, "An Undergraduate Curriculum in Numerical Analysis", Communications of the ACM 7, 4 (April 1964), 214.

Korfhage [119] recognizes the special place of logic in the education of a computer scientist and proposes a sequence of four courses in the area; Introduction to Logic and Algorithms, Logical Design, Mathematical Logic, and Computability and Algorithms.

Muller [141] also addresses the problems of introducing logic into the educational program of the computer scientist, but he does it in terms of a sequence in logical design and switching theory. In a critique of this paper Garner, while recognizing the importance of the material, raises a question as to whether the material would be better covered in an interdisciplinary approach.

Gorn [86], with little discussion, outlines a graduate course in mechanical languages.

Atchison and Hamblen [18], in the concluding article of the sequence, consider existing programs in computer science. A survey they conducted in 1963 identified 28 such programs with another 18 anticipated. The programs were identified by a variety of names including:

Computer Science
Systems Engineering
Systems Analysis
Information Processing
Data Processing
Information Science
Information Systems Science
Computer Oriented Mathematics
Applied Sciences
Applied Mathematics[16]

16. William F. Atchison and John W. Hamblen, "Status of Computer Science Curriculum in Colleges and Universities". Communications of the ACM 7, 4 (April 1964), 227.

Computer Science was the most commonly selected and the one felt most appealing to the area of study. It was noted that the programs arose from or were housed in mathematics, engineering (usually electrical but occasionally industrial) or business administration, with the majority associated with mathematics:

> At this point in time, most existing computer programs have come from the first of these, namely mathematics. Most of these programs still have a very strong orientation towards mathematics. Many of them are, in fact, little different from a degree program in numerical analysis where the computer has been used in problem solution. Others have developed to a point where they have a strong emphasis in advanced programming and the development of computer languages. Some are laying down a broader base by including in their preparation of computer oriented personnel such things as logic, Boolean algebra, theory of automata and artificial intelligence. Others have a very strong emphasis on the statistical side.
>
> There are people who contend that after all computer science is nothing but a portion of applied mathematics. There is certainly no question but what the rigorous thought-processes that are required in mathematics are likewise required in the computer field. This is not to say, however, that an excellent mathematician will be a good computer man or conversely. It should be remarked that there are mathematics departments which already have or are planning to incorporate some computer training in their calculus sequence. The view is frequently expressed that some computer science will crystallize out of mathematics, as has statistics.[17]

In September 1967 the COSINE Committee of the Commission on Engineering Education published Computer Sciences in Electrical Engineering [46], recommending an undergraduate course program. The report in condensed form appeared in

17. Ibid., 227.

the March 1968 issue of IEEE Spectrum. Though originally designed as an enhancement of an electrical engineering curriculum, this report has had impact on computer science curricula in general, as has subsequent work of the COSINE Committee which will be reviewed in subsequent chapters. Throughout the work of the COSINE Committee liaison was maintained with C^3S through E. J. McCluskey and William Viavant.

The COSINE Committee did not regard its mission as the same as C^3S, and did not equate electrical engineering and computer science:

> Clearly it would be unreasonable to equate computer sciences with electrical engineering, or to regard it as a subset of the latter. Nevertheless, the close relation between the two is presenting the electrical engineering departments with a special responsibility for the training of the large number of computer engineers and scientists who would be needed by industry, government, business, and educational institutions in the years ahead.[18]

The report then goes on to reflect on how the existence of computer science and computing systems is impacting instruction in electrical engineering:

> The emergence of computer sciences as a highly important field of study, coupled with the growing shift in emphasis in information processing technology from the analog and continuous to the digital and the discrete, is creating an urgent need for a major reorganization of electrical engineering curricula. Such a reorganization must, in the first place, accommodate the needs of students who wish to major in computer

18. COSINE Committee, Computer Science in Electrical Engineering (Commission on Engineering Education, Washington, D. C., September 1967), p. 5.

> sciences within electrical engineering. Second, it must bring into balance the treatment of continuous and digital systems, and provide all electrical engineering students with a background in digital systems comparable to that which they currently acquire in continuous systems. Third, it must result in a much wider and more effective use of the digital computer as a tool for system analysis and design in all engineering courses.[19]

All these areas are addressed in the report, however, the Computer Science Program in Electrical Engineering is of primary interest to this study.

In specifying a computer science program in electrical engineering, the committee first specifies the aims of the program:

> a) It must provide the student with a thorough understanding of computer systems and their use that is based on fundamental principles of long term value rather than the salient facts of contemporary practice.
> b) It must give the student a background in the relevant discrete mathematics (set theory, mathematical logic, and algebra) including familiarity with methods of deduction as applied to abstract models relevant to the field of computation.
> c) It must give the student access to a variety of subjects covering specialized and advanced aspects of computer science.
> d) It must provide the student with sufficient technical and general knowledge so that he can readily broaden his education through continuing study, and remain adaptable to the changing demands of society throughout his professional life.[20]

To meet these objectives a curriculum is suggested by subject areas. The subject areas do not necessarily each imply a three semester hour course:

---

19. Ibid., p. 6.

20. Ibid., p. 9.

Subject Areas for a Computer Science Program in Electrical Engineering[21]

Category A: Basic Subject Areas
- A-1. Programming Principles
- A-2. Computation Structures
- A-3. Introduction to Discrete Mathematics
- A-4. Machines, Languages and Algorithms

Category B: Recommended Elective Subject Areas
- B-1. Digital Devices and Circuits
- B-2. Switching Theory and Logical Design
- B-3. Programming Systems
- B-4. Operating Systems
- B-5. Numerical Methods
- B-6. Optimization Techniques
- B-7. Circuit and System Theory
- B-8. Information Theory and Coding
- B-9. Functional Analysis
- B-10. Combinatorics and Applications
- B-11. Probability and Statistics
- B-12. Symbol Manipulation and Heuristic Programming

Areas A-1, A-2, and A-3 correspond to courses B1, Introduction to Computing, B2, Computers and Programming and B3, Introduction to Discrete Structures of "Curriculum '68", though the descriptions in the COSINE report seem to indicate a somewhat higher level. Area A-4 as described in the report covers much of the same material as courses I7, Sequential Machines, A1, Formal Languages and Syntactic Analysis, and A7, Theory of Computability.

From the elective subject areas B-2 corresponds to course I6 Switching Theory, B-3 to courses I1 Data Structures and I2 Programming Languages, B-5 to the courses B4 Numerical Calculus, I8 Numerical Analysis I and I9 Numerical Analysis II, area B-12 corresponds to A9 Artificial Intelligence and

21. Ibid., p. 10.

Heuristic Programming. The collection of other courses, covering more engineering topics, cover I3 Computer Organization and A2 Advanced Computer Organization.

It is of note then that much of the material recommended in the COSINE report parallels that of "Curriculum '68". The main differences appear in the advanced areas, and in the lack of a course or area corresponding directly to Compiler Construction. The core of the two programs are virtually the same, however the work reported on by the COSINE Committee does not place the area of Data Structures in as fundamental a position as did C3S.

The other report to be considered in this group is the Recommendations of the Undergraduate Mathematics Program for Work in Computing [39], published in May, 1964 by the Panel on Mathematics for the Physical Sciences and Engineering of the Commission on the Undergraduate Program in Mathematics (CUPM).

This program was designed to enhance a mathematics major, and came about from the recognition that many mathematics students are going into some area of computing. This Committee's view of the relationship between mathematics and computer science is specified:

> Out of the solution of such problems as these has emerged a field of study called Computer Science, embracing such topics as numerical analysis, theory of programming, theory of automata, switching theory, etc. Computer Science is closely related to mathematics; indeed, numerical analysis is a branch of that subject,

> while many of the problems arising from computers are intimately associated with questions in combinatorial mathematics, abstract algebra, and symbolic logic. But an even more fundamental relationship also exists. The inherent structure of a computer forces the attendent disciplines to strive for the type of generality, abstraction and close attention to logical detail that is characteristic of mathematical arguments. Research workers in Computer Science must have a knowledge of the spirit and techniques of mathematics. Although they do not need to be mathematicians, they must think like mathematicians.
>
> For these reasons a university or college organization charged with the responsibility for research and training in Computer Science should be closely linked to mathematics within the institution. (It goes without saying that it should also be coordinated with groups charged with providing computation services to the academic community.) It is to the combined group of mathematicians and computer scientists that we direct our report.[22]

The report recommends a three hour introductory course in computer science covering the description of a computer, the description of a programming language, and problem solving. This course would not be a part of the standard mathematics curriculum, but instead would be the base for a technical sequence in computer science. Though not described in as much detail, the course proposed corresponds to course B1, Introduction to Computing, of "Curriculum '68".

With this course specified the report goes on to recommend a program for a mathematics major with an option in computing:

CUPM Proposed Mathematics Program with Option in Computing[23]

A. The introductory course in Computer Science is a Requirement (3 semester hours)

22. Committee on the Undergraduate Program in Mathematics (CUPM), Recommendations of the Undergraduate Mathematics Program for Work in Computing (CUPM, Berkeley, California, May, 1964), pp. 1-2.

23. Ibid., 7.

B. Required mathematics courses
   1. Beginning analysis (12 semester hours)
   2. Linear Algebra (3 semester hours)
   3. Probability and statistics (3 semester hours)
   4. Algebraic structures (3 semester hours)
   5. Advanced calculus (6 semester hours)
   6. Numerical analysis (3 semester hours)

C. Electives (3 semester hours each). A minimum of six semester hours. With the exception of course 8 (Logic), the Introduction to Computer Science is a prerequisite for each of the following:
   7. Numerical analysis
   8. Logic
   9. Information processing
   10. Machine organization
   11. Theory of automata
   12. Advanced programming
   13. Combinatorics
   14. Systems simulation

It can be seen that the program described here, selecting the machine organization and advanced programming electives would roughly cover the core of "Curriculum '68" with Data Structures, course I1, and Systems Programming, course I4 being the notable exceptions.

Summary

"Curriculum '68" may be seen as an expansion and refinement of the 1965 report of C^3S. The earlier report spent a great deal more time on the definition of computer science, its justification as a discipline, and its relationship to other disciplines, especially mathematics. The 1965 report was not the first nor last time these questions were addressed, and further work in these areas are covered by Arsac [13], Gorn [85, 87], Hammer [99, 100], Yovits [189] and Zadeh [190].

The 1965 report of C^3S can be seen as something of an outgrowth of a panel presentation at the 1963 ACM meeting

and reported in 1964 in the Communications of the ACM. Within these articles one finds a definition and specification of computer science; course and program descriptions emphasizing the first course, numerical analysis, logic and logic design; and descriptions of existing programs in higher education in the United States. Within these course and program descriptions emphasis was placed on the role of programming projects and exercises as an integral portion of the programs, and the close ties of computer science education to mathematics is noted.

Two other reports were prepared during the same time frame as the work of C^3S. The COSINE Committee presented a program within electrical engineering that emphasizes subject areas rather than specific courses. CUPM presents a program for mathematicians desiring some exposure to computer science. In both cases the programs recommended, while differing somewhat in detail, contain virtually the same material as found in the undergraduate program of "Curriculum '68". The most noticable exception is that neither the work of COSINE nor CUPM put the emphasis on the data structure area that is found in "Curriculum '68".

Chapter 4 - Relation to Existing Programs

To assess the impact of "Curriculum '68" and the areas requiring attention, several studies of computer science curricula will be considered. The first of these is the "Inventory of Computers in United States Higher Education 1969-70" completed in 1972 by the Computer Science Project of the Southern Regional Education Board, under the direction of John Hamblen. The material cited here is reduced from the summarized raw data of the study. The survey is reported in part in [97] and [98], while [96] reports an earlier similar survey.

In this study, information regarding the organization and resources of university computer centers was gathered. In addition, data were collected on degree programs in computer science, and on computer science course offerings. 770 degree programs in computer science were listed as being offered in 1971-2. These were listed under 13 different names, with only one unidentifiable. 286 or 37.14% of these programs were offered at schools granting the associate degree as their highest degree. The greatest number of programs were listed under the name of computer science (43.34%) with data processing comprising 32.60%, though these were primarily associate degree programs offered at institutions offering at most the associate degree. Table 4-1 summarizes the number of degree granting departments by name.

Computer Science Related Degrees Offered 1971-2 Table 4-1

Department Name	Degree				
	Associate	Bachelor's	Master's	Doctorate	Total
Computer Engineering	0	2	1	1	4
Computer Programming	41(37)	7	0	1	49(37)
Computer Science	42(29)	133	100	59	334(29)
Computer Technology	22(11)	9	3	2	36(11)
Data Processing	219(205)	26(1)	5	1	251(206)
Information and Control Science	0	2	3	2	7
Information Processing	0	1	0	0	1
Information Science	0	5	10	6	21
Information Systems	7(3)	10	6	4	27(3)
Systems Analysis	1	3	3	1	8
Systems Engineering	0	8	12	7	27
Systems and Information Science	0	0	1	1	2
Systems Science	0	1	0	1	2
Unknown	0	1	0	0	1
Total	332(285)	208(1)	144	86	770(286)

note - parenthesized quantities refer to institutions offering at most the associate degree

Within these programs, 9935 degrees were awarded in the academic year 1970-71. 5151 or 51.85% of these degrees were awarded by institutions offering at most the associate degree. 56.51% of the degrees awarded were associate degrees, 25.35% bachelor's, 15.54% master's, 2.60% doctorates. Degrees awarded in 1970-71 are given in Table 4-2.

In terms of enrollment in 1969-70, these programs accounted for a total of 55,999 students; 33,447 or 59.73% of these were enrolled at institutions offering at most the associate degree. 87.84% of the students were undergraduates, while 12.16% were graduate students. These figures refer to students enrolled in the respective majors. Enrollment figures for 1969-70, by departmental title are given in Table 4-3.

The SREB Study considers courses offered in computer science areas, whether they were part of a degree program or not. The following descriptors were used for the identification of courses:[1]

Computer Appreciation
Problem Solving in the Humanities
Introductory Course in Some Compiler Language such as FORTRAN, PL/I, BASIC, ALGOL
Introductory Course in a Language such as COBOL
Problem Solving Course in Engineering or Scientific Applications
Introductory Course in Programming
Advanced Course in Programming
First Course in Computer Design
Advanced Course in Computer Design (Computer Architecture)
Programming Languages (Structure, etc.)
Operating Systems (Systems Programming, etc.)
Data Structures
Automata Theory

1. From the instructions for the survey forms.

Degrees awarded in Computer Science and Related Programs 1970-71 Table 4-2

Department	Degree				
	Associate	Bachelor's	Master's	Doctorate	Total
Computer Engineering	0	11	3	0	14
Computer Programming	819(697)	32	0	0	851(697)
Computer Science	435(354)	1537	1249	198	3419(354)
Computer Technology	295(125)	88	9	2	394(125)
Data Processing	3971(3870)	395(41)	19	2	4387(3911)
Information and Control Science	0	10	3	14	27
Information Processing	0	0	0	0	0
Information Science	0	78	66	16	160
Information Systems	94(64)	133	43	6	276(64)
Systems Analysis	0	31	27	0	58
Systems Engineering	0	149	98	14	261
System and Information Science	0	0	27	6	33
Systems Science	0	48	0	0	48
Unknown	0	7	0	0	7
Total	5614(5110)	2519(41)	1544	258	9935(5151)

note - parenthesized quantities refer to institutions offering at most the associate degree

Enrollment in Computer Science Related Degree Programs 1969-70 Table 4-3

Department	Undergraduate	Graduate
Computer Engineering	85	9
Computer Programming	4786(3868)	63(44)
Computer Science	10909(2428)	4725
Computer Technology	2791(1009)	53
Data Processing	27712(25551)	159(46)
Information and Control Science	0	585
Information Processing	0	0
Information Science	546	240
Information Systems	1198(501)	241
Systems Analysis	120	132
Systems Engineering	893	402
Systems and Information Science	0	199
Systems Science	75	0
Unknown	76	0
Total	49191(33357)	6808(90)

note - Parenthesized quantities refer to institutions offering at most the associate degree

Simulation Applications (non-econometric)
Simulation Languages and/or Techniques
Artificial Intelligence (Heuristic Programming)
Hybrid Computers
Discrete Structures
Computer Organization
Switching Theory
Sequential Machines
Compiler Construction
Formal Languages and Syntactic Analysis
Introductory Computer Organization
Advanced Computer Organization
Information Storage and Retrieval
Computer Graphics
Computability
Introductory Numerical Analysis (or methods)
Advanced Numerical Analysis
Introductory Operations Research
Advanced Operations Research
Data Communications
Process Control
Data Preparation (Keypunch, verifier, data entry, ...)
Unit Record (EAM)
Introductory Course in Data Processing
Advanced Course in Data Processing
Problem Solving Course in Business Applications
Advanced Business Applications (Marketing, Finance, Production, etc.)
Introductory Systems Analysis (Business)
Large Scale Information Processing Systems
File Organization
Data Base Management
Information Systems Design (Introductory)
Information Systems Design (Advanced)
Management of Computer Centers and of Information Processing Departments
Management Information Systems
Computers and Accounting (Including Auditing)
Econometric Simulation Applications
Other
Description not coded.

A variety of information was gathered for each of the courses listed under these descriptions. This data includes the following: number of times offered in 1969-70, 1970-71, and 1971-72; the requirement status of required for majors in computer science, required for minors in computer science, service course for science majors, service course for

non-science majors; the mode of operation of the course; average section size for the course; enrollment for 1969-70; and the cost of computing services for the course.

9971 courses were offered or on the books in 1971-2. These courses enrolled a total of 554,499 students in the 1969-70 school year. Of these courses 2838 or 28.46% were offered at institutions offering at most the associate degree, and courses at these institutions enrolled 177,230 or 31.96% of the students in 1969-70.

Table 4-4 presents the data for the courses of "Curriculum '68". Table 4-5 presents the data for courses which may best be classified as service courses. Table 4-6 presents the data for courses which may be classified as data processing courses. Table 4-7 presents the data for courses falling into none of the above classifications.

Courses comparable to those of "Curriculum '68" comprised 52.11% of the courses listed, and enrolled 52.81% of the students enrolled in computer related courses in 1969-70. 18.65% of the courses could be grouped under the category of service courses, and these enrolled 17.34% of the students. 82.00% of the courses were listed as either 3-4 quarter hours or 3-4 semester hours credit. Only 9.38% of the responses were classified as "other" or "description not coded".

In Fall 1971, Engel [71] conducted a survey on the impact of "Curriculum '68" for C^3S. Concentrating on established programs, the questionaire was circulated to the approximately

Course Data for Courses of "Curriculum '68" Table 4-4

Course "Curriculum '68"	Offered as on books 71-72	Required Major CS	Required Minor CS	Service Science	Service Non Science	Enrollment 69-70
B1. Introduction to Computing[1]	1778 (633)	781 (412)	80 (12)	487 (91)	414 (63)	188945 (41289)
B2. Computers and Programming[2]	728 (311)	443 (271)	31 (1)	146 (18)	77 (9)	27897 (10830)
B3. Introduction to Discrete Structures	71 (1)	45 (1)	6	13	3	1441 (80)
B4. Numerical Methods[3]	405 (38)	173 (33)	27	162 (3)	25	12882 (1645)
I1. Data Structures	146 (8)	91 (8)	14	20	14	4190 (480)
I2. Programming Languages	239 (56)	139 (54)	24	49 (2)	19	8667 (2128)
I3. Computer Organization[4]	221 (34)	135 (29)	21	31 (4)	12 (1)	9615 (2382)
I4. Systems Programming[5]	326 (92)	222 (87)	19 (1)	44 (1)	19 (3)	11773 (4040)
I5. Compiler Construction	135 (7)	83 (7)	5	26	5	2880 (169)
I6. Switching Theory	121	66	2	36	3	5219
I7. Sequential Machines	36	14	2	12	0	862
I8.-I9. Numerical Analysis I & II[6]	267 (11)	115 (11)	10	96	12	4187 (230)
A1. Formal Languages and Syntactic Analysis	107 (8)	66 (7)	0	15 (1)	11	1681 (125)

Table 4-4 (continued)

Course "Curriculum '68"	Offered as on books 71-72	Required Major CS	Required Minor CS	Service Science	Service Non Science	Enrollment 69-70
A2. Advanced Computer Organization	69 (5)	40 (5)	3	7	6	1746 (70)
A3. Analog and Hybrid Computing[7]	46 (1)	16 (1)	1	22	0	956 (16)
A4. System Simulation[8]	209 (6)	89 (6)	6	56	31	4588 (165)
A5. Information Organization and Retrieval[9]	76 (1)	32 (1)	2	16	21	1549 (27)
A6. Computer Graphics	46 (4)	13 (2)	3	17 (1)	3	851 (71)
A7. Theory of Computability	58	33	1	14	2	858
A8. Large Scale Information Processing Systems	35 (5)	15 (4)	0	12	1	553 (196)
A9. Artificial Intelligence and Heuristic Programming	77 (1)	37	2	16 (1)	3	1479 (8)

notes - 1. Course B1 did not have an equivalent in the survey descriptors. This is a combination of "Introductory Course in some compiler language such as FORTRAN, PL/I, BASIC, ALGOL" and "Introductory Course in Programming".
2. Course B2 did not have an equivalent in the survey descriptors. This is for "Advanced Course in Programming".
3. Listed under "Introductory Numerical Analysis (or Methods)".
4. Combined "Computer Organization" and "Introductory Computer Organization".

Table 4-4 (continued)

5. Listed under "Operating Systems (Systems Programming etc.)".
6. Not listed as two courses only "Advanced Numerical Analysis".
7. Listed under "Hybrid Computers".
8. Course A4 did not have an equivalent in the survey descriptors. This is a combination of "Simulation Applications (non-econometric)", and "Simulation Languages and Techniques".
9. Listed under "Information Storage and Retrieval".

Parenthesized quantities refer to material on institutions offering at most the associate degree.

Course Data for Service Courses — Table 4-5

Course Descriptor	Offered as on books 71-72	Required Major CS	Required Minor CS	Service Science	Service Non Science	Enrollment 69-70
Computer Appreciation	208 (56)	50 (21)	10 (3)	26 (1)	114 (29)	18112 (6350)
Problem Solving in Humanities	88 (1)	8	1	21	55 (1)	2318 (41)
Introductory Course in a language such as COBOL	456 (273)	310 (247)	7	33 (6)	100 (18)	27410 (16137)
Problem Solving Course in Engineering or Scientific Applications	456 (71)	91 (32)	7	297 (36)	41 (3)	18664 (2364)
Problem Solving Course in Business Applications	228 (115)	129 (96)	6 (1)	16 (5)	71 (11)	12212 (4102)
Advanced Business Applications (Marketing, Finance, Production, etc.)	95 (30)	48 (27)	2	10	25 (2)	3752 (739)
Introductory Systems Analysis (Business)	257 (154)	196 (145)	6	8	44 (9)	9987 (5534)
Computers and Accounting (including auditing)	46 (12)	15 (8)	4	2	23 (4)	2377 (411)
Econometric Simulation Applications	26 (3)	7 (3)	0	7	11	1320 (75)

note - Parenthesized quantities refer to institutions offering at most the associate degree

Course Data for Data Processing Courses — Table 4-6

Course Descriptor	Offered as on books 71-72	Required Major CS	Required Minor CS	Service Science	Service Non Science	Enrollment 69-70
Data Preparation (keypunch, verifier, data entry,...)	65 (56)	31 (27)	2 (2)	6 (6)	27 (22)	6358 (3615)
Unit Record (EAM)	196 (151)	144 (126)	2 (2)	4 (3)	44 (19)	14889 (12165)
Introductory Course in Data Processing	457 (246)	246 (190)	12 (2)	25 (6)	191 (51)	77538 (47108)
Advanced Course in Data Processing	109 (42)	61 (31)	3	18 (3)	24 (5)	5399 (2148)

note - Parenthesized quantities refer to institutions offering at most the associate degree

Course Data for Other Courses

Table 4-7

Course Descriptor	Offered as on books 71-72	Required Major CS	Required Minor CS	Service Science	Service Non Science	Enrollment 69-70
First Course in Computer Design	135 (15)	73 (13)	10 (1)	43 (1)	6 (1)	5697 (823)
Advanced Course in Computer Design (Computer Architecture)	88 (1)	43 (1)	6	22	3	2515 (50)
Automata Theory	131 (1)	75	5	31	2 (1)	2121 (250)
Advanced Computer Organization	69 (5)	40 (5)	3	7	6	1746 (70)
Introductory Operations Research	140 (13)	59 (12)	6	50 (1)	22	6687 (407)
Advanced Operations Research	77 (3)	32 (2)	4	29 (1)	6	1307 (94)
Data Communications	44 (11)	21 (7)	3	14 (2)	5 (2)	1406 (344)
Process Control	39 (6)	18 (6)	1	16	2	959 (212)
File Organization	24 (8)	17 (8)	2	3	1	615 (192)
Data Base Management	18 (5)	10 (5)	0	1	5	682 (92)
Information Systems Design (Introductory)	140 (65)	101 (64)	3	12	22 (1)	4408 (2302)
Information Systems Design (Advanced)	92 (40)	69 (38)	1	9	12 (1)	1957 (893)

Table 4-7 (continued)

Course Descriptor	Offered as on books 71-72	Required Major CS	Required Minor CS	Service Science	Service Non Science	Enrollment 69-70
Management Information Systems	103 (23)	49 (22)	4	11	37 (1)	4195 (589)
Management of Computer Centers or Information Processing Departments	53 (29)	40 (27)	1	3 (1)	7 (1)	1614 (688)

note - Parenthesized quantities refer to institutions offering at most the associate degree

50 institutions offering a Ph. D. in Computer Science which appeared on the so called "Forsythe List". Of these twenty-six responded. The material requested concentrated primarily on the relationship of existing courses to "Curriculum '68", and by no means attempted to determine the broad view of computing in American Colleges and Universities that was attempted in the Hamblen study.

The first area in which response was requested was in a course by course comparison of the curriculum of the schools to "Curriculum '68". This data is shown in Table 4-8.

Twenty-one responses were given to the question of whether or not the prerequisite structure shown in "Curriculum '68" is adequate. Nineteen responded that it was, while only two indicated it was not, with one of these saying the program was "too heavily oriented to numerical prerequisites".

Comments were requested for suggestions for change in "Curriculum '68". Two were negative:

> "Not suitable for our courses which rather divide the subject differently."
> "The ACM is totally unimaginative, I do not like it."[2]

The other comments received supplied more specific suggestions for change:

> "A1 (watered down) should be a prerequisite for I5."
> "One year of calculus before starting the computer science major."
> "A few of the definite prerequisites could be replaced by desirable prerequisites."

2. Gerald L. Engel, "Input From ACM Curriculum Committee on Computer Science", SIGCSE Bulletin 3, 4 (December 1971), 31.

Relation of Courses in Existing Programs to Courses of "Curriculum '68"[3] Table 4-8

Course	Offered as Specified	Similar Course Offered	Not Offered	Comments
B1. Introduction to Computing	9	16	1	
B2. Computers and Programming	8	15	3	
B3. Introduction to Discrete Structures	6	13	7	Two had course available in mathematics, one had course available in EE
B4. Numerical Calculus	7	13	6	Three had course available in mathematics
I1. Information Structures	9	15	2	
I2. Programming Languages	5	17	4	One planned
I3. Computer Organization	5	17	4	One planned, one had course available in EE
I4. Systems Programming	4	20	2	
I5. Computer Construction	8	18	0	
I6. Switching Theory	6	8	12	Eight had course available in EE
I7. Sequential Machines	4	12	10	Three had course available in EE, one planned
I8. Numerical Analysis I	8	15	3	Three had course available in mathematics

3. Ibid., 30-31.

Table 4-8 (continued)

Course	Offered as Specified	Similar Course Offered	Not Offered	Comments
I9. Numerical Analysis II	7	15	4	Three had course available in mathematics
A1. Formal Languages and Syntactic Analysis	5	18	3	
A2. Advanced Computer Organization	2	18	6	One had course offered in EE
A3. Analog and Hybrid Computing	1	4	21	Seven had course offered in EE, four did not have necessary equipment, one did not have necessary staff
A4. System Simulation	1	12	13	One had course offered in EE, two had course offered in operations research, one had course offered in IE, four did not have necessary staff
A5. Information Organization and Retrieval	4	13	9	Two did not have necessary staff, one had course offered in Library School
A6. Computer Graphics	3	7	16	One had course offered in EE, two did not have necessary equipment, two did not have necessary staff
A7. Theory of Computability	1	24	1	

Table 4-8 (continued)

Course	Offered as Specified	Similar Course Offered	Not Offered	Comments
A8. Large-Scale Information Processing Systems	1	9	16	Two planned, two did not have necessary staff
A9. Artificial Intelligence and Heuristics	5	16	5	One had course offered in EE

"I1 should precede I2, I1 is not needed for A1."
"Do not need B1 as a prerequisite for B3."
"Delete B3. It is not clear why this should be a specific prerequisite to any course. We haven't needed I6 and I7."
"See no reason to have I1 as a required prerequisite of A1, even though it may be desirable."
"A5 should concentrate on file organization, it would then require I4 as a prerequisite."[4]

Of the twenty-six institutions, fourteen offered undergraduate degrees in computer science. All fourteen of the institutions required course B1 for an undergraduate major, and all but one required B2. The rest of the courses listed in the "core" of "Curriculum '68" were required in at least half the program with B3 required at eight institutions, B4 required at nine institutions, I1 required at eight institutions, I2 required at ten institutions, I3 required at seven institutions and I4 required at eight institutions. Of this eight course core only one institution required all eight, and only one required just one of the courses. Two required four, two required five, five required six, and two required seven. The other courses listed in "Curriculum '68" appeared as requirement to a lesser degree: I5 appeared four times, I6 four times, I7 once, I8 four times, I9 twice, A1 twice, A3 once, A4 once, A5 once, A8 once, and A9 once. Only A2 - Advanced Computer Organization, A6 - Computer Graphics, and A7 - Theory of Computability failed to appear as undergraduate requirements.

"Curriculum '68" suggested courses in calculus, differential

4. Ibid., 31.

equations, linear algebra, and probability. Twelve of the institutions required calculus for an undergraduate major, eight required differential equations, nine required linear algebra, and five required probability. Various other mathematics was required; three required statistics, one logic, and two advanced calculus. Six required advanced algebra, but of these six, four did not require course B3 in the requirements for the major.

All of the institutions recommended some form of technical electives for computer science majors. The most common situation, occurring at six institutions, was for this matter to be determined by the student and his advisor. Mathematics was recommended as a technical elective at four institutions, statistics at one, electrical engineering at five, industrial engineering at two, philosophy at one, physics at two, business at two, economics at one, additional computer science at four, and linguistics at three.

The question was posed as to what programming languages must be learned in the undergraduate program. One institution responded it was not important. Of the other thirteen, approximately four languages were required as an average (this includes assembly and machine language). Thirteen required FORTRAN, ten an assembler, eight ALGOL, five SNOBOL, five PL/I, three COBOL, two BASIC, one APL, one LISP, and one a machine language.

Twenty-three of the institutions indicated they offered

service courses. Three departments offered only one service course, six departments offered two, six offered three, six offered four, one offered seven, and one offered eleven.

Among service courses offered were the following titles:

Survey of Computer Science
Introduction to Computer Science
Algorithmic Approach to Computing
Computers and Society
Introduction to Computing for Business Students
Non-numerical Programming
Hybrid Computing
Higher Level Languages and Their Applications
Machine Organization and Programming
Programming for Engineers
Programming for the Biological and Agricultural Sciences
Programming for Students without Mathematical Background
Numerical Methods for Engineers
Programming for Social Sciences
Introduction to Computing for the Humanities
Information Storage and Retrieval
Data Processing[5]

Finally, questions were raised about the master's program. Twenty-five institutions responded to these inquiries.

In general, admissions to the master's program required a bachelor's degree with a B average. Twelve institutions listed specific computer science courses for entrance, with one institution requiring a major. Of those not requiring a major, the requirements ranged from two to five computer science courses. Thirteen institutions made specific mathematics requirements, the usual being a calculus sequence, linear algebra, and selected advanced courses.

While the above represents the minimum entrance requirements,

5. Ibid., 36.

the question was also raised as to what the "usual" background was for students entering the program. The "usual" background indicated was a bachelor's degree in mathematics, electrical engineering, or one of the physical sciences. Four institutions indicated a bachelor's degree in computer science was "normal" and almost all the institutions indicated at least two courses in computer science was "usual".

A listing of the core requirements for the master program was also requested. The answers were too diverse for statistical interpretation, and are individually listed in table 4-9.

Walker [181] conducted a survey of computer science departments in the Fall of 1972. Questionaires were circulated to 493 schools in the United States and Canada asking about various aspects of the computer science curricula. Approximately 60% responded with 158 responding to questions on the undergraduate program, 95 on the masters program and 58 on doctoral programs.

To give some perspective to the results the type of institution was asked on the questionaire:

type of institution[6]	% responding
junior college	1
vocational school	0
four year college	15
College offering a limited amount of graduate work	18
University	66

6. Terry M. Walker, "Computer Science Curricula Survey", SIGCSE Bulletin 5, 4 (December 1973), 28.

Core Courses of Master's Program[7] Table 4-9

Institution	Requirements
1	No specific required courses
2	Programming languages, numerical analysis, computer design, logic and computational theory, and systems
3	Varies
4	Machine organization, introduction to sequential machines, advanced programming techniques, numerical analysis, non-numerical algorithms
5	Hardware, numerical analysis, computers and programming, switching theory
6	No specific core
7	Artificial intelligence, fundamentals of computer mathematics, theory of computation, programming systems, computer systems
8	Numerical analysis, theory of computation, data structures, systems programming, compiler construction
9	Undergraduate program plus systems, computer languages and information processing
10	Numerical analysis, computer organization, introduction to linguistics, information storage and retrieval, systems programming, engineering, psychology
11	Compilers and computer languages, algorithm specification, computability, survey of computer algorithms
12	None listed
13	Numerical solution of differential equations, numerical linear algebra, advanced language structure and theory, advanced hardware concepts
14	Programming systems, logic and computability, switching theory, compiler design, systems theory, operations research, evaluation of computer systems

7. Gerald L. Engel, "Input From ACM Curriculum Committee on Computer Science", SIGCSE Bulletin 3, 4 (December 1971), 35-36.

Table 4-9 (continued)

Institution	Requirements
15	Switching theory, computer organization, compiler construction, theory of computation, numerical analysis
16	Data structures, programming languages, computer organization, theory of computability, numerical analysis
17	None listed
18	Modern algebra for engineering, numerical analysis, programming systems design, theory and design of digital machines
19	Switching theory, sequential machines, formal languages, systems programming, advanced computer organization, programming languages, compiler construction
20	Programming, structures, automata, artificial intelligence
21	Data representation and manipulation, data processing and file management, programming systems, software engineering laboratory, numerical analysis
22	Two of four courses in three of the following areas: numerical analysis, language and information processing, computer systems, theory of computing
23	Continuous systems, discrete systems, optimization and evaluation, structure of algorithmic languages, computer systems
24	Data structures, programming languages, compiler construction, operating systems, information retrieval, theory of computing, numerical analysis
25	Computer systems and organization, language translation and compiler construction, list processing and string manipulation languages

The respondents were also asked to reflect on the usefulness of the reports of the ACM Curriculum Committee:

Usefulness of C^3S reports[8]	% responding
Very helpful	34
Moderately helpful	44
Not very helpful	9
Not considered	14

In regards to courses information was requested on the following titles. Designations in parentheses are the corresponding courses of "Curriculum '68", with M referring to mathematics courses recommended:[9]

Computer Appreciation
Introduction to Data Processing
Introduction to Computer Science (B1)
Calculus (M)
Differential Equations (M)
Linear Algebra (M)
Discrete Structures (B3)
Probability and Statistics (M)
Operations Research
Machine and/or Assembly Language (B2)
Data Structures (I1)
Computer Organization and Design (I3)
Programming Languages (I2)
Numerical Analysis (I8)
Advanced Numerical Analysis (I9)
Compiler Construction (I5)
Information Theory and Coding
Computers and Society
Operating Systems and Design (I4)
Information Retrieval
History of Computer Science
Time-sharing Systems Design
Symbol Manipulation and Text Processing
Simulation and Modeling (A4)
Computer Graphics (A6)
Process Control
Real-time Systems Design
Data Processing and File Management
Computer Center Administration

8. Terry M. Walker, "Computer Science Curricula Survey", SIGCSE Bulletin 5, 4 (December 1973), 28.

9. Ibid., 20.

Legal Aspects of Computers
Artificial Intelligence and Heuristics (A9)
Theory of Computability (A7)
Sequential Machines (I7)
Formal Languages (A1)
Large-Scale Information Processing Systems (A8)
Switching Theory (I6)
Analog and Hybrid Computers (A3)
Programming Techniques
Management Information Systems

Under each course area heading the question was asked if the course area was (1) required, (2) allowed (but not required), (3) not allowed, or (4) no courses offered. The bachelor's, master's, and doctoral programs were considered separately. Since these descriptions referred to "course areas", a one-to-one correspondence to the courses of "Curriculum '68" is, of necessity, somewhat rough. Nothing clearly identifiable as Numerical Calculus (B4) or Advanced Computer Organization (A2) appeared in the list. Table 4-10 presents the course data for courses similar to those of "Curriculum '68". Table 4-11 presents the data for mathematics courses. Table 4-12 presents course data, for additional courses considered in the survey. The programming languages required at the various levels are summarized in Table 4-13.

The survey asked what service courses were offered by the responding departments. This information is given in Table 4-14.

A list of possible objectives was given for the various programs, and the respondents were asked to rank them in order of importance. For the undergraduate program the

"Curriculum '68" Courses[10] Table 4-10

Course	Undergraduate (%)				Master's (%)				Doctoral (%)			
	Required	Elective	Not Allowed	Not Offered	Required	Elective	Not Allowed	Not Offered	Required	Elective	Not Allowed	Not Offered
B1. Introduction to Computer Science	77	7	3	14	17	11	55	17	13	18	50	20
B2. Machine and/or Assembly Language	82	13	1	5	41	31	25	3	33	39	25	4
B3. Discrete Structures	43	34	1	22	30	45	12	13	23	53	16	9
I1. Data Structures	58	22	0	21	41	47	8	4	33	56	7	4
I2. Programming Languages	72	25	0	3	43	47	6	4	40	54	4	2
I3. Computer Organization and Design	54	30	0	15	45	48	3	3	39	53	7	2
I4. Operating Systems Design	31	45	1	24	30	64	0	6	30	70	0	0
I5. Compiler Construction	20	53	1	25	22	75	0	3	30	70	0	0
I6. Switching Theory	19	36	1	44	11	73	2	14	15	76	4	5
I7. Sequential Machines	12	40	1	48	13	72	0	15	21	71	0	7

10. Ibid., 20, 22, 24.

Table 4-10 (continued)

Course	Undergraduate (%)				Master's (%)				Doctoral (%)			
	Required	Elective	Not Allowed	Not Offered	Required	Elective	Not Allowed	Not Offered	Required	Elective	Not Allowed	Not Offered
I8. Numerical Analysis	47	47	1	5	28	65	7	0	23	67	11	0
I9. Advanced Numerical Analysis	7	70	2	21	10	87	0	3	16	82	0	2
A1. Formal Languages	18	50	1	31	25	66	0	9	30	63	0	7
A3. Analog and Hybrid Computers	5	36	1	59	4	53	0	43	4	66	0	30
A4. Simulation and Modeling	13	60	1	26	14	74	0	13	11	81	0	9
A5. Information Retrieval	14	45	0	41	7	66	0	28	11	67	0	23
A6. Computer Graphics	1	32	1	66	5	49	0	46	4	63	0	34
A7. Theory of Computability	7	38	1	54	17	64	0	18	28	65	0	7
A8. Large-Scale Information Processing Systems	5	33	1	61	5	56	0	39	5	65	0	29
A9. Artificial Intelligence and Heuristics	1	40	1	57	6	80	0	14	7	86	0	7

Mathematics Courses[11]

Table 4-11

Course Area	Undergraduate (%)				Master's (%)				Doctoral (%)			
	Required	Elective	Not Allowed	Not Offered	Required	Elective	Not Allowed	Not Offered	Required	Elective	Not Allowed	Not Offered
Calculus	80	16	1	3	29	24	41	6	21	31	40	5
Differential Equations	44	50	1	4	16	52	29	2	16	51	29	4
Linear Algebra	55	41	1	3	31	49	17	3	32	46	20	2
Probability and Statistics	53	46	0	1	25	68	6	1	19	68	9	4

11. Ibid.

Additional Courses[12] Table 4-12

Course Area	Undergraduate (%)				Master's (%)				Doctoral (%)			
	Required	Elective	Not Allowed	Not Offered	Required	Elective	Not Allowed	Not Offered	Required	Elective	Not Allowed	Not Offered
Computer Appreciation	14	19	12	55	5	11	37	48	2	9	38	51
Introduction to Data Processing	38	21	9	32	12	17	51	20	7	20	47	25
Operations Research	20	59	0	21	14	72	3	10	5	81	5	9
Information Theory and Coding	9	48	1	43	8	69	0	23	11	82	0	7
Computers and Society	2	40	3	55	2	27	18	53	0	30	15	56
History of Computer Science	11	11	0	78	3	8	6	83	0	11	5	84
Time-sharing Systems Design	7	37	0	57	9	57	0	33	5	70	0	25
Symbol Manipulation and Text Processing	10	37	0	53	6	64	1	29	11	73	2	14
Process Control	1	24	0	75	1	38	0	61	4	47	0	49
Real-time Systems Design	6	29	0	66	5	52	0	43	4	60	0	36
Data Processing and File Management	21	51	1	27	10	59	1	29	11	63	0	27
Computer Center Administration	5	18	0	76	1	25	1	73	2	22	0	76

12. Ibid.

Table 4-12 (continued)

Course Area	Undergraduate (%)				Master's (%)				Doctoral (%)			
	Required	Elective	Not Allowed	Not Offered	Required	Elective	Not Allowed	Not Offered	Required	Elective	Not Allowed	Not Offered
Legal Aspects of Computers	0	9	0	91	2	8	1	88	4	9	0	88
Programming Techniques	47	32	0	21	17	48	5	30	16	54	7	23
Management Information Systems	15	42	1	42	4	54	1	40	2	53	0	45

Programming Languages In Which Proficiency Is Expected Upon Graduation[13] Table 4-13

Language	Undergraduate (%)	Master's (%)	Doctoral (%)
Some Machine or Assembly Language	80	77	78
ALGOL	22	43	48
BASIC	24	22	19
FORTRAN	90	82	86
COBOL	54	25	17
PL/I	36	37	36
APL	15	21	19
MAD	1	0	0
GPSS	14	16	7
LISP	8	22	31
SNOBOL	15	25	34
SIMSCRIPT	4	6	3
Others	17	9	9

13. Ibid., 21, 22, 24.

Service Courses[14]

Table 4-14

Course	Offering (%)
Computer Appreciation	40
Introductory Computer Science	78
FORTRAN Programming	84
COBOL Programming	48
APL Programming	11
PL/I Programming	24
ALGOL Programming	8
BASIC Programming	29
Computers and Society	28
Machine and/or Assembly Language Programming	59
Other	23

14. Ibid., 25.

following ranking occurred from highest to lowest:[15]

1) Prepare a person for a job as a systems analyst
2) Prepare a person to pursue a graduate degree in computer science
3) Prepare a person for a job as a scientific programmer
4) Prepare a person for a job designing computer software systems (tied with 5)
5) Prepare a person for a job as a commercial programmer (tied with 4)
6) Prepare a person for a job teaching computer science at the secondary school level (tied with 7)
7) Prepare a person for a job as a data processing manager (tied with 6)
8) Prepare a person for a job designing computer hardware systems

In terms of objectives for the master's program, the ranks were as follows, from highest to lowest:[16]

1) Prepare a person for a job designing computer software systems
2) Prepare a person for a job as a systems analyst
3) Prepare a person to pursue a doctoral degree in computer science
4) Prepare a person for a job as a scientific programmer
5) Prepare a person for a job teaching computer science at the college or junior college level
6) Prepare a person for a job as a commercial programmer
7) Prepare a person for a job designing computer hardware systems
8) Prepare a person for a job as a data processing manager
9) Prepare a person for a job teaching computer science at the secondary school level

For the doctoral programs there was the following ranking, from highest to lowest:[17]

1) Prepare a person for a job teaching computer science at the college or university level
2) Prepare a person for a position conducting full-time computer research
3) Prepare a person for a job designing computer software systems

15. Ibid., 19-20.

16. Ibid., 21.

17. Ibid., 23.

4) Prepare a person for a job as a systems analyst
5) Prepare a person for a job designing computer hardware systems
6) Prepare a person for a job as a scientific programmer
7) Prepare a person for a job as a data processing manager
8) Prepare a person for a job as a commercial programmer

An additional survey of computer science curricula was conducted by Vickers [179]. This data is quite difficult to analyze and adds little to the information in the studies referenced. Additional information is also available in descriptions of specific programs found in department brochures and in papers such as Barnes and Gotterer [23], Bauer [26], Cowan and Roden [56], Gorsline and Green [88], Mapp [126], Mathis and Kerr [131], Rahimi and Hedges [151], Roth [157], Schwenkel [162], and Semple [165]. These papers and descriptions, however, do not give the overall perspective and comparison needed in this study, and the three surveys cited remain the fundamental sources regarding curriculum implementation.

While the studies we have considered are concerned with what is offered in a curriculum, they do not evaluate programs. To do this three surveys of graduates of computer science programs will be considered. Though this is not an exhaustive evaluation of graduates of such programs, it does represent all that has been published regarding this aspect of computer science education.

Barnes and Gotterer [24] in 1971 reported several statistics relating to students having completed a bachelor's degree at The Pennsylvania State University. This work reflected on

the professional activities of the 35 respondents to the questionaire. These individuals are graduates of the program from its inception in 1967 to 1970. Of particular importance is the ranking of courses taken as to their value to the professional. Each course was to be ranked from 0 - 9 with 9 very valuable and 0 almost useless. These results are reported in Table 4-15.

Of interest also is that 34 of the respondents indicated that if the opportunity again presented itself they would major in Computer Science at the Pennsylvania State University.

A companion study was conducted by Gotterer and Barnes [89] on graduates of the master's program at The Pennsylvania State University for the years 1966 - 1972. Responses were obtained from 70 of the 136 graduates of this time period. The course rating scheme used in the study of the bachelor's program was also used in this study and is reported in Table 4-16. "Curriculum '68" equivalents are not included since the higher level courses do not have clear equivalences. Again the rating was 0 useless and 9 most useful.

Some additional questions are of interest in explaining the results of the ratings. The respondents were asked for reasons for considering a course valuable:[18]

18. Malcolm H. Gotterer and Bruce H. Barnes, "The Computer Science M. S. Graduate", SIGCSE Bulletin 5, 1 (February 1973), 109.

Undergraduate Course Rankings[19] Table 4-15

Course Title	Approximate "Curriculum '68" Equivalent	Number Taking	Average Rating
Introduction to Programming	B1	32	6.6
Assembly Programming	B2	32	7.3
Numerical Calculus	B4	14	3.6
Introduction to Data Processing	Not Applicable	5	6.8
Information Structures	I1	30	6.6
Systems Programming	I4	32	8.0
Structure of Programming Languages	I2	32	5.5
Graph Theory	Not Applicable	21	3.0
Numerical Computation	I8	26	3.0
Matrix Computation	I9	13	2.8
Logic and Computability	Not Applicable	10	3.5

19. Bruce H. Barnes and Malcolm H. Gotterer, "Attributes of Computer Professionals", in Theodore C. Willoughby (ed.), Proceedings of the Ninth Annual Computer Personnel Research Conference (ACM, New York, 1971), pp. 172, 174.

Master's Course Rankings[20] Table 4-16

Course	Number of Responses	Average Rating
Information Structures	29	7.31
Systems Organization and Programming	49	7.69
The Structure of Programming Languages	41	6.12
Combinatorics and Graph Theory	27	4.14
Information Theory and Error-Correcting Codes	8	4.87
Numerical Computations	55	3.31
Matrix Computations	61	3.16
Logic and Computability	25	3.96
Mathematical Machine Theory	16	4.00
Theory of Automata	33	3.78
Algebraic Theory of Automata	10	3.70
Structure of Artificial Languages	57	6.14
Systems Programming	59	7.06
Nonarithmetic Programming	54	5.77
Information Processing Systems	58	5.91
Theory of Graphs and Networks	15	4.33
Information Retrieval	13	5.89
Numerical Analysis I	10	4.60
Numerical Analysis II	8	4.62
Combinatorial Systems	2	7.00

20. Malcolm H. Gotterer and Bruce H. Barnes, "The Computer Science M. S. Graduate", SIGCSE Bulletin 5, 1 (February 1973), 108.

Table 4-16 (continued)

Course	Number of Responses	Average Rating
Approximation Theory	4	2.22
Numerical Solution of Partial Differential Equations	5	4.40
Algebraic Coding Theory	3	3.33
Theory of Formal Languages and Automata I	11	5.36
Theory of Formal Languages and Automata II	6	4.50

Reason	No. of Responses
Can use in Present Job	41
Good Foundation or Background	16
Course was "realistic"	5
Personal Interest	4
Gave Insight into Computing	4
Permitted Research Opportunity	3
Liked Professor	3
Other	9

The respondents were also asked to give reasons for negative responses:[21]

Reason	No. of Responses
Not Practical or Useable	44
Too Theoretical	10
Lack of Interest	9
No Relevance	7
Poorly Taught	4
Other	7

Finally the respondents were asked to suggest courses that should be added to the Curriculum:[22]

Course	No. of Responses
Systems Programming	12
More Depth in Basic Languages	7
COBOL	7
Operations Research	5
Data Processing Techniques	7
Hardware Characteristics and Utilization	7
Data Communications	6
Information System Design	5
System Analysis and Design	5
Other	9

As with the bachelor's degree program, the vast majority of the respondents indicated that if they had it to do over again, they would enter the master's program at The Pennsylvania State University.

A similar study of the Undergraduate program at The Ohio

21. Ibid., 109.

22. Ibid.

State University is reported by Kalmey [114]. Information was requested from the graduates of the program from 1968 to 1973. 88, or about 20% responded.

The 0 - 9 (0 useless, 9 most useful) rating scale of courses was used in this study also. Mathematics and Statistics courses involved in the curriculum were also considered with computer science courses. Ohio State offers undergraduate degrees in the College of Engineering, the College of Arts and Science, and the College of Administrative Science. The responses were partitioned among these various programs. The results are given in Table 4-17.

Additional material was requested to identify areas the respondent felt were important to him and the degree of emphasis needed in these areas. The results are reported in Table 4-13.

Respondents were asked what additional courses they would like to see in the curriculum. These results are reported in Table 4-19.

As with the Penn State studies there is an apparent feeling among the graduates of such programs that greater emphasis should be placed on application areas, and in theoretical areas, on ways in which theory relates to practical applications.

Summary

The relation of "Curriculum '68" to existing programs is seen through three studies of existing curricula, and

Undergraduate Program Course Evaluation[23] Table 4-17

Course	Program						Total	
	Engineering		Arts and Sciences		Admin. Science			
	Responses	Rating	Responses	Rating	Responses	Rating	Responses	Rating
Intermediate Digital Computer Programming	27	7.6	46	7.8	5	6.8	78	7.7
Computer Systems Programming I	26	7.0	40	7.7	3	7.1	69	7.4
Data Structures	10	7.2	20	7.5	-	-	30	7.4
Computer Programming and Data Processing II	14	6.5	35	7.2	6	6.8	56	7.0
Computer Systems Programming II	15	6.8	29	7.1	1	5.0	45	7.0
Digital Computer Programming I	24	7.1	41	7.0	3	5.6	68	7.0
Computer Programming and Data Processing I	9	7.6	21	6.6	6	6.8	36	6.6
Survey of Programming Languages	22	6.7	32	6.0	4	7.8	58	6.4
Modeling of Information Systems	9	6.6	17	6.1	1	5.0	27	6.2
Individual Studies (Graphics, Mini, etc.)	11	6.5	18	5.9	1	7.0	30	6.2
Programming Languages	11	5.8	19	5.7	1	7.0	31	5.8

23. Donald L. Kalmey, "Profile of a Computer and Information Science B. S. Graduate", SIGCSE Bulletin 6, 1 (February 1974), 43.

Table 4-17 (continued)

Course	Program						Total	
	Engineering		Arts and Sciences		Admin. Science			
	Responses	Rating	Responses	Rating	Responses	Rating	Responses	Rating
Linear Optimization Techniques in Information Processing	11	5.9	16	5.7	1	5.0	28	5.8
Compiler Design and Implementation	8	6.1	20	5.4	1	9.0	29	5.7
Digital Computer Organization	13	5.4	18	5.5	-	-	31	5.5
Algebraic Structures	22	5.6	34	5.3	1	5.0	57	5.4
Mathematical Statistics I	27	5.4	39	5.2	6	6.3	72	5.4
Modern Methods of Information Storage and Retrieval	10	5.5	19	5.1	2	6.0	31	5.3
Fundamental Concepts of Computer and Information Science	11	5.6	26	5.5	3	1.0	40	5.2
Linear Algebra	27	5.4	40	5.1	2	5.0	69	5.2
Mathematical Statistics II	27	5.2	28	5.1	6	5.6	61	5.2
Survey of Numerical Methods	27	5.4	44	4.9	2	6.5	73	5.1

Table 4-17 (continued)

Course	Program						Total	
	Engineering		Arts and Sciences		Admin. Science			
	Responses	Rating	Responses	Rating	Responses	Rating	Responses	Rating
Introduction to Information Storage and Retrieval	28	3.9	44	5.6	6	4.1	78	4.9
Differential Equations	26	5.9	40	4.3	3	4.3	69	4.9
Advanced Calculus	15	6.0	26	4.1	1	5.0	42	4.8
Numerical Linear Algebra	9	5.1	20	3.6	-	-	29	4.0
Numerical Solution to Ordinary Differential Equations	6	4.8	14	2.7	1	0	21	3.2

Areas of Importance and Degree of Coverage[24] Table 4-18

Program	Area	Operating Systems	Numerical Analysis	Data Management	Programming Languages	Computer Organization	Finite Automata	Graphics	Simulation	Information Storage Retrieval	Mathematics	Business
Engineering	Need More	15	1	17	3	5	7	10	16	7	1	19
	O.K.	12	25	11	23	19	11	13	9	20	19	11
	Need Less	0	1	0	1	1	5	4	1	0	7	0
Arts and Science	Need More	33	3	36	10	20	8	12	21	16	2	33
	O.K.	16	29	10	37	26	16	13	13	27	29	12
	Need Less	0	15	0	1	1	16	10	5	6	18	0
Administrative Science	Need More	3	0	4	1	3	1	0	1	3	0	1
	O.K.	2	5	2	4	3	3	1	3	2	5	4
	Need Less	1	1	0	1	0	2	4	1	1	1	1
Totals	Need More	51	4	57	14	28	16	22	38	26	3	53
	O.K.	30	59	23	64	48	30	27	25	49	53	27
	Need Less	1	17	0	3	2	23	18	7	7	26	1

24. Ibid., 44.

Additional Courses Recommended for the Curriculum[25] Table 4-19

Course	College			Total
	Engineering	Arts and Science	Administrative Science	
PL/I	2	1	2	5
COBOL	3	2	2	7
Data Communication and Teleprocessing	2	5	-	7
Business	4	-	-	4
Simulation	2	1	-	3
Projects	2	1	-	3
Logic Design and Architecture	4	1	-	5
Utilities and Job Control Language	-	2	-	2
Operations Research	-	-	1	1
Data Base Structure	-	1	3	4
Hands-on Experience	1	2	-	3

25. Ibid.

three studies of graduates of computer science programs. The three studies of existing programs show a strong relationship between courses offered and those of "Curriculum '68", though, of course, there is no implication of a causal relationship. The Hamblen study of all post-secondary institutions shows that better than 52% of the courses listed in computer science in 1971-72 were comparable to courses in "Curriculum '68". Engel's 1971 study shows that most of the courses of "Curriculum '68" are offered as specified or in similar form in the twenty-six institutions involved in the study. These results are also verified in the Walker survey.

The first two of the basic courses B1, Introduction to Computing and B2, Computers and Programming are offered in most cases. Course B3, Introduction to Discrete Structures, does not seem to have as much acceptance. Hamblen reported that only 71 such courses were available in 1971-72 as compared with 1778 for B1 and 728 for B2. In Engel's study 19 of the surveyed departments offered the same or a similar course, but B3 was the least commonly offered of the basic courses. Walker's study also showed B3 to be the least offered of the basic courses, being required in 43% of the institutions. Course B4, Numerical Calculus was offered in more cases than Discrete Structures with 405 courses reported in the Hamblen Survey, however, its offering was considerably below that of courses B1 and B2.

Considering the intermediate level courses of "Curriculum '68", I1, Data Structures, I2, Programming Languages, I3,

Computer Organization, I4, Systems Programming, I5, Compiler Construction, I8, Numerical Analysis I and I9, Numerical Analysis II were offered with the greatest frequency. In all three reports I6, Switching Theory and I7, Sequential Machines had the smallest offerings; in the Hamblen study 121 courses similar to I6 were offered and 36 similar to I7; in Walker's study 55% offered I6, 52% I7; and in Engel's study 14 of 26 offered I6 and 16 of 26 offered I7. In Engel's study the other seven intermediate level courses were offered with consistency at the institutions surveyed, I2, I3, and I9 offered in 22 cases, I8 in 23, I1 and I4 in 24 and I5 in 26. For these same courses in Walker's Survey I2 was the most popular, being offered in 97% of the institutions with I5 the least popular being offered in 73% of the institutions. In Hamblen's study these seven courses followed a fairly wide range, I4 offered in 326 cases, I8 and I9 in 269, I2 in 239, I3 in 221, I1 in 146, and I5 in 135. It is interesting to note that I4, System Programming while most popular in the Hamblen list was in position 6 in Walker's study, while programming languages, third in Hamblen's list was first on Walker's. In both cases the Data Structures courses was in the middle, though its popularity, in terms of offering, increases from the Hamblen to the Walker study.

The three studies indicate a high degree of popularity in the offerings of the advanced courses A4, System Simulation, A1, Formal Languages and Syntactic Analysis, A9, Artificial Intelligence and Heuristic Programming, A5, Information

Organization and Retrieval, A2, Advanced Computer Organization, and A7, Theory of Computability. Of these, course A4, System Simulation presents some difficulty in that it was not clearly identifiable as such in the Hamblen or Walker study, and was reported as offered in only 13 of the 26 institutions in the Engel study. 58 offerings of Theory of Computability, Course A7, were reported in the Hamblen study, however it was reported in 25 of the 26 institutions in the Engel study and in 81% of master's programs (93% of Doctoral programs) in Walker's work. Courses A3, Analog and Hybrid Computing, A6, Computer Graphics, and A8, Large Scale Information Processing Systems were low in terms of number of offerings in all three studies.

Courses other than those listed in "Curriculum '68" appeared in the surveys of Hamblen and Walker. In Engel's survey, a listing of service courses was obtained, but other information on these, and information regarding other courses not in "Curriculum '68" was not considered.

In terms of service courses a number of different courses were listed as offered by the responding departments. In Walker's survey, including 283 responses, Computer Appreciation was offered in 40% of the institutions, and Computers and Society in 28%. The Introduction to Computer Science (Course B1) was used as a service course at 78% of the responding institutions and a course similar to B2 served as a service course at 59% of the institutions. In addition a variety of language courses are reported under service courses the

most popular being FORTRAN (84%) and COBOL (48%).

In Hamblen's survey all the courses listed in "Curriculum '68" were also listed as serving as service courses. In addition special applications courses were listed in engineering applications (456), business applications (228), and the Humanities (88). 208 courses in Computer Appreciation were listed, more than half of which served as service courses for non-science students. 257 Introductory Systems Analysis courses were also listed though these were heavily (154) in two year schools where they were required in the major program.

In the data processing area, Hamblen's survey lists four courses two of which, Data Preparation, and Unit Record are primarily offered in two year schools. The other two, however, were more generally offered. The Introductory Course in Data Processing was offered at 457 institutions (246 two year institutions) and was required for a computer science major in 56 programs. An advanced course in data processing was offered at 109 schools (42 two year institutions), and required in 30 computer science programs. The Introduction to Data Processing was considered in the Walker study, being listed as required or as an elective in 59% of the undergraduate programs, 29% of the master's programs and 27% of the doctoral programs.

Both the Hamblen and Walker survey list a variety of the courses being taught which do not have corresponding courses in "Curriculum '68". Hamblen's survey reports courses

in hardware such as A First Course in Computer Design, Advanced Course in Computer Design and Advanced Computer Organization; courses in the general area of information systems such as Introductory Information Systems Design, Advanced Information Systems Design, Management Information Systems, Data Base Management, File Organization, Introductory Operations Research, and Advanced Operations Research; and other courses such as Automata Theory, Data Communications, Process Control, and Management of Computer Centers or Information Processing Departments. To these courses, Walker's survey indicates that courses including Information Theory and Coding, History of Computer Science, Time-Sharing Systems Design, Symbol Manipulation and Text Processing, Real-Time Systems Design, Legal Aspects of Computers and Programming Techniques were offered at a significant number of institutions.

Though a full understanding of the actual structure of the undergraduate program is difficult to determine from the data of the three studies, the material of the Engel survey and the Walker survey seem to indicate a concentration of the requirements of many programs around the core of "Curriculum '68". Exceptions seem to be in the areas of discrete structures, numerical calculus, and a lack of the requirement of the intermediate level courses. The mathematics required is quite close to the recommendations in many of the institutions. Since the graduate programs of "Curriculum '68" are not specified in detail, it is not possible to come to firm conclusions regarding these programs from the data of the surveys.

While the number of offerings represent one measure of the strength or weakness of the programs, another is the reactions of the graduates of the program looking back at their experience. In the Barnes and Gotterer paper concerning bachelor degree graduates, in response to the value of course work to the professional career of the graduate there is a rather clear break with the programming related courses Introduction to Programming, Assembly Programming, Introduction to Data Processing, Information Structures and Systems Programming rated high, while mathematically oriented courses including Numerical Calculus, Graph Theory, Numerical Computation and Logic and Computability rated low. The course Structure of Programming Languages was the only one receiving an effectively neutral rating. Similar reactions appeared in the Gotterer and Barnes review of graduates of the master's program. Considering only those courses receiving more than six responses, highest rankings went to programming related areas such as Information Structures, Systems Organization and Programming, the Structure of Programming Languages, Structure of Artificial Languages, and Systems Programming. More or less neutral rankings went to Nonarithmetic Programming, Information Processing Systems, Information Retrieval, and Theory of Formal Languages and Automata. Low rankings again appeared for the more mathematically oriented courses such as Combinatorics and Graph Theory, Information Theory and Error-Correcting Codes, Numerical Computations, Matrix Computations, Logic and Computability, Mathematical Machine Theory, Theory of Automata,

Algebraic Theory of Automata, Theory of Graphs and Networks, Numerical Analysis, and Algebraic Coding Theory.

Kalmey's study of graduates of the undergraduate program reveals something of the same pattern. The highest rankings as to value to the graduate's professional life went to programming oriented courses including, Intermediate Computer Programming, Computer Systems Programming I and II, Data Structures, Computer Programming and Data Processing I and II, Digital Computer Programming I, Survey of Programming Languages, Modeling of Information Systems, and Individual Studies. The courses receiving low ratings were primarily the mathematics courses Differential Equations, Advanced Calculus, Numerical Linear Algebra, and Numerical Solution to Ordinary Differential Equations and also Introduction to Information Storage and Retrieval. Effectively neutral rankings went to the courses Programming Languages, Linear Optimization Techniques in Information Processing, Compiler Design and Implementation, Digital Computer Organization, Algebraic Structures, Mathematical Statistics I and II, Modern Methods of Information Storage and Retrieval, Fundamental Concepts of Computer and Information Science and Survey of Numerical Methods.

The recommendations of the graduates of additional work that should be offered reflects a desire for more practical material. In the Gotterer and Barnes study work is suggested in systems programming, additional language instruction, COBOL, operations research, hardware, information systems

design and systems analysis and design. In Kalmey's study much the same trend occurs with requests for material involving PL/I, COBOL, data communication and teleprocessing, business, simulation, special projects, logic design and architecture, utilities and job control language, operations research, data base structure, and hands-on experience.

Chapter 5 - Individual Courses

Chapter 4 indicates several areas where course recommendations were not made in "Curriculum '68", and where activity is occuring, and, conversely, there were courses recommended whose utility, based on experience, is open to some question. These matters will be considered again in Chapter 7. Additionally, courses have been proposed in curriculum work conducted subsequent to the publication of "Curriculum '68"; these are discussed in Chapter 6.

In this chapter, consideration will be given to courses which have been specified and discussed in the literature subsequent to the publication of "Curriculum '68". These reports primarily represent experience in the teaching of the material, and additional consideration of the areas of the curriculum which, for one reason or another, proved difficult to implement.

Included under this heading are courses dealing with a variety of areas of the curriculum including the first course, discrete structures, service courses, operating systems, computer organization, minicomputers, systems analysis, software engineering, computer design and architecture, program debugging, advanced programming, and computers and society.

Discussion of the first course have been quite extensive and have inevitably also raised questions relating to the teaching of service courses. Among the questions raised are

whether such courses should be taught separately to different groups of students or combined, whether they should be exclusively taught by the Computer Science Department or by various departments, which languages should be included in such courses, the general content of the first course, and pedagogical issues.

Brady [31] points out the advantages of programming courses being offered in a number of different departments. In a panel at the First SIGCSE Symposium, Sterling and Pollock [171] described an introductory sequence for all students, while deCampo [59] describes special courses for humanists. In the same panel Brillinger and Cowan [32] describe the importance of a consolidated program emphasizing student oriented software.

At the Second SIGCSE Symposium Adams and Haden [1] reported a need to split introductory courses in programming while offering common courses in "computer appreciation". Ralston [152] stresses that the particular language selected for teaching the first course is not a significant factor.

The discussions of the first course continued at the Third SIGCSE Symposium. Fisher, Hankley, and Wallentine [77] and Salton [160] discussed programs in which certain aspects of the first course are common to all students, with additional work concentrating on special material appropriate to smaller groups. In the Fourth SIGCSE Symposium the ideas encompassed in structured programming are considered in relation to teaching introductory programming; Gries [93] and Kernighan

and Plauger [118]. In this same source Sistare and Sondak [168] describe an individually paced approach to the presentation of the first course.

Van Dam, Strauss, McGowan, and Morse [177], also in the Fourth SIGCSE Symposium, reported on a survey of introductory programming courses. Twenty-four institutions were contacted regarding various aspects of the course and the following summary of such courses was reached:

> Using the survey's results, let us now outline the "typical" introductory programming course at major American Universities. It is a large, service course taught by the Computer Science department. The principal course goal is for the students to become proficient in a single programming language. This high level language is either FORTRAN (in the guise of WATFIV or WATFOR) or Pl/I (typically the PL/C or PLAGO subsets). As perceived by the instructors, the other university departments are very satisfied with both the goals and the results of the introductory programming course. Also the university is very flexible in budgeting enough computer time (money to meet the course's needs).
>
> To achieve the course's objective each student must write about 300 total lines of code. This coding represents about 6 different programming assignments. At least one assignment requires writing over 75 lines of code. We estimate that a student averages over 10 hours per week on the course even though the usual turnaround time is less than 1/2 hour.[1]

Additional material has been published more specifically directed to the service aspects of programming courses. Goddard [84] in the Proceedings of the Third SIGCSE Symposium considers the problem of computer work, in this case two courses, used to partially fulfill language requirements

1. Andries van Dam, Clark M. Strauss, Chalres McGowan and Jean Morse, "A Survey of Introductory and Advanced Programming Courses", SIGCSE Bulletin 6, 1 (February 1974), 175.

in doctoral programs. In the same source, Bateman and Pitts [25] address the same question again resulting in two courses, the first similar to course B1 of "Curriculum '68" and the second, advanced programming in a higher level language appropriate to the students area. Also in the Third SIGCSE Symposium Proceedings Willoughby [186] presents data on the attitudes of business students taking a programming course taught in a Computer Science Department.

Related closely to the question of introductory courses and first courses is that of alternatives to a course like B1 of "Curriculum '68". One such alternative which has received a good deal of attention, at least by using books published as a measure, is Computers and Society or Computer Appreciation.

In the Proceedings of the Second SIGCSE Symposium Lee [122] describes his course, and summarizes the underlying philosophy of such courses:

> Irrespective of one's personal position on the role of computers in society, it is indeed desirable that all college graduates in the coming years have a realistic even though minimal understanding of how computers work and how they may be directed to implement and maintain almost any desired social system. Consequently, the primary purpose of this course on computers in society is to give an elementary but sound fundamental understanding of how computers work, what they can do, what applications of computer technology currently exist or are now in research consideration, and the relationships of these applications to the role of man in society. Thus, the course is conceived as a citizen's social problems course in which much of the time will be devoted to documentation of the claim that society is undergoing a computer revolution and to illumination of this position by the presentation of several problem

areas resulting from computer applications.[2]

Lee goes on to describe the course which includes a little programming, an overview of computing, an overview of non-numeric computation, and discussion of the development and future of computer applications.

Horowitz, Morgan and Shaw [106] describe a course on Computers and Society dealing with political, economic, cultural, social and moral issues. This course however was designed for computer science students and its purpose is to give the computer science student an awareness of the implications of this chosen vocation. Horowitz and Horowitz [107] at the Third SIGCSE Symposium, describe another approach in which computer science students and other students are combined in a team taught interdisciplinary course in Computers and Society.

Such courses and material is also referenced in Austing and Engel [22], and the Wheaton Conference [121] which are considered in Chapter 6. In the former report material on implications of computing is included in a second course with programming as a prerequisite. In the latter case, it was felt that such issues are well handled by inclusion of material in existing courses, and where resources are short there is no pressing need to introduce the material as a special course.

---

2. Hans E. Lee, "Computers in Society--A Course Description, Purpose and Rationale", SIGCSE Bulletin 4, 1 (March 1972), 97.

Related to programming courses generally and the first course in particular is the question of correcting or debugging a program. Heilman and Ashby [104] observe that while this is an important area to the practicing computer scientist, it is neglected in most curricula. Mathis [132] specifies and discusses a course in debugging which also includes discussions of various levels of debugging techniques as well as methods for preparation of less error-prone programs.

The Discrete Structures Course (Course B3 of "Curriculum '68") has received considerable attention. Although Fischer [76] claims the course specified in "Curriculum '68" is effectively taught in most institutions, three papers, Engel and Jones [73], Yeh, Good and Musser [188], and Tremblay and Manohar [176], have addressed this course. These papers have attempted to delimit the course specifications given in "Curriculum '68" and show how the course can be fit into and motivated at the elementary level. Tremblay and Manohar conclude that the topics of discrete mathematics are broad enough, and of enough significance to the computer science student to require the equivalent of two semesters. While Yeh, Good and Musser suggest a one semester course in discrete structures, they, at the same time, specify a second course in computational analysis which builds on the discrete structures course, and covers the topics of computational analysis, computational correctness, and computing time analysis.

In addition to commenting on the Discrete Structures course of "Curriculum '68", Fischer [76] considers other

courses in the theoretical areas. Switching Theory (I6) and Sequential Machines (I7), he feels are also well specified in terms of current practice. The advanced courses A1 (Formal Languages and Syntactic Analysis) and A7 (Theory of Computability) are in need of attention. Course A1 could be made more practical in nature by encompassing topics in language research such as parsing methods and abstract families of languages. The material within A7 with some modifications, such as removal of recursive function theory and introduction of computational complexity, analysis of algorithm and program schemata, could be split into two courses. The first course would be at the senior-first-year graduate level and carries a suggested title of "Introduction to Computability Theory and Formal Languages":

> This course is designed to introduce the advanced undergraduate or first-year graduate to formal systems of computation (including formal languages) and to give him a basic understanding of the nature of the unsolvability phenomena which pervade attempts to deduce correctness of programs equivalence of algorithms, etc.[3]

The other course, remaining at the advanced level would concentrate on computational complexity, analysis of algorithms and program schemata.

The area of Operating Systems has received a good deal of coverage, including the detailed report of the COSINE Committee Task Force on Operating Systems [52]. This report

3. P. C. Fischer, "Theory of Computing in Computer Science Education", Proceedings of the AFIPS 1972 SJCC (AFIPS Press, Montvale, New Jersey, 1972), p. 860.

gives detailed specifications for eight modules covering the various aspects of the subjects; Introduction, Procedures, Processes, Memory Management, Name Management, Protection, Resource Allocation, and Pragmatic Aspects. Denning [61] considers ways in which this material can be put together into a course, and its place within the Curriculum. Additionally, he compares the work of the COSINE Task Force on Operating Systems with similar course work in "Curriculum '68":

> The course itself is related to ACM's systems programming course (ACM Course I-4) but differs in at least two significant ways. First the ACM outline suggests a "descriptive", case-study approach wheras this course is organized along conceptual lines. Second, ACM's course emphasizes techniques of systems programming wheras this course emphasizes the principles of system organization and operation. This shift in emphasis is made possible by new developments, since the ACM report predates the appearance of much of the modeling and analysis material on which this course is based.[4]

Work involving the presentation of operating systems courses were reported prior to the publication of the COSINE Committee recommendations. Included in these is Engel's report on the C^{3}S Programming System Workshop of 1969 [69] which considered courses offered at the University of Pennsylvania, University of Michigan, University of Iowa, Purdue University, Stanford University, the University of Texas, Cornell University, Carnegie-Mellon University, the University of California, Berkeley, and the University of Illinois. Included

4. Peter J. Denning, Operating Systems Principles and Undergraduate Computer Science Curricula (Computer Science Laboratory, Department of Electrical Engineering, Princeton University, Technical Report TR-99, September 1971), p. 10.

in this report are course outlines and sample examinations. The First SIGCSE Symposium also had discussions of this topic with Denning [60] reporting on work that would be closely related to the development of the COSINE recommendations, and Graham [90] reporting on a case-study approach.

Reported work in the operating systems, or systems programming course area subsequent to the publication of the COSINE Committee Report has been mainly concerned with student oriented software for such courses. Among the systems so described are the HMS 5050 [185], ATOPSS [175], SLIC [27], ASSIST [128, 129], POPSS [180], and MIX [91]. In all these cases the integration of these pedagogical aids to the operating systems area is stressed.

Related to the development of the pedagogical software, is the question of special computing equipment for a computer science laboratory in which the students have a machine, independent of the central computing facility, for experimentation and research. Hunt [109] reported on the situation at the University of Washington at the First SIGCSE Symposium. In this case the department has a XDS Sigma 5 dedicated to student and faculty research. The general philosophy and requirement for a dedicated departmental computer, or computer science laboratory is further explored in two papers presented at the Second SIGCSE Symposium; Eckhouse [65] and Stark [170].

The question of minicomputers and their relationship to the computer science laboratory is addressed in the April 1972 report of the Task Force on Minicomputers of the COSINE

Committee [53]. Though originally aimed at Departments of Electrical Engineering, the report holds considerable relevance to computer science education:

> Today's electrical engineer must be familiar with the characteristics and operations of minicomputers. This means that every electrical engineering department should have at least one minicomputer available for use as part of the undergraduate laboratory program. Cost is no longer a limiting factor since it is possible to obtain a basic system for approximately six thousand dollars. Once a basic system is installed it can be slowly expanded by adding peripherals as the use of the system grows.[5]

The task force, in justifying this, notes the following eight educational objectives, of which they feel, only the first two can be met by the usual central computing facility:

> 1) To provide training and experience using high level languages
> 2) To provide training and programming experience using machine code and assembly level coding
> 3) To teach the student the fundamentals of machine organization
> 4) To provide the student with an experimental facility to test the ideas and concepts presented in courses on Operating Systems
> 5) To provide the students with an experimental facility to test the ideas and concepts learned in courses dealing with interfacing and data processing
> 6) To provide the student with an experimental facility to test out ideas and concepts learned in courses on digital process control, digital testing and equipment monitoring
> 7) To provide students with an opportunity to understand by hands on experimentation, the relationships and interactions between hardware and software and the problems presented by real-time programming problems
> 8) To provide the student with an opportunity to utilize the computer as a system element.[6]

5. COSINE Committee, Minicomputers in the Digital Laboratory Program (National Academy of Engineering, Washington, D. C., April 1972), 22.

6. Ibid., p. 5.

In the Proceedings of the Third SIGCSE Symposium Marsland and Tartar [127] discuss their experience in teaching a course in minicomputer systems utilizing a laboratory situation similar to that described by the COSINE Task Force.

The COSINE Committee's task force on the Computer Organization Course published a report in October 1968 [48]. The recommendations outlined in this report give more detailed course outlines, but cover roughly the same material as is covered in Course I3 (Computer Organization) and Course A2 (Advanced Computer Organization) of "Curriculum '68".

Closely related to questions of computer organization is computer architecture. At the Fourth SIGCSE Symposium, Sloan [169] reported on the current status of offerings in this area. Using the results of a 1972 COSINE survey, he identified two courses B2 (Computers and Programming) and I3 (Computer Organization) of "Curriculum '68" as roughly equivalent to COSINE Courses Machine Structure and Machine Language Programming, and Computer Organization respectively. In the survey of 151 Electrical Engineering Departments it was found that 55.4% offered the course equivalent to B2 with 35.1% having the course offered by another department, usually computer science, and 68.3% offered the course I3 with 26.8% having the course offered by another department, again usually computer science. Sloan went on to discuss ways in which the courses differed from the given recommendations, noting that they seemed to fall into five basic catagories:

1) Introductory computer engineering courses with a computer architecture flavor;
2) Software-oriented computer organization courses;
3) Hardware-oriented computer organization courses;
4) Case study courses
5) Topical seminars[7]

In this same volume, Thomas [174] discussed the objectives and placement of courses in computer architecture in the curriculum, while Clark [36] outlines the position and content of this material in the doctoral program. Ellis and Wann [68] at the Second SIGCSE Symposium described how specially designed hardware macromodules can be used as a pedagogical aid in teaching computer architecture.

Consideration has been given to the problems of preparation of reliable software. Much of this work is now considered under the title of structured programming, but has also been referenced as software engineering.[8] Parnas [145] described how material of this type fits into a curriculum. He includes a description of the general term:

> The term "software engineering" is often used to denote the building of commonly used systems programs such as assemblers, compilers, and operating systems. In the design of this course I have taken a much broader

7. M. E. Sloan, "Computer Architecture in U. S. and Canadian Electrical Engineering Departments", SIGCSE Bulletin 6, 1 (February 1974), 113.

8. A conference on Software Engineering and its impact on education was held in Annapolis in 1969. No reports were published on this conference. References in the area include two reports of the NATO Science Committee, Software Engineering, Peter Naur and Brian Randell (ed.), January 1969, and Software Engineering Techniques, J. N. Buxton and B. Randell (ed.), April 1970.

> view. I take the view that programming is taught in our basic courses as a solo activity. Such courses teach programming techniques that are suitable for use by a single person constructing a program which will not be touched by other people. In contrast, I feel that the essential characteristic of a software engineering task is that many people will be involved with the product. Either several people will cooperate in producing it, or it will be used or modified by persons other than the original writer. The course emphasizes procedures which are optional and might be superfluous for solo programming tasks, but are important if several people are involved.[9]

The integration of business oriented courses into a computer science curriculum is addressed in a paper by Ein-Dor and Lyons [66]. After presenting some statistics showing that a large number of the graduates of computer science programs go into business oriented fields, the authors propose four courses; Optimization Technique, Probability Models and Queing Theory, Introduction to Simulation Languages and Experiments, and System Simulation. The topics presented in this paper are further addressed by the work of the ACM Curriculum Committee on Computer Education for Management which is referenced in Chapter 6.

Summary

The literature of computer science education contains a number of references to specific courses, providing expansion on the course materials given in the curriculum studies.

The first course has received considerable study

9. D. L. Parnas, "A Course on Software Engineering Techniques", SIGCSE Bulletin 4, 1 (March 1972), 154.

focusing on questions of pedagogy, and the use of this course as a service course. Recently attention has been given to the introduction of the ideas of programming style and structure in this course. Alternatives to the first course, at least as a service course, have been proposed in computer appreciation courses.

Discrete structures as proposed in "Curriculum '68" has presented some problems in implementation, and as a result has received attention in the literature. Issues discussed have focused on the placement of the course in the curriculum, methods of motivation and whether or not the material needed requires more than one three semester hour course for coverage.

Operating systems has received considerable study with the COSINE Committee report redefining this course in terms of developments and experience since the publication of "Curriculum '68". Considerable effort has also been evident in the development of special software systems for such courses.

Questions of the utility of the central university computing system for use in computer science courses have been raised, and special laboratory computers for the departments have been suggested. Related to this, the introduction of minicomputers, to serve this laboratory need, and as an object of study in themselves, has been proposed.

Consideration has been given to courses in computer organization and computer architecture. These considerations

have focused on the position of such courses in the curriculum, course content, and pedagogical aids.

Work has been evident in many other aspects of computer science education with reported activities in the areas of business oriented courses, software engineering, and theory courses.

Chapter 6 - Subsequent Work

The publication of "Curriculum '68" did not complete formal work on curriculum in computer science. C^3S became a standing committee of the Association for Computing Machinery, and has been involved in several projects since publication of "Curriculum '68". In addition another committee of ACM has considered academic work in information systems, while other groups have also been involved in curriculum in computer science. It will be the purpose of this chapter to look at these various efforts.

It was noted in Chapter 2 that C^3S did not regard its work as complete with the publication of "Curriculum '68" and indeed planned to address such problems as programs for smaller colleges, junior colleges and technical schools; the relationship of computer science to other disciplines; program implementation problems; graduate programs; and that a program of consultation and visitation would be sponsored by the Committee. These objectives have all been met to varying degrees with the exception of programs for junior colleges and technical schools. It was generally found that the problems of these institutions were quite different from those of the four year schools and ACM has formed a separate committee on computer science in junior colleges. Initial work in this area is reported by Connelly [44].

Though not a direct activity of C^3S, several members of the Committee, joined with other professionals in 1968 to form the ACM Special Interest Group on Computer Science

Education (SIGCSE):

> The objectives and purposes which led to the formation of SIGCSE can be stated in terms of its goal: to encourage and assist in the development of effective academic programs and courses in computer science.
>
> In accordance with this stated goal, SIGCSE has attempted to do the following:
>
> 1) Collect and disseminate information concerning courses and programs offered at the college level.
>
> 2) Organize and present technical panel sessions at national meetings and when appropriate, organize independent technical symposia.
>
> 3) Channel useful information developed by SIGCSE members to the attention of other ACM educational activities and committees.
>
> 4) Publish in the SIGCSE Bulletin such program descriptions, course syllabi, problem sets, and other items such as news and discussion items which are of particular relevance to its members. In addition, SIGCSE also plans to publish the proceeding of its special technical symposium.[1]

Since its founding, the SIGCSE Bulletin has been a most significant contribution to computer science education. SIGCSE has held four well attended, high quality symposia with the proceedings of each being major contributions to the field; additional symposia are planned as an annual event.

Returning to the sponsored activities of C^3S; within "Curriculum '68" reference is made to future work involving doctoral programs:

> In 1966 Professor Thomas Hull was asked by ACM to examine the question of doctoral programs in computer science. After discussion with the members of this Committee and with many other interested persons,

1. Robert M. Aiden, "Purpose, Goals, and Activities of SIGCSE", Proceedings of the IFIP World Conference on Computer Education 1970 (Science Associates/International, New York, 1970), p. II/153.

> Professor Hull decided to solicit a series of articles on the research and teaching areas which might be involved in doctoral programs. Each article is to be written by an expert in the particular subject area, such as programming languages, systems programming, computer organization, numerical mathematics, automata theory, large systems, and artificial intelligence. Each article is to attempt to consider all aspects of the subject area which might be helpful to those developing a graduate program, including as many of the following topics as possible:
>
> a) Definition of the subject area, possibly in terms of an annotated bibliography.
>
> b) Prerequisites for work in the area at the doctoral level.
>
> c) Outlines of appropriate graduate courses in the area.
>
> d) Examples of questions for qualifying examinations in the area.
>
> e) Indication of suitable thesis topics and promising directions for research in the area.
>
> f) The extent to which the subject area ought to be required of all doctoral students in computer science.
>
> These articles are scheduled for publication in Communications of the ACM and it is hoped that they will stimulate further articles on doctoral programs.[2]

Four articles appeared in the series: "Automata, Formal Languages, Abstract Switching, and Computability in a Ph. D. Computer Science Program", Robert McNaughton [136], "Computational Linguistics in a Ph. D. Computer Science Program", Susumu Kuno and Anthony G. Oettinger [120], "The Role of Programming in a Ph. D. Computer Science Program", Bruce W. Arden [7], and "Information Science in a Ph. D. Computer Science Program", G. Salton [159].

McNaughton briefly discusses the nature and definitions of automata, formal languages, abstract switching and

2. Curriculum Committee on Computer Science, "Curriculum '68, Recommendations for Academic Programs in Computer Science", Communications of the ACM 11, 3 (March 1968), 165.

computability. Seven courses are suggested which are labeled level 1 (junior-senior, first year graduate) and level 2 (seniors and all graduate students):

1) Introduction to Logic Design and Switching Theory (level 1)
2) Introduction to Computability, Formal Languages and Automata (level 1)
3) Theory of Computability (level 2)
4) Theory of Finite Automata (level 2)
5) Automata as Computational Models (level 2)
6) Computational Theory of Formal Languages (level 2)
7) Abstract Switching Theory (level 2)

The courses are referenced with a bibliography of 27 items.

Kuno and Oettinger first specify the scope of computational linguistics:

> Computational linguistics comprises (1) mathematical characterizations of natural languages, (2) developments of computer programs useful for linguistic research, and (3) the application of linguistic techniques to computer problems.[3]

A ten course program of study accompanied by an extensive bibliography is presented to cover the material in this field. The courses are listed as basic (freshman to junior level), intermediate (junior to graduate level) and advanced (advanced graduate level). It is noted that since the field itself is interdisciplinary, the courses overlap other branches of computer science:

1) Introduction to Computational Linguistics (basic)
2) Computers in Humanitic Research (basic)
3) Formal Theories of Grammar (intermediate)
4) Mathematical Models of Grammar and Syntactic Analysis (intermediate)

3. Susumu Knuo and A. G. Oettinger, "Computational Linguistics in a Ph. D. Computer Science Program" Communications of the ACM 11, 12 (December 1968), 831.

5) Information Organization and Retrieval (intermediate)
6) Introduction to Transformational Grammar (intermediate)
7) Topics in Transformational Grammar (advanced)
8) Computational Semantics (advanced)
9) Application of Linguistic Techniques to Computer Problems (advanced)
10) Advanced Course in Computational Philogy (advanced)

Arden describes three courses:

1) Applications Programming
2) Language Processing
3) System Design

These courses are to include the programming material involved in an advanced degree program. An extensive bibliography on programming is included, as is a table of some one hundred sixty five terms of which "subject matter suggested by these words should be familiar to an expert in programming and programming languages".[4]

Salton abstracts his work as follows:

> The report contains recommendations on a sample course curriculum in the general area of information organization and information systems designed in a Ph. D. Computer Science Program. The subject area is first briefly described, followed by a listing of some desirable graduate-level courses. Suitable bibliographies are appended.[5]

The courses specified are:

1) Data Structures and Information Organization
2) Time-Sharing Computer Organization
3) Language Structure and Syntactic Analysis
4) Text Analysis and Automatic Classification
5) Information Retrieval System Design
6) Automatic Text Processing Systems

---

4. Bruce W. Arden, "The Role of Programming in a Ph. D. Computer Science Program", Communications of the ACM 12, 1 (January 1969), 33.

5. G. Salton, "Information Science in a Ph. D. Computer Science Program", Communications of the ACM 12, 2 (February 1969), 111.

It should be noted that included with each of these four reports was the following statement by P. Calingaert, then education editor for the Communications of the ACM:

> ...The purpose of the series is to provide some guidelines for what constitutes a "good" doctoral program. These articles contain the opinions and recommendations of experts in the subject areas. However, unlike "Curriculum '68", they reflect neither the views of official ACM nor the deliberations of a committee....[6]

C^3S, or individuals serving on the committee, have been involved in programs for smaller colleges and universities, consulting activities on curriculum and related matters, and further work on the master's program. Much of the work of discussing programs and course content problems, have properly fallen on SIGCSE.

In 1970-71, a subcommittee of C^3S chaired by O. W. Rechard [154] addressed problems of smaller colleges:

> A sub-committee of the ACM Curriculum Committee has been established with the goal of studying the problem of Computer Science instruction in the four-year undergraduate colleges, and recommending a course of action designed to alleviate the matriculation problem described above. This proposal seeks funds for the work of the sub-committee and for an experimental summer program designed to introduce faculty members from small colleges to the structure and content of a group of selected undergraduate courses from schools with strong degree programs.[7]

In its initial activities information was gathered on the nature of programs in computer science in small colleges [70]:

6. Peter Calingaert, "Editor's Note", Communications of the ACM 12, 2 (February 1969), 111.

7. Ottis Rechard, "ACM Curriculum Committee Proposal", SIGCSE Bulletin 2, 2 (June-July 1970), 28.

> The small college with an interest in computing offers two or three courses in computer science. Probably a FORTRAN course, then courses like B1 and B2 of "Curriculum '68", and possibly a special topics course. The courses are being taught by one man, holding a masters degree in mathematics and probably teaching a mathematics course also. He has, at best, one course in computer science in his background, and serves as a consultant to those on campus interested in using the computer. He may also be associated with the operation of the computer center of his college.
>
> Within the next two years additional computer science courses will be added to the curriculum, bringing the total offering to close to 18 semester hours.
>
> There is a definite feeling that current graduate programs are inadequate for the preparation of this stand-alone computer specialist, and that a Doctor of Arts program, or a special masters program with a good deal of emphasis on application areas is needed.[8]

To address the problem of increasing the quality of offering in computer science, the subcommittee, with support from the National Science Foundation sponsored an institute at Purdue University in the summer of 1971. In this program four fundamental courses, discrete structures, programming languages, operating systems, and data structures were presented to fifty-three individuals from small colleges. The courses were taught as normally done, except that they were compressed into a four week per course time frame. It was anticipated that the attendees would take the material presented to them and offer it at their home institutions thus forming a strong core for a computer science program.

8. Gerald L. Engel, "Computer Science Instruction in Small Colleges - An Initial Report", SIGCSE Bulletin 3, 2 (June 1971), 8.

Evaluations of the program indicated that indeed the material was used as anticipated, and that the program was most successful.

Difficulties in finding funding for further institute programs, and the problems of reaching enough people with such programs led to a study by Austing and Engel [22] to recommend appropriate programs in computer science for small schools:

> This report gives recommendations for the content implementation and operation of a program of computer science courses specifically directed to small colleges. In no way does this material represent a major program in computer science. It does describe a program for those schools with limited resources, but with an interest, enthusiasm and desire for some course offerings in computer science. Those institutions interested in computer science and with resources necessary for a major program in this field should refer to the existing reports of ACM's Committee on Curriculum in Computer Science (C^3S), and other curriculum studies. Institutions which desire to complement computer science offerings with a set of courses in computational mathematics should consider the report of the Committee on the Undergraduate Program in Mathematics.[9]

The report emphasizes a program that would be within the reach of institutions with limited resources, includes an extensive library list, comments on staff and equipment requirements, and considers how the courses can be designed to meet a variety of objectives for an institution which cannot afford both a series for computer science students, and a series for departments using computer science:

---

9. Richard H. Austing and Gerald L. Engel, "A Computer Science Course Program for Small Colleges", Communications of the ACM 16, 3 (March 1973), 139.

> Four courses are described and suggestions are made for additional study and courses for students interested in further work. No names have been given to the four courses, but they correspond roughly to the areas of algorithms and programming (Course 1), application of computers and their impact on society (Course 2), machine and systems organization (Course 3), and file and data organization (Course 4). Though these courses in a real sense represent a coherent program, they are structured so as to allow a student with limited objectives and limited time to pick and choose those parts most relevant to his needs.[10]

C^3S has also continued to be concerned with graduate programs in Computer Science, and a program of study leading to a master's degree in computer science was reported by Melkanoff [137] at the Third SIGCSE Symposium in 1973, and again by Melkanoff, Barnes and Engel [138] at the 1973 National Computer Conference. This program modifies the master's recommendations of "Curriculum '68", based on experience:

> The proposed M. S. program is based on the following assumptions regarding the destination of its graduates:
>
> Some 10 to 15% of the graduates will work in areas related to the scientific aspects of Computer Science.
>
> Some 35 to 40% of the graduates will work in the area of computer system design (hardware and software).
>
> Some 35 to 40% of the graduates will work in those areas of systems design where the computer plays a key part.
>
> Some 10 to 15% of the graduates will work in areas unrelated to Computer Science or computers.[11]

The program suggested requires nine three semester hour courses, at least five of which are graduate. A thesis is optional. A high level of competence in at least one

10. Ibid., 139-140.

11. M. A. Melkanoff, "An M. S. Program in Computer Science", SIGCSE Bulletin 5, 1 (February 1973), 77.

programming language is required, as well as "Proficiency in a fundamental core of knowledge common to all subfields of Computer Science. This should cover the material taught in 4 to 6 courses (upper division or graduate) and include formal linguistics, programming languages, computer architecture, and systems programming."[12] A major should be completed in one of the established field of computer science which include theory, programming languages, systems, architecture and design, and artificial intelligence, the major selected in some sense designed to meet the students ultimate objectives. In addition it was strongly recommended that a minor be taken through a coherent sequence of courses in some application area.

It was also observed that C^3S, upon completion of work with "Curriculum '68", planned to develop a program of visitation and consultation. In 1968-69 this was handled through the ACM Visiting Scientist Program which gave way in the latter part of 1969 to the ACM College Consulting Service, funded by the National Science Foundation and under the direction of W. Viavant. In the period 1969-1973 nearly one hundred institutions were visited by computer professionals to offer various comments and suggestions on computer science curriculum, computer uses in education and facilities. The materials that encompassed the reports on these visits has

12. Ibid., 78.

been collected and a final report on this program will be published soon.

The activities described above indicate the formal and published material of the C3S subsequent to the publication of "Curriculum '68". In addition to this work, the committee has met regularly to discuss current issues in computer science education, and to supply input for additional work through the ACM Education Board. Included in these activities is a study leading to recommendations for certification of computer science teachers for secondary schools which will be carried out by members of C3S and the ACM Secondary School Committee, under the direction of the ACM Education Board.

C3S has maintained an interest in the areas of service courses with G. Stokes chairing a subcommittee studying the needs in this area. It is anticipated that once these are collected, further course descriptions to meet the service course need, based on experience, can be prepared.

The area of Computer Appreciation courses, or Computers and Society courses has become quite active in the past few years. R. Austing is chairman of a subcommittee to review this area and prepare necessary recommendations.

Continuing discussions have gone on regarding the nature of work needed on "Curriculum '68". The following comments represent a cross section of comments made at the various sessions:

"Computer and society courses are appearing with great frequency. Something is needed in the way of a general interest type course."

"Curriculum '68" is closely tied to mathematics. Is this necessary?"

"Topics like telecommunications and systems analysis are not mentioned."

"What effect does high school background in computing have on the first course?"

"At the junior-senior level, business-oriented courses are missing these should cover information structures, file and communication systems, systems analysis, and systems simulation."

"Little thought is given to operational aspects of computing."

"More is needed of the relationship of electrical engineering courses to computer science."

"Course bibliographies are much in need of updating."

"Several courses need reorganization and several courses could be expanded and/or split into two courses."

"Service courses need to be recommended."

"'Curriculum '68' came under early and continuing heavy criticism from within and without academia for its almost total neglect of the pragmatics of the job market."

"New curriculum should reflect the advances in Computer Science and include materials on topics that have come of age since "Curriculum '68" such as semantics, structured programming, theory of programming languages and mini-computers."

"A curricula report documenting undergraduate courses and undergraduate degree programs in Computer Science is needed."

"Any new version of "Curriculum '68" should stress more than before, applications of Computer Science."[13]

13. From the minutes of C3S meetings.

Though not themselves being direct C^3S activities, two additional meetings deserve mention. The 1970 ACM National Meeting featured some ten sessions on education including various aspects of computer uses in education and the application of computers to training. Also included were four sessions on Computer Science Education:

General Computer Education
Moderator - William F. Atchison, University of Maryland
Participants
Glen Bonham, Ontario Department of Education
Robert Korfhage, Southern Methodist University
Werner Rheinboldt, University of Maryland
William Viavant, University of Utah

Undergraduate Education
Moderator - Preston C. Hammer, Pennsylvania State University
Participants
Samuel D. Conte, Purdue University
D. D. Cowan, University of Waterloo
Gerald Engel, Hampden-Sydney College
Thomas Schoen, University of Dayton

Graduate Education
Moderator - George E. Forsythe, Stanford University
Participants
Thomas Bredt, Stanford University
Saul Gorn, University of Pennsylvania
Robert Spinrad, Xerox Data Systems
Peter Wegner, Brown University

Organizing for Computer Science Education
Moderator - Earl J. Schweppe, University of Kansas
Participants
William F. Atchison, University of Maryland
Aaron Finerman, SUNY, Stonybrook
George E. Forsythe, Stanford University
Preston C. Hammer, Pennsylvania State University

While not presenting much new, these panels did nicely summarize the problems facing various aspects of computer science education. What is significant is that, unlike most panel

sessions, edited transcripts of the sessions appear in the Proceedings of the Conference [19, 101, 81, 164].

The other activity was a meeting of small college educators at Wheaton College in 1972 to discuss problems of computer science education in small schools. This work is reported by La France and Roth [121]. At this session the existing studies on curriculum were taken, and, based on them, recommendations were made regarding appropriate material for colleges of the type represented (church related liberal arts colleges with enrollments of up to 1500 students). Effectively the Workshop recommended that a small school should seriously consider offering the equivalent of the courses outlined in the Austing and Engel report, and in addition a course in systems analysis. As demand warrents courses in the areas of language theory, theory of computing and simulation could be added, and appropriate course outlines are included. It was recommended that the mathematics departments of the colleges should also be strongly encouraged to offer courses in discrete structures, numerical mathematics, and combinatorics.

Curriculum development in areas related to computer science also continued in the period subsequent to the publication of "Curriculum '68". ACM was involved not only in the activities of C^3S, but also in the Curriculum Committee on Computer Education for Management (C^3EM) chaired by Daniel Teichroew. C^3EM has prepared and published curriculum

recommendations for both undergraduate and graduate programs in information systems, these programs are described in September 1971 [173], May 1972 [15], and December 1973 [55] Communications of the ACM, the Proceedings of the 1972 ACM National Conference [16, 54], and the Third SIGCSE Technical Symposium [17].

The graduate level recommendations were prepared by a subcommittee chaired by R. L. Ashenhurst:

> This report presents recommendations for a graduate professional program in information systems development at the Master's level. The program is intended for the education of individuals who will develop complex information systems. Concomitantly, recommendations are given for information system specialization within existing Master's degree programs.
>
> As documented in the position paper there is a widely felt need for individuals who can bring to bear the relevant computer technology on the information requirements of particular organizations. To meet this need requires the introduction of new professional programs and the modification of existing ones in institutions of higher learning.
>
> A body of knowledge exists for both organizational functions and information technology, but this knowledge is currently offered in diverse areas of graduate education. The curriculum in information system development presented here represents an attempt to integrate this knowledge and add new definition and perspective to the field.[14]

Accompanied by detailed course descriptions and extensive bibliographies, thirteen courses are presented, divided into four basic groups:

---

14. R. Ashenhurst (ed.), "Curricula Recommendations for Graduate Professional Programs in Information Systems", Communications of the ACM 15, 5 (May 1972), 365.

Course Group A: Analysis of Organizational Systems
- A1. Introduction to Systems Concepts
- A2. Organizational Functions
- A3. Information Systems for Operations and Management
- A4. Social Implications of Information Systems

Course Group B: Background for Systems Development
- B1. Operations Analysis and Modeling
- B2. Human and Organizational Behavior

Course Group C: Computer and Information Technology
- C1. Information Structures
- C2. Computer Systems
- C3. File and Communication Systems
- C4. Software Design

Course Group D: Development of Information Systems
- D1. Information Analysis
- D2. System Design
- D3. System Development Projects[15]

In many ways the program described was intended to meet objections raised to "Curriculum '68" regarding the relevance of the material to the real applications of computers. The primary interaction of this program and that of a computer science department is within the Group C courses. Ashenhurst described some of the problems involved in interaction and differences between programs in computer science and information systems:

> Computer science education today is a mixture of the "theoretical" and "applied". The theoretical, or basic, aspects are not directly relevant to education for work in information systems, although, contrary to some belief, the kind of precise thinking engendered is by no means an unwelcome attribute in an information systems developer. In fact, it is in the applied, or pragmatic aspects of computer science that the habits of thought instilled in the student may be most inconsistent with the needs of information systems development.

15. Ibid., 373.

> This paradox is brought about because of the "do it yourself" attitude toward programming often inculcated by computer science programs. The student uses the computer as a "personal problem solver", as a tool which he or she employs to obtain results in a given area of interest. The difficulties which are generally associated with communicating these results in their essence to others, let alone making it possible for others to use the programs to obtain similar ones, are too common to require elaboration here.
>
> The main area in which some of the information systems philosophy has penetrated computer science is that of the development of computer systems, in operating systems courses. This is not surprising, because a computer system is in fact a special type of information system, one whose "subject matter" is the computer hardware system and its associated pool of software resources, programs and data sets. Most of the computer science work on applicational techniques, however, takes little cognizance of the information systems point of view.[16]

In December 1973 recommendations for undergraduate programs in information systems were published [55]. This work was under the direction of J. Daniel Couger. The undergraduate program is described in terms of two options; an organizational option stressing the use of computers, and technological option for preparation of an individual for employment in information processing.

Eleven courses are specified, again with detailed course descriptions and bibliographies:

UB1 Operations Analysis and Modeling
UB2 Human and Organizational Behavior
UC1 Information Structures
UC2 Computer Systems
UC3 File and Communication Systems
UC4 Software Design

16. Robert L. Ashenhurst, "Implications for Computer Science Departments of the ACM Information Systems Curriculum", SIGCSE Bulletin 5, 1 (February 1973), 3.

UC8 Programming Structures and Techniques
UC9 Computerware
UA8 Systems Concepts and Implications
UD8 Information Systems Analysis
UD9 System Design and Implementation[17]

The first six of these courses are undergraduate versions of courses described in the Master's program. The other courses are combinations of material from the master's program. Course UC8 is of some special interest in that it represents an approach at a second course in programming.

In addition to the curriculum activities of ACM in the post "Curriculum '68" era, other organizations were active in curriculum planning related to computing and computer science. The Computer Sciences in Electrical Engineering (COSINE) Committee, of the Commission on Engineering Education, in addition to work reported in Chapter 5, produced a number of other reports involving computer science to varying degrees; Some Specifications for a Computer-Oriented First Course in Electrical Engineering [47], Some Specifications for an Undergraduate Course in Digital Subsystems [49], Impact of Computers on Electrical Engineering Education - A View from Industry [50], and Digital Systems Laboratory Courses and Laboratory Developments [51].

The Committee on the Undergraduate Program in Mathematics (CUPM) upgraded their 1964 recommendations for work in computing

17. J. Daniel Couger, "Curricula Recommendations for Undergraduate Programs in Information Systems", Communications of the ACM 16, 12 (December 1973), 732.

into a program in computational mathematics appearing in May 1971 [43]. The report comments on the need for this work:

> More recently, three trends have become noticeable. First, there appears to have developed a strong tendency on the part of computer science programs to minimize prerequisite requirements in traditional mathematics, particularly analysis, and also to underemphasize or even to disregard most areas of scientific computing. Second, many disciplines, including in particular the biological, social, and behavioral sciences, have become increasingly mathematical, giving rise to a need in these fields for expanded education in mathematics and in scientific computing. Finally the computer has begun to have a direct effect upon mathematics courses themselves. New courses particularly in computer-oriented applied mathematics, are being introduced into many mathematics curriculum, and traditional courses are being modified and taught with a computer orientation. As an example of the latter we cite only the teaching of calculus. Approximately 100 schools now offer a course in calculus using the text, Calculus, A Computer Oriented Presentation, published by the Center for Research in College Instruction in Science and Mathematics....
>
> These three trends all indicate a need for the mathematics community to accept a responsibility for mathematical or scientific computing, and to broaden educational opportunities toward a more encompassing "mathematical science" in which students may explore the areas of overlap between pure and computational mathematics, as well as computer science. There is thus a need for innovative undergraduate programs which provide for a wide range of options, different opportunities for graduate studies and a variety of future careers.[18]

The differentiation between the program outlined in this report and a computer science program is clearly stated:

> The present report is the result of such a reappraisal by the CUPM Panel on Computing. From the

18. Committee on the Undergraduate Program in Mathematics (CUPM), Recommendations for an Undergraduate Program in Computational Mathematics (CUPM, Berkeley, California, 1971), pp. 2-3.

> outset it was evident that the aims of this report should be different from those of the earlier work, since its intended audience is different. The present report does not address itself to the training of computer scientists. Instead, its concern is for the education of mathematicians who will know how to use and apply computers. Programs in computational mathematics necessarily have different objectives than do programs in computer science.[19]

The report goes on to recommend that the following courses would be appropriate to add to a program in mathematics:

Computer Science Courses
- C1 Introduction to Computing
- C2 Computer Organization and Programming
- C3 Programming Languages and Data Structures

Computational Mathematics Courses
- CM1 Computational Models and Problem Solving
- CM2 Introduction to Numerical Computation
- CM3 Combinatorial Computing
- CM4 Differential Equations and Numerical Methods

A strong relationship is seen between these courses and those of "Curriculum '68": C1 to B1 of C'68; C2 to B2 of C'68; C3 to a combination of I1 and I2 of C'68; CM2 to B4 of C'68; CM3 to B3 of C'68; and CM4 to I8 of C'68. The primary differences seem to be in the level of offerings. Course CM1 does not have a direct parallel in "Curriculum '68" and presents something of a second course in programming with emphasis in computer applications.

As was mentioned above, C^3S has had an interest in programs for secondary schools and indeed is beginning work jointly with the ACM Committee on Secondary Schools to draw up certification recommendations.

19. Ibid., p. 3.

Two other groups have published recommendations involving computer science education as it applies to the pre-college level. The Committee on Computer Education of the Conference Board of the Mathematical Sciences published in April 1972 "Recommendations Regarding Computers in High School Education" [38]. This report presented a series of seven recommendations regarding what computer education should be offered in the schools and how it should be achieved:

> A1. We recommend the preparation of a junior high school course in "computer literacy" designed to provide students with enough information about the nature of a computer so that they can understand the roles which computers play in our society.
>
> A2. We recommend that the process of preparing the text materials for the above course be such as to provide wide and rapid dissemination of information about the availability and feasibility of the course.
>
> B. We recommend that text materials for a number of other courses be prepared, including an introduction to computing, as a followup to the computer literacy course, some modules which integrate computing into high school mathematics courses, and other modules which utilize computers in simulating the behavior of physical or social phenomena and which enable the use of computers in the study of courses outside mathematics.
>
> C. We recommend the development of special programs for high school students showing unusual aptitude and promise in computer science.
>
> D. We recommend a major effort aimed at making vocational computer training more generally available and at the same time improving the quality of such training.
>
> E. We recommend that the National Science Foundation provide financial support for the development of a variety of programs for the training of teachers and of teachers of teachers of high school courses involving computers.

F. We recommend the establishment of a clearinghouse for information about high school computer education.[20]

Activity in the secondary school area has also been conducted by the Working Group on Secondary School Education (WG 3.1) of the International Federation of Information Processing (IFIP), Technical Committee for Education (TC-3). This group published an outline guide for teachers in August 1970 [110], revised September 1971 [111], "intended for those people who are concerned with the planning of computer courses for the training of teachers".[21] Various aspects of computing are introduced in such a way as to lend themselves to course development for teacher training courses, as well as ideas for classroom development. It is planned that this work will be expanded into a series on computing in secondary schools.

Additional material involving curriculum planning has been undertaken and reported by individuals not directly involved in the activities of the curriculum planning committees. In virtually all cases the ideas expressed have been incorporated in the deliberations of the various committees and subcommittees, and as such will not be directly considered here. The reader is referred to several papers appearing in the bibliography,

20. Committee on Computer Education, Recommendations Regarding Computers in High School Education (Conference Board of the Mathematical Sciences, Washington, D. C., April 1972), pp. 1-2.

21. IFIP Technical Committee for Education, Computer Education in Secondary Schools; An Outline Guide (IFIP, Amsterdam, The Netherlands, 1971), p. 4.

and especially those of S. Amarel [4] and P. Wegner [182, 183, 184].

Summary

Subsequent to the publication of "Curriculum '68", considerable work has gone on relating to computer science education. A significant development in this period was the formation of the ACM Special Interest Group on Computer Science Education (SIGCSE), which provides a continuing organization for the presentation and exchange of ideas in the field.

C^3S has continued its activities following the publication of the report. Under the sponsorship of this Committee, a series on doctoral programs was published, a summer institute program for smaller schools was conducted, course recommendations for smaller schools were prepared, and guidelines for masters programs were prepared. Additionally, C^3S has regularly met to discuss current issues in computer science education, and presently has subcommittees preparing materials relating to service courses, and courses in computer appreciation.

Additional curriculum work has been conducted within ACM by the Curriculum Committee on Computer Education for Management (C^3EM). This Committee has prepared guidelines for both graduate and undergraduate programs in information systems which integrate materials associated with computer

science with materials associated with business.

Curriculum study and development has also gone on outside ACM with the COSINE Committee continuing its study and review of computer science in electrical engineering and CUPM expanding its earlier work into a program in computational mathematics.

Chapter 7 - Impact of "Curriculum '68"

In chapters 1 to 3 "Curriculum '68" was discussed as were the immediate critiques of the report, work preceding the publication of the recommendations, and related work to "Curriculum '68". In chapters 4 to 6 implementation of the recommendations were considered, as were discussion of work on specific courses and work on C^3S and other groups involved in computer science curriculum. In this chapter these materials will be drawn together to determine the strengths and weaknesses of "Curriculum '68", and to see how the subsequent reports and recommendations serve to correct the weaknesses. The areas needing additional attention will then be determined.

The data from the curriculum surveys of Chapter 4 indicate that the courses of "Curriculum '68" are offered with significant frequency at educational institutions in the United States. The data also indicate, however, that certain courses require additional consideration. These include B3, Discrete Structures, B4, Numerical Calculus, I6, Switching Theory, I7, Sequential Machines, A4, System Simulation, A3, Analog and Hybrid Computing, A6, Computer Graphics, and A8, Large Scale Information Processing Systems. It is interesting to note that courses B3, I6, and I7 represent most of the theoretical component in the undergraduate program of "Curriculum '68", indicating that experience dictates that this entire

component needs further evaluation. Course B4 appears to need study regarding its placement in the curriculum and the relationship to the other courses in numerical mathematics. Course A4 appears to need additional focus as to content, direction and placement. Courses A3, A6, and A8 seem to suffer from a lack of equipment or personnel in most cases, but appear to raise the question of their necessity in detailed specification in a limited curriculum. Course I1, though offered with more frequency than the others, is offered with the lowest frequency of the core courses of "Curriculum '68". Considering the critical position of this course in "Curriculum '68", it too requires additional study.

It must also be noted, that in considering the courses of "Curriculum '68", the date of publication of the report, and the subsequent development of additional instructional materials, that there is a need, in the case of all the courses, to evaluate and further develop the referencing materials.

Atchison observed at the C^3S meeting at the 1971 FJCC[1] that C^3S had, in the construction of "Curriculum '68" sidestepped the issue of service courses in favor of what was then the more pressing issue of definition of the field of computer science. From the data of these studies of curriculum

1. Minutes of C^3S Meeting, 1971 FJCC.

implementation it is clear that service courses represent a major undertaking of the departments, and hence consideration of this area must be made in a review of "Curriculum '68". Areas that need attention include the appropriateness of a course like B1 as a service course, the nature and content of computer appreciation or computer and society courses, the role of the special application courses, the role of the special language courses, and course work above a one course offering in the service area.

Experience shows that work in the data processing area is offered and even required in a considerable number of programs, yet it was not considered in "Curriculum '68". This, then, is an area which needs attention.

Experience also shows that a large number of courses in computer topics not considered in "Curriculum '68" are being taught. It is interesting to note that many of these courses are offered with greater frequency than some of the courses of "Curriculum '68", and as such must be evaluated in considerations of revisions and updatings of "Curriculum '68". These include courses like Computer Design, Systems Design, Data Base Management, File Organization, Operations Research, Data Communications, and Process Control.

The consistency with which the graduates report the importance of the practical, programming oriented courses is significant. In many ways this repeats the thoughts expressed in discussions of industry reactions to computer

science programs such as the remarks of Hamming considered in Chapter 2, and the panel sessions at the Second SIGCSE Symposium and the Fourth SIGCSE Symposium. These are summarized by Dodd [64] in his position statement from the latter:

> Today, undergraduate Computer Science Education teaches mechanics without teaching problem solving. Typical curricula include courses in assembler language, compiler theory, list processing, and automata theory. Every MS degree holder, and most BS degree holders, know Polish notation, and have written parts of compilers. However, few of them have ever learned to write a program that can be easily enhanced or respond to changes as new management (instructor) requirements are set forth. Even fewer can read a program and describe what it does or debug a system consisting of ten or more modules.[2]

When one considers that Walker's study indicates that preparation of systems analysts was the highest ranked objective of undergraduate programs and second highest ranked objective for a master's program, while preparation of computer software systems designers was the highest ranked objective of the master's programs, these results are even more significant.

The evaluation of the graduates, especially when combined with the comments from industry, indicates a weakness in "Curriculum '68" and programs related to it in the practical programming oriented areas. These areas must be carefully considered in a revision of "Curriculum '68".

Chapter 5 looks at specific course development as reported in the literature since the publication of "Curriculum '68". The fact that such material was published points to the

2. George Dodd, "Position Statement on Industry Reaction to Computer Science Education", SIGCSE Bulletin 6, 1 (February 1974), 79.

fact that it was to share certain ideas in curriculum not widely known, and hence serves to overcome some of the shortcomings of "Curriculum '68". In the process of evaluation of "Curriculum '68" working toward revision, this work must be carefully considered as representing potentially significant contributions to the revision.

The first course is extensively discussed from the standpoint of the computer science curriculum, as well as from that of a service course, and that of a combined course to meet the objectives of both groups. The ideas of programming structure and style are also brought into these discussions.

Work in the areas of computers and society and computer appreciation courses is considered. The discrete structures course receives considerable attention as might be expected by a course having a key role in the curriculum recommendations while showing far from universal acceptance in curricula implementation. Additionally a new course in computational analysis and further discussion of additional theory courses are considered.

Operating systems or systems programming has considerable coverage including the extensive report of the COSINE Committee. Related to this is the discussion of pedagogical software and of special computing facilities teaching computer science.

Additional courses considered include computer organization, computer architecture software engineering and business

related courses.

Chapter 6 considered subsequent work of C^3S and other groups involved in curriculum planning. In some cases this demonstrates areas where C^3S is addressing or has addressed some of the shortcomings of "Curriculum '68", in others it represents studies that must be evaluated and potential solutions to such shortcomings.

Aspects of the doctoral program have been addressed in the Communications of the ACM. A subcommittee of C^3S has looked at the special problems of computer science at small colleges and recommended a sequence of four courses to serve as a core for students planning to go on in computing, and also to meet various service needs. A subcommittee is studying the area of service courses and another computers and society courses. The master's program has been considered further and an initial set of recommendations has been presented.

The ACM Curriculum Committee on Computer Education for Management has prepared recommendations for both graduate and undergraduate programs in information systems designed to meet and overcome some of the objections that computer science programs are not responding to the need of business. Armstrong [10] summarizes the objectives of this committee's work in the Proceedings of the Second SIGCSE Symposium:

> As a member of the ACM Curriculum Committee on Computer Science Education for Management, I have been involved in the development of a curriculum for a professional Master's degree in Information Systems. It is

> the opinion of this committee that their curriculum proposal represents a viable approach to the current and projected needs of organizations - within the current university framework. Computer Science Departments will be called upon to support certain aspects of this curriculum - as will the business schools, and the operations research people. The desire is for an integrated approach - with the additional infusion of the experience of industry practitioners. A challenge exists in integrating experience into such a program in a meaningful fashion without violating the aims of the university.[3]

Additionally the COSINE Committee of the Commission on Engineering Education has been active in preparation of various aspects of computer science and how it relates to electrical engineering, while the Committee on the Undergraduate Program in Mathematics has prepared recommendations for appropriate computer science work in a mathematics program.

Work has been undertaken in the secondary school area by the International Federation of Information Processing and by the Conference Board of the Mathematical Sciences. ACM has recently formed a group composed from members of C^3S and the ACM Committee on Secondary School Programs to investigate and recommend guidelines for the training of secondary school teachers.

Summary and conclusions

"Curriculum '68" has proved an important source in the planning and development of academic programs in computer science. One only need look at the recent listings of current

3. Russell M. Armstrong, "Industry's Need and Computer Science Departments", SIGCSE Bulletin 4, 1 (March 1972), 73.

offerings to see this influence, and this cannot measure all of the impact which must also include the various cases where materials have been introduced after a study of the report indicated a different approach was necessary.

Like all documents, however, "Curriculum '68" has its shortcomings and is in need of revision, modification, and updating, though the evidence of experience indicates that a complete new set of recommendations is probably not necessary. On the other hand, some items such as updating of bibliographies is obviously necessary.

Other areas, including service courses, the relationship to business data processing, teacher training programs, and courses reflecting advances in computer technology must be considered. In considering these areas, clearly certain areas and topics of interest and importance at the time of the preparation of the "Curriculum '68" must be reconsidered.

Pedagogical questions such as the role of the ideas expressed in the recent developments in programming style and structure, developments in the areas of pedagogical languages and their implications on equipment requirements and course development, and the availability of minicomputers and their implication on the curriculum must be evaluated.

Specific courses recommended in the report which have not been widely accepted, or which have not achieved the key position prescribed in the prerequisite structure of "Curriculum '68" must be reevaluated.

At the meeting of C^3S at the 1974 National Computer Conference it was determined that all efforts should be made to make "Curriculum '68" more flexible and timely. To this end commentaries are to be prepared on the business area, the operating systems area, the hardware area, the service area, the information retrieval area, the theoretical area, the computer engineering area, and the relationship of two year programs to the university programs.

Computer Science Education since "Curriculum '68" has a rich if not exhaustive literature, covering approaches and activities in many areas that represent omissions and shortcomings of "Curriculum '68". In the development of these commentaries and the revision of the report, it is essential that the experience reflected in this literature be fully utilized.

Curriculum recommendations serve a variety of needs among which are guidelines to schools developing programs, and models by which information on courses and programs can be explained and measured. Granted the youth of the computer field, and that there is a continued need for computer science graduates at least at the bachelor's and master's level[4] such recommendations are important. Work on an update and revision of "Curriculum '68" is necessary and will be a most welcome addition to the literature of Computer Science Education.

4. John W. Hamblen, Computer Manpower - Supply and Demand - by States, Information Systems Consultants, R. R. 1 box 256A, St. James, Missouri, 1973.

Bibliography

1. Adams, J. M. and D. H. Haden. "Introductory Service Courses in the Computer Science Curriculum". SIGCSE Bulletin 4, 1 (March 1972), 49-52.

2. Adams, J. Mack, William H. Inmon and Jim Shirley. "PL/I in the Computer Science Curriculum". SIGCSE Bulletin 4, 1 (March 1972), 116-126.

3. Aiken, Robert M.. "Purpose, Goals, and Activities of SIGCSE". Proceedings of the World Conference on Computer Education 1970. Science Associates/International, New York, 1970, II/153-II/156.

4. Amarel, Saul. "Computer Science: A Conceptual Framework for Curriculum Planning". Communications of the ACM 14, 6 (June 1971), 391-401.

5. Amarel, Saul. "A Set of Goals and Approaches for Education in Computer Science". Proceedings of the AFIPS 1972 SJCC. AFIPS Press, Montvale, New Jersey, 1972, 841-846.

6. Arden, Bruce W.. "On Introducing Digital Computing". Communications of the ACM 7, 4 (April 1964), 212-214.

7. Arden, Bruce W.. "The Role of Programming in a Ph. D. Computer Science Program". Communications of the ACM 12, 1 (January 1969), 31-37.

8. Arden, Bruce W., Larry K. Flanigan and Bernard A. Galler. "An Advanced System Programming Course". Proceedings of the IFIP Congress 71. North-Holland Publishing, Amsterdam, The Netherlands, 1972, 1510-1514.

9. Armstrong, Russell M. and Emmett K. Platt. "Business and The University Computer Science Department: The Left-Hand Side of a Dialogue". SIGCSE Bulletin 2, 3 (November 1970), 6-8.

10. Armstrong, Russell M.. "Industry's Need and Computer Science Departments". SIGCSE Bulletin 4, 1 (March 1972), 73.

11. Armstrong, Russell M.. "Continuing Education in Information System Development". Proceedings of the ACM 1972 Annual Conference. ACM, New York, 1972, 130-133.

12. Armstrong, Russell M.. "Industry's Need and Computer Science Departments". SIGCSE Bulletin 4, 3 (October 1972), 41-43.

13. Arsac, Jacque J.. "Informatics and Computer Education". Proceedings of the World Conference on Computer Education 1970. Science Associates/International, New York, 1970, I/69-I/72.

14. Ashenhurst, Robert L.. "Balance in Computer Science Education". Proceedings of the World Conference on Computer Education 1970. Science Associates/International, New York, 1970, II/157-II/160.

15. Ashenhurst, R. (ed.). "Curricula Recommendations for Graduate Professional Programs in Information Systems". Communications of the ACM 15, 5 (May 1972), 363-398.

16. Ashenhurst, Robert L.. "Curriculum Recommendations for a Master's Program in Information Systems Development". Proceedings of the ACM 1972 Annual Conference. ACM, New York, 1972, 134-137.

17. Ashenhurst, Robert L.. "Implications for Computer Science Departments of the ACM Information Systems Curriculum". SIGCSE Bulletin 5, 1 (February 1973), 2-5.

18. Atchison, William F. and John W. Hamblen. "Status of Computer Science Curricula in Colleges and Universities". Communications of the ACM 7, 4 (April 1964), 225-227.

19. Atchison, William F. (Chairman). "General Computer Education: Edited Session from ACM 70". in R. W. Bemer (ed.), Computers and Crisis. ACM, New York, 1971, 96-101.

20. Atchison, William F. "Computer Science Preparation for Secondary School Teachers". SIGCSE Bulletin 5, 1 (February 1973), 45-47.

21. Austing, Richard H. "A Lower Division Course Sequence in Computer Science". Proceedings of the World Conference on Computer Education 1970. Science Associates/International, New York, 1970, II/167-II/170.

22. Austing, Richard H. and Gerald L. Engel. "A Computer Science Course Program for Small Colleges". Communications of the ACM 16, 3 (March 1973), 139-147.

23. Barnes, Bruce H. and Malcolm H. Gotterer. "Undergraduate Education in Computer Science: Some Fundamental Problems and their Solution". Proceedings of the World Conference on Computer Education 1970. Science Associates/International, New York, 1970, II/171-II/174.

24. Barnes, Bruce H. and Malcolm H. Gotterer. "Attributes of Computer Professionals". in Theodore Willoughby (ed.),

Proceedings of the Ninth Annual Computer Personnel Research Conference. ACM, 1971, 167-176.

25. Bateman, Barry L. and Gerald N. Pitts. "Computer Science as a Foreign Language Substitute". SIGCSE Bulletin 5, 1 (February 1973), 132-133.

26. Bauer, Michael A.. "A Student-Designed Undergraduate Program". SIGCSE Bulletin 2, 3 (November 1970), 13-17.

27. Beidler, John A.. "A Machine Independent Course in Processor Organization and Assembler Language Programming". SIGCSE Bulletin 5, 1 (February 1973), 149-152.

28. Belzer, Jack. "Education in Information Science". Journal of the American Society for Information Science 21, 4 (July-August 1970), 269-273.

29. Benson, Robert. "The Computer Science/Industry Gap: The Educational Issues". SIGCSE Bulletin 4, 3 (October 1972), 38-41.

30. Blount, S. E. and L. Fein. "The Practical Aspect of Computer Science--Discussion". Communications of the ACM 16, 1 (January 1973), 45-46.

31. Brady, Allen H.. "The Introductory and Service Courses in Computing: Some Experiences and a Critical Assessment". SIGCSE Bulletin 2, 2 (June-July 1970), 31-37.

32. Brillinger, P. C. and D. D. Cowan. "A Complete Package for Introducing Computer Science". SIGCSE Bulletin 2, 3 (November 1970), 118-126.

33. Brown, D. C.. "The Project, and the Future of Computing Science Courses". The Computer Journal 16, 4 (November 1973), 380-381.

34. Caffrey, John. "Computers in Higher Education". in D. D. Bushnell and D. W. Allen (eds.), The Computer in American Education. John Wiley and Sons, Inc., New York, 1967, 216-225.

35. Charp, Sylvia. "Computer Programming Courses in Secondary Schools". in D. D. Bushnell and D. W. Allen (eds.), The Computer in American Education. John Wiley and Sons, Inc., New York, 1967, 137-155.

36. Clark, Douglas. "Hardware Systems in the Core Curriculum of a Computer Science Ph. D. Program". SIGCSE Bulletin 6, 1 (February 1974), 106-110.

37. Coates, C. L. et al. "An Undergraduate Computer Engineering Option for Electrical Engineering". Proceedings of the IEEE 59, 6 (June 1971), 854-860.

38. Committee on Computer Education. Recommendations Regarding Computers in High School Education. Conference Board of the Mathematical Sciences, Washington, D. C., April 1972.

39. Committee on the Undergraduate Program in Mathematics (CUPM). Recommendations on the Undergraduate Mathematics Program for Work in Computing. CUPM, Berkeley, California, 1964.

40. Committee on the Undergraduate Program in Mathematics (CUPM). A General Curriculum in Mathematics for Colleges. CUPM, Berkeley, California, 1965.

41. Committee on the Undergraduate Program in Mathematics (CUPM). A Curriculum in Applied Mathematics. CUPM, Berkeley, California, 1966.

42. Committee on the Undergraduate Program in Mathematics (CUPM). Recommendations in the Undergraduate Mathematics Program for Engineers and Physicists. CUPM, Berkeley, California, 1967.

43. Committee on the Undergraduate Program in Mathematics (CUPM). Recommendations for an Undergraduate Program in Computational Mathematics. CUPM, Berkeley, California, 1971.

44. Connolly, Frank W.. "A Community/Junior College View of Curriculum '68". SIGCSE Bulletin 5, 1 (February 1973), 68-69.

45. Conte, Sam. "History and Activities of the ACM Curriculum Committee". in William Viavant (ed.), Proceedings of the Park City Conference on Computers in Undergraduate Education. The University of Utah, Salt Lake City, 1969, 38-50.

46. COSINE Committee. Computer Science in Electrical Engineering. Commission on Engineering Education, Washington, D. C., September 1967.

47. COSINE Committee. Some Specifications for a Computer-Oriented First Course in Electrical Engineering. Commission on Engineering Education, Washington, D. C., September 1968.

48. COSINE Committee. An Undergraduate Electrical Engineering Course on Computer Organization. Commission on Engineering Education, Washington, D. C., October 1968.

49. COSINE Committee. Some Specifications for an Undergraduate Course in Digital Subsystems. National Academy of Engineering, Washington, D. C., November 1968.

50. COSINE Committee. Impact of Computers on Electrical Engineering Education - A View from Industry. National Academy of Engineering, Washington, D. C., September 1969.

51. COSINE Committee. Digital Systems Laboratory Courses and Laboratory Developments. National Academy of Engineering, Washington, D. C., March 1971.

52. COSINE Committee. An Undergraduate Course on Operating Systems Principles. National Academy of Engineering, Washington, D. C., April 1972.

53. COSINE Committee. Minicomputers in the Digital Laboratory Program. National Academy of Engineering, Washington, D. C., April 1972.

54. Couger, J. D.. "The Undergraduate Program in Information Systems Development". Proceedings of the ACM 1972 Annual Conference. ACM, New York, 1972, 138-144.

55. Couger, J. Daniel. "Curriculum Recommendations for Undergraduate Programs in Information Systems". Communications of the ACM 16, 12 (December 1973), 727-749.

56. Cowan, D. D. and R. B. Roden. "A Large-Scale Undergraduate Programme in Computer Science". SIGCSE Bulletin 2, 3 (November 1970), 18-23.

57. Curriculum Committee on Computer Science (C3S). "An Undergraduate Program in Computer Science, Preliminary Recommendations". Communications of the ACM 8, 9 (September 1965), 543-552.

58. Curriculum Committee on Computer Science (C3S). "Curriculum '68, Recommendations for Academic Programs in Computer Science". Communications of the ACM 11, 3 (March 1968), 151-197.

59. de Campo, Leila. "Introducing the Computer at a Small Liberal Arts College". SIGCSE Bulletin 2, 3 (November 1970), 113-117.

60. Denning, Peter J.. "Principles of Computer System Organization". SIGCSE Bulletin 2, 3 (November 1970), 45-55.

61. Denning, Peter J.. Operating Systems Principles and Undergraduate Computer Science Curricula. Technical Report TR-99, Computer Science Laboratory, Department of Electrical Engineering, Princeton University, September 1971.

62. Denning, Peter J.. "Operating Systems Principles and Undergraduate Computer Science Curricula". Proceedings of the AFIPS 1972 SJCC. AFIPS Press, Montvale, New Jersey, 1972, 849-855.

63. Dennis, Jack B.. "Inter-relating Hardware and Software in Computer Science Education". Proceedings of the AFIPS 1969 SJCC. AFIPS Press, Montvale, New Jersey, 1969, 537-538.

64. Dodd, George. "Position Statement on Industry Reaction to Computer Science Education". SIGCSE Bulletin 6, 1 (February 1974), 79.

65. Eckhouse, Richard H.. "The Computer Science Laboratory". SIGCSE Bulletin 4, 1 (March 1972), 42-45.

66. Ein-Dor, Phillip and Norman Lyons. "Systems Analysis in Computer Science Education". SIGCSE Bulletin 2, 5 (December 1970), 16-21.

67. Elliott, Roger W.. "Master's Level Computer Science Curricula". Communications of the ACM 11, 7 (July 1968), 507-508.

68. Ellis, Robert A. and Donald F. Wann. "Teaching Computer Design Using Macromodules". SIGCSE Bulletin 4, 1 (March 1972), 160-162.

69. Engel, G. L.. "Programming Systems Workshop". SIGCSE Bulletin 1, 3 (October 1969), 10-27.

70. Engel, Gerald L.. "Computer Science Instruction in Small Colleges - An Initial Report". SIGCSE Bulletin 3, 2 (June 1971), 8-18.

71. Engel, Gerald L.. "Input from ACM Curriculum Committee on Computer Science". SIGCSE Bulletin 3, 4 (December 1971), 30-39.

72. Engel, Gerald L. and Bruce H. Barnes. "The Effect of Environment on Computer Science Education". SIGCSE Bulletin 4, 1 (March 1972), 13-18.

73. Engel, G. L. and N. D. Jones. "Discrete Structures in the Undergraduate Computer Science Curriculum". SIGCSE Bulletin 5, 1 (February 1973), 56-59.

74. Finerman, A.. University Education in Computing Science. Academic Press, New York, 1968.

75. Finerman, Aaron and Anthony Ralston. "Undergraduate Programs in Computing Science in the Tradition of Liberal Education". Proceedings of the World Conference on Computer Education 1970. Science Associates/International, New York, 1970, II/195-II/200.

76. Fischer, P. C.. "Theory of Computing in Computer Science Education". Proceedings of the AFIPS 1972 SJCC. AFIPS Press, Montvale, New Jersey, 1972, 857-864.

77. Fisher, P., W. Hankley and V. Wallentine. "Separation of Introductory Programming and Language Instruction". SIGCSE Bulletin 5, 1 (February 1973), 9-14.

78. Forsythe, George E.. "An Undergraduate Curriculum in Numerical Analysis". Communications of the ACM 7, 4 (April 1964), 214-215.

79. Forsythe, G.. "A University Educational Program in Computer Science". Communications of the ACM 10, 1 (January 1967), 3-11.

80. Forsythe, George E.. "Let's not Discriminate Against Good Work in Design or Experimentation". Proceedings of the AFIPS 1969 SJCC. AFIPS Press, Montvale, New Jersey, 1969, 538-539.

81. Forsythe, George E. (Chairman). "Graduate Education: Edited Session from ACM 70". in R. W. Bemer (ed.), Computers and Crisis. ACM, New York, 1971, 109-117.

82. Galler, B. A., R. Wagman and J. Bravatto. "CRISP: An Interactive Student Registration System". Proceedings of the ACM 1973 Annual Conference. ACM, New York, 1973, 283-289.

83. Gluckson, Fred A.. "Position Statement on Industry Reaction to Computer Science Education". SIGCSE Bulletin 6, 1 (February 1974), 79.

84. Goddard, Alton R.. "Structure and Content of Service Courses in Computer Science for Other Disciplines". SIGCSE Bulletin 5, 1 (February 1973), 15-17.

85. Gorn, Saul. "The Computer and Information Sciences: A New Basic Discipline". SIAM Review 5, 2 (April 1963), 150-155.

86. Gorn, Saul. "Mechanical Languages: A Course Specification". Communications of the ACM 7, 4 (April 1964), 219-222.

87. Gorn, Saul. "The Computer and Information Sciences and the Community of Discipline". Behavioral Science 12, 6 (November 1967), 433-452.

88. Gorsline, George W. and Duff Green. "Computer Science Education Through a Rearview Mirror: Experience with Curriculum '68 at Virginia Polytechnic Institute and State University". SIGCSE Bulletin 5, 1 (February 1973), 102-105.

89. Gotterer, Malcolm H. and Bruce H. Barnes. "The Computer Science M. S. Graduate". SIGCSE Bulletin 5, 1 (February 1973), 106-109.

90. Graham, Robert M.. "Teaching Systems Programming and Software Design, Problems and Solutions". SIGCSE Bulletin 2, 3 (November 1970), 56-60.

91. Greenawalt, E. M. and D. I. Good. "The MIX Computer as an Educational Tool". Proceedings of the ACM 1972 Annual Conference. ACM, New York, 1972, 302-309.

92. Gregory, R. T.. "Review of Curriculum '68". Computing Reviews (Review Number 14,390) 9, 6 (June 1968), 304-305.

93. Gries, David. "What Should we Teach in an Introductory Programming Course?". SIGCSE Bulletin 6, 1 (February 1974), 81-89.

94. Hale, John E.. "Remarks by Representatives of Computer Manufacturers". in William Viavant (ed.), Proceedings of the Park City Conference on Computers in Undergraduate Education. The University of Utah, Salt Lake City, 1969, 107-111.

95. Hamblen, John W.. "Computer Science and Related Degree Programs in U. S. Higher Education". SIGCSE Bulletin 1, 4 (December 1969), 9-13.

96. Hamblen, John W.. "Computer and Related Degree Programs in U. S. Higher Education Through June 30, 1967". Proceedings of the World Conference on Computer Education 1970. Science Associates/International, New York, 1960, II/201-II/216.

97. Hamblen, John W.. "Using Computers in Higher Education: Past Recommendations, Status and Needs". Communications of the ACM 14, 11 (November 1971), 709-712.

98. Hamblen, John W.. "Degree Programs in Computer Science, Data Processing, etc. Offered by Institutions of Higher Education During 1969-70, 1970-71, and 1971-72". SIGCSE Bulletin 4, 2 (July 1972), 29-39.

99. Hammer, Preston C.. "Computer Science and Mathematics". Proceedings of the World Conference on Computer Education 1970. Science Associates/International, New York, 1970, I/65-I/67.

100. Hammer, Preston C.. "Undergraduate Computer Science Education". SIGCSE Bulletin 2, 3 (November 1970), 1-5.

101. Hammer, P. C. (Chairman). "Undergraduate Education: Edited Session from ACM 70". in R. W. Bemer (ed.), Computers and Crisis. ACM, New York, 1971, 102-108.

102. Hamming, R. W.. "One Man's View of Computer Science". Journal of the ACM 16, 1 (January 1969), 3-12.

103. Hebenstreit, J.. "A Curriculum in Computer Science Oriented Toward Computer Design". Proceedings of the IFIP Congress 1968 (Applications 2, Booklet G). North-Holland Publishing, Amsterdam, The Netherlands, 1968, 53-57.

104. Heilman, Robert L. and Gordon P. Ashby. "Re-evaluation of Debugging in the Computer Science Curriculum". SIGCSE Bulletin 3, 4 (December 1971), 15-18.

105. Heimer, Ralph T. and Lars C. Jansson. "Teacher Training in Computer Education". SIGCSE Bulletin 5, 1 (February 1973), 48-50.

106. Horowitz, E., H. L. Morgan and A. C. Shaw. "Computers and Society: A Proposed Course for Computer Scientists". Communications of the ACM 15, 4 (April 1972), 257-261.

107. Horowitz, E. and M. C. Horowitz. "Computers and Society: An Interdisciplinary Approach". SIGCSE Bulletin 5, 1 (February 1973), 134-137.

108. Hosch, Frederick A.. "Some Comments on the Role of Computer Science Education". SIGCSE Bulletin 5, 3 (September 1973), 13-17.

109. Hunt, Earl. "The Computer Science Teaching Laboratory at the University of Washington". SIGCSE Bulletin 2, 3 (November 1970), 30-33.

110. IFIP Technical Committee for Education. Computer Education in Secondary Schools, An Outline Guide for Teachers. IFIP, Amsterdam, The Netherlands, 1970.

111. IFIP Technical Committee for Education. Computer Education in Secondary Schools, An Outline Guide (revised edition). IFIP, Amsterdam, The Netherlands, 1971.

112. Johnson, David L.. "Computers in the Engineering College Curriculum". Journal of Engineering Education 58, 8 (April 1968), 909-911.

113. Joubert, G. R. and L. Ottilie Stander. "A New Approach to the Presentation of Computer Science Courses". The Computer Bulletin 14, 10 (October 1970), 342-343.

114. Kalmey, Donald L.. "Profile of a Computer and Information Science B. S. Graduate". SIGCSE Bulletin 6, 1 (February 1974), 40-45.

115. Kandel, Abraham. "Computer Science - A Vicious Circle". Communications of the ACM 15, 6 (June 1972), 470-471.

116. Kandel, Abraham. "Computer Science - Seminars for Undergraduates". Communications of the ACM 16, 7 (July 1973), 442.

117. Keenan, Thomas A.. "Computers and Education". Communications of the ACM 7, 4 (April 1964), 205-209.

118. Kerighan, B. W. and P. J. Plauger. "Programming Style". SIGCSE Bulletin 6, 1 (February 1974), 90-96.

119. Korfhage, Robert R.. "Logic for the Computer Sciences". Communications of the ACM 7, 4 (April 1964), 216-218.

120. Kuno, Susumu and A. G. Oettinger. "Computational Linguistics in a Ph. D. Computer Science Program". Communications of the ACM 11, 12 (December 1968), 831-836.

121. La France, Jacques and R. Waldo Roth. "Computer Science for Liberal Arts Colleges". SIGCSE Bulletin 5, 1 (February 1973), 70-76.

122. Lee, Hans E.. "Computers in Society--A Course Description, Purpose and Rationale". SIGCSE Bulletin 4, 1 (March 1972), 97-102.

123. Leininger, C. W.. "Computer Related Studies at a College of Arts, Sciences and Education". SIGCSE Bulletin 4, 3 (October 1972), 19-20.

124. Lynn, M. Stewart. "Computer Science Education - The Need for Interaction". Proceedings of the AFIPS 1972 SJCC. AFIPS Press, Montvale, New Jersey, 1972, 847-848.

125. Magleby, Kay. "Remarks by Representatives of Computer Manufacturers". in William Viavant (ed.), Proceedings of the Park City Conference on Computers in Undergraduate Education. The University of Utah, Salt Lake City, 1969, 112-121.

126. Mapp, George A.. "A Proposal for a B. S. in Information Systems". SIGCSE Bulletin 5, 1 (February 1973), 91-96.

127. Marsland, T. A. and J. Tartar. "A Course in Minicomputer Systems". SIGCSE Bulletin 5, 1 (February 1973), 153-156.

128. Mashey, J. R., G. M. Campbell and C. Forney. "ASSIST - A Self Modifiable Assembler for Instructional Purposes". Proceedings of the ACM 1972 Annual Conference. ACM, New York, 1972, 310-312.

129. Mashey, John R.. "ASSIST: Three Years Experience with a Student-Oriented Assembler". SIGCSE Bulletin 5, 1 (February 1973), 157-165.

130. Mathis, R. F. and M. C. Yovits. "Computer and Information Science in Engineering Education". Engineering Education 61, 4 (January 1971), 340-342.

131. Mathis, Robert F. and Douglas S. Kerr. "Development of a Multifacetted Undergraduate Program in Computer and Information Science". SIGCSE Bulletin 4, 1 (March 1972), 8-12.

132. Mathis, Robert F.. "Teaching Debugging". SIGCSE Bulletin 6, 1 (February 1974), 59-63.

133. Matula, David W.. "The Emergence of Computational Arithmetic as a Component of the Computer Science Curriculum". SIGCSE Bulletin 2, 3 (November 1970), 41-44.

134. McFarlan, F. Warren and Richard L. Nolan. "Curriculum Recommendations for Graduate Professional Programs in Information Systems: Recommended Addendum on Information Systems Administration". Communications of the ACM 16, 7 (July 1973), 439-441.

135. McGee, Pat. "Computer Science Graduates - Industry/ University Gap?". SIGCSE Bulletin 4, 3 (October 1972), 44-56.

136. McNaughton, Robert. "Automata, Formal Languages, Abstract Switching and Computability in a Ph. D. Computer Science Program". Communications of the ACM 11, 11 (November 1968), 738-740, 746.

137. Melkanoff, M. A.. "An M. S. Program in Computer Science". SIGCSE Bulletin 5, 1 (February 1973), 77-82.

138. Melkanoff, M. A., B. H. Barnes and G. L. Engel. "The Masters Degree Program in Computer Science". Proceedings of the AFIPS 1973 NCC. AFIPS Press, Montvale, New Jersey, 1973, 563.

139. Menninga, Larry D.. "Introducing Practical Experience into Curriculum 68 Through Integration of Courses". SIGCSE Bulletin 6, 1 (February 1974), 152-154.

140. Minsky, Marvin. "Speculations about Man and Machines". in William Viavant (ed.), Proceedings of the Park City Conference on Computers in Undergraduate Education. The University of Utah, Salt Lake City, 1969, 145-185.

141. Muller, David E.. "The Place of Logical Design and Switching Theory in the Computer Curriculum". Communications of the ACM 7, 4 (April 1964), 222-225.

142. Naur, Peter. "Datalogy, the Science and Data and Data Processes and its Place in Education". Proceedings of the IFIP Congress 1968 (Applications 2, Booklet G). North-Holland Publishing, Amsterdam, The Netherlands, 1968, 48-52.

143. Oliver, James R.. "The Need to Upgrade Computer Science Curricula". SIGCSE Bulletin 5, 4 (December 1973), 14-18.

144. Organick, E. I.. "Review of Curriculum '68". Computing Reviews (Review Number 14,389) 9, 6 (June 1968), 303-304.

145. Parnas, D. L.. "A Course on Software Engineering Techniques". SIGCSE Bulletin 4, 1 (March 1972), 154-159.

146. Perlis, A. J.. "Programming of Digital Computers". Communications of the ACM 7, 4 (April 1964), 210-214.

147. Perlis, Alan J.. "Identifying and Developing Curricula in Software Engineering". Proceedings of the AFIPS 1969 SJCC, AFIPS Press, Montvale, New Jersey, 1969, 540-541.

148. Piore, E. R.. "Challenges to Progress in Computing". SIGCSE Bulletin 1, 3 (October 1969), 7-9.

149. Pitts, Gerald N. and Roy S. Ellzey. "Computer Science - A Professional Degree". SIGCSE Bulletin 5, 4 (December 1973), 8-11.

150. President's Science Advisory Committee. Computers in Higher Education. The White House, Washington, D. C., February 1967.

151. Rahimi, M. A. and H. G. Hedges. "Evolution of a Computer Science Academic Program in a College of Education". SIGCSE Bulletin 5, 1 (February 1973), 110-114.

152. Ralston, Anthony. "FORTRAN and the First Course in Computer Science". SIGCSE Bulletin 3, 4 (December 1971), 24-29.

153. Rechard, Ottis W.. "The Computer Sciences in Colleges and Universities". in D. D. Bushnell and D. W. Allen (eds.), The Computer in American Education. John Wiley and Sons, Inc., New York, 1967, 156-164.

154. Rechard, Ottis W.. "ACM Curriculum Committee Proposal". SIGCSE Bulletin 2, 2 (June-July 1970), 28-30.

155. Rosen, S.. "Review of Curriculum '68". Computing Reviews (Review Number 14,391) 9, 6 (June 1968), 305-306.

156. Rosin, Robert F.. "Teaching about Programming". Communications of the ACM 16, 7 (July 1973), 435-439.

157. Roth, R. Waldo. "A Computer Science Curriculum for a Liberal Arts College". SIGCSE Bulletin 4, 3 (October 1972), 20-35.

158. Rubinoff, Morris. "The Computer and Information Sciences Programs at the University of Pennsylvania". Proceedings of the World Conference on Computer Education 1970. Science Associates/International, New York, 1970, II/377-II/381.

159. Salton, G.. "Information Science in a Ph. D. Computer Science Program". Communications of the ACM 12, 2 (February 1969), 111-117.

160. Salton, Gerard. "Introductory Programming at Cornell". SIGCSE Bulletin 5, 1 (February 1973), 18-20.

161. Sampson, Jeffrey R.. "An Introductory Adaptive Systems Course for Undergraduate Computer Science Majors". SIGCSE Bulletin 6, 1 (February 1974), 148-151.

162. Schwenkel, Frieder. "The Undergraduate Computer Science Program at the University of Notre Dame". SIGCSE Bulletin 2, 5 (December 1970), 11-15.

163. Schweppe, Earl (Chairman). "Summary of Workshop Meeting, Curriculum and Programs". in William Viavant (ed.), Proceedings of the Park City Conference on Computers in Undergraduate Education. The University of Utah, Salt Lake City, 1969, 187-192.

164. Schweppe, Earl J. (Chairman). "Organizing for Computer Science Education: Edited Session from ACM 70". in R. W. Bemer (ed.), Computers and Crisis. ACM, New York, 1971, 118-125.

165. Semple, Wolsey A.. "Evolution of a Computer Science Program". SIGCSE Bulletin 5, 1 (February 1973), 115-118.

166. Setzer, Valdermer W. and Charles H. Warlick. "A Unified Approach to Compiler Theory and Construction". Proceedings of the IFIP Congress 71. North-Holland Publishing, Amsterdam, The Netherlands, 1972, 1523-1529.

167. Shaw, Alan C. and Nelson H. Weiderman. "A Multiprogramming System for Education and Research". Proceedings of the IFIP Congress 71. North-Holland Publishing, Amsterdam, The Netherlands, 1972, 1505-1509.

168. Sistare, John H. and Norman E. Sondak. "Introduction to Digital Computer Programming - An IPI Approach". SIGCSE Bulletin 6, 1 (February 1974), 184-194.

169. Sloan, M. E.. "Computer Architecture in U. S. and Canadian Electrical Engineering Departments". SIGCSE Bulletin 6, 1 (February 1974), 111-115.

170. Stark, Richard H.. "Computer Science Needs its Laboratory". SIGCSE Bulletin 4, 1 (March 1972), 46-48.

171. Sterling, T. and S. Pollack. "Experience with a 'Universal' Introductory Course in Computer Science". SIGCSE Bulletin 2, 3 (November 1970), 106-112.

172. Tartar, J. and J. P. Penny. "Undergraduate Education in Computing Science - Some Immediate Problems". SIGCSE Bulletin 4, 1 (March 1972), 1-7.

173. Teichroew, D. (ed.). "Education Related to the Use of Computers in Organizations". Communications of the ACM 14, 9 (September 1971), 573-588.

174. Thomas, Richard T.. "Computer Architecture in the Computer Science Curriculum". SIGCSE Bulletin 6, 1 (February 1974), 116-120.

175. Tracz, Will. "The Use of ATOPSS: For Presenting Elementary Operating System Concepts". SIGCSE Bulletin 6, 1 (February 1974), 74-78.

176. Tremblay, J. P. and R. Manohar. "A First Course in Discrete Structures with Applications to Computer Science". SIGCSE Bulletin 6, 1 (February 1974), 155-160.

177. van Dam, Andries, Charles M. Strauss, Charles McGowan, and Jean Morse. "A Survey of Introductory and Advanced Programming Courses". SIGCSE Bulletin 6, 1 (February 1974), 174-183.

178. Viavant, William (ed.). Proceedings of the Park City Conference on Computers in Undergraduate Education. University of Utah, Salt Lake City, 1969.

179. Vickers, F. D.. "Data on Computer Science Departments/ Curricula". SIGCSE Bulletin 3, 4 (December 1971), 40-45.

180. Walker, Justin C. and Charles E. Hughes. "POPSS - A Parametric Operating System Simulator". SIGCSE Bulletin 5, 1 (February 1973), 166-172.

181. Walker, Terry M.. "Computer Science Curricula Survey". SIGCSE Bulletin 5, 4 (December 1973), 19-28.

182. Wegner, Peter. "Problems of Computer Science Education in Small Colleges". SIGCSE Bulletin 3, 3 (September 1971), 15-18.

183. Wegner, Peter. "A View of Computer Science Education". American Mathematical Monthly 79, 2 (February 1972), 168-179.

184. Wegner, Peter. "A View of Computer Science Education". Proceedings of the IFIP Congress 71. North-Holland Publishing, Amsterdam, The Netherlands, 1972, 1515-1522.

185. Weinberg, Bernhard and Leonard H. Weiner. "A Systems Programming Course Using the HMS 5050, A Counterfeit, Hands-on, Large-Scale Computer System". SIGCSE Bulletin 6, 1 (February 1974), 64-73.

186. Willoughby, Theodore C.. "Student Attitudes Toward Computers". SIGCSE Bulletin 5, 1 (February 1973), 145-148.

187. Wright, Albert H.. "Education of the Computer Professional". Proceedings of the World Conference on Computer Education 1970. Science Associates/International, New York, 1970, II/149-II/152.

188. Yeh, Raymond T., Donald I. Good and David R. Musser. "New Directions in Teaching the Fundamentals of Computer Science--Discrete Structures and Computational Analysis". SIGCSE Bulletin 5, 1 (February 1973), 60-67.

189. Yovits, M. C.. "Information Science: Toward the Development of a True Scientific Discipline". American Documentation 20, 4 (October 1969), 369-376.

190. Zadeh, L. A.. "Computer Science as a Discipline". Journal of Engineering Education 58, 8 (April 1968), 913-916.

191. Zadeh, L.. "Applied Computer Science". Proceedings of the AFIPS 1969 SJCC. AFIPS Press, Montvale, New Jersey, 1969, 539-540.

Vita

Gerald Lawrence Engel was born in Cleveland, Ohio, on July 5, 1942. He attended Cleveland Heights High School graduating in 1960. He received the B. S. degree Magna Cum Laude from Hampden-Sydney College in 1964 and the M. A. degree from Louisiana State University in 1965.

Mr. Engel has taught at Randolph-Macon College and Hampden-Sydney College in addition to serving as an Instructor at The Pennsylvania State University. He has also been affiliated with the U. S. Naval Weapons Laboratory.

He is presently Head of the Department of Computing and Statistics at The Virginia Institute of Marine Science, holding an appointment as Associate Professor in the School of Marine Science of the College of William and Mary, and the Department of Marine Science of the University of Virginia.

In August 1964, he married the former Doris E. Smith of Roanoke, Virginia. They have two children.