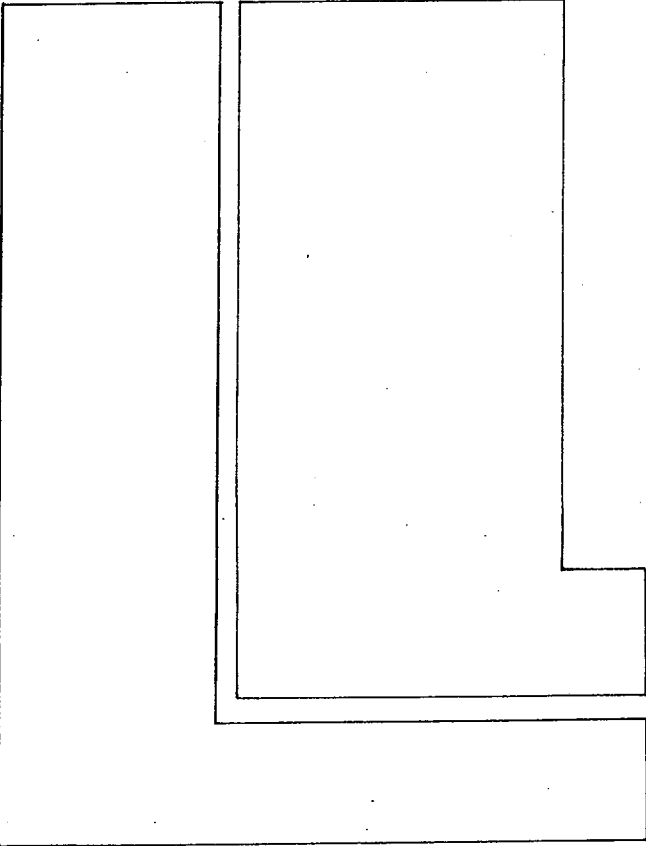


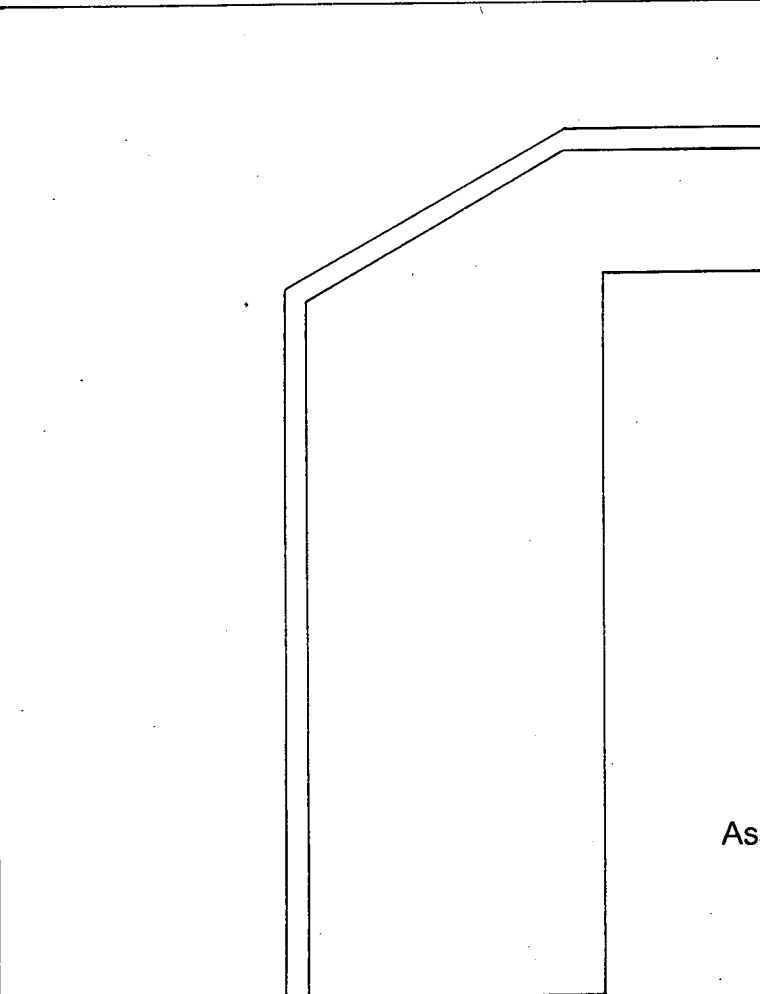
**ACM Recommended
Curricula for
Computer Science and
Information Processing
Programs in Colleges
and Universities,
1968-1981**

Compiled by the Committees
on Computer Curricula
of the
ACM Education Board

Association for Computing Machinery

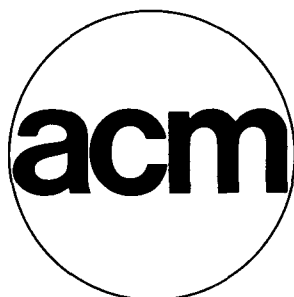


**ACM Recommended
Curricula for
Computer Science and
Information Processing
Programs in Colleges
and Universities,
1968-1981**



Compiled by the Committees
on Computer Curricula
of the
ACM Education Board

Association for Computing Machinery



**Association for Computing Machinery
1133 Avenue of the Americas
New York, NY 10036**

**Price: ACM Members: \$15.00
Non-members: \$20.00**

Copies may be ordered, prepaid, from:

**ACM Order Department
P.O. Box 64145
Baltimore, MD 21264**

Use order #201813

Copyright © 1981 by the Association for Computing Machinery

ISBN: 0-89791-058-3

CONTENTS

	Page
Curriculum 68: Recommendations for Academic Programs in Computer Science	1
Curriculum Recommendations for Graduate Professional Programs in Information Systems	49
A Computer Science Course Program for Small Colleges	85
Curriculum Recommendations for Undergraduate Programs in Information Systems	95
Curriculum 78: Recommendations for the Undergraduate Program in Computer Science	119
Recommendations for Master's Level Programs in Computer Science	139
Educational Programs in Information Systems	149
Recommendations and Guidelines for an Associate Level Degree Program in Computer Programming	159

PREFACE

When we enter the twenty-first century, computers will have influenced our lives more than any other technology known to civilization. The need for knowledge about computers, computer technology, and the theoretical aspects of computers has grown exponentially. Since its founding in 1947, the Association for Computing Machinery has attempted to address this need for computer knowledge.

The study of computers is an endeavor demanding careful, thorough, and organized development. Recognizing the need for a comprehensive model curriculum for the growing number of institutions offering computer science courses, ACM, in the mid-sixties, established the Curriculum Committee on Computer Science to make recommendations and provide guidelines to institutions. Chaired by Dr. William Atchison of the University of Maryland, this dedicated group of forward-thinking people published preliminary recommendations in September 1965. With financial assistance from the National Science Foundation, they published their final report, "Recommendations for Academic Programs in Computer Science," in *Communications of the ACM* in March 1968. This curriculum, known as "Curriculum '68," formed the basis of formal computer science study in colleges and universities for the next ten years.

Circumstances in the world of computers change constantly. As soon as a new idea emerges, as soon as a new principle is stated, as soon as a new technology is developed, it is outdated. Such is the case with curricula. As soon as "Curriculum '68" was published, many realized that there were other avenues to take, and as a consequence, other curricula were developed.

Since many small colleges could not afford a computer science program similar to the one outlined in "Curriculum '68," a small college curriculum was developed. The need for information systems curricula, quite different from computer science curricula, was recognized; those curricula were published in 1972, 1973, and 1981. In addition, specialized curricula for Associate Degree programs in Community and Junior Colleges and Doctoral level programs in Health Computing were developed, and "Curriculum '68" was updated by "Curriculum '78".

This volume includes all the curricula published as of June 1981, plus the final version of the "Curriculum on Computer Programming for Community and Junior Colleges" which appeared in draft version in the June 1977 *SIGCSE Bulletin*. These curricula are printed in order of publication dates. We realize that some of the recent curricula included will soon be updated — nothing is absolute, nothing is final. ACM's efforts in the development of curricula will not end, they will be ongoing.

Two new curricula, *A Model Curriculum for Doctoral-Level Programs in Health Computing* and *Recommendations and Guidelines for Vocation-Technical Career Programs for Computer Personnel in Operations* were published in July 1981 and are available from ACM separately.

ACM is aware that there are demands for education about computers in areas other than formal, theoretical programs. In the middle of 1981, when the computer's impact on society is so clear, we realize computer study cannot be confined to the post-secondary area. Work is now in progress to develop viable programs for the secondary and elementary schools.

No author of a preface should close without giving thanks. Over the past 15 years, each ACM administration has supported curriculum activity; those administrations are to be thanked. Also, thanks should be given to the staffs of ACM Headquarters who helped publish each curriculum, the authors of each curriculum published, as well as to the committees that supported them. I would like to list all those who have given their time and energy to make these curricula a reality, but I fear if I mention one name, than I shall forget another. Therefore, a blanket thank you is given to each person who has contributed to the development of these curricula.

William B. Gruener, Chairman
ACM Curriculum Committee
on Computer Education
November 1981

CURRICULUM 68

Recommendations for Academic Programs in Computer Science

A REPORT OF THE ACM CURRICULUM COMMITTEE ON COMPUTER SCIENCE

Dedicated to the Memory of Silvio O. Navarro

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 science is emphasized. Attention is given to the organization, staff requirements, computer resources, and other facilities needed to implement computer science educational programs.

KEY WORDS AND PHRASES: computer science courses, computer science curriculum, computer science education, computer science academic programs, computer science graduate programs, computer science undergraduate programs, computer science course bibliographies

CR CATEGORIES: 1.52

Preface

The Curriculum Committee on Computer Science (C³S) was initially formed in 1962 as a subcommittee of the Education Committee of the Association for Computing Machinery. In the first few years of its existence this subcommittee functioned rather informally by sponsoring a number of panel discussions and other sessions at various national computer meetings. The Curriculum Committee became an independent committee of the ACM in 1964 and began an active effort to formulate detailed recommendations for curricula in computer science. Its first report, "An Undergraduate Program in Computer Science—Preliminary Recommendations" [1], was published in the September 1965 issue of *Communications of the ACM*.

The work of the Committee during the last two years has been devoted to revising these recommendations on undergraduate programs and developing recommendations for graduate programs as contained in this report. The primary support for this work has been from the National Science Foundation under Grant Number GY-305, received in July 1965.

The Committee membership during the preparation of this report was:

William F. Atchison, University of Maryland (Chairman)
Samuel D. Conte, Purdue University
John W. Hamblen, SREB and Georgia Institute of Technology
Thomas E. Hull, University of Toronto
Thomas A. Keenan, EDUCOM and the University of Rochester
William B. Kehl, University of California at Los Angeles
Edward J. McCluskey, Stanford University
Silvio O. Navarro,* University of Kentucky
Werner C. Rheinboldt, University of Maryland
Earl J. Schweppe, University of Maryland (Secretary)
William Viavant, University of Utah
David M. Young, Jr., University of Texas

* Dr. Navarro was killed in an airplane crash on April 3, 1967.

In addition to these members many others have made valuable contributions to the work of the Committee. Their names and affiliations are listed at the end of this report. Robert Ashenurst and Peter Wegner have given especial assistance in the preparation of this report.

CONTENTS

1. Introduction
 2. Subject Classification
 3. Description of Courses
 4. Undergraduate Programs
 5. Master's Degree Programs
 6. Doctoral Programs
 7. Service Courses, Minors, and Continuing Education
 8. Implementation
- References
Acknowledgments
Appendix. Course Outlines and Bibliographies

1. Introduction

Following the appearance of its Preliminary Recommendations [1], the Curriculum Committee on Computer Science received many valuable comments, criticisms, and suggestions on computer science education. From these, the advice of numerous consultants, and the ideas of many other people, the Committee has prepared this report, "Curriculum 68," which is a substantial refinement and extension of the earlier recommendations. The Committee hopes that these new recommendations will stimulate further discussion in this area and evoke additional contributions to its future work from those in the computing profession. The Committee believes strongly that a continuing dialogue on the process and goals of education in computer science will be vital in the years to come.

In its Preliminary Recommendations the Committee devoted considerable attention to the justification and description of "computer science" as a discipline. Although debate on the existence of such a discipline still continues, there seems to be more discussion today on what this discipline should be called and what it should include. In a recent letter [2], Newell, Perlis, and Simon defend the name "computer science." Others, wishing perhaps to take in a broader scope and to emphasize the information being processed, advocate calling this discipline "information science" [3] or, as a compromise, "the computer and information sciences" [4]. The Committee has decided to use the term "computer science" throughout this report, although it fully realizes that other names may be used for essentially the same discipline.

In attempting to define the scope of this discipline, the Committee split computer science into three major subject divisions to which two groups of related areas were then added. Using this as a framework, the Committee developed a classification of the subject areas of computer science and some of its related fields, and this classification is presented in Section 2.

As was the case for its Preliminary Recommendations, the Committee has devoted considerable effort to the development of descriptions, detailed outlines, and bibliographies for courses in computer science. Of the sixteen courses proposed in the earlier recommendations, eleven have survived in spirit if not in detail. Two of the other five courses 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. Thus this report contains detailed information—in the form of catalog descriptions and prerequisites in Section 3 and detailed outlines and annotated bib-

liographies in the Appendix—on a total of twenty-two courses.

Another important issue which concerned the Curriculum Committee is the extent to which undergraduate programs as opposed to graduate programs in computer science ought to be advocated. Certainly, both undergraduate and graduate programs in "computer science" do now exist, and more such programs operate under other names such as "information science" or "data processing" or as options in such fields as mathematics or electrical engineering. A recent survey [5], supported by the National Science Foundation and carried out by the Computer Sciences Project of the Southern Regional Education Board, contains estimates of the number of such degree programs operating in 1964-1965 and projections of the number planned to be operating by 1968-1969. These estimates and projections can be summarized as follows:

Program name	Program level					
	Bachelor's		Master's		Doctoral	
	1964 1965	1968 1969	1964 1965	1968 1969	1964 1965	1968 1969
Computer Science	11	92	17	76	12	38
Data Processing	6	15	3	4	1	2
Information Science	2	4	12	17	4	13
Similar Programs	25	40	29	40	21	28

The information contained in these figures is interesting for two reasons. First, it shows that the number of computer science degree programs will continue to grow rapidly even if some of the programs now being planned do not come into being. Second, it shows a strong tendency to use the name "computer science," although the availability of academic work in computing is not limited to institutions having a department or a program operating under that title.

A major purpose of the Committee's recommendations on undergraduate programs and master's degree programs given in Sections 4 and 5 is to provide a sense of direction and a realizable set of goals for those colleges and universities which plan to provide computer science education for undergraduate and/or graduate students. The discussion in Section 6 of how guidelines for doctoral programs may be developed is very general, mainly because this is a difficult area in which to make detailed recommendations.

The importance of service courses, minors, and continued education in computer science has also been of concern to the Committee. Although detailed work still needs to be done, some preliminary discussion of the needs in these areas is given in Section 7. In Section 8 some of the problems of implementing an educational program in computer science are discussed.

In general the difficulties in establishing such programs are formidable; the practical problems of finding qualified faculty, of providing adequate laboratory facilities, and of beginning a program in a new area where there are few textbooks are severe. These problems are magnified for baccalaureate programs in comparison with graduate programs, where the admission can be more closely controlled.

The demand for substantially increased numbers of persons to work in all areas of computing has been noted in a report of the National Academy of Sciences-National Research Council [6] (commonly known as the "Rosser Report") and in a report of the President's Science Advisory Committee [7] (often called the "Pierce Report"). Although programs based on the recommendations of the Curriculum Committee can contribute substantially to satisfying this demand, such programs will not cover the full breadth of the need for personnel. For example, 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 applications 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. Subject Classification

The scope of academic programs and curricula in computer science will necessarily vary from institution to institution as dictated by local needs, resources, and objectives. To provide a basis for discussion, however, it seems desirable to have a reasonably comprehensive system for classifying the subject areas within computer science and related fields. Although any such system is somewhat arbitrary, it is hoped that any substantial aspect of the computer field, unless specifically excluded for stated reasons, may be found within the system presented here. The subject areas within computer science will be classified first; those shared with or wholly within related fields will be discussed later in this section.

Computer Science. The subject areas of computer science are grouped into three major divisions: "information structures and processes," "information processing systems," and "methodologies." The subject areas contained in each of these divisions are given below together with lists of the topics within each subject area.

In addition to this Committee, several other organizations have set forth guidelines to aid educational institutions in the establishment of programs pertinent to the needs of today's computer-oriented technology. Prominent among these are the reports of the Committee on the Undergraduate Program in Mathematics (CUPM) of the Mathematical Association of America [8], the COSINE Committee of the Commission on Engineering Education [9], and the Education Committee of the British Computer Society [10]. Also, the ACM Curriculum Committee on Computer Education for Management, chaired by Daniel Teichroew, is now beginning to consider educational matters related to the application of computers to "management information systems." The Curriculum Committee has benefited greatly from interchanging ideas with these other groups. In addition, the entire Committee was privileged to take part in "The Graduate Academic Conference in Computing Science" [11] held at Stony Brook in June 1967.

Computer science programs, in common with those of all disciplines, must attempt to provide a basis of knowledge and a mode of thinking which permit continuing growth on the part of their graduates. Thus, in addition to exposing the student to a depth of knowledge in computer science sufficient to lay the basis for professional competence, such programs must also provide the student with the intellectual maturity which will allow him to stay abreast of his own discipline and to interact with other disciplines.

I. INFORMATION STRUCTURES AND PROCESSES

This subject division is concerned with representations and transformations of information structures and with theoretical models for such representations and transformations.

1. **DATA STRUCTURES:** includes the description, representation, and manipulation of numbers, arrays, lists, trees, files, etc.; storage organization, allocation, and access; enumeration, searching and sorting; generation, modification, transformation, and deletion techniques; the static and dynamic properties of structures; algorithms for the manipulation of sets, graphs, and other combinatoric structures.
2. **PROGRAMMING LANGUAGES:** includes the representation of algorithms; the syntactic and semantic specification of languages; the analysis of expressions, statements, declarations, control structures, and other features of programming languages; dynamic structures which arise during execution; the design, development and evaluation of languages; program efficiency and the simplification of programs; sequential transformations of program structures; special purpose languages; the relation between programming languages, formal languages, and linguistics.
3. **MODELS OF COMPUTATION:** includes the behavioral and structural analysis of switching circuits and sequential machines; the properties and classification of automata; algebraic automata theory and model theory; formal languages and formal grammars; the classification of languages by recognition devices; syntactic analysis; formal

specification of semantics; syntax directed processing; decidability problems for grammars; the treatment of programming languages as automata; other formal theories of programming languages and computation.

II. INFORMATION PROCESSING SYSTEMS

This subject division is concerned with systems having the ability to transform information. Such systems usually involve the interaction of hardware and software.

1. **COMPUTER DESIGN AND ORGANIZATION:** includes types of computer structure—von Neumann computers, array computers, and look-ahead computers; hierarchies of memory—flip-flop registers, cores, disks, drums, tapes—and their accessing techniques; micro-programming and implementation of control functions; arithmetic circuitry; instruction codes; input-output techniques; multiprocessing and multiprogramming structures.
2. **TRANSLATORS AND INTERPRETERS:** includes the theory and techniques involved in building assemblers, compilers, interpreters, loaders, and editing or conversion routines (media, format, etc.).
3. **COMPUTER AND OPERATING SYSTEMS:** includes program monitoring and data management; accounting and utility routines; data and program libraries; modular organization of systems programs; interfaces and communication between modules; requirements of multi-access, multiprogram and multiprocess environments; large scale systems description and documentation; diagnostic and debugging techniques; measurement of performance.
4. **SPECIAL PURPOSE SYSTEMS:** includes analog and hybrid computers; special terminals for data transmission and display; peripheral and interface units for particular applications; special software to support these.

III. METHODOLOGIES

Methodologies are derived from broad areas of applications of computing which have common structures, processes, and techniques.

1. **NUMERICAL MATHEMATICS:** includes numerical algorithms and their theoretical and computational properties; computational error analysis (for rounding and truncation errors); automatic error estimates and convergence properties.
2. **DATA PROCESSING AND FILE MANAGEMENT:** includes techniques applicable to library, biomedical, and management information systems; file processing languages.
3. **SYMBOL MANIPULATION:** includes formula operations such as simplification and formal differentiation; symbol manipulation languages.
4. **TEXT PROCESSING:** includes text editing, correcting, and justification; the design of concordances; applied linguistic analysis; text processing languages.
5. **COMPUTER GRAPHICS:** includes digitizing and digital storage; display equipment and generation; picture compression and image enhancement; picture geometry and topology; perspective and rotation; picture analysis; graphics languages.
6. **SIMULATION:** includes natural and operational models; discrete simulation models; continuous change models; simulation languages.
7. **INFORMATION RETRIEVAL:** includes indexing and classification; statistical techniques; automatic classification; matching and search strategies; secondary outputs such as abstracts and indexes; selective dissemination systems; automatic question answering systems.
8. **ARTIFICIAL INTELLIGENCE:** includes heuristics; brain models; pattern recognition; theorem proving; problem solving; game playing; adaptive and cognitive systems; man-machine systems.
9. **PROCESS CONTROL:** includes machine tool control; experiment control; command and control systems.
10. **INSTRUCTIONAL SYSTEMS:** includes computer aided instruction.

Related Areas. In addition to the areas of computer science listed under the three divisions above, there are many related areas of mathematics, statistics, electrical engineering, philosophy, linguistics, and industrial engineering or management which are essential to balanced computer science programs. Suitable courses in these areas should be developed cooperatively with the appropriate departments, although it may occasionally be desirable to develop some of these courses within the computer science program.

Since it is not feasible in this report to list all of the areas which might be related to a computer science program, let alone indicate where courses in these areas should be taught, the following listing is somewhat restricted. It is grouped into two major divisions: "mathematical sciences" and "physical and engineering sciences."

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

No attempt has been made to include within this classification system all the subject areas which make use of computer techniques, such as chemistry and economics; indeed, to list these would require inclusion of a major portion of the typical university catalog. Furthermore, the sociological, economic, and educational implications of developments in computer science are not discussed in this report. These issues are undoubtedly important, but they are not the exclusive nor even the major responsibility of computer science. Indeed, other departments such as philosophy and sociology should be urged to cooperate with computer scientists in the development of courses or seminars covering these topics, and computer science students should be encouraged to take these courses.

3. Description of Courses

The computer science courses specified in this report are divided into three categories: "basic," "intermediate," and "advanced." The basic courses are intended to be taught primarily at the freshman-sophomore level, whereas both the intermediate and the advanced courses may be taught at the junior-senior or the graduate level. In general, the intermediate courses are strongly recommended as part of undergraduate programs. The advanced courses are classified as such either because of their higher level of prerequisites and required maturity or because of their concern with special applications of computer science.

In addition to more elementary computer science courses, certain courses in mathematics are necessary, or at least highly desirable, as prerequisites for some of the proposed courses. More advanced mathematics courses may be included as supporting work in the programs of some students. Because of the considerable variation in the level and content of mathematics courses among (and even within) schools, the courses described by the Committee on the Undergraduate Program in Mathematics (CUPM) in the report, "A General Curriculum in Mathematics for Colleges" [12] have been used to specify the prerequisites for the proposed courses in computer science and requirements for degrees. Other pertinent mathematics courses are described in the CUPM reports, "Recommendations on the Undergraduate Mathematics Pro-

gram for Engineers and Physicists" [13] and "A Curriculum in Applied Mathematics" [14].

The titles and numbers of all the courses proposed in this report and the pertinent courses recommended by CUPM are shown in Figure 1 along with the prerequisite structure linking these courses. The courses described below, which make up the core of the undergraduate program, are also singled out in Figure 1. The relatively strong prerequisite structure proposed for these core courses allows their content to be greatly expanded from what a weaker structure would permit. The Committee recognizes that other—perhaps weaker—prerequisite structures might also be effective and that the structure shown will change along with the course content as computer science education develops. Prerequisites proposed for the advanced courses are subject to modification based on many orientations which these courses may be given at individual institutions.

Most of the courses have been designed on the basis of three semester hours of credit. Laboratory sessions, in which the more practical aspects of the material can be presented more effectively than in formal lectures, have been included where appropriate. The proposed number of hours of lecture and laboratory each week and the number of semester hours of credit for the course are shown in parentheses in the catalog descriptions below. For example, (2-2-3) indicates two hours of lecture and two hours of laboratory per week for a total of three semester hours of credit.

Course Catalog Descriptions and Prerequisites

For each of the courses listed below, a brief statement on the approach which might be taken in teaching it is given in the Appendix along with the detailed outlines of its proposed contents and annotated bibliographies of pertinent source materials and textbooks.

The first course is designed to provide the student with the basic knowledge and experience necessary to use computers effectively in the solution of problems. It can be a service course for students in a number of other fields as well as an introductory course for majors in computer science. Although no prerequisites are listed, it is assumed that the student will have had a minimum of three years of high school mathematics. All of the computer science courses which follow will depend upon this introduction.

Course B1. Introduction to Computing (2-2-3)

Algorithms, programs, and computers. Basic programming and program structure. Programming and computing systems. Debugging

and verification of programs. Data representation. Organization and characteristics of computers. Survey of computers, languages, systems, and applications. Computer solution of several numerical and nonnumerical problems using one or more programming languages.

The second course is intended to lay a foundation for more advanced study in computer science. By familiarizing the student with the basic structure and language of machines, the content of this course will give him a better understanding of the internal behavior of computers, some facility in the use of assembly languages, and an ability to use computers more effectively—even with procedure-oriented languages.

Course B2. Computers and Programming (2-2-3)

Prerequisite: Course B1.

Computer structure, machine language, instruction execution, addressing techniques, and digital representation of data. Computer

THE CORE COURSES OF THE PROPOSED UNDERGRADUATE PROGRAM

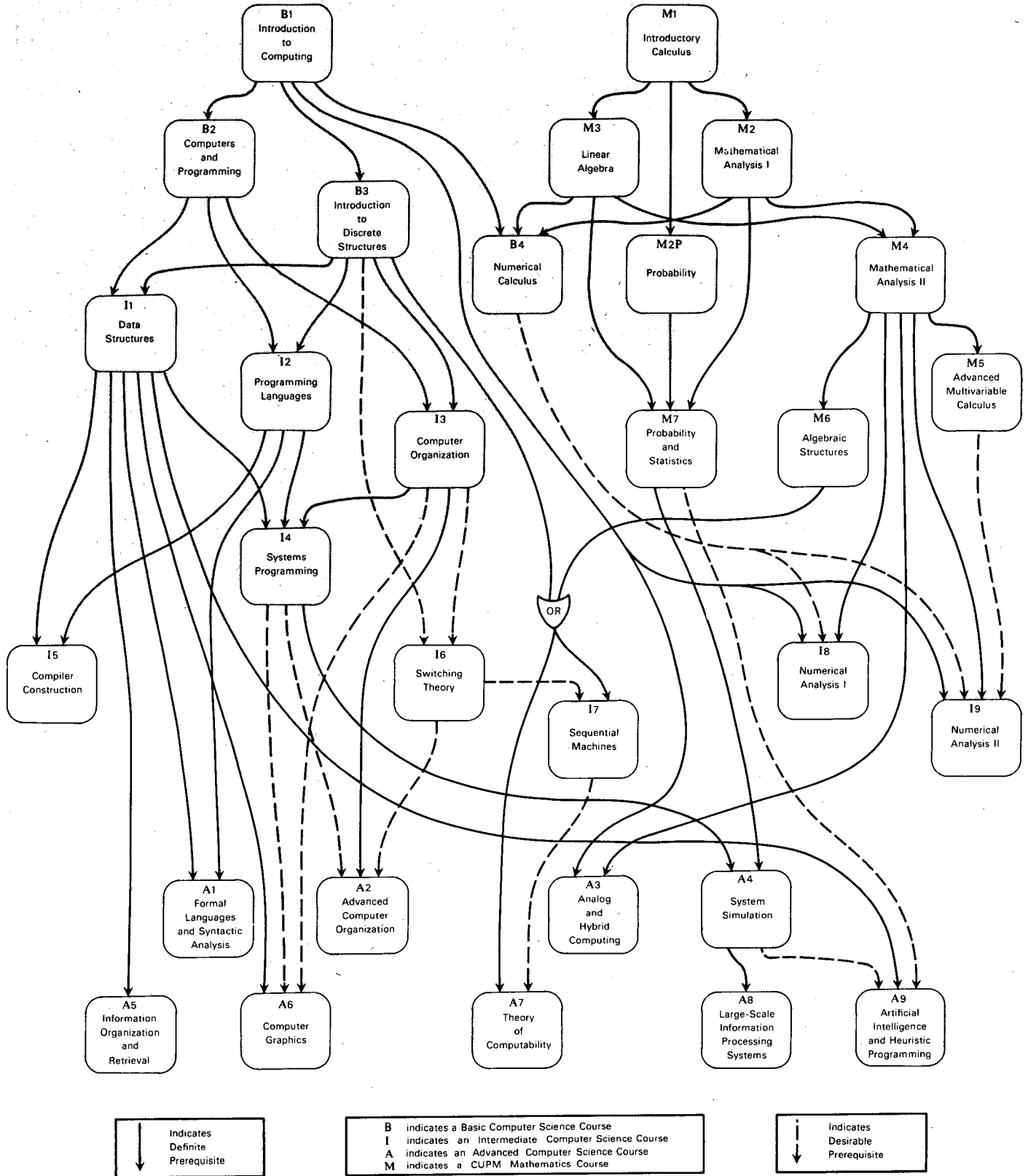


FIG. 1. Prerequisite structure of courses

systems organization, logic design, micro-programming, and interpreters. Symbolic coding and assembly systems, macro definition and generation, and program segmentation and linkage. Systems and utility programs, programming techniques, and recent developments in computing. Several computer projects to illustrate basic machine structure and programming techniques.

This course introduces the student to those fundamental algebraic, logical, and combinatoric concepts from mathematics needed in the subsequent computer science courses and shows the applications of these concepts to various areas of computer science.

Course B3. Introduction to Discrete Structures (3-0-3)

Prerequisite: Course B1.

Review of set algebra including mappings and relations. Algebraic structures including semigroups and groups. Elements of the theory of directed and undirected graphs. Boolean algebra and propositional logic. Applications of these structures to various areas of computer science.

This course provides the student with an introduction to the basic numerical algorithms used in scientific computer work—thereby complementing his studies in beginning analysis—and affords him an opportunity to apply the programming techniques he has learned in Course B1. Because of these aims, many of the standard elementary numerical analysis courses now offered in mathematics departments cannot be considered as substitutes for this course.

Course B4. Numerical Calculus (2-2-3)

Prerequisites: Courses B1, M2, and M3.

An introduction to the numerical algorithms fundamental to scientific computer work. Includes elementary discussion of error, polynomial interpolation, quadrature, linear systems of equations, solution of nonlinear equations, and numerical solution of ordinary differential equations. The algorithmic approach and the efficient use of the computer are emphasized.

This course is concerned with one of the most fundamental—but often inadequately recognized—areas of computer science. Its purpose is to introduce the student to the relations which hold among the elements of data involved in problems, the structures of storage media and machines, the methods which are useful in representing structured data in storage, and the techniques for operating upon data structures.

Course I1. Data Structures (3-0-3)

Prerequisites: Courses B2 and B3.

Basic concepts of data. Linear lists, strings, arrays, and orthogonal lists. Representation of trees and graphs. Storage systems and structures, and storage allocation and collection. Multilinked structures. Symbol tables and searching techniques. Sorting (ordering) techniques. Formal specification of data structures, data structures in programming languages, and generalized data management systems.

The following intermediate course is designed to present a systematic approach to the study of programming languages and thus provide the student with the knowledge necessary to learn and evaluate such languages.

Course I2. Programming Languages (3-0-3)

Prerequisites: Courses B2 and B3.

Formal definition of programming languages including specification of syntax and semantics. Simple statements including precedence, infix, prefix, and postfix notation. Global properties of algorithmic languages including scope of declarations, storage allocation, grouping of statements, binding time of constituents, subroutines, coroutines, and tasks. List processing, string manipulation, data description, and simulation languages. Run-time representation of program and data structures.

The following course discusses the organization, logic design, and components of digital computing systems. It can be thought of as a continuation of the hardware concepts introduced in Course B2.

Course I3. Computer Organization (3-0-3) or (3-2-4)

Prerequisites: Courses B2 and B3.

Basic digital circuits, Boolean algebra and combinational logic, data representation and transfer, and digital arithmetic. Digital storage and accessing, control functions, input-output facilities, system organization, and reliability. Description and simulation techniques. Features needed for multiprogramming, multiprocessing, and real-time systems. Other advanced topics and alternate organizations.

The following course is concerned primarily with the software organization—and to a lesser extent the hardware—of computer systems which support a wide variety of users. It is intended to bring together the concepts and techniques developed in the previous courses on data structures, programming languages, and computer organization by considering their role in the design of general computer systems. The problems which arise in multiaccessing, multiprogramming, and multiprocessing are emphasized.

Course I4. Systems Programming (3-0-3)

Prerequisites: Courses I1, I2, and I3.

Review of batch process systems programs, their components, operating characteristics, user services and their limitations. Implementation techniques for parallel processing of input-output and interrupt handling. Overall structure of multiprogramming systems on multiprocessor hardware configurations. Details on addressing techniques, core management, file system design and management, system accounting, and other user-related services. Traffic control, interprocess communication, design of system modules, and interfaces. System updating, documentation, and operation.

The following course is intended to provide a detailed understanding of the techniques used in the design and implementation of compilers.

Course I5. Compiler Construction (3-0-3)

Prerequisites: Courses I1 and I2.

Review of program language structures, translation, loading, execution, and storage allocation. Compilation of simple expressions and statements. Organization of a compiler including compile-time and run-time symbol tables, lexical scan, syntax scan, object code generation, error diagnostics, object code optimization techniques, and overall design. Use of compiler writing languages and bootstrapping.

This course introduces the theoretical principles and mathematical techniques involved in the design of digital system logic. A course compatible with the content and approach of this course is frequently taught in departments of electrical engineering.

Course I6. Switching Theory (3-0-3) or (2-2-3)

Prerequisites: Courses B3 (desirable) and I3 (desirable, as it would allow more meaningful examples to be used).

Switching algebra, gate network analysis and synthesis, Boolean algebra, combinational circuit minimization, sequential circuit analysis and synthesis, sequential circuit state minimization, hazards and races, and elementary number systems and codes.

This theoretical course is especially recommended for undergraduate students planning to do graduate work in computer science. It is also an appropriate course for electrical engineers and may sometimes be available from or jointly developed with an electrical engineering department.

Course I7. Sequential Machines (3-0-3)

Prerequisites: Courses B3 or M6, and I6 (desirable).

Definition and representation of finite state automata and sequential machines. Equivalence of states and machines, congruence, reduced machines, and analysis and synthesis of machines. Decision problems of finite automata, partitions with the substitution property, generalized and incomplete machines, semigroups and machines, probabilistic automata, and other topics.

The following two courses in numerical analysis are intended to be mathematically rigorous and at the same time computer-oriented.

Course I8. Numerical Analysis I (3-0-3)

Prerequisites: Courses B1, B4 (desirable), and M4.

A thorough treatment of solutions of equations, interpolation and approximations, numerical differentiation and integration, and numerical solution of initial value problems in ordinary differential equations. Selected algorithms will be programmed for solution on computers.

Course I9. Numerical Analysis II (3-0-3)

Prerequisites: Courses B1, B4 (desirable), M4, and M5 (desirable).

The solution of linear systems by direct and iterative methods, matrix inversion, the evaluation of determinants, and the calculation of eigenvalues and eigenvectors of matrices. Application to bound-

ary value problems in ordinary differential equations. Introduction to the numerical solution of partial differential equations. Selected algorithms will be programmed for solution on computers.

The following course serves as an introduction both to the theory of context-free grammars and formal languages, and to syntactic recognition techniques for recognizing languages specified by context-free grammars.

Course A1. Formal Languages and Syntactic Analysis (3-0-3)

Prerequisites: Courses I1 and I2.

Definition of formal grammars: arithmetic expressions and precedence grammars, context-free and finite-state grammars. Algorithms for syntactic analysis: recognizers, backtracking, operator precedence techniques. Semantics of grammatical constructs: reductive grammars, Floyd productions, simple syntactical compilation. Relationship between formal languages and automata.

The following advanced course in computer organization is centered around the comparison of solutions to basic design problems which have been incorporated in a number of quite different computers.

Course A2. Advanced Computer Organization (3-0-3)

Prerequisites: Courses I3, I4 (desirable), and I6 (desirable).

Computer system design problems such as arithmetic and nonarithmetic processing, memory utilization, storage management, addressing, control, and input-output. Comparison of specific examples of various solutions to computer system design problems. Selected topics on novel computer organizations such as those of array or cellular computers and variable structure computers.

This course is designed to give the computer science student some experience with analog, hybrid, and related techniques. It could also be very valuable as a service course.

Course A3. Analog and Hybrid Computing (2-2-3)

Prerequisites: Courses B1 and M4. (The CUPM mathematical analysis courses include some differential equations; more may be needed.)

Analog, hybrid and related digital techniques for the solution of differential equations. Analog simulation languages. Scaling methods. Operational characteristics of analog components. Digital differential analyzers. Analog-to-digital and digital-to-analog conversion. Stability problems. Modeling methods. Use of analog and hybrid equipment and of digital simulation of continuous systems.

The following course is concerned with the simulation and modeling of discrete systems on a computer. Since simulation is one of the most common applications of computers and is used to a great extent in the design of computing machines and systems, students of computer science should become acquainted with simulation techniques and their use.

Course A4. System Simulation (3-0-3)

Prerequisites: Courses I4 and M7.

Introduction to simulation and comparison with other techniques. Discrete simulation models, and introduction to, or review of, queuing theory and stochastic processes. Comparison of discrete change simulation languages. Simulation methodology including generation of random numbers and variates, design of simulation experiments for optimization, analysis of data generated by simulation experiments, and validation of simulation models and results. Selected applications of simulation.

The purpose of the following course is to provide an introduction to natural language processing, particularly as it relates to the design and operation of automatic information systems. Included are techniques for organizing, storing, matching, and retrieving structured information on digital computers, as well as procedures useful for the optimization of search effectiveness.

Course A5. Information Organization and Retrieval (3-0-3)

Prerequisite: Course I1.

Structure of semiformal languages and models for the representation of structured information. Aspects of natural language processing on digital computers. The analysis of information content by statistical, syntactic, and logical methods. Search and matching techniques. Automatic retrieval systems, question-answering systems. Production of secondary outputs. Evaluation of retrieval effectiveness.

The objective of the following course is to study the problems of handling graphic information, such as line drawings, block diagrams, handwriting, and three-dimensional surfaces, in computers. Input-output and representation-storage of pictures will be introduced from the hardware and software points of view. The course is intended to serve both the student interested in specializing in computer graphics per se and the student who seeks to apply graphic techniques to his particular computing work.

Course A6. Computer Graphics (2-2-3)

Prerequisites: Courses I1, I3 (desirable), and I4 (desirable).

Display memory, generation of points, vectors, etc. Interactive versus passive graphics. Analog storage of images on microfilm, etc. Digitizing and digital storage. Pattern recognition by features, syntax tables, random nets, etc. Data structures and graphics software. The mathematics of three-dimensions, projections, and the hidden-line problem. "Graphical programs," computer-aided design and instruction, and animated movies.

The following course uses abstract machines as models in the study of computability and computational complexity. Emphasis is placed on the multi-tape Turing machine as a suitable model, but other models are also considered.

Course A7. Theory of Computability (3-0-3)

Prerequisites: Courses B3 or M6, and I7 (desirable).

Introduction to Turing machines, Wang machines, Shepherdson-Sturgis, and other machines. Gödel numbering and unsolvability results, the halting problem, Post's correspondence problem, and relative uncomputability. Machines with restricted memory access, limited memory, and limited computing time. Recursive function theory and complexity classification. Models of computation including relationships to algorithms and programming.

The following course is intended for students who are interested in the application of information technology in large-scale information processing systems. The term "information processing system" is used here to include the hardware, software, procedures, and techniques that are assembled and organized to achieve some desired objectives. Examples of such large-scale information processing systems are business data processing systems, information storage and retrieval systems, command and control systems, and computer centers.

Course A8. Large-scale Information Processing Systems (3-0-3)

Prerequisites: Course A4, and a course in operations research or optimization theory.

Organization of major types of information processing systems. Data organization and storage structure techniques. Designing "best" systems by *organizing files and segmenting problems into computer programs to make efficient use of hardware devices*. Documentation methods and techniques for modifying systems. Use of optimization and simulation as design techniques. Communication problems among individuals involved in system development.

The following course introduces the student to those nonarithmetical applications of computing machines that: (1) attempt to achieve goals considered to require human mental capabilities (artificial intelligence); (2) model highly organized intellectual activity (simulation of cognitive behavior); and (3) describe purposeful behavior of living organisms or artifacts (self-organizing systems). Courses in this area are often taught with few prerequisites, but by requiring some or all of the prerequisites listed here this course could be taught at a more advanced level.

Course A9. Artificial Intelligence and Heuristic Programming (3-0-3)

Prerequisites: Courses I1, A4 (desirable), and M7 (desirable); and some knowledge of experimental and theoretical psychology would also be useful.

Definition of heuristic versus algorithmic methods, rationale of heuristic approach, description of cognitive processes, and approaches to mathematical invention. Objectives of work in artificial intelligence, simulation of cognitive behavior, and self-organizing systems. Heuristic programming techniques including the use of list processing languages. Survey of examples from representative application areas. The mind-brain problem and the nature of intelligence. Class and individual projects to illustrate basic concepts.

4. Undergraduate Programs

As indicated in the Introduction, there has been considerable discussion on the desirability of undergraduate degree programs in computer science. Many who favor these programs believe that an undergraduate computer science "major" is as natural today as a major in established fields such as mathematics, physics, or electrical engineering. Many who oppose these programs feel that, although undergraduate courses in computer science should be available for support of work in other areas, to offer an undergraduate degree in computer science may encourage too narrow a specialization at the expense of breadth. They point out that the lack of such breadth may be a serious handicap to a student desiring to do graduate work in computer science, and they contend that it would be better for the student to major in some established discipline while taking a number of computer science courses as supporting work. To meet these objections the Committee has made every effort to present a curriculum which includes a broad representation of basic concepts and an adequate coverage of professional techniques.

The number of undergraduate degree programs now in existence or in the planning stages—approximately one third of all Ph.D. granting institutions in the United States either have such computer science programs now or expect to have them by 1970 [5]—indicates that a discussion of the desirability of such programs is much less relevant than the early development of guidelines and standards for them. The Committee feels strongly, however, that schools should exercise caution against the premature establishment of undergraduate degree programs. The pressures created by large numbers of students needing to take courses required for the degree could easily result in a general lowering of standards exactly when it is vital that such programs be established and maintained only with high standards.

The variation of undergraduate program requirements among and within schools dictates that this Committee's recommendations must be very general. It is fully expected that each individual school will modify these recommendations to meet its specific circumstances, but it is hoped that these modifications will be expansions of or changes in the emphasis of the basic program proposed, rather than reductions in quantity or quality. The requirements recommended herein have been kept to a minimum in order to allow the student to obtain a "liberal education" and to enable individual programs to add additional detailed requirements. Since the liberal education requirements of each school are already well established, the Committee has not considered making recommendations on such requirements.

The Committee's recommendations for an under-

graduate computer science curriculum are stated in terms of computer science course work, programming experience, mathematics course work, technical electives, and possible areas of specialization. Some suggestions are also given as to how the courses might fit chronologically into a semester-by-semester schedule.

Computer Science Courses. The basic and intermediate course requirements listed below emphasize the first two "major subject divisions"—namely, "information structures and processes" and "information processing systems"—described in Section 2 of this report. These courses should give the student a firm grounding in the fundamentals of computer science.

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
- 11. Data Structures
- 12. Programming Languages
- 13. Computer Organization
- 14. Systems Programming

and at least two of the courses:

- 15. Compiler Construction
- 16. Switching Theory
- 17. Sequential Machines
- 18. Numerical Analysis I
- 19. Numerical Analysis II

Programming Experience. Developing programming skill is by no means the main purpose of an undergraduate program in computer science; nevertheless, such skill is an important by-product. Therefore such a program should insure that the student attains a reasonable level of programming competence. This can be done in part by including computer work of progressive complexity and diversity in the required courses, but it is also desirable that each student participate in a "true-to-life" programming project. This might be arranged through summer employment, a cooperative work-study program, part-time employment in computer centers, special project courses, or some other appropriate means.

Mathematics Courses. The Committee feels that an academic program in computer science must be well based in mathematics since computer science draws so heavily upon mathematical ideas and methods. The recommendations for required mathematics courses given below should be regarded as minimal; obviously additional course work in mathematics would be essential for students specializing in numerical applications.

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

Technical Electives. Assuming that a typical four-year curriculum consists of 124 semester hours, a number of technical electives beyond the requirements listed above should be available to a student in a computer science program. Some of these electives might be specified by the program to develop a particular orientation or minor. Because of the temptation for the student to overspecialize, it is suggested that a limit be placed on the number of computer science electives a student is allowed to take—for example, three such courses might be permitted. For many students it will be desirable to use the remaining technical electives to acquire a deeper knowledge of mathematics, physical science, electrical engineering, or some other computer-related field.

Students should be carefully advised in the choice of their electives. In particular, those preparing for graduate school must insure that they will be qualified for admission into the program of their choice. Those seeking a more “professional” education can specialize to some extent through the proper choice of electives.

Areas of Specialization. Although undue specialization is not appropriate at the undergraduate level, the technical electives may be used to orient undergraduate programs in a number of different directions. Some of the possible orientations, along with appropriate courses (of Section 3) and subject areas (of Section 2) from which the optional and elective courses might be taken, are given below.

APPLIED SYSTEMS PROGRAMMING

Optional courses

- 15. Compiler Construction
- 16. Switching Theory

Electives from courses

- A2. Advanced Computer Organization
- A5. Information Organization and Retrieval
- A6. Computer Graphics

Electives from areas

- IV.8 Combinatorial Mathematics
- IV.9 Mathematical Logic
- IV.11 Probability and Statistics
- IV.12 Operations Analysis

COMPUTER ORGANIZATION AND DESIGN

Optional courses

- 16. Switching Theory
- 17. Sequential Machines

Electives from courses

- A2. Advanced Computer Organization
- A4. System Simulation
- A8. Large-scale Information Processing Systems

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

- 18. Numerical Analysis I
- 19. 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

- 15. Compiler Construction
- 16. 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

Semester Chronology. Any institution planning an undergraduate program based on the recommendations of this report should work out several complete four-year curricula to insure that the required courses mesh in an orderly manner with electives and with the “general education” requirements of the institution. This will help the school to take into account local circumstances such as having very few entering freshmen who can begin college mathematics with the calculus.

Table I gives some examples of how a student in computer science might be scheduled for the minimum set of courses recommended for all majors.

TABLE I

Year	Semester	First example	Second example	Third example
Freshman	First	M1, B1	Basic Math.	Basic Math.
	Second	M2, B2	M1, B1	Basic Math.
Sophomore	First	M3, B3	M2, B2	M1, B1
	Second	M4, B4	M3, B3	M2, B2
Junior	First	M2P, I1	M4, B4	M2P, B3, B4
	Second	M5, I2	M2P, I1	M3, I1, I2
Senior	First	I3, I8	M7, I2, I3	M6, I3, I6
	Second	I4, I9	I4, I5, I6	M7, I4, I7

5. Master's Degree Programs

The recommendations given in this section concern undergraduate preparation for graduate study in computer science, requirements for a Master of Science degree in computer science, and some possible areas of concentration for students who are at the master's degree level.

Undergraduate Preparation. 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 of Section 3.)

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.)

A student with a bachelor's degree in computer science, such as recommended in Section 4, can have taken all these prerequisites as basic and supporting courses and can also have taken further work which overlaps with some of the subject matter to be treated at the master's level. Although such a student will be able to take more advanced graduate work in computer science to satisfy his master's requirements, he may need to take more supporting work than a student whose undergraduate degree was in some other field. A student with an undergraduate degree in mathematics, physical science, or electrical engineering can easily qualify for such a program if he has taken adequate supporting courses in computer science. Other applicants should have no more than a few deficiencies in order to qualify.

In the near future many of the potential students, because of having completed their undergraduate work some time ago, will not have had the opportunity to meet these requirements. Liberal policies should therefore be established so that promising students can make up deficiencies.

Degree Requirements. Each student's program of study for the master's degree should have both breadth and depth. In order to obtain breadth, the student should take course work from each of the three subject divisions of computer science described in Section 2. To obtain depth, he should develop an area of concen-

tration in which he would write a master's thesis or complete a master's project (if required).

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 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.

In order that the student may perform in his required course work at the graduate level he must acquire a knowledge of related areas, such as mathematics and the physical sciences, either as part of his undergraduate preparation or as part of his graduate program. Computer science as a discipline requires an understanding of mathematical methods and an ability to use mathematical techniques beyond the specific undergraduate preparation in mathematics recommended above. Hence, a student who does not have a "strong" mathematics background should take either further courses in mathematics, or he should take computer science courses which contain a high mathematical content.

If Courses I1, I2, I3, and I4 of Section 3 are taught at a sufficiently high level, they can be used to satisfy the "breadth" requirements for the first two subject divisions listed above. In any case, the student who has taken such courses as part of his undergraduate program could take more advanced courses in these areas so that the requirement for two courses in each subject division might be relaxed somewhat. This might permit such a student to take more supporting work outside computer science.

Areas of Concentration. The "depth" requirement will often involve courses from fields other than computer science, so that a student may have to take additional courses in these fields just to meet prerequisites unless he has anticipated this need in his undergraduate preparation. In any event, the particular courses a student selects from each of the three subject divisions of computer science should be coordinated with his area of concentration. To illustrate how this might be done, six possible concentrations are shown below together with lists of the subject areas (of Section 2) from which appropriate courses might be selected for each of the concentrations. The characterization of courses in terms of subject areas instead of explicit content effectively gives a list of suggested topics which can be drawn upon in designing master's level courses suited to the needs of individual institutions.

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
- III.3 Symbol Manipulation
- III.6 Simulation
- IV.7 Optimization Theory
- IV.9 Mathematical Logic

APPLIED HARDWARE

- I.1 Data Structures
- I.3 Models 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.7 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

The requirement of a master's thesis or other project has been left unspecified since general institutional requirements will usually determine this. It is strongly recommended, however, that a master's program in computer science contain some formal provision for insuring that the student gains or has gained project experience in computer applications. This could be effected by requiring that students carry out either individually or cooperatively a substantial assigned task involving analysis and programming, or better, that students be involved in an actual project on campus or in conjunction with other employment.

This proposed program embodies sufficient flexibility to fulfill the requirements of either an "academic" degree obtained in preparation for 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 master's degree programs.

6. Doctoral Programs

Academic programs at the doctoral level reflect the specific interests of the faculty and, hence, vary from university to university. Therefore, the Committee cannot expect to give recommendations for such doctoral programs in as great a detail as has been done for the undergraduate and master's degree programs. The large number of institutions planning such programs and the variety of auspices under which they are being sponsored, however, suggest that a need exists for guidelines as to what constitutes a "good" doctoral program. While recommendations on doctoral programs

will not be given at this time, the problem of how to obtain such guidelines has been of considerable interest to the Committee.

One possible source of such guidelines is the existing doctoral programs. A description of the program at Stanford University [15] has already been published in *Communications of the ACM* and descriptions of many other programs are available from the universities concerned. Information based on a number of such programs is contained in the report of the June 1967, Stony Brook Conference [11]. This report also contains

a list of thesis topics currently being pursued or recently completed. In the future the Curriculum Committee hopes to encourage wide dissemination of the descriptions of existing programs and research topics. Perhaps it can take an active role in coordinating the interchange of such information.

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, 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 develop-

ing 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.

7. Service Courses, Minors, and Continuing Education

Though it is now generally recognized that a significant portion of our undergraduate students needs some knowledge of computing, the amount and type of computing knowledge necessary for particular areas of study are still subject to considerable discussion. The Pierce Report [7] estimates that about 75 percent of all college undergraduates are enrolled in curricula where some computer training would be useful. This estimate, based on figures compiled by the US Office of Education, involves dividing the undergraduate student population into three groups. The first group, about 35 percent of all undergraduates, consists of those in scientific or professional programs having a substantial quantitative content (e.g. mathematics, physics, and engineering). At least some introductory knowledge of computing is already considered highly desirable for almost all of these students. The second group, some 40 percent, is made up of those majoring in fields where an understanding of the fundamentals of computing is steadily becoming more valuable (e.g. business, behavioral sciences, education, medicine, and library science). Many programs in these areas are already requiring courses in computing, and most are expected to add such requirements in the future. The third group, roughly 25 percent, comprises those undergraduates who are majoring in areas which do not necessarily depend on the use of computers (e.g. music, drama, literature, foreign languages, liberal arts, and fine arts). There are many persons who maintain that even these students could benefit from a course which would give them an appreciation of this modern technology and its influence on the structure of our society.

The extent and nature of the courses on computing

needed for these three groups of students should be given further careful study, but the existence of a substantial need for service courses in computer science seems undeniable. Students in the more quantitative fields are usually well-equipped to take the basic courses designed for the computer science major. In particular, Course B1 should serve as an excellent introductory course for these students and, depending upon their interests, Course B2 or B4 might serve as a second course. When these students develop a greater interest in computing, they should normally be able to select an appropriate "minor" program of study from the courses described in Section 3. In developing a minor program careful consideration should be given to the comparative values of each course in the development of the individual student.

Some special provisions appear to be necessary for students in the second and third groups described above. A special version of Course B1 which would place more emphasis on text processing and other nonnumeric applications might be more appropriate for students in the second group. However, it is important that this course provides adequate preparation for such courses as B2 and B3, since many of these students might be expected to take further courses in computer science. It may also be desirable to develop courses giving primary emphasis to the economic, political, sociological, and other implications of the growing use of computer technology. Such courses would not be considered substitutes for basic technical courses such as B1, but they could serve the needs of the third group of students.

Professional programs at all levels offer limited op-

portunity for courses outside their highly structured curricula, and they also present special problems. In some cases it may be necessary to develop special courses for students in such programs or to integrate work on computing into existing courses. Those preparing for graduate professional programs will often find it desirable to include some of the basic computer science courses in their undergraduate work.

The responsibility for developing and conducting the basic service courses in computer science should be concentrated within the academic structure and combined with the operation of educational programs in computer science. By properly aggregating students from similar fields, those responsible for planning academic courses can make them more generally applicable and broadly directed. Under this arrangement teachers can be used more effectively and course content can more easily be kept current with the rapidly moving developments in the field. Also, students who find a need or a desire to delve further into computer science are more likely to have the necessary background to take advanced

courses. On the other hand, it must be recognized that some departments will have many situations where special applications of the computer can best be introduced in their own courses. Certainly those responsible for the basic computer science service courses must be sensitive to the needs of the students for whom these courses are intended.

Finally, the need for continuing education in computer science must be recognized. Much of the course material discussed in this report did not exist ten or fifteen 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.

8. Implementation

In educational institutions careful consideration should be given to the problems of implementing a computer-related course of study—be it a few introductory or service courses, an undergraduate degree program, or a graduate degree program. Some of these problems involve organization, staff requirements, and physical facilities (including computing services). Although individual ways of providing a favorable environment for computer science will be found in each school, the following discussion is intended to call attention to the extent of some of these problems.

Organization for Academic Programs. It should be realized that the demands for education in computer science are strong. If some suitable place in the institutional structure is not provided for courses and programs in computer science to be developed, they will spring up within a number of existing departments and a possible diffusion of effort will result as has been experienced with statistics in many universities.

If degree programs in computer science are to be offered, it is desirable to establish an independent academic unit to administer them. Such a unit is needed to provide the appropriate mechanisms for faculty appointments and promotions, for attention to continuing curriculum development, and for the allocation of resources such as personnel, budget, space, and equipment. This academic unit should also be prepared to provide general service courses and to cooperate in developing computer-oriented course work in other departments and professional schools.

Many universities have established departments of computer science as part of their colleges of arts and

sciences, and some have established divisions of mathematical sciences, which include such departments as mathematics, applied mathematics, statistics, and computer science. Other institutions have located computer science departments in colleges of engineering and applied science. Academic units in computer science have also been affiliated with a graduate school, associated with more than one college, or even established independent of any college in a university.

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.

Staff Requirements. Degree programs in computer science require a faculty dedicated to this discipline—that is, individuals who consider themselves computer scientists regardless of their previous academic training. Although graduate programs in computer science are now producing a limited number of potential faculty members, the demand for such people in industry and government and the competition for faculty among universities are quite intense. Hence educational institutions will have to obtain most of their computer science faculty from other sources—at least for the immediate future. Many faculty members in other departments of our universities have become involved with computing and have contributed to its development to the point that they are anxious to become part of a com-

puter science program. Within industry and government there are also people with the necessary academic credentials who are willing to teach the technology they have helped develop. Thus, extensive experience and academic work in computer science, accompanied by academic credentials in a related area such as mathematics, electrical engineering, or other appropriate disciplines, can serve as suitable qualifications for staff appointments in a computer science program. Joint appointments with other academic departments or with the computing center can help fill some of the need, but it is desirable that a substantial portion of the faculty be fully committed to computer science. Moreover, there is some critical size of faculty—perhaps the equivalent of five full-time positions—which is needed to provide a reasonable coverage of the areas of computer science discussed in Section 2.

Since relatively few good textbooks are available in the computer sciences, the computer science faculty will need to devote an unusually large part of its time to searching the literature and developing instructional materials. This fact should be taken into consideration in determining teaching loads and staff assignments.

Physical Facilities. Insofar as physical facilities are concerned, computer science should generally be included among the laboratory sciences. Individual faculty members may need extra space to set up and use equipment, to file cards and voluminous computer listings, and otherwise to carry out their teaching and research. In addition to normal library facilities, special collections of material, such as research reports, computer manuals, and computer programs, must be obtained and facilities made available for their proper storage and effective use. Space must also be provided for keypunches, remote consoles to computers, and any other special equipment needed for education and/or research. Laboratory-type classrooms must be available to allow students, either as individuals or as groups, to spread out and study computer listings.

It is no more conceivable that computer science courses—let alone degree programs—can exist without a computer available to students than that chemistry and physics offerings can exist without the associated laboratory equipment. Degree programs require regular access to at least a medium-sized computer system of sufficient complexity in configuration to require the use of an operating system. The total operating costs 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 class day.

Space for data and program preparation and program checking must be provided, and the logistics of handling hundreds and possibly thousands of student programs per day must be worked out so that each student has frequent access to the computer with a minimum of waiting and confusion. Although many of these same facilities must be provided for students and faculty other than those in computer science, the computer science program is particularly dependent on these services. It is simply false economy to hamper the use of expensive computing equipment by crowding it into unsuitable space or in some other way making it inaccessible.

The study and development of systems programs will require special forms of access to at least medium-scale computing systems. This will place an additional burden on the computer center and may possibly require the acquisition of completely separate equipment for educational and research purposes. In advanced programs it is likely that other specialized equipment will be necessary to handle such areas as computer graphics, numeric control of machines, process control, simulation, information retrieval systems, and computer-assisted instruction.

Although assistance in financing computer services and equipment can be obtained from industry and from federal and state governments, the Committee feels that universities should provide for the costs of equipment and services for computer science programs just as they provide for costs of other laboratory sciences. Based on its knowledge of costs at a number of schools the Committee estimates that computer batch processing of student jobs for elementary courses presently costs an average of about \$30 per semester hour per student, whereas the Pierce Report [7] estimates that it costs colleges about \$95 per chemistry student per year for a single chemistry laboratory course. Although computer costs are decreasing relative to capacity, it is expected that students will be able to use more computer time effectively in the future as computers become more accessible through the use of such techniques as time-sharing. On the basis of these estimates and expectations, future computer costs for academic programs may well approach faculty salary costs.

Relation of the Academic Program to the Computing Center. As indicated above, the demands which an academic program in computer science places on a university computing center are more than routine. Computer and programming systems must be expanded and modified to meet the growing and varied needs of these programs as well as the needs of the other users. The service function of a computer center must therefore be enhanced by an activity which might be described as "applied computer science." In a complementary way, it is appropriate for a computer science faculty to be deeply involved in the application of computers,

particularly in the development of programming systems. For these reasons, the activities of a computer center and a computer science department should be closely coordinated. The sharing of staff through joint appointments helps facilitate such cooperation, and it is almost necessary to provide such academic appointments in order to attract and retain certain essential computer center personnel.

It should be realized, however, that the basic philosophies of providing services and of pursuing 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 a 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 having only transient value. Considerable and constant care must be taken to maintain a balance between these extremes.

REFERENCES

1. Association for Computing Machinery, Curriculum Committee on Computer Science. An undergraduate program in computer science—preliminary recommendations. *Comm. ACM* 8, 9 (Sept. 1965), 543-552.
2. NEWELL, A., PERLIS, A. J., AND SIMON, H. A. Computer science. (Letter to the Editor). *Science* 157, 3795 (22 Sept. 1967), 1373-1374.
3. University of Chicago. Graduate programs in the divisions, announcements 1967-1968. U. of Chicago, Chicago, pp. 167-169.
4. GORN, S. The computer and information sciences: a new basic discipline. *SIAM Review* 5, 2 (Apr. 1963), 150-155.
5. HAMBLIN, J. W. Computers in higher education: expenditures, sources of funds, and utilization for research and instruction 1964-65, with projections for 1968-69. (A report on a survey supported by NSF). Southern Regional Education Board, Atlanta, Ga., 1967.
6. ROSSER, J. B., ET AL. Digital computer needs in universities and colleges. Publ. 1233, National Academy of Sciences-National Research Council, Washington, D. C., 1966.
7. President's Science Advisory Committee. Computers in higher education. The White House, Washington, D. C., Feb. 1967.
8. Mathematical Association of America, Committee on the Undergraduate Program in Computer Science (CUPM). Recommendations on the undergraduate mathematics program for work in computing. CUPM, Berkeley, Calif., May 1964.
9. Commission on Engineering Education, COSINE Committee. Computer sciences in electrical engineering. Commission in Engineering Education, Washington, D. C., Sept. 1967.
10. British Computer Society, Education Committee. Annual education review. *Comput. Bull.* 11, 1 (June 1967), 3-73.
11. FINERMAN, A. (Ed.) *University Education in Computing Science*. (Proceedings of the Graduate Academic Conference in Computing Science, Stony Brook, New York, June 5-8, 1967) ACM Monograph, Academic Press, New York, 1968.
12. Mathematical Association of America, Committee on the Undergraduate Program in Mathematics (CUPM). A general curriculum in mathematics for colleges. CUPM, Berkeley, Calif., 1965.
13. ——. Recommendations in the undergraduate mathematics program for engineers and physicists. CUPM, Berkeley, Calif., 1967.
14. ——. A curriculum in applied mathematics. CUPM, Berkeley, Calif., 1966.
15. FORSYTHE, G. E. A university's education program in computer science. *Comm. ACM* 10, 1 (Jan. 1967), 3-11.

Acknowledgments

The following people have served as consultants to the Committee on one or more occasions or have given considerable other assistance to our work.

Richard V. Andree, University of Oklahoma
Robert L. Ashenurst, University of Chicago
Bruce H. Barnes, Pennsylvania State University
Robert S. Barton, University of Utah
J. Richard Buchi, Purdue University
Harry Cantrell, General Electric Company
Mary D'Imperio, Department of Defense
Arthur Evans, Massachusetts Institute of Technology
David C. Evans, University of Utah
Nicholas V. Findler, State University of New York at Buffalo
Patrick C. Fischer, University of British Columbia
George E. Forsythe, Stanford University
Bernard A. Galler, University of Michigan
Saul Gorn, University of Pennsylvania
Preston C. Hammer, Pennsylvania State University
Richard W. Hamming, Bell Telephone Laboratories
Harry D. Huskey, University of California at Berkeley
Peter Z. Ingerman, Radio Corporation of America
Donald E. Knuth, California Institute of Technology
Robert R. Korfhage, Purdue University
Donald J. Laird, Pennsylvania State University
George E. Lindamood, University of Maryland
William C. Lynch, Case Institute of Technology
M. Douglas McIlroy, Bell Telephone Laboratories
Robert McNaughton, Rensselaer Polytechnic Institute
Michel Melkanoff, University of California at Los Angeles
William F. Miller, Stanford University
Anthony G. Oettinger, Harvard University
Elliott I. Organick, University of Houston
Robert H. Owens, University of Virginia
Charles P. Reed, Jr., Georgia Institute of Technology
Saul Rosen, Purdue University
Daniel Teichroew, Case Institute of Technology
Andries van Dam, Brown University
Robert J. Walker, Cornell University
Peter Wegner, Cornell University

Written comments on the Committee's work, contributions to course outlines, and other assistance have been rendered by the following:

Bruce W. Arden, University of Michigan
John E. Bakken, Midwest Oil Corporation
Larry L. Bell, Auburn University
Robert D. Brennan, International Business Machines Corp.
Yaohan Chu, University of Maryland
Charles H. Davidson, University of Wisconsin
Harold P. Edmundson, University of Maryland
Charles W. Gear, University of Illinois
Robert T. Gregory, University of Texas
Keith Hastings, University of Toronto
Carl F. Kossack, University of Georgia
Ralph E. Lee, University of Missouri at Rolla
George Mealy, Massachusetts Institute of Technology
Harlan D. Mills, International Business Machines Corp.
Jack Minker, University of Maryland and Auerbach Corp.
Jack Noland, General Electric Company
James C. Owings, Jr., University of Maryland
David L. Parnas, Carnegie-Mellon University
Charles R. Pearson, J. P. Stevens and Co.
Tad Pinkerton, University of Michigan
Roland L. Porter, Los Angeles, California
Anthony Ralston, State University of New York at Buffalo
Roy F. Reeves, Ohio State University
John R. Rice, Purdue University
Gerard Salton, Cornell University
Gordon Sherman, University of Tennessee
Vladimir Slamecka, Georgia Institute of Technology
Joseph F. Traub, Bell Telephone Laboratories

Numerous other people have contributed to the work of the Committee through informal discussions and other means. The Committee is grateful for all of the assistance it has received and especially for the cooperative spirit in which it has been given.

Appendix. Course Outlines and Bibliographies

For each of the twenty-two courses described in Section 3, this Appendix contains a brief discussion of the approach to teaching the course, a detailed outline of the content of the course, and a bibliography listing material which should be useful to the teacher and/or the student in the course. The amount of attention which might be devoted to the various topics in the content of some of the courses is indicated by percentage or by number of lectures. Whenever possible, each bibliographic entry is followed by a reference to its review in *Computing Reviews*. The format used for these references is CR-xyvi-n, where xy indicates the year of the review, v the volume number, i the issue number, and n the number of the review itself. Most of the bibliographic entries are followed by a brief annotation which is intended to indicate the way in which the item would be useful and perhaps to clarify the subject of the item. In some cases the title is sufficient for this purpose, and no annotation is given. In other cases the items are simply keyed in various ways to the sections of the content to which they apply. Although an effort has been made to cite a wide variety of texts and reference materials for each course, space and other considerations have prevented the listing of all books and papers which might bear on the topics treated.

Course B1. Introduction to Computing (2-2-3)

APPROACH

This first course in computing concentrates on the solution of computational problems through the introduction and use of an algorithmic language. A single such language should be used for most of the course so that the students may master it well enough to attack substantial problems. It may be desirable, however, to use a simple second language of quite different character for a problem or two in order to demonstrate the wide diversity of the computer languages available. Because of its elegance and novelty, SNOBOL can be used quite effectively for this purpose. In any case, it is essential that the student be aware that the computers and languages he is learning about are only particular instances of a widespread species.

The notion of an algorithm should be stressed throughout the course and clearly distinguished from that of a program. The language structures should be carefully motivated and precisely defined using one or more of the formal techniques available. Every effort should be made to develop the student's ability to analyze complex problems and formulate algorithms for their solution. Numerous problems should be assigned for computer solution, beginning early in the course with several small projects to aid the student in learning to program, and should include at least one major project, possibly of the student's own choosing. Careful verification of program operation and clear program documentation should be emphasized.

CONTENT

This outline reflects an order in which the material might be presented; however, the order of presentation will be governed by the choice of languages and texts as well as individual preferences. In particular, the treatment of some of the topics listed below might be distributed throughout the course. Although not specifically listed in the following outline, programming and computer projects should constitute an important part of the content of this course.

1. *Algorithms, Programs, and Computers.* The concept and properties of algorithms. Flowcharts of algorithms and the need for precise languages to express algorithms. The concept of a program, examples of simple programs, and description of how computers execute programs. Programming languages including the description of their syntax and semantics. (10%)

2. *Basic Programming.* Constants, identifiers, variables, subscripts, operations, functions, and expressions. Declarations, substi-

tution statements, input-output statements, conditional statements, iteration statements, and complete programs. (10%)

3. *Program Structure.* Procedures, functions, subroutine calling, and formal-actual parameter association. Statement grouping, nested structure of expressions and statements, local versus global variables, run-time representation, and storage allocation. Common data, segmenting, and other structural features. (10%)

4. *Programming and Computing Systems.* Compilers, libraries, loaders, system programs, operating systems, and other information necessary for the student to interact with the computer being used. (5%)

5. *Debugging and Verification of Programs.* Error conditions and messages, techniques of debugging, selection of test data, checking of computer output, and programming to guard against errors in data. (5%)

6. *Data Representation.* Systems of enumeration and binary codes. Representation of characters, fixed and floating-point numbers, vectors, strings, tables, matrices, arrays, and other data structures. (10%)

7. *Other Programming Topics.* Formatted input and output. Accuracy, truncation, and round-off errors. Considerations of efficiency. Other features of language(s) being considered. (10%)

8. *Organization and Characteristics of Computers.* Internal organization including input-output, memory-storage, processing and control. Registers, arithmetic, instruction codes, execution of instruction, addressing, and flow of control. Speed, cost and characteristics of various operations and components. (10%)

9. *Analysis of Numerical and Nonnumerical Problems.* Applications of algorithm development and programming to the solution of a variety of problems (distributed throughout the course). (15%)

10. *Survey of Computers, Languages, Systems, and Applications.* The historical development of computers, languages, and systems including recent novel applications of computers, and new developments in the computing field. (10%)

11. *Examinations.* (5%)

ANNOTATED BIBLIOGRAPHY

In addition to the materials listed here, there are numerous books and manuals on specific computer languages which would be appropriate as part of the textual material for this course. Very few books, however, place sufficient emphasis on algorithms and provide the general introductory material proposed for this course.

1. ARDEN, B. W. *An Introduction to Digital Computing.* Addison-Wesley, Reading, Mass., 1963, 389 pp. CR-6345-4551.

This text uses MAD and emphasizes the solution of numerical problems, although other types of problems are discussed. Numerous examples and exercises.

2. FORTE, A. *SNOBOL3 Primer*. M.I.T. Press, Cambridge, Mass., 1967, 107 pp.

An elementary exposition of SNOBOL3 which might well be used to introduce a "second" language. Many exercises and examples. (SNOBOL4 is now becoming available.)

3. GALLER, B. A. *The Language of Computers*. McGraw-Hill, New York, 1962, 244 pp. CR-6341-3574.

Emphasizes "discovering" the structure of algorithms needed for the solution of a varied set of problems. The computer language features necessary to express these algorithms are carefully motivated. The language introduced is primarily based on MAD, but FORTRAN and ALGOL are also discussed.

4. GRUENBERGER, F. The teaching of computing (Guest editorial). *Comm. ACM* 8, 6 (June 1965), 348 and 410. CR-6565-8074.

Conveys eloquently the philosophy which should be used in developing and teaching an introductory computing course.

5. GRUENBERGER, F. AND JAFFRAY, G. *Problems for Computer Solution*. Wiley, New York, 1965, 401 pp. CR-6671-8757.

Contains a collection of problems appropriate for computer solution by students. Student is guided into the analysis of the problems and the development of good computational solutions, but actual computer programs for the solutions are not given.

6. HULL, T. E. *Introduction to Computing*. Prentice-Hall, Englewood Cliffs, N. J., 1966, 212 pp.

Text on fundamentals of algorithms, basic features of stored-program computers, and techniques involved in implementing algorithms on computers. Presents a complete description of FORTRAN IV with examples of numerical methods, nonnumerical applications, and simulations. Numerous exercises.

7. MARCOVITZ, A. B. AND SCHWEPPE, E. J. *An Introduction to Algorithmic Methods Using the MAD Language*. Macmillan, New York, 1966, 433 pp. CR-6781-11,199.

Emphasizes algorithms and their expression as programs, characteristics of computers and computer systems, formal definition of computer languages, and accuracy and efficiency of programs. Numerous examples and exercises.

8. PERLIS, A. J. Programming for digital computers. *Comm. ACM* 7, 4 (Apr. 1964), 210-211.

Description of course developed by Perlis at Carnegie Institute of Technology which has strongly influenced the course proposed here.

9. RICE, J. K. AND RICE, J. R. *Introduction to Computer Science: Problems, Algorithms, Languages and Information*, Preliminary edition. Holt, Rinehart and Winston, New York, 1967, 452 pp.

Presentation revolves around the theme of "problem solving," emphasizing algorithms, languages, information representations, and machines necessary to solve problems. Problem solution methods classified, and many sample problems included. The nature of errors and uncertainty is considered. Detailed appendix on FORTRAN IV by E. Desautels.

10. School Mathematics Study Group. *Algorithms, Computation and Mathematics*, rev. ed. Stanford University, Stanford, Calif., 1966. *Student Text*, 453 pp., *Teacher's Commentary*, 301 pp.; *Algol Supplement: Student Text*, 133 pp., *Teacher's Commentary*, 109 pp.; *Fortran Supplement: Student Text*, 132 pp., *Teacher's Commentary*, 102 pp. Available from A. C. Vroman, Inc., 367 South Pasadena, Pasadena, Calif. A *MAD Language Supplement* by E. I. Organick is available from Ulrich's Book Store, 549 E. University Avenue, Ann Arbor, Mich.

Although developed for high school students and teachers, this work contains much material appropriate for this course. Develops an understanding of the relationship between mathematics, computing, and problem solving. Basic text uses English and flow charts to describe algorithms; supplements introduce the computer language and give these algorithms in ALGOL, FORTRAN, and MAD.

Course B2. Computers and Programming (2-2-3)

APPROACH

This course is designed to introduce the student to basic computer organization, machine language programming, and the use of assembly language programming systems. A particular computer, machine language and programming system should be used extensively to illustrate the concepts being taught and to give the student actual experience in programming. However, it is important that the course not degenerate into mere training in how to program one machine. Alternative machine languages, machine organization, and programming systems should be discussed and compared. Emphasis should be placed on the overall structure of the machines and programming techniques considered. A "descriptive" presentation of various computer features and organizations may be very effective; nevertheless, it is recommended that a precise language be introduced and used to describe computer organizations and instruction execution (as the Iverson notation has been used to describe the IBM System/360).

CONTENT

The following outline indicates a possible order in which the material for this course might be taught, but other arrangements might be equally suitable depending upon the choice of text, availability of computing facilities, and preferences of the instructor. Computer projects—although not specifically listed below—should be an essential part of the course content.

1. *Computer Structure and Machine Language*. Organization of computers in terms of input-output, storage, control, and processing units. Register and storage structures, instruction format and execution, principal instruction types, and machine language programming. Machine arithmetic, program control, input-output operations, and interrupts. Characteristics of input-output and storage devices. (10%)

2. *Addressing Techniques*. Absolute addressing, indexing, indirect addressing, relative addressing, and base addressing. Memory mapping functions, storage allocation, associative addressing, paging, and machine organization to facilitate modes of addressing. (5%)

3. *Digital Representation of Data*. Bits, fields, words, and other information structures. Radices and radix conversion, representation of integer, floating-point, and multiple-precision numbers in binary and decimal form, and round-off errors. Representation of strings, lists, symbol tables, arrays and other data structures. Data transmission, error detection and correction. Fixed versus variable word lengths. (10%)

4. *Symbolic Coding and Assembly Systems*. Mnemonic operation codes, labels, symbolic addresses and address expressions. Literals, extended machine operations, and pseudo operations. Error flags and messages, updating, and program documentation. Scanning of symbolic instructions and symbol table construction. Overall design and operation of assemblers. (10%)

5. *Selected Programming Techniques (chosen from among the following)*. Techniques for sorting, searching, scanning, and converting data. String manipulation, text editing, and list processing. Stack management, arithmetic expression recognition, syntactic recognition, and other compilation techniques. (10%)

6. *Logic Design, Micro-programming, and Interpreters*. AND, OR, and NOT elements, design of a half-adder and an adder, storage and delay elements, and design of an arithmetic unit. Parallel versus serial arithmetic, encoding and decoding logic, and micro-programming. Interpreters, simulation, and emulation. Logical equivalence between hardware and software. (5%)

7. *Macros*. Definition, call, and expansion of macros. Nested and recursive macro calls and definitions. Parameter handling, conditional assembly, and assembly time computations. (10%)

8. *Program Segmentation and Linkage*. Subroutines, coroutines, and functions. Subprogram loading and linkage, common data linkage, transfer vectors, and parameters. Dynamic storage allocation,

overlays, re-entrant subprograms, and stacking techniques. Linkage using page and segment tables. (10%)

9. *Computer Systems Organization*. Characteristics and use of tapes, disks, drums, cores, data-cells, and other large-volume devices in storage hierarchies. Processing unit organization, input-output channels and devices, peripheral and satellite processors, multiple processor configurations, computer networks, and remote access terminals. (10%)

10. *Systems and Utility Programs*. Loaders, input-output systems, monitors, and accounting programs. Program libraries. Organization, documentation, dissemination, and maintenance of system programs. (10%)

11. *Recent Developments*. Selected topics in computer organization, technology, and programming systems. (5%)

12. *Examinations*. (5%)

ANNOTATED BIBLIOGRAPHY

Whereas many of the books on "computer programming" might seem to be appropriate texts or references for this course, only a few even begin to approach the subject as proposed for this course. Most books deal with specific machines, actual or hypothetical, but very few discuss computer organization from any general point of view or consider the techniques of symbolic programming by any method other than examples. A few of the many books which deal with specific machines have been included in this list, but no manufacturers' manuals have been listed even though they may be used effectively as supplemental material.

1. BROOKS, F. P., JR., AND IVERSON, K. E. *Automatic Data Processing*. Wiley, New York, 1963, 494 pp. CR-6673-9523.

On computing fundamentals, machine language organization and programming using IBM 650 as the principal example.

2. DAVIS, G. B. *An Introduction to Electronic Computers*. McGraw-Hill, New York, 1965, 541 pp.

Informally written text containing a general introduction to computing, rather complete coverage of FORTRAN and COBOL, and considerable material on machines and machine language programming.

3. FISCHER, F. P., AND SWINDLE, G. F. *Computer Programming Systems*. Holt, Rinehart and Winston, New York, 1964, 643 pp. CR-6455-6299.

Part I is concerned with machine oriented programming and programming systems using IBM 1401 as the illustrative computer.

4. FLORES, I. *Computer Programming*. Prentice-Hall, Englewood Cliffs, N. J., 1966, 386 pp. CR-6674-10,060.

Covers machine language and software techniques using the Flores Assembly Program (FLAP) for illustrative purposes.

5. HASSITT, A. *Computer Programming and Computer Systems*. Academic Press, New York, 1967, 374 pp. CR-6784-12,355.

Discusses various features of computer organization and programming languages using examples from a number of machines including IBM 1401, 1620, 7090 and System/360, and CDC 1604 and 3600.

6. IVERSON, K. E. *A Programming Language*. Wiley, New York, 1962, 286 pp. CR-6671-9004.

Introduces a language used extensively for description of computers as well as for description of computer programs. Contains material on machine organization, sorting and data structures.

7. STARK, P. A. *Digital Computer Programming*. Macmillan, New York, 1967, 525 pp.

Presents machine language and symbolic programming for a 24-bit computer.

8. STEIN, M. L., AND MUNRO, W. D. *Computer Programming: A Mixed Language Approach*. Academic Press, New York, 1964, 459 pp. CR-6455-6140.

A text on computer organization and assembly language programming using CDC 1604 as the basic computer.

9. WEGNER, P. *Programming Languages, Information Structures*

and *Machine Organization*. McGraw-Hill, New York, 1968, about 410 pp.

Covers machine languages, multiprogramming, assembler construction and procedure-oriented languages. Programming languages are treated as information structures.

Course B3. Introduction to Discrete Structures (3-0-3)

APPROACH

The theoretical material should be introduced in a mathematically precise manner with all concepts and results being amply motivated and being illustrated with examples from computer science. The student should be given extensive homework assignments of both a theoretical and a programming nature which further the understanding of the applications of the concepts in computer science.

CONTENT

Since the material listed below is more than can normally be offered in a one-semester three-credit course on this level, care must be taken to select those topics which will support the more advanced courses as they are developed at each particular school. The description in each of the four sections is divided into two parts labeled (a) Theory and (b) Applications, but in practice the material in both parts would be intermixed.

1. Basic Set Algebra.

a. Theory: Sets and basic set algebra. Direct products. Mappings, their domains and ranges, and inverse mappings. Finite and denumerable sets. Relations including order relations. Set inclusion as partial ordering. Equivalence relations, equivalence classes, partition of sets, congruences. The preservation of relations under mappings. Finite sets and their subsets. Permutations, combinations, and related combinatorial concepts.

b. Applications: Examples of sets. The Peano axioms for the set of integers. Congruences and ordering relations over the integers. Relations over the integers defined by arithmetic operations. The set of all subsets of an n-element set and the set of all n-digit binary numbers. The set of all strings over a finite alphabet. Languages over an alphabet as subsets of the set of all strings over the alphabet. Algorithms for listing combinations, compositions, or partitions. Algorithms for ranking combinations.

2. Basic Algebraic Structures.

a. Theory: Operations on a set. Algebraic structures as sets with particular functions and relations defined on it. Groups, subgroups, cyclic groups, and other examples of groups. The concepts of homomorphism and isomorphism on a set with operations. Semigroups and semigroups of transformations. Definition and general discussion of examples of structures with several operations, e.g. fields and possibly lattices.

b. Applications: Computer use for working group theoretic problems, e.g. with permutation groups as they occur as input transformation in switching networks. The semigroup of all words over a fixed finite alphabet under the operation of concatenation. The letters of the alphabet as generators. Pair algebra.

3. Boolean Algebra and Propositional Logic.

a. Theory: The axioms of set algebra. Axiomatic definition of Boolean algebras as algebraic structures with two operations. Duality. Basic facts about Boolean functions. Propositions and propositional functions. Logical connectives. Truth values and truth tables. The algebra of propositional functions. The Boolean algebra of truth values. Conjunctive and disjunctive normal forms.

b. Applications: Boolean algebra and switching circuits. Basic computer components. Decision tables.

4. Graph Theory.

a. Theory: Directed and undirected graphs. Subgraphs, chains, circuits, paths, cycles, connectivity, trees. Graphs and their rela-

tion to partial orderings. Graph isomorphisms. Cyclomatic and chromatic numbers. The adjacency and the incidence matrices. Minimal paths. Matchings of bipartite graphs. Elements of transport networks.

b. Applications: Flow charts and state transition graphs. Connectivity in flow charts. Syntactic structure of arithmetic expressions as trees. Graph theoretic examples in coding theory. Algorithms for determining cycles and minimal paths. Basic elements of list structures. Accessing problems. Graphs of a game. Matching algorithms and some related applications.

ANNOTATED BIBLIOGRAPHY

1. BECKENBACH, E. F. (Ed.) *Applied Combinatorial Mathematics*. Wiley, New York, 1964, 608 pp.
A collection of articles on a broad spectrum of topics. Not directly suitable as a text, but an excellent source of ideas and an important reference.
2. BERGE, C. *Theory of Graphs and Its Applications*. Wiley, New York, 1962, 244 pp.
A good presentation of directed and undirected graph theory, with some attention to algorithms. The work suffers from many misprints and errors which have been carried over into the English translation. A general reference text for this course.
3. BIRKHOFF, G., AND BARTEE, T. *Modern Applied Algebra*, Preliminary edition, *Parts I and II*. McGraw-Hill, New York, 1967.
Preliminary edition available only in limited quantities, but the full text expected by the fall of 1968. Appears to be very close in spirit to the material proposed for this course, but the content is more algebraically oriented and includes little on graphs.
4. SACKER, R., AND SAATY, T. *Finite Graphs and Networks: An Introduction with Applications*. McGraw-Hill, New York, 1965, 294 pp.
A good work on graph theory with a very nice collection of applications. Useful as source and reference for the graph theory part of this course.
5. GROSSMAN, I., AND MAGNUS, W. *Groups and Their Graphs*. Random House, New York, 1965, 195 pp. CR-6564-8003.
An elementary but very well written discourse on basic connections between group and graph theory.
6. HARARY, F., NORMAN, R. Z., AND CARTWRIGHT, D. *Structural Models: An Introduction to the Theory of Directed Graphs*. Wiley, New York, 1965, 415 pp. CR-6566-8421.
Excellent on directed graphs and probably the best source book on that field. Should be an important reference for the corresponding portion of this course.
7. HOHN, F. *Applied Boolean Algebra*, 2nd ed. Macmillan, New York, 1966, 273 pp.
Very good introduction to basic facts of Boolean algebra and especially its applications in electrical engineering. Important reference for the corresponding portion of this course.
8. KEMENY, J., MIRKIL, H., SNELL, J., AND THOMPSON, G. *Finite Mathematical Structures*. Prentice-Hall, Englewood Cliffs, N. J., 1959, 487 pp.
A text for physical science and engineering students who have completed the calculus. First two chapters on compound statements, sets, and functions should be particularly useful.
9. KEMENY, J., SNELL, J., AND THOMPSON, G. *Introduction to Finite Mathematics*, 2nd ed. Prentice-Hall, Englewood Cliffs, N. J., 1966, 352 pp.
Freshman-sophomore level text designed primarily for students in biological and social sciences. Follows CUPM recommendations for the mathematical education of such students. First three chapters on compound statements, sets and subsets, partitions, and counting cover similar material as proposed for this course.
10. KORFHAGE, R. *Logic and Algorithms: With Applications to the Computer and Information Sciences*. Wiley, New York, 1966, 194 pp. CR-6782-11,339.

A fine new text introducing those basic topics from mathematical logic important in computer science—for instance Boolean algebra, Turing machines, and Markov algorithms. Written in the spirit which should pervade this course.

11. LEDERMAN, W. *Introduction to the Theory of Finite Groups*. Interscience, New York, 1953, 160 pp.
A very readable introduction to finite groups. Particularly interesting to this course is the chapter on permutation groups.
12. MACLANE, S., AND BIRKHOFF, G. *Algebra*. Macmillan, New York, 1967, 598 pp.
A substantially revised and updated version of *A Survey of Modern Algebra*, which has been a classic text on modern algebra. Should be one of the main references for the algebraic parts of this course.
13. ORE, O. *Graphs and Their Uses*. Random House, New York, 1963, 131 pp.
An introduction to the elementary concepts of graph theory. Very pleasant to read.
14. RIORDAN, J. *An Introduction to Combinatorial Analysis*. Wiley, New York, 1958, 244 pp.
One of the best source books on enumerative combinatorial analysis. However, it is too advanced for use as a text in a course of this type.
15. RYSER, H. *Combinatorial Mathematics*. Wiley, New York, 1963, 154 pp. CR-6562-7371.
An excellent introduction to such topics as (0,1) matrices, Latin-squares, and block-design, but containing almost no graph theory.
16. WHITESITT, J. E. *Boolean Algebra and Its Applications*. Addison-Wesley, Reading, Mass., 1961, 182 pp.
An introductory text designed for readers with a limited mathematical background.

Course B4. Numerical Calculus (2-2-3)

APPROACH

In this course the emphasis is placed upon building algorithms for the solution of numerical problems, the sensitivity of these algorithms to numerical errors, and the efficiency of these algorithms. In the laboratory portion of the course the student is to complete a substantial number of computational projects using a suitable procedure-oriented language.

CONTENT

1. *Basic Concepts of Numerical Error*. Significant digit arithmetic rounding procedures. Classification of error, evaluation of expressions and functions.
2. *Interpolation and Quadrature*. Polynomial interpolation, elements of difference calculus, Newton and Lagrange formulas, Aitken's interpolation method, quadrature formulas, Romberg integration, numerical differentiation, and the inherent error problems.
3. *Solution of Nonlinear Equations*. Bisection method, successive approximations including simple convergence proofs, linearization and Newton's method, method of false-position. Applications to polynomial equations. Generalization to iterative methods for systems of equations.
4. *Linear Systems of Equations*. Solution of linear systems and determinant evaluation by elimination procedures. Roundoff errors and ill-conditioning. Iterative methods.
5. *Numerical Solution of Ordinary Differential Equations*. Euler's method, modified Euler's method, simplified Runge-Kutta.

ANNOTATED BIBLIOGRAPHY

Listed below are some of the books which might be used as texts and/or references for this course. Most of the books cover the following topics: solution of polynomial and other nonlinear equations;

interpolation, numerical quadrature, and numerical differentiation; ordinary differential equations; and linear algebra. Significant deviations from these topics are indicated by the annotation.

1. CONTE, S. D. *Elementary Numerical Analysis: An Algorithmic Approach*. McGraw-Hill, New York, 1965, 278 pp.

Designed as a text for a one-semester, three-hour course for engineering and science undergraduate students. Machine-oriented treatment with many illustrative examples including flow charts and FORTRAN programs. Except for the chapter on differential equations, a knowledge of basic calculus and of programming in a procedure-oriented language is sufficient background. Numerous exercises.

2. JENNINGS, W. *First Course in Numerical Methods*. Macmillan, New York, 1964, 233 pp. CR-6671-9036.

Designed as a text for a one-semester course for advanced undergraduate students in science and engineering. Brief treatment of the standard topics. Presupposes calculus, differential equations, some experience with the computer, and, for later chapters, matrices. Some exercises.

3. MACON, N. *Numerical Analysis*. Wiley, New York, 1963, 161 pp.

Designed as a text for a one-semester first course in numerical analysis. Emphasis is more on the mathematical aspects rather than the computational aspects although there is an introductory chapter on the elements of computing, flow charting, and FORTRAN programming. For the early chapters calculus provides sufficient background. For later chapters an elementary knowledge of matrix theory, differential equations, and advanced calculus is recommended. Examples and exercises.

4. MCCORMICK, J. M., AND SALVADORI, M. G. *Numerical Methods in FORTRAN*. Prentice-Hall, Englewood Cliffs, N.J., 1964, 324 pp. CR-6676-10,883.

Designed as a text either for an elementary course in numerical analysis at the junior-senior level or for a course in programming. First part presents the methods without reference to programming techniques. There are 320 examples and problems. The last part contains 53 completely worked illustrative FORTRAN programs. Presupposes beginning analysis.

5. MCCRACKEN, D., AND DORN, W. S. *Numerical Methods and FORTRAN Programming*. Wiley, New York, 1964, 457 pp. CR-6562-7107.

Designed as a text for a four semester-hour course in science or engineering at the sophomore-senior level. Emphasis on practical methods—for example, the treatment of simultaneous linear algebraic equations does not make use of matrices. Chapters on various aspects of FORTRAN are interspersed with chapters on numerical methods. Includes a brief chapter on partial differential equations. Presupposes beginning analysis. Examples and exercises.

6. MILNE, W. E. *Numerical Calculus*. Princeton University Press, Princeton, N. J., 1949, 393 pp.

Written in 1949 in the early days of computing, this is a very useful reference even though the treatment is oriented toward manual computation and though some of the methods have been superseded. Presupposes a knowledge of calculus and differential equations. Examples and exercises.

7. NIELSEN, K. L. *Methods in Numerical Analysis*, 2nd ed. Macmillan, New York, 1956 and 1964, 382 pp. CR-6455-6333.

Designed as a textbook for a practical course for engineers. Primary emphasis on the use of desk calculators and tables. Presupposes calculus. Examples and exercises.

8. PENNINGTON, R. H. *Introductory Computer Methods and Numerical Analysis*. Macmillan, New York, 1965, 452 pp. CR-6565-8060.

Designed as a text for a one-year elementary course for scientists and engineers to be taken immediately after integral calculus. The first part treats digital computers and programming. Numerical methods are then discussed from a computer viewpoint with the aid of flow diagrams. Little knowledge of computing is assumed. For some of the topics a knowledge of matrices and

ordinary differential equations would be helpful. Many examples and exercises.

9. SINGER, J. *Elements of Numerical Analysis*. Academic Press, New York, 1964, 395 pp. CR-6561-6959.

Designed as a text for junior undergraduate students in mathematics. Treatment geared more to manual computation than to the use of computers. Presupposes beginning analysis and, for some parts, differential equations and advanced calculus. Examples and exercises.

10. STIEFFEL, E. L. *An Introduction to Numerical Mathematics*, transl. by W. C. and C. J. Rheinboldt. Academic Press, New York, 1963, 286 pp. CR-6455-6335.

Appropriate for a junior-senior level course in mathematics, science, and engineering. Emphasis is on the algorithmic approach, although there are only a few flow charts and specific references to programs. A wide variety of topics and methods is treated. Basic calculus is required for the early chapters, but for later chapters familiarity with ordinary differential equations is desirable. Examples are given. There is a separate problem supplement with 36 exercises.

Course I1. Data Structures (3-0-3)

APPROACH

This course is intended to present the data structures which may be used in computer storage to represent the information involved in solving problems. However, emphasis should be placed on treating these data structures independently of the applications in which they are embedded. Each data structure should be motivated carefully in terms of the operations which may conveniently be performed, and illustrated with examples in which the structure is useful. The identification of the natural relations between entities involved in problems and alternate representations of information should be stressed. Computer storage structures should also be described and classified according to their characteristics, and the interaction between data structures and storage structures should be studied.

The student should be required to apply the techniques presented to problems which illustrate a wide variety of data structures. Solutions to a number of these problems should be programmed and run on a computer.

CONTENT

More material is listed here than can normally be covered in a one-semester course. The instructor should carefully select material which gives the student a broad introduction to this subject, but which fits together pedagogically. It may be desirable to develop an advanced course to cover some of these topics more completely.

1. *Basic Concepts of Data*. Representation of information as data inside and outside the computer. Bits, bytes, fields and data items. Records, nodes and data elements. Data files and tables. Names, values, environments, and binding times of data. Use of pointer or linkage variables to represent data structure. Identifying entities about which data is to be maintained, and selecting data nodes and structures which are to be used in problem solution. Storage media, storage structures, encoding of data and transformations from one medium and/or code to another. Alternative representations of information and data. Packing, unpacking, and compression of data. Data formats, data description languages, and specification of data transformations.

2. *Linear Lists and Strings*. Stacks, last-in-first-out, first-in-first-out, double-ended, and other linear lists. Sequential versus linked storage allocation. Single versus double linkage. Circular lists. Character strings of variable length. Word packing, part-word addressing, and pointer manipulation. Insertion, deletion and accessing of list elements.

3. *Arrays and Orthogonal Lists*. Storage of rectangular arrays in one-dimensional media. Storage mapping functions, direct and in-

direct address computation, space requirements, set-up time, accessing time, and dynamic relocation time. Storage and accessing triangular arrays, tetrahedral arrays, and sparse matrices.

4. *Tree Structures*. Trees, subtrees, ordered trees, free trees, oriented trees and binary trees. Representation of trees using binary trees, sequential techniques, or threaded lists. Insertion, deletion, and accessing elements of trees. Relative referencing, finding successors and predecessors, and walking through trees. Examples of tree structures such as algebraic formulas, arrays, and other hierarchic data structures (PL/I and COBOL).

5. *Storage Systems and Structures*. Behavioral properties of unit record (card), random access (core), linear (tape), and intermediate (disk, drum, etc.) storage media and devices including cost, size, speed, reusability, inherent record and file structure, and deficiencies and interrelation of these properties. Influence of machine structure—in particular addressing—on data structuring. Hierarchies of storage, virtual memory, segmentation, paging, and bucketing. Influence of data structures and data manipulation on storage systems. Associative structures, both hardware and software.

6. *Storage Allocation and Collection*. Static versus dynamic allocation. Sequential versus linked allocation. Last-in-first-out data versus data of unrelated life times. Uniform block size and available space lists. Variable block size and stratified available space lists. Explicit release of available storage. Coalescing adjacent free space and compacting occupied space or data. Accessing disciplines for movable data, unmovable anchors, and updating of pointers. Reference counts and list borrowing. Garbage collection by surveying active data.

7. *Multilinked Structures*. Use of different types of data nodes or elements. Use of different types of linkage to sequence, adjoin, or associate data elements and to build hierarchies of data structures. Sublists, list names, list heads, and attribute lists. Multidimensional linked lists and mixed list structures. Accessing, insertion, deletion and updating. Relative referencing, finding successors and predecessors, and walking through structures. Representation of graphs and networks. Structures used for string manipulation and list processing languages.

8. *Sorting (Ordering) Techniques*. Radix sorting, radix exchange sorting, merge sorting, bubble sorting, address table sorting, topological sorting and other sorting methods. Comparative efficiency of sorting techniques. Effect of data structures and storage structures on sorting techniques.

9. *Symbol Tables and Searching*. Linear, stack, tree and scatter structured tables, and table lookup techniques. Hash code algorithms. Use of index lists and associative techniques. Comparison of search strategies in terms of speed and cost. Batching and ordering of requests to remote storage to minimize number of accesses. TRIE memory as an example of structure organized for searching.

10. *Data Structures in Programming Languages*. Compile-time and run-time data structures needed to implement source language data structures of programming languages. Linkage between partially executed procedures, data structures for coroutines, scheduled procedures, and other control structures, and storage management of data structures in procedure-oriented languages. Examples of higher level languages which include list processing and other data structuring features.

11. *Formal Specification of Data Structures*. Specification of syntax for classes of data structures. Predicate selectors and constructors for data manipulation, data definition facilities, programs as data structures, computers as data structures and transformations, formal specification of semantics, and formal systems viewed as data structures.

12. *Generalized Data Management Systems*. Structures of generalized data management systems: directory maintenance, user languages (query), data description maintenance, and job management. Embedding data structures in generalized data management systems. Examples of generalized data management systems and comparison of system features.

Although a great deal of material is available in this area, very little of it is appropriate for classroom use.

1. Association for Computing Machinery. ACM sort symposium, Nov. 29-30, 1962, Princeton, N. J. *Comm. ACM* 6, 5 (May 1963), 194-272.
Seventeen papers on various aspects of sorting.
2. Association for Computing Machinery. Papers presented at the ACM Storage Allocation Symposium, June 23-24, 1961, Princeton, N. J. *Comm. ACM* 4, 10 (Oct. 1961), 416-464.
Eleven papers on various techniques of storage allocation.
3. Association for Computing Machinery. Proceedings of the ACM Symposium on Symbolic and Algebraic Manipulation, Washington, D. C., Mar. 29-31, 1966. *Comm. ACM* 9, 8 (Aug. 1966), 547-643.
Eleven papers some of which discuss applications of data structuring techniques. One paper by Knowlton describes the list language L⁶.
4. CLIMENSON, W. D. File organization and search techniques. In C. A. Cuadra (Ed.), *Annual Review of Information Science and Technology*, Vol. 1, (Amer. Doc. Inst., Ann. Rev. ser.), Interscience, New York, 1966, pp. 107-135. CR-6783-11,900.
Surveys file organizations and data structures with particular emphasis on developments during 1965. Provides framework for some of the material covered by this course. An extensive bibliography.
5. COHEN, J. A use of fast and slow memories in list-processing languages. *Comm. ACM* 10, 2 (Feb. 1967), 82-86.
Describes a paging scheme which keeps the "most often called pages in the fast memory" and involves a slow down of 3 to 10 as compared with in-core operations.
6. Control Data Corporation. *3600/3800 INFOL Reference Manual*. Publication No. 60170300, CDC, Palo Alto, Calif., July 1966.
Describes the *INFORMATION Oriented Language* which is designed for information storage and retrieval applications.
7. DAHL, O.-J., AND NYGAARD, K. SIMULA—an ALGOL-based simulation language. *Comm. ACM* 9, 9 (Sept. 1966), 671-678.
Contains interesting data and control structures.
8. D'IMPERIO, M. Data structures and their representation in storage. In M. Halpern (Ed.), *Annual Review in Automatic Programming*, Vol. 5, Pergamon Press, New York, spring 1968.
Defines certain basic concepts involved in the representation of data and processes to be performed on data. Analyzes a problem and describes nine different solutions involving different data structures. Discusses ten list processing languages and gives examples of their data and storage structures.
9. FITZWATER, D. R. A storage allocation and reference structure. *Comm. ACM* 7, 9 (Sept. 1964), 542-545. CR-6561-6933.
Describes a method of structuring and referencing dynamic structures in AUTOCODER for the IBM 7070/72/74.
10. General Electric Company. *Integrated Data Store—A New Concept in Data Management*. Application Manual AS-CPB-483A, Revision of 7-67, GE Computer Division, Phoenix, Ariz., 1967.
Describes a sophisticated data management system which uses paging and chaining to develop complex data structures.
11. GRAY, J. C. Compound data structures for computer-aided design: a survey. Proc. ACM 22nd Nat. Conf., 1967, Thompson Book Co., Washington, D. C., pp. 355-365.
Considers requirements of a data structure software package and surveys a number of such packages.
12. HELLERMAN, H. Addressing multidimensional arrays. *Comm. ACM* 5, 4 (Apr. 1962), 205-207. CR-6235-2619.
Surveys direct and indirect methods for accessing arrays.
13. IVERSON, K. E. *A Programming Language*. Wiley, New York, 1962, 286 pp. CR-6671-9004.

- Contains considerable material on data structures, graphs, trees, and sorting, as well as a language for describing these.
14. KLEIN, M. M. Scheduling project networks. *Comm. ACM* 10, 4 (Apr. 1967), 225-234. CR-6784-12,275.
Discusses project networking and describes the C-E-I-R critical path algorithm.
 15. KNUTH, D. E. *The Art of Computer Programming, Vol. 1, Fundamental Algorithms*. Addison-Wesley, Reading, Mass., 1968, 634 pp.
Chap. 2 on "Information Structures" contains the first comprehensive classification of data structures to be published. Each structure considered is carefully motivated and generously illustrated. Includes a brief history of data structuring and an annotated bibliography.
 16. LANDIN, P. J. The mechanical evaluation of expressions. *Comput. J.* 6, 4 (Jan. 1964), 308-320. CR-6456-6677.
Presents a mathematical language based on Church's λ -notation and uses it to describe computational structures such as expressions and lists.
 17. LAWSON, H. W., JR. PL/I list processing. *Comm. ACM* 10, 6 (June 1967), 358-367.
Discusses the list processing facilities in PL/I.
 18. MADNICK, S. E. String processing techniques. *Comm. ACM* 10, 7 (July 1967), 420-424.
Presents and evaluates six techniques for string data storage structures. One of these techniques is used for an implementation of SNOBOL on an IBM System/360.
 19. MARRON, B. A., AND DE MAINE, P. A. D. Automatic data compression. *Comm. ACM* 10, 11 (Nov. 1967), 711-715.
Describes a three-part compressor which can be used on "any" body of information to reduce slow external storage requirements and to increase the rate of information transmission through a computer.
 20. MEALY, G. H. Another look at data. Proc. AFIPS 1967 Fall Joint Comput. Conf., Vol. 31, Thompson Book Co., Washington, D. C., pp. 525-534.
Sketches a theory of data based on relations. Includes some rather precise definitions of concepts such as data structure, list processing, and representation.
 21. MINKER, J., AND SABLE, J. File organization and data management. In C. A. Cuadra (Ed.), *Annual Review of Information Science and Technology, Vol. 2*, (Amer. Doc. Inst., Ann. Rev. ser.). Interscience, New York, 1967, pp. 123-160.
Surveys file organizations and generalized data management systems developed during 1966. Describes linkage types, data structures, storage structures, and how data structures have been mapped into storage structures. Extensive bibliography.
 22. MORRIS, R. Scatter storage techniques. *Comm. ACM* 11, 1 (Jan. 1968), 38-44.
Surveys hashing schemes for symbol table algorithms.
 23. ROSEN, S. (Ed.) *Programming Languages and Systems*. McGraw-Hill, New York, 1967, 734 pp.
Part 4 of this collection contains papers on IPL-V, COMIT, SLIP, SNOBOL, LISP and a comparison of list-processing computer languages.
 24. ROSS, D. T. The AED free storage package. *Comm. ACM* 10, 8 (Aug. 1967), 481-492.
Describes a storage allocation and management system for the mixed n-component elements ("beads") needed for "plex programming."
 25. SALTON, G. Data manipulation and programming problems in automatic information retrieval. *Comm. ACM* 9, 3 (Mar. 1966), 204-210. CR-6674-10,078.
Describes a variety of representations for tree structured data and examines their usefulness in retrieval applications.
 26. SAVITT, D. A., LOVE, H. H., JR., AND TROOP, R. E. ASP: a new concept in language and machine organization. Proc. 1967 Spring Joint Comput. Conf., Vol. 30, Thompson Book Co., Washington, D. C., pp. 87-102.
Describes the data bases used in the "Association-Storing Processor." These structures are complex in organization and may vary dynamically in both organization and content.
 27. SCHORR, H., AND WAITE, W. M. An efficient machine-independent procedure for garbage collection in various list structures. *Comm. ACM* 10, 8 (Aug. 1967), 501-506.
Reviews and compares past garbage collection methods and presents a new algorithm.
 28. STANDISH, T. A. A data definition facility for programming languages. Ph.D. Thesis, Carnegie Institute of Technology, Pittsburgh, Pa., 1967.
Presents a descriptive notation for data structures which is embedded in a programming language.
 29. WEGNER, P. (Ed.) *Introduction to Systems Programming*. Academic Press, New York, 1965, 316 pp. CR-6455-6300.
Contains a collection of papers of which the following are of special interest for this course: Illiffe, pp. 256-275; Jenkins, pp. 283-293; and Burge, pp. 294-312.
 30. WEGNER, P. *Programming Languages, Information Structures, and Machine Organization*. McGraw-Hill, New York, 1968, about 410 pp.
Introduces information structures and uses them in describing computer organization and programming languages.

Course 12. Programming Languages (3-0-3)

APPROACH

This course is intended to survey the significant features of existing programming languages with particular emphasis on the underlying concepts abstracted from these languages. The relationship between source programs and their run-time representation during evaluation will be considered, but the actual writing of compilers is to be taught in Course 15.

CONTENT

There are four basic parts of this course: the structure of simple statements; the structure of algorithmic languages; list processing and string manipulation languages; and topics in programming languages.

Part A. Structure of Simple Statements. (10 lectures)

1. Informal syntax and semantics of arithmetic expressions and statements, translation between infix, prefix, and postfix notation, and the use of pushdown stores for translation and execution of arithmetic expressions and statements. Precedence hierarchy of arithmetic operations, relational operators, and Boolean operators. Backus normal form representation of syntax of arithmetic statements and the semantics of arithmetic statements. (6 lectures)

2. Precedence relations, precedence grammars, and syntactic analysis of precedence grammars. Application to arithmetic expressions, code generation, error diagnostics and error correction for syntactic arithmetic expression compilation. (4 lectures)

Part B. Structure of Algorithmic Languages. (20 lectures)

3. Review of program constituents, branching statements and loops. (2 lectures)

4. Grouping of statements, declarations, "types" of program constituents, nomenclature, scopes, local and nonlocal quantities, independent blocks (FORTRAN), and nested blocks (ALGOL). (2 lectures)

5. Function and statement type procedures, formal parameters and actual parameters, and call by value, name and reference. Binding time of program constituents, recursive procedures, and side effects during execution of procedures. (3 lectures)

6. Storage allocation for independent blocks (FORTRAN) and storage allocation for nested blocks and procedures using a run-time

pushdown store. Overall structure of an ALGOL-style compiler. (3 lectures)

7. Coroutines, tasks, interrupt specification, and classification of control structures in procedure-oriented languages. (2 lectures)

8. Syntactic specification of procedures, blocks and statements. Formal semantics corresponding to syntactic specification. Survey of principal concepts of syntactic analysis. (5 lectures)

9. Generalized arrays. Data definition facilities, pointer-valued variables, and list creation and manipulation using pointer-valued variables. Templates and controlled storage allocation. Distinction between data specification by a data template and the creation of instances of a specified data structure. (3 lectures)

Part C. List Processing and String Manipulation Languages. (7 lectures)

10. List structures, basic operations on list structures, LISP-like languages, machine-oriented list processing languages (IPL-V), embedding of list operations in algorithmic languages (SLIP), dynamic storage allocation for list languages, and garbage collection. (5 lectures)

11. String structures, operations on strings, and functions which have strings as arguments and strings as their values. (SNOBOL). (2 lectures)

Part D. Topics in Programming Languages. (8 lectures)

12. Additional features of programming languages, simulation languages, algebraic manipulation languages, and languages with parallel programming facilities. (2-6 lectures)

13. Formal description of languages and their processors. The work of Floyd, Wirth, and others. (2-6 lectures)

14. Other topics selected by the instructor.

ANNOTATED BIBLIOGRAPHY

- American Standards Association X3.4.1 Working Group. Toward better documentation of programming languages. *Comm. ACM* 6, 3 (Mar. 1963), 76-92.
A series of papers describing the documentation of significant current programming languages.
- Association for Computing Machinery. Proceedings of the ACM programming languages and pragmatics conference, San Dimas, Calif., August 8-12, 1965. *Comm. ACM* 9, 3 (Mar. 1966), 137-232.
Includes a number of papers applicable to this course.
- Association for Computing Machinery. Proceedings of the ACM symposium on symbolic and algebraic manipulation, Washington, D. C., March 29-31, 1966. *Comm. ACM* 9, 8 (Aug. 1966), 547-643.
A number of languages for symbolic and algebraic manipulation are described in this special issue.
- DAHL, O.-J., AND NYGAARD, K. SIMULA—an ALGOL-based simulation language. *Comm. ACM* 9, 9 (Sept. 1966), 671-678.
Describes a language encompassing ALGOL, but having many additional features including those needed for simulation.
- GALLER, B. A., AND PERLIS, A. J. A proposal for definitions in ALGOL. *Comm. ACM* 10, 4 (Apr. 1967), 204-219.
Describes a generalization of ALGOL which allows new data types and operators to be declared.
- GOODMAN, R. (Ed.) *Annual Review in Automatic Programming, Vols. 1, 2, 3, 4*. Pergamon Press, New York, 1960 to 1965. CR-6123-0811, CR-6235-2602, and CR-6564-7901.
These volumes contain several papers which are applicable to this course.
- HALSTEAD, M. H. *Machine-Independent Computer Programming*. Spartan Books, New York, 1962.
Contains both internal and external specifications of the NELIAC programming language.
- IEEE Computer Group. The special issue on computer languages. *IEEE Trans. EC-13*, 4 (Aug. 1964), 343-462.
Contains articles on ALGOL, FORTRAN, FORMAC, SOL and other computer languages.
- International Business Machines. PL/I Language Specification. Form C28-6571-4, IBM System/360 Operating System, IBM Corporation, White Plains, N. Y., 1967.
A specification of the PL/I language.
- International Standards Organization Technical Committee 97, Subcommittee 5. Survey of programming languages and processors. *Comm. ACM* 6, 3 (Mar. 1963), 93-99.
An international survey of current and imminent programming languages.
- KNUTH, D. E. The remaining trouble spots in ALGOL 60. *Comm. ACM* 10, 10 (Oct. 1967), 611-618.
This paper lists the ambiguities which remain in ALGOL 60 and which have been noticed since the publication of the Revised ALGOL 60 Report in 1963.
- MARKOWITZ, H. M., KARR, H. W., AND HAUSNER, B. *SIMSCRIPT: A Simulation Programming Language*. Prentice-Hall, Englewood Cliffs, N. J., 1963, 138 pp.
A description of the SIMSCRIPT simulation language. There is a new SIMSCRIPT 1.5 supplement now available which describes a generalization of the original language.
- MOOERS, C. N. TRAC, a procedure-describing language for the reactive typewriter. *Comm. ACM* 9, 3 (Mar. 1966), 215-219. CR-6674-10,079.
Describes a language for the manipulation of text from an on-line typewriter.
- NAUR, P. (Ed.) Revised report on the algorithmic language, ALGOL 60. *Comm. ACM* 6, 1 (Jan. 1963), 1-17. CR-6016-0323.
The Backus normal form notation was developed to help describe the syntax of ALGOL in the original version of this report (*Comm. ACM* 3, 5 (May 1960), 299-314).
- PERLIS, A. J. The synthesis of algorithmic systems—first annual A. M. Turing lecture. *J. ACM* 14, 1 (Jan. 1967), 1-9. CR-6782-11,512.
A stimulating talk on the nature of programming languages and the considerations which should underlie their future development.
- ROSEN, S. (Ed.) *Programming Systems and Languages*. McGraw-Hill, New York, 1967, 734 pp.
This collection of papers contains many of the important references for this course. In particular, Parts 1 and 2 of the collection are useful for Parts A and B of the course and Part 4 of the collection is useful for Part C of the course.
- SHAW, C. J. A comparative evaluation of JOVIAL and FORTRAN IV. *Automatic Programming Inf., No. 22*. Technical College, Brighton, England, Aug. 1964, 15 pp. CR-6562-7265.
A descriptive point-by-point comparison of these two languages. Concerned mainly with the features of the languages rather than their processors.
- SHAW, C. J. A programmer's look at JOVIAL, in an ALGOL perspective. *Datamation* 7, 10 (Oct. 1961), 46-50. CR-6233-1933.
An interesting article showing how ALGOL and JOVIAL evolved from ALGOL 58 and how they differ.
- USA Standards Institute. Standards X3.9-1966, FORTRAN and X3.10-1966. Basic FORTRAN. USASI, 10 East 40th Street, New York, N. Y. 10016, 1966.
Standard definitions of essentially FORTRAN II and FORTRAN IV. These also appeared in almost final form in *Comm. ACM* 7, 10 (Oct. 1964), 591-625.
- WEGNER, P. *Programming Languages, Information Structures, and Machine Organization*. McGraw-Hill, New York, 1968, about 410 pp.
Develops a unified approach to the study of programming languages emphasizing the treatment of such languages as infor-

mation structures. First two chapters devoted to machine organization, machine language, and assembly language, but much of Chap. 3 and essentially all of Chap. 4 devoted to the topics of this course.

21. WIRTH, N. A generalization of ALGOL. *Comm. ACM* 6, 9 (Sept. 1963), 547-554. CR-6451-5030.

Proposes a generalization of ALGOL which involves the elimination of "type" declarations and the replacement of procedure declarations by an assignment of a so-called "quotation."

22. WIRTH, N., AND WEBER, H. EULER—a generalization of ALGOL and its formal definition, Parts I and II. *Comm. ACM* 9, 1 (Jan. 1966), 13-23, and 2 (Feb. 1966), 89-99.

Develops a method for defining programming languages which introduces a rigorous relationship between structure and meaning. The structure of a language is defined by a phrase structure syntax and the meaning is defined in terms of the effects which the execution of a sequence of interpretation rules has upon a fixed set of variables called the "environment."

Course 13. Computer Organization (3-0-3) or (3-2-4)

APPROACH

This course is intended to introduce the student to the basic ideas of computer elements and logic design techniques and to the principles of computer systems organization. Emphasis should be placed on the various alternative possibilities which must be considered in arriving at a computer design; choices such as character or word organized data, serial or parallel data transmission, synchronous or asynchronous control should be compared and evaluated.

In addition to using block diagrams, it is recommended that a formal descriptive language for computer specification be introduced and used to provide a uniform method for the presentation of much of the material. The student should carry out a detailed computer design project and evaluate his design by simulation, if possible. A laboratory in which simple logic elements may be combined to perform digital functions is also desirable.

CONTENT

The following topics are to be covered, although not necessarily in the order listed.

1. *Basic Digital Circuits.* Switches, relays, transistors, diodes, magnetic cores, circuits of individual elements, and integrated circuits. (These topics may be spread throughout the course.) (5%)

2. *Boolean Algebra and Combinational Logic.* Boolean values, variables, operations, expressions and equations. Logic elements such as AND, OR, NOT, NAND and NOR. Correspondence between Boolean functions and combinations of logic elements. (5%)

3. *Data Representation and Transfer.* Flip-flops, registers, core storage, and other memory elements. Review of number representations, binary versus binary-coded decimal representation, and integer versus floating-point representation. Weighted and nonweighted codes, redundancy, and coding of character information. Coders and decoders. Clearing, gating and other transfer considerations. (10%)

4. *Digital Arithmetic.* Counters, comparators, parity checkers, and shift registers. Half and full adders. Serial versus parallel adders. Subtraction and signed magnitude versus complemented arithmetic. Multiplication and division algorithms. Integer versus floating-point arithmetic. Double precision arithmetic. Elementary speed-up techniques for arithmetic. (10%)

5. *Digital Storage and Accessing.* Structure of core memory, memory control, data buses, and address buses. Addressing and accessing methods including index registers, indirect addressing, base registers, and other techniques. Overlapping, interleaving, protection, dynamic relocation, and memory segmentation methods. Characteristics of drum, disk, tape, and other surface recording media and devices. Data flow in multimemory systems and hierarchies of storage. (10%)

6. *Control Functions.* Synchronous versus asynchronous control. Time pulse distributors, controlled delay techniques, and Gray code control sequencers. Instruction repertoire, decoding networks, and sequencing methods. Centralized, decentralized, and micro-programmed control units. Internal, external and trapping interrupts. Interrupt sensing, priority, selection, recognition, and processing. Input-output control. (10%)

7. *Input-Output Facilities.* Characteristics of input-output devices and their controllers. Relationship between input-output devices, main storage, auxiliary storage, buffers, data channels, and multiplexers. Serial versus parallel transmission. Low speed, high speed, and burst mode data flow. (10%)

8. *System Organization.* Overall organization of modules into a system. Interface between modules. Word-oriented versus character-oriented machines. Simplex and multiprocessor machines. Special purpose computers. Relationship between computer organization and software. (15%)

9. *Reliability.* Error detection and correction. Diagnostics and preventive maintenance. Start-up, power-down, and recovery procedures. (5%)

10. *Description and Simulation Techniques.* Definition of a formal computer description language which would be used in discussing most of the other topics listed for the course. Use of a computer simulator to design and test simple computers or computer modules. (10%)

11. *Selected Topics.* Multiple arithmetic units, instruction overlapping, and look-ahead techniques. Discussion of alternate organizations including highly parallel machines. (5%)

12. *Examinations.* (5%)

ANNOTATED BIBLIOGRAPHY

As indicated, several of the books listed below might possibly be used as texts for this course, but it probably would be good to supplement any of them with additional material. Only a few of the many references on computer description languages and programs for simulating computer designs are listed; no annotations are given for these.

General textbooks

1. BARTEE, T. C. *Digital Computer Fundamentals.* McGraw-Hill, New York, 1960, 1966, 401 pp. CR-6676-10,647.
Not advanced enough but a very useful supplement for circuits and equipment.
2. BARTEE, T. C., LEBOW, I. L., AND REED, I. S. *Theory and Design of Digital Systems.* McGraw-Hill, New York, 1962, 324 pp. CR-6344-4416.
Very mathematical and somewhat out-of-date. An interesting reference.
3. BRAUN, E. L. *Digital Computer Design—Logic, Circuitry, and Synthesis.* Academic Press, New York, 1963, 606 pp. CR-6453-5484.
Somewhat out-of-date for a text but useful as a reference.
4. BUCHHOLZ, W. *Planning a Computer System.* McGraw-Hill, New York, 1962, 336 pp. CR-6346-4786.
Good reference on systems concepts but somewhat dated.
5. Burroughs Corporation. *Digital Computer Principles.* McGraw-Hill, New York, 1962, 507 pp.
Restricted scope (engineering oriented) and dated, but could be used as a reference.
6. CHU, Y. *Digital Computer Design Fundamentals.* McGraw-Hill, New York, 1962, 481 pp. CR-6343-4198.
Good reference which contains a wealth of material on logic design.
7. FLORES, I. *Computer Logic.* Prentice-Hall, Englewood Cliffs, N. J., 1960, 458 pp. CR-6122-0641 and CR-6124-0936.
Dated and unorthodox but possibly useful for supplementary reading.
8. FLORES, I. *The Logic of Computer Arithmetic.* Prentice-Hall,

Englewood Cliffs, N. J., 1963, 493 pp. CR-6452-5458.

Very detailed, unorthodox treatment of computer arithmetic.

9. GSCHWIND, H. W. *Design of Digital Computers*. Springer-Verlag, New York, 1967.
A possible text.
10. HELLERMAN, H. *Digital Computer System Principles*. McGraw-Hill, New York, 1967, 424 pp.
A possible text. Uses Iverson notation throughout. Would have to be supplemented on circuits and equipment as well as novel organizations.
11. MALEY, G. A., AND SKIKO, E. J. *Modern Digital Computers*. Prentice-Hall, Englewood Cliffs, N. J., 1964, 216 pp. CR-6561-7081.
A possible reference. Somewhat dated but contains a good description of the IBM 7090 and 7080 machines.
12. MURTHA, J. C. Highly parallel information processing systems. In F. L. Alt (Ed.), *Advances in Computers*, Vol. 7. Academic Press, New York, 1966, pp. 1-116. CR-6782-11,678.
A useful reference on highly parallel systems.
13. PHISTER, M., JR. *Logical Design of Digital Computers*. Wiley, New York, 1958, 408 pp.
Somewhat dated. Relies heavily on sequential circuit theory and concentrates on serial, clocked machines.
14. RICHARDS, R. K. *Arithmetic Operations in Digital Computers*. D. Van Nostrand, Princeton, N. J., 1955, 397 pp.
Somewhat dated but still a good reference for arithmetic.
15. RICHARDS, R. K. *Electronic Digital Systems*. Wiley, New York, 1966, 637 pp. CR-6676-10,649.
An interesting reference for reliability and design automation. Discusses telephone systems and data transmission.

References on computer description languages

16. CHU, Y. An ALGOL-like computer design language. *Comm. ACM* 8, 10 (Oct. 1965), 607-615. CR-6672-9315.
17. FALKHOFF, A. D., IVERSON, K. E., AND SUSSENGUTH, E. H. A formal description of System /360. *IBM Syst. J.* 3, 3 (1964), 198-263.
18. GORMAN, D. F., AND ANDERSON, J. P. A logic design translator. Proc. AFIPS 1962 Fall Joint Comput. Conf., Vol. 22. Spartan Books, New York, pp. 251-261.
19. IVERSON, K. E. *A Programming Language*. Wiley, New York, 1962, 286 pp. CR-6671-9004.
20. McCLURE, R. M. A programming language for simulating digital systems. *J. ACM* 12, 1 (Jan. 1965), 14-22. CR-6563-7634.
21. PARNAS, D. L., AND DARRINGER, J. A. SODAS and a methodology for system design. Proc. AFIPS 1967 Fall Joint Comput. Conf., Vol. 31, Thompson Book Co., Washington, D. C., pp. 449-474.
22. WILBER, J. A. A language for describing digital computers. M.S. Thesis, Report No. 197, Dept. of Comput. Sci., U. of Illinois, Urbana, Ill., Feb. 15, 1966.

Course I4. Systems Programming (3-0-3)

APPROACH

This course is intended to bring the student to grips with the actual problems encountered in systems programming. To accomplish this it may be necessary to devote most of the course to the study of a single system chosen on the basis of availability of computers, systems programs, and documentation.

The course should begin with a thorough review of "batch" processing systems programming, emphasizing loading and subroutine linkage. The limitations of these systems should be used to motivate the more complex concepts and details of multiprogramming and multiprocessor systems. The theoretical concepts and

practical techniques prescribed in Course II should be used to focus on the data bases, their design for the support of the functions of the key system components (hardware and software), and the effective interrelation of these components. Problem assignments should involve the design and implementation of systems program modules; the design of files, tables or lists for use by such modules; or the critical use and evaluation of existing system programs. Other problems might involve the attaching or accessing of procedure or data segments of different "ownership" that are resident in a single file system or the development of restricted accessing methods (i.e. privacy schemes) and other such techniques.

CONTENT

This description has been written with MULTICS in mind as the system chosen for central study, but the description can be modified to fit any reasonably comprehensive system. There is considerably more material listed here than can normally be covered in one semester, so that careful selection of topics should be made or the course should be extended to two semesters.

1. *Review of Batch Process Systems Programs*. Translation, loading, and execution. Loader languages. Communication between independent program units. Limitations imposed by binding at pre-execution times. Incremental linkage.

2. *Multiprogramming and Multiprocessor Systems*. General introduction to the structure of these systems, the techniques involved in their construction and some of the problems involved in their implementation.

3. *Addressing Techniques*. Review of indexing and indirect addressing. Relocation and base registers. Two-dimensional addressing (segmentation). Segmented processes. Concepts of virtual memory. Effective address computation. Modes of access control. Privileged forms of accessing. Paging. Physical register (address) computation, including use of associative memories.

4. *Process and Data Modules*. Concept of a process as a collection of procedure and data components (segments). Process data bases. Controlled sharing of segments among two or more processes. Intersegment linking and segment management. Interprocedure communication. Process stacks. Levels of isolation within a process (rings of protection).

5. *File System Organization and Management*. File data bases and their storage structures. Accessing, protection and maintenance of files. Storage and retrieval of segments and/or pages from files in secondary storage (segment and page control, directory control, and core management). Search strategies.

6. *Traffic Control*. State words. Running, ready, blocked, and inactive processes. Process switching. Priority control of waiting processes. Scheduling algorithms. Pseudo processes. System tables for process management.

7. *Explicit Input-Output References*. Auxiliary (secondary) memory references. Communication with peripheral devices. Management of input-output and other request queues. Effects of data rates on queue management.

8. *Public and Private Files*. On line and off line memory. Automatic shifting of data among devices in the storage hierarchy and "flushing" of online memory, i.e. multilevel storage management. File backup schemes and recovery from system failures.

9. *Other Topics*. Some of the following topics may be studied if time permits. They might also be covered in subsequent seminars.

a. System accounting for facilities employed by the user. Special hardware features for metering different uses. Accounting for system overhead. Factors which determine system overhead.

b. Characteristics of large systems. Overall discussion of large system management including effect of binding times for system and user process variables. Other selected topics on large systems such as the effect of new hardware components (e.g. mass memories) on the overall system design.

c. Foreground and background processes. Foreground-initiated

background processes. Remote job control. Hierarchical job control. Broadcasting.

d. Microprogramming as an equivalent of various hardware and/or software component subprograms in a computing system.

e. Command languages. Commands of a multiprogramming system. Command language interpreters.

f. Provisions for dynamic updating of the operating system without shutdown.

g. Operating behavior, e.g. system startup, (graceful) degradation, and shutdown.

ANNOTATED BIBLIOGRAPHY

In addition to the following sources of information, there are many manuals available from manufacturers which describe specific systems programs for a wide range of computers.

1. CHOROFAS, D. N. *Programming Systems for Electronic Computers*. Butterworths, London, 1962, 188 pp. CR-6566-8553.

Chapters 14, 15, and 16 contain a general discourse on the control and diagnostic functions of operating systems.

2. CLARK, W. A., MEALY, G. H., AND WITT, B. I. The functional structure of OS/360. *IBM Syst. J.* 5, 1 (1966), 3-51.

A general description in three parts of the operating system for the IBM System/360. Part I (Mealy), Introductory Survey; Part II (Witt), Job and Task Management; and Part III (Clark), Data Management.

3. DESMONDE, W. H. *Real-Time Data Processing Systems. Introductory Concepts*. Prentice-Hall, Englewood Cliffs, N. J., 1964, 192 pp. CR-6562-7236.

An elementary survey of the design and programming of real-time data processing systems based on three IBM systems: Sabre, Mercury, and Gemini.

4. ERDWINN, J. D. (Ch.) Executive control programs—Session 8. Proc. AFIPS 1967 Fall Joint Comput. Conf., Vol. 31, Thompson Book Co., Washington, D. C., pp. 201-254.

Five papers on control programs for a variety of circumstances.

5. FISCHER, F. P., AND SWINDLE, G. F. *Computer Programming Systems*. Holt, Rinehart and Winston, New York, 1964, 643 pp. CR-6455-6299.

Sets out to "discuss the entire field of computer programming systems," but in reality considers primarily the systems programs for the IBM 1401; "other computer systems are mentioned only where a particular characteristic of a programming system, found on that computer, warrants discussion." Only IBM computer systems and (with a very few exceptions) only IBM literature are referenced.

6. FLORES, I. *Computer Software*. Prentice-Hall, Englewood Cliffs, N. J., 1965, 464 pp. CR-6671-8995.

Elementary and conversational text primarily concerned with assembly systems using FLAP (Flores Assembly Program) as its example. Some material on service programs, supervisors and loaders.

7. GLASER, E. (Ch.) A new remote accessed man-machine system—Session 6. Proc. AFIPS 1965 Fall Joint Comput. Conf., Vol. 27, Pt. 1, Spartan Books, New York, pp. 185-241. (Reprints available from the General Electric Company.)

Six papers on the MULTICS system.

8. HEISTAND, R. E. An executive system implemented as a finite-state automaton. *Comm. ACM* 7, 11 (Nov. 1964), 669-677. CR-6562-7282.

Describes the executive system for the 473L command and control system. The system was considered as a finite automaton and the author claims this approach forced a modularity on the resulting program.

9. LEONARD, G. F., AND GOODROE, J. R. An environment for an operating system. Proc. ACM 19th Nat. Conf., 1964, Association for Computing Machinery, New York, pp. E2.3-1 to E2.3-11. CR-6561-6546.

An approach to computer utilization involving the extension

of the operations of a computer with software so as to provide a proper environment for an operating system.

10. MARTIN, J. *Design of Real-Time Computer Systems*. Prentice-Hall, Englewood Cliffs, N. J., 1967, 629 pp.

A general text covering many aspects of real-time data processing systems including design, applications, management, and operation.

11. MARTIN, J. *Programming Real-Time Computer Systems*. Prentice-Hall, Englewood Cliffs, N. J., 1965, 386 pp.

Based on some of the early systems such as Sage, Project Mercury, Sabre, and Panamac. A general coverage designed for managers, systems analysts, programmers, salesmen, students.

12. MILLER, A. E. (Ch.) Analysis of time-shared computer system performance—Session 5. Proc. ACM 22nd Nat. Conf., 1967, Thompson Book Co., Washington, D. C., pp. 85-109.

Three papers on measurement of time-shared system performance.

13. M.I.T. Computation Center. *Compatible Time-Sharing System: A Programmer's Guide*, 2nd ed. M.I.T. Press, Cambridge, Mass., 1965.

A handbook on the use of CTSS which contains valuable information and guidelines on the implementation of such systems.

14. Project MAC. *MULTICS System Programmer's Manual*. Project MAC, M.I.T., Cambridge, Mass., 1967, (limited distribution).

A description of and guide to systems programming for MULTICS.

15. ROSEN, S. (Ed.) *Programming Systems and Languages*. McGraw-Hill, New York, 1967, 734 pp.

Collection of important papers in the area of which Part 5 (Operating Systems) is particularly relevant to this course.

16. ROSENBERG, A. M. (Ch.) Program structures for the multiprogramming environment—Session 6A. Proc. ACM 21st Nat. Conf., 1966, Thompson Book Co., Washington, D. C., pp. 223-239.

Two papers: one on program behavior under paging; the other on analytic design of look-ahead and program segmenting systems.

17. ROSENBERG, A. M. (Ch.) Time-sharing and on-line systems—Session 7. Proc. ACM 22nd Nat. Conf., 1967, Thompson Book Co., Washington, D. C., pp. 135-175.

Three papers on various topics related to the subject.

18. SALTZER, J. H. Traffic control in a multiplexed computer system. M.I.T. Ph.D. Thesis, June 1966. (Also available as Project MAC publication MAC-TR-30.)

On traffic control in the MULTICS system.

19. SMITH, J. W. (Ch.) Time-shared scheduling—Session 5A. Proc. ACM 21st Nat. Conf., 1966, Thompson Book Co., Washington, D. C., pp. 139-177.

Four papers on time-sharing which are more general than the session title indicates.

20. THOMPSON, R. N., AND WILKINSON, J. A. The D825 automatic operating and scheduling program. Proc. AFIPS 1963 Spring Joint Comput. Conf., Vol. 23, Spartan Books, New York, pp. 41-49. CR-6453-5699.

A general description of an executive system program for handling a multiple computer system tied to an automatic input-output exchange containing a number of input-output control modules. Discusses many of the problems encountered in such systems and the general plan of attack in solving these problems.

21. WEGNER, P. (Ed.) *Introduction to System Programming*. Academic Press, New York, 1965, 316 pp. CR-6455-6300.

Contains a collection of papers of which the following are of special interest for this course: Gill, pp. 214-226; Howarth, pp. 227-238; and Nash, pp. 239-249.

Course I5. Compiler Construction (3-0-3)

APPROACH

This course is to emphasize the techniques involved in the analysis of source language and the generation of efficient object code. Although some theoretical topics must be covered, the course should have the practical objective of teaching the student how compilers may be constructed. Programming assignments should consist of implementations of components of a compiler and possibly the design of a simple but complete compiler as a group project.

CONTENT

There is probably more material listed here than can reasonably be covered, so some selection will be necessary.

1. Review of assembly techniques, symbol table techniques, and macros. Review of syntactic analysis and other forms of program recognition. Review of compilation, loading, and execution with emphasis on the representation of programs in the loader language.

2. One-pass compilation techniques. Translation of arithmetic expressions from postfix form to machine language. Efficient use of registers and temporary storage.

3. Storage allocation for constants, simple variables, arrays, temporary storage. Function and statement procedures, independent block structure, nested block structure, and dynamic storage allocation.

4. Object code for subscripted variables, storage mapping functions, and dope vectors. Compilation of sequencing statements.

5. Detailed organization of a simple complete compiler. Symbol tables. Lexical scan on input (recognizer), syntax scan (analyzer), object code generators, operator and operand stacks, output subroutines, and error diagnostics.

6. Data types, transfer functions, mixed mode expressions and statements.

7. Subroutine and function compilation. Parameters called by address, by name and by value. Subroutines with side effects. Restrictions required for one pass execution. Object code for transmission of parameters. Object code for subroutine body.

8. Languages designed for writing compilers: TMG (McClure), COGENT (Reynolds), GARGOYLE (Garwick), META II (Schorre), and TGS-II (Cheatham).

9. Bootstrapping techniques. Discussion of a meta-compiler in its own language.

10. Optimization techniques. Frequency analysis of use of program structures to determine most important features for optimization.

11. Local optimization to take advantage of special instructions. Loading registers with constants, storing zeros, changing sign, adding to memory, multiplication or division by two, replacement of division with multiplication by a constant, squaring, raising to integer powers, and comparing to zero. Subscript optimization.

12. Expression optimization. Identities involving minus signs, common subexpression evaluation and other techniques. Minimization of temporary storage and the number of arithmetic registers in multiple register machines.

13. Optimization of loops. Typical loops coded several ways. Index register optimization in the innermost loop. Classification of loops for optimization purposes.

14. Problems of global optimization. Determination of flowchart graph of program. Analysis of program graphs. Rearrangement of computation to do as little as possible in innermost loop. Factoring of invariant subexpressions. Object code for interfaces between flow blocks.

ANNOTATED BIBLIOGRAPHY

1. ACM Compiler Symposium. Papers presented at the ACM Compiler Symposium, November 17-18, 1960, Washington, D. C., *Comm. ACM* 4, 1 (Jan. 1961), 3-84.

Contains a number of relevant papers including one by R. W. Floyd entitled "An Algorithm for Coding Efficient Arithmetic Operations" and one by P. Z. Ingerman on "Thunks."

2. ARDEN, B. W., GALLER, B. A., AND GRAHAM, R. M. An algorithm for translating Boolean expressions. *J. ACM* 9, 2 (Apr. 1962), 222-239. CR-6341-3567.

Description of code generation in the MAD Compiler.

3. BRINCH-HANSEN, P., AND HOUSE, R. The COBOL compiler for the Siemens 3003. *BIT* 6, 1 (1966), 1-23.

Describes the design of a ten-pass compiler with extensive error detection.

4. CHEATHAM, T. E., JR. The TGS-II translator generator system. Proc. IFIP Congress, New York, 1965, Vol. 2, Spartan Books, New York, pp. 592-593.

A report on the "current position" of Computer Associates "translator generator system."

5. CHEATHAM, T. E., JR. *The Theory and Construction of Compilers*. Document CA-6606-0111, Computer Associates, Inc., Wakefield, Mass., June 2, 1966, limited distribution.

Notes for course AM 295 at Harvard, fall 1967.

6. CHEATHAM, T. E., JR., AND SATTLEY, K. *Syntax-directed compiling*. Proc. AFIPS 1964 Spring Joint Comput. Conf., Vol. 25, Spartan Books, New York, pp. 31-57. CR-6455-6304.

An introduction to top-down syntax directed compilers.

7. CONWAY, M. E. Design of a separable transition-diagram compiler. *Comm. ACM* 6, 7 (July 1963), 396-408. CR-6451-5024.

Describes the organization of a COBOL compiler. The methods are largely applicable to construction of compilers for other languages such as ALGOL.

8. CONWAY, R. W., AND MAXWELL, W. L. CORC—the Cornell computing language. *Comm. ACM* 6, 6 (June 1963), 317-321.

Description of a language and compiler which are designed to provide extensive error diagnostics and other aids to the programmer.

9. ERSHOV, A. P. ALPHA—an automatic programming system of high efficiency. *J. ACM* 13, 1 (Jan. 1966), 17-24. CR-6673-9720.

Describes a compiler for a language which includes most of ALGOL as a subset. Several techniques for optimizing both the compiler and the object code are presented.

10. ERSHOV, A. P. *Programming Programme for the BESM computer*, transl. by M. Nadler. Pergamon Press, New York, 1959, 158 pp. CR-6235-2595.

One of the earliest works on compilers. Introduced the use of stacks and the removal of common subexpressions.

11. ERSHOV, A. P. On programming of arithmetic operations. *Comm. ACM* 1, 8 (Aug. 1958), 3-6 and 9 (Sept. 1958), 16.

Gives an algorithm for creating rough machine language instructions in pseudoform and then altering them into a more efficient form.

12. FREEMAN, D. N. Error correction in CORC. Proc. AFIPS 1964 Fall Joint Computer Conf., Vol. 26, Part I, Spartan Books, New York, pp. 15-34.

Discusses techniques of correcting errors in programs written in the Cornell computing language.

13. GARWICK, J. V. GARGOYLE, a language for compiler writing. *Comm. ACM* 7, 1 (Jan. 1964), 16-20. CR-6453-5675.

Describes an ALGOL-like language which uses syntax-directed methods.

14. GEAR, C. W. High speed compilation of efficient object code. *Comm. ACM* 8, 8 (Aug. 1965), 483-488. CR-6671-9000.

Describes a three-pass compiler which represents a compromise between compilation speed and object code efficiency. Primary attention is given to the optimization performed by the compiler.

15. GRIES, D., PAUL, M., AND WIEHLE, H. R. Some techniques used in the ALCOR ILLINOIS 7090. *Comm. ACM* 8, 8 (Aug. 1965), 496-500. CR-6566-8556.
Describes portions of an ALGOL compiler for the IBM 7090.
16. HAWKINS, E. N., AND HUXTABLE, D. H. R. A multipass translation scheme for ALGOL 60. In R. Goodman (Ed.), *Annual Review in Automatic Programming, Vol. 3*, Pergamon Press, New York, 1963, pp. 163-206.
Discusses local and global optimization techniques.
17. HORWITZ, L. P., KARP, R. M., MILLER, R. E., AND WINOGRAD, S. Index register allocation. *J. ACM* 13, 1 (Jan. 1966), 43-61. CR-6674-10,068.
A mathematical treatment of the problem. Useful in compiler writing.
18. International Computation Centre (Eds.) *Symbolic Languages in Data Processing, Proceedings of the Symposium in Rome, March 26-31, 1962*, Gordon and Breach, New York, 1962, 849 pp.
The twelve papers listed under "Construction of Processors for Syntactically Highly Structured Languages" in this volume are particularly of interest for this course.
19. KNUTH, D. E. A history of writing compilers. *Comput. Autom.* 11, 12 (Dec. 1962), 8-18.
Describes some of the early techniques used in writing American compilers.
20. McCLURE, R. M. TMG—a syntax directed compiler. Proc. ACM 20th Nat. Conf., 1965, Association for Computing Machinery, New York, pp. 262-274.
The compiler writing system described in this paper was designed to facilitate the construction of simple one-pass translators for some specialized languages. It has features which simplify the handling of declarative information and errors.
21. NAUR, P. The design of the GIER ALGOL compiler. In R. Goodman (Ed.), *Annual Review in Automatic Programming, Vol. 4*, Pergamon Press, New York, 1964, pp. 49-85. CR-6564-7904.
Describes a multipass compiler written for a computer with a small high-speed memory.
22. RANDELL, B., AND RUSSELL, L. J. *ALGOL 60 Implementation*. Academic Press, New York, 1964, 418 pp. CR-6565-8246.
Contains a survey of ALGOL implementation techniques and a description of an error-checking and debugging compiler for the KDF9 computer.
23. REYNOLDS, J. C. An introduction to the COGENT programming system. Proc. ACM 20th Nat. Conf., 1965, Association for Computing Machinery, New York, pp. 422-436.
Describes the structure and major facilities of a compiler-compiler system which couples the notion of syntax-directed compiling with that of recursive list processing.
24. ROSEN, S. (Ed.) *Programming Systems and Languages*. McGraw-Hill, New York, 1967, 734 pp.
A collection of papers of which the following are of special interest for this course: Backus, et al., pp. 29-47; Bauer and Samelson, pp. 206-220; Dijkstra, pp. 221-227; Kanner, Kosinski, and Robinson, pp. 228-252; Rosen, Spurgeon, and Donnelly, pp. 264-297; and Rosen, pp. 306-331.
25. SCHORRE, D. V. META II, a syntax-oriented compiler writing language. Proc. ACM 19th Nat. Conf., 1964, Association for Computing Machinery, New York, pp. D1.3-1 to D1.3-11. CR-6561-6943.
Describes a compiler writing language in which its own compiler can be written.
26. WEGNER, P. (Ed.) *Introduction to System Programming*. Academic Press, New York, 1965, 316 pp. CR-6455-6300.
Contains a collection of papers of which the following are of special interest for this course: Pyle, pp. 86-100; Wegner, pp. 101-121; Randell, pp. 122-136; Huxtable, pp. 137-155; Hoare, pp. 156-165; and d'Agapeyeff, pp. 199-214.

Course I6. Switching Theory (3-0-3) or (2-2-3)

APPROACH

This course should present the theoretical foundations and mathematical techniques concerned with the design of logical circuits. Examples should be chosen to illustrate the applicability to computers or other digital systems whenever possible. Facility with Boolean algebra and appreciation for the effects of delays should be developed. Some laboratory experiments are highly desirable.

CONTENT

1. Review of nondecimal number systems. Introduction to unit-distance, error-correcting, and other codes.
2. Development of switching algebra and its relation to Boolean algebra and propositional logic. Brief discussion of switching elements and gates. Analysis of gate networks. Truth tables and completeness of connectives.
3. Simplification of combinational networks. Use of map and tabular techniques. The prime implicant theorem. Threshold logic.
4. Different modes of sequential circuit operation. Flow table, state diagram, and regular expression representations. Clocked circuits. Flip-flop and feedback realizations.
5. Synthesis of sequential circuits. State minimization and internal variable assignments for pulse and fundamental mode circuits. Race considerations. Iterative and symmetric networks.
6. Effects of delays. Static, dynamic, and essential hazards.

ANNOTATED BIBLIOGRAPHY

As is indicated, several of the books listed below could be used as texts for this course, but it probably would be desirable to supplement any of them with additional material.

General textbooks

1. CALDWELL, S. H. *Switching Circuits and Logical Design*. Wiley, New York, 1958, 686 pp.
The classic book on relay-oriented switching theory.
2. HARRISON, M. A. *Introduction to Switching and Automata Theory*. McGraw-Hill, New York, 1965, 499 pp. CR-6671-9109.
A mathematical and abstract reference for advanced topics.
3. HIGONNET, R. A., AND GREY, R. A. *Logical Design of Electrical Circuits*. McGraw-Hill, New York, 1958, 220 pp.
Almost exclusively devoted to relay networks.
4. HUMPHREY, W. S., JR. *Switching Circuits with Computer Applications*. McGraw-Hill, New York, 1958, 264 pp.
A somewhat out-of-date undergraduate level text.
5. KRIEGER, M. *Basic Switching Circuit Theory*. Macmillan, New York, 1967, 256 pp. CR-6784-12,510.
Basic elementary treatment which does not discuss hazards or codes.
6. McCLUSKEY, E. J., JR. *Introduction to the Theory of Switching Circuits*. McGraw-Hill, New York, 1965, 318 pp. CR-6673-9834.
A possible text for this course.
7. McCLUSKEY, E. J., JR., AND BARTEE, T. C. (Eds.) *A Survey of Switching Circuit Theory*. McGraw-Hill, New York, 1962, 205 pp. CR-6342-3958.
A collection of papers. Weak as a text since there are no problems. Possibly of some value as reading to illustrate different approaches.
8. MALEY, G. A., AND EARLE, J. *The Logic Design of Transistor Digital Computers*. Prentice-Hall, Englewood Cliffs, N. J., 1963, 322 pp. CR-6345-4582.
Despite its title, this book covers a considerable amount of switching theory. Emphasis is on NOR circuits and asynchronous systems, and on techniques rather than theorems. Numerous examples.

9. MARCUS, M. P. *Switching Circuits for Engineers*. Prentice-Hall, Englewood Cliffs, N. J., 1962, 296 pp. CR-6341-3681.
Broad but not too mathematical coverage of switching theory.
 10. MILLER, R. E. *Switching Theory, Vol. 1, Combinational Circuits*. Wiley, New York, 1965, 351 pp. CR-6565-8369.
Highly mathematical and somewhat advanced for an undergraduate course. Interesting discussion of the effects of delays.
 11. PRATHER, R. E. *Introduction to Switching Theory: A Mathematical Approach*. Allyn and Bacon, Boston, 1967, 496 pp.
A highly mathematical and broad coverage of both combinatorial switching theory and sequential machine theory.
 12. TORNG, H. C. *Introduction to the Logical Design of Switching Systems*. Addison-Wesley, Reading, Mass., 1965, 286 pp. CR-6456-6806.
General elementary coverage including a discussion of switching elements and magnetic logic. Outmoded discussion of iterative (cascaded) networks. Many computer-related examples.
 13. WARFIELD, J. N. *Principles of Logic Design*. Ginn and Co., Boston, 1963, 291 pp. CR-6451-5136.
Covers some elementary switching theory in the context of computer logic.
- More specialized books*
14. CURTIS, V. A. *A New Approach to the Design of Switching Circuits*. D. Van Nostrand, Princeton, N. J., 1962, 635 pp. CR-6346-4818.
Devoted mainly to decomposition theory for combinational circuits. Useful as a reference in this area and as a source of examples since it contains many detailed sample problems.
 15. DERTOUZOS, M. L. *Threshold Logic: A Synthesis Approach*. M.I.T. Press, Cambridge, Mass., 1965, 256 pp. CR-6676-10,929.
Concentrates on the characterization and application of threshold elements in terms of logical design.
 16. HU, S. T. *Threshold Logic*. University of California Press, Berkeley, Calif., 1965, 338 pp.
A comprehensive reference which also contains some of the author's original research.
 17. LEWIS, P. M., AND COATES, C. L. *Threshold Logic*. Wiley, New York, 1967, 483 pp.
Emphasizes single and multigate networks for controlled sensitivity.
 18. PHISTER, M., JR. *Logical Design of Digital Computers*. Wiley, New York, 1958, 401 pp.
Covers the application of sequential circuit theory to design of computer logic. Considers only clocked circuits and (for the most part) serial operation.

Course I7. Sequential Machines (3-0-3)

APPROACH

This is to be a rigorous theoretical course. The material is about evenly devoted to three major points of view: the structural aspects of sequential machines, the behavioral aspects of sequential machines, and the variants of finite automata. Machines with unbounded storage such as Turing machines and pushdown-store automata are to be covered in course A7.

CONTENT

1. Definition of finite automata and sequential machines. Various methods of representing automata including state tables, state diagrams, set theoretic methods, and sequential nets. (3 lectures)
2. Equivalent states and equivalent machines. Reduction of states in sequential machines. (3 lectures)
3. Right-invariant and congruence relations. The equivalence of nondeterministic and deterministic finite automata. Closure prop-

erties of languages definable by finite automata and the Kleene-Myhill theorem on regular languages. (4 lectures)

4. Decision problems of finite automata. Testing equivalence, acceptance of a nonempty set, acceptance of an infinite set. (3 lectures)
5. Incomplete sequential machines. Compatible states and algorithms for constructing minimal state incomplete sequential machines. (3 lectures)
6. The state assignment problem. Series-parallel decompositions. Coordinate assignments. (2 lectures)
7. Algebraic definition of a sequential machine. Homomorphisms of monoids. (2 lectures)
8. Partitions with the substitution property. Homomorphism decompositions into a series-composition. (2 lectures)
9. Decomposition of permutation automata. (1 lecture)
10. The decomposition of finite-state automata into a cascade of permutation-reset automata using the method of set systems (covers). (2 lectures)
11. Final series-parallel decomposition into a cascade of two-state automata and simple-group automata. (2 lectures)
12. Brief introduction to undecidability notions. The halting problem. (2 lectures)
13. Multitape nonwriting automata. (4 lectures)
14. Generalized sequential machines. (2 lectures)
15. Subsets of regular languages. Group-free automata. (3 lectures)
16. Regular expressions. Algebra, derivatives and star height. (5 lectures)
17. Probabilistic automata. (3 lectures)

ANNOTATED BIBLIOGRAPHY

Except for two survey articles, only books are included in the following list. Since this field has developed within the last fifteen years, much of the material is still in the periodical literature.

1. CAIANIELLO, E. R. (Ed.) *Automata Theory*. Academic Press, New York, 1966, 343 pp. CR-6676-10,935.
A collection of research and tutorial papers on automata, formal languages, graph theory, logic, algorithms, recursive function theory, and neural nets, which, because of varying interest and difficulty, might be useful for supplementary reading by ambitious students.
2. FISCHER, P. C. Multitape and infinite-state automata—a survey. *Comm. ACM* 8, 12 (Dec. 1965), 799-805. CR-6675-10,561.
A survey of machines which are more powerful than finite automata and less powerful than Turing machines. Also an extensive bibliography.
3. GILL, A. *Introduction to the Theory of Finite-State Machines*. McGraw-Hill, New York, 1962, 207 pp. CR-6343-4207.
An automata theory approach to finite-state machines which is somewhat engineering oriented and written at a fairly elementary level.
4. GINSBURG, S. *An Introduction to Mathematical Machine Theory*. Addison-Wesley, Reading, Mass., 1962, 137 pp. CR-6452-5431.
A text on the behavior of the sequential machines of Huffman-Moore-Mealy, abstract machines of Ginsburg, and tape recognition devices of Rabin and Scott.
5. GLUSHKOV, V. M. *Introduction to Cybernetics*, transl. by Scripta Technica, Inc. Academic Press, New York, 1966, 324 pp.
A translation of the Russian text which assumes only a limited background. Approaches subject from somewhat different point of view than most Western texts. Contains much relevant material.
6. HARRISON, M. *Introduction to Switching and Automata Theory*, McGraw-Hill, New York, 1965, 499 pp. CR-6671-9109.
This text for engineers and mathematicians develops the foundations of both switching and automata theory in abstract

mathematical terms. Emphasis is on switching theory. Coverage includes sequential machines, regular events, definite events, probabilistic machines, and context-free languages.

- HARTMANIS, J., AND STEARNS, R. E. *Algebraic Structure Theory of Sequential Machines*. Prentice-Hall, Englewood Cliffs, N. J., 1966, 211 pp. CR-6782-11,635.

The first thorough treatment of the structure theory of sequential machines and its applications to machine synthesis and machine decomposition. A research monograph selected from a series of papers by the authors and not written as a text. Practically no exercises.

- HENNIE, F. C., III. *Iterative Arrays of Logical Circuits*. M.I.T. Press, Cambridge, Mass., and Wiley, New York, 1961, 242 pp. CR-6232-1733.

Currently the most complete treatise on iterative arrays.

- KAUTZ, W. H. (Ed.) *Linear Sequential Switching Circuits—Selected Technical Papers*. Holden-Day, San Francisco, 1965, 234 pp. CR-6674-10,205.

A collection of papers on linear sequential machines.

- McNAUGHTON, R. The theory of automata—a survey. In F. L. Alt (Ed.), *Advances in Computing, Vol. 2*, Academic Press, New York, 1961, pp. 379–421. CR-6342-3920.

Most of the areas of automata theory are included with the exception of switching theory and other engineering topics.

- MILLER, R. E. *Switching Theory, Vol. 2, Sequential Circuits and Machines*. Wiley, New York, 1965, 250 pp. CR-6783-12,120.

Highly mathematical and somewhat advanced as a text for an undergraduate course.

- MINSKY, M. *Computation: Finite and Infinite Machines*. Prentice-Hall, Englewood Cliffs, N. J., 1967, 317 pp.

The concept of an “effective procedure” is developed in this text. Also treats algorithms, Post productions, regular expressions, computability, infinite and finite-state models of digital computers, and computer languages.

- MOORE, E. F. (Ed.) *Sequential Machines: Selected Papers*. Addison-Wesley, Reading, Mass., 1964, 266 pp.

This collection of classical papers on sequential machines includes an extensive bibliography by the editor.

- PRATHER, R. E. *Introduction to Switching Theory: A Mathematical Approach*. Allyn and Bacon, Boston, 1967, 496 pp.

A mathematical and broad treatment of both combinatorial switching theory and sequential machines.

- SHANNON, C. E., AND MCCARTHY, J. (Eds.) *Automata Studies*. Princeton University Press, Princeton, N. J., 1956, 285 pp.

A collection of many of the early papers on finite automata, Turing machines, and synthesis of automata which stimulated the development of automata theory. Philosophical papers, in addition to mathematical papers, are included, since the aim of the collection is to help explain the workings of the human mind.

Courses 18 and 19. Numerical Analysis I and II (3-0-3) and (3-0-3)

APPROACH

The numerical methods presented in these courses are to be developed and evaluated from the standpoint of efficiency, accuracy, and suitability for high-speed digital computing. While other arrangements of the material in these courses are possible, the ones suggested here do allow the two courses to be taught independently of one another.

CONTENT OF COURSE 18

- Solution of Equations*. Newton’s method and other iterative methods for solving systems of equations. Aitken’s δ^2 process. Newton-Bairdstow method. Muller’s method and Bernoulli’s method

for polynomial equations. Convergence conditions and rates of convergence for each method.

- Interpolation and Approximation*. Polynomial interpolation. Lagrange’s method with error formula. Gregory-Newton and other equal interval interpolation methods. Systems of orthogonal polynomials. Least-squares approximation. Trigonometric approximation. Chebyshev approximation.

- Numerical Differentiation and Quadrature*. Formulas involving equal intervals. Romberg integration. Extrapolation to the limit. Gaussian quadrature.

- Solution of Ordinary Differential Equations*. Runge-Kutta methods. Multistep methods. Predictor-corrector methods. Stability.

CONTENT OF COURSE 19

- Linear Algebra*. Rigorous treatment of elimination methods and their use to solve linear systems, invert matrices, and evaluate determinants. Compact schemes. Methods for solving the eigenvalue-eigenvector problem including the power method, the inverse power method, Jacobi’s method, Givens’ method and Householder’s method. Roundoff analysis and conditioning.

- Numerical Solution of Boundary Value Problems in Ordinary Differential Equations*.

- Introduction to the Numerical Solution of Partial Differential Equations*. Computational aspects of finite difference methods for linear equations. Determination of grids. Derivation of difference equations. Solution of large linear systems by iterative methods such as simultaneous displacements, successive displacements, and successive overrelaxation.

ANNOTATED BIBLIOGRAPHY

Listed below are some but by no means all of the books which could be used as texts and/or references for these courses. The more general texts normally include solution of polynomial and other nonlinear equations; interpolation, numerical quadrature, and numerical differentiation; ordinary differential equations; and linear algebra. Significant deviations from these are indicated by the annotations.

Besides listing books which might be used as texts for part or all of these courses, the following includes books for those desiring to go deeper into the various areas. In particular, Refs. 1, 3, 10, 17, and 18 have been included for linear algebra; Refs. 2 and 16 for partial differential equations; Ref. 15 for the solution of nonlinear equations; and Ref. 7 for ordinary differential equations.

- FADDEEV, D. K., AND FADDEEVA, V. N. *Computational Methods of Linear Algebra*, transl. by R. C. Williams. W. H. Freeman, San Francisco, 1963, 621 pp. CR-6016-0374.

An excellent reference on the theory of computational methods in linear algebra. Does not treat the theory of computational errors. Introductory chapter could serve as a text for a course in linear algebra. Beginning analysis and an elementary knowledge of complex variables is assumed. Examples but no exercises.

- FORSYTHE, G. E., AND WASOW, W. R. *Finite-Difference Methods for Partial Differential Equations*. Wiley, New York, 1960, 444 pp.

A fundamental reference on the numerical solution of partial differential equations by finite-difference methods. Provides a thorough treatment of hyperbolic, parabolic, and elliptic equations. Orientation is toward the use of high-speed computers, but it is not intended as a guide for programmers. For most of the book, advanced calculus and linear algebra provide sufficient background. Previous knowledge of partial differential equations not required. Some illustrative examples but no exercises.

- FOX, L. *An Introduction to Numerical Linear Algebra*. Oxford University Press, New York, 1964, 295 pp. CR-6456-6723.

A basic reference on computational methods in linear algebra. Designed for engineers and scientists as well as mathematicians. Emphasis on the principles involved rather than the details of applications to computers. Intended to prepare the reader for a more advanced book such as Wilkinson’s *The Algebraic Eigenvalue Problem*. Introductory chapter on matrix algebra. Illustrative examples and exercises.

4. FRÖBERG, C. E. *Introduction to Numerical Analysis*. Addison-Wesley, Reading, Mass., 1965, 340 pp. CR-6671-9037.
Designed as a text for an undergraduate numerical analysis course. Includes, in addition to the standard topics, partial differential equations (briefly), approximation by Chebyshev polynomials and other functions, Monte Carlo methods, and linear programming. Emphasis on modern methods well-adapted for computers. Mathematically rigorous treatment with detailed error analysis given in typical cases. Presupposes differential and integral calculus and differential equations. Illustrative examples and exercises.
5. HAMMING, R. W. *Numerical Methods for Scientists and Engineers*. McGraw-Hill, New York, 1962, 411 pp. CR-6236-3367.
Excellent as a reference. Provides interesting and different point of view. Treats interpolation and approximation; numerical differentiation and integration; and ordinary differential equations by polynomial and other methods such as Fourier methods, and exponentials. Brief treatments of nonlinear equations and linear algebra, simulation, and Monte Carlo methods. Presupposes beginning analysis, Fourier series, mathematical statistics, feed-back circuits, noise theory. Illustrative examples and exercises.
6. HENRICI, P. *Elements of Numerical Analysis*. Wiley, New York, 1964, 328 pp.
Designed as a text for a one-semester course in numerical analysis. Covers the standard topics except linear algebra. Emphasis on numerical analysis as a mathematical discipline. A distinction is made between algorithms and theorems. Introductory chapters on complex variables and difference equations. Beginning analysis (12 semester hours) and ordinary differential equations are assumed. Illustrative examples and exercises.
7. HENRICI, P. *Discrete Variable Methods in Ordinary Differential Equations*. Wiley, New York, 1962, 407 pp. CR-6341-3733.
A basic reference on the numerical methods for solving ordinary differential equations. Designed as a text for a senior-level course on ordinary differential equations. Includes a mathematically rigorous treatment of various methods. Emphasis is on the study of discretization errors and round-off errors. Presupposes differential equations, advanced calculus, linear algebra, and elementary complex variables (though large parts of the book do not require all of these topics). Illustrative examples and exercises.
8. HILDEBRAND, F. B. *Introduction to Numerical Analysis*. McGraw-Hill, New York, 1956, 511 pp.
A good book for supplementary reading though written in 1956. Gives primary emphasis to methods adapted for desk calculators. Includes standard topics except for linear algebra. Separate chapters on least-squares, polynomial approximation, Gaussian quadrature, and approximation of various types. Beginning analysis sufficient background for most of the book. An extensive set of exercises.
9. HOUSEHOLDER, A. S. *Principles of Numerical Analysis*. McGraw-Hill, New York, 1953, 274 pp.
Good for supplementary reading. Designed as mathematical textbook rather than a compendium of computational rules. Published in 1953, the book includes many methods applicable only to hand computation though it was written with computers in mind. Includes the standard topics except ordinary differential equations. Presupposes beginning analysis plus some knowledge of probability and statistics. Some exercises.
10. HOUSEHOLDER, A. S. *The Theory of Matrices in Numerical Analysis*. Blaisdell, New York, 1964, 257 pp.
Good for supplementary reading. Considers the development and appraisal of computational methods in linear algebra from the theoretical point of view. Does not develop specific computer flowcharts or programs. Presupposes a knowledge of matrix algebra. Illustrative examples and exercises.
11. ISAACSON, E., AND KELLER, H. B. *Analysis of Numerical Methods*. Wiley, New York, 1966, 541 pp. CR-6783-11.966.
A very well written and rather comprehensive text presenting a careful analysis of numerous important numerical methods with a view toward their applicability to computers. With an appropriate selection of material the book lends itself well to use as a text; otherwise, it is an excellent reference.
12. MILNE, W. E. *Numerical Solution of Differential Equations*. Wiley, New York, 1953, 275 pp.
Since it was written in 1953, much of this material has been superseded by more recent work; yet it remains very suitable for supplemental reading. Ordinary and partial differential equations are treated as well as some problems in linear algebra. Many of the methods are adapted for hand computation rather than for computers. Beginning analysis should provide sufficient background for most of the book. Illustrative examples and some exercises.
13. RALSTON, A. *A First Course in Numerical Analysis*. McGraw-Hill, New York, 1965, 578 pp. CR-6671-9035.
Designed as a text for a one-year course in numerical analysis (though not all of the material could be covered) to be taken by graduate students and advanced undergraduate students, primarily in mathematics. Although numerical analysis is treated as a full-fledged branch of applied mathematics, orientation is toward the use of digital computers. Basic topics in numerical analysis covered thoroughly. Separate chapters devoted to functional approximation by least-squares techniques and by minimum-maximum error techniques. Presupposes beginning analysis, advanced calculus, orthogonal polynomials, and complex variables. A course in linear algebra is assumed for the chapters in that area. An extensive set of illustrative examples and exercises.
14. TODD, J. (Ed.) *A Survey of Numerical Analysis*. McGraw-Hill, New York, 1962, 589 pp. CR-6236-3368.
Written by a number of authors. Some of the early chapters have been used in connection with introductory courses. Because of its breadth of coverage it is especially suited as a reference for these courses. Besides the usual topics, there are separate chapters on orthogonalizing codes, partial differential equations, integral equations, and problems in number theory. The prerequisites vary with the chapters but for early chapters beginning analysis and linear algebra would suffice. Exercises given in some of the early chapters.
15. TRAUB, J. F. *Iterative Methods for the Solution of Equations*. Prentice-Hall, Englewood Cliffs, N. J., 1964, 310 pp. CR-6672-9339.
A good reference on the numerical solution of equations and (briefly) systems of equations by iteration algorithms. The methods are treated with rigor, though rigor in itself is not the main object. Contains a considerable amount of new material. Many illustrative examples.
16. VARGA, RICHARD. *Matrix Iterative Analysis*. Prentice-Hall, Englewood Cliffs, N. J., 1962, 322 pp. CR-6343-4236.
An excellent reference giving theoretical basis behind methods for solving large systems of linear algebraic equations which arise in the numerical solution of partial differential equations by finite-difference methods. Designed as a text for a first-year graduate course in mathematics.
17. WILKINSON, J. H. *Rounding Errors in Algebraic Processes*. Prentice-Hall, Englewood Cliffs, N. J., 1964, 161 pp. CR-6455-6341.
Studies the cumulative effect of rounding errors in computations involving large numbers of arithmetic operations performed by digital computers. Special attention given to problems involving polynomials and matrices. A very important reference for a computer-oriented course in numerical analysis.
18. WILKINSON, J. H. *The Algebraic Eigenvalue Problem*. Clarendon Press, Oxford, England, 1965, 662 pp.
A basic reference on computational methods in linear algebra. Provides a thorough treatment of those methods with which the author has had direct numerical experience on the computer. Treats the methods theoretically and also from the standpoint of rounding errors. Presupposes beginning analysis, linear algebra, and elementary complex variables. Illustrative examples but no exercises.

Course A1. Formal Languages and Syntactic Analysis (3-0-3)

APPROACH

This course combines the theoretical concepts which arise in formal language theory with their practical application to the syntactic analysis of programming languages. The objective is to build a bridge between theory and practical applications, so that the mathematical theory of context-free languages becomes meaningful to the programmer and the theoretically oriented student develops an understanding of practical applications. Assignments in this course should include both computer programming assignments and theorem proving assignments.

CONTENT

The following topics should be covered, but the organization of the material and relative emphasis on individual topics is subject to individual preference.

1. Definition of a formal grammar as notation for specifying a subset of the set of all strings over a finite alphabet, and of a formal language as a set specified by a formal grammar. Production notation for specifying grammars. Recursively enumerable, context-sensitive, context-free, and finite-state grammars. Examples of languages specified by grammars such as a^nb^n , $a^nb^nc^n$.

2. Specification of arithmetic expressions and arithmetic statements as context-free grammars. Use of context-free grammars as recognizers. Use of recognizers as a component in compilation or interpretive execution of arithmetic statements.

3. Syntactic analysis, recognizers, analyzers and generators. Top-down and bottom-up algorithms. The backtracking problem and the reduction of backtracking by bounded context techniques. Theory of bounded context analysis and LR(k) grammars.

4. Precedence and operator precedence techniques. The algorithms of Floyd, Wirth and Weber, etc. Semantics of precedence grammars for arithmetic statements and simple block structure.

5. Top-down and bottom-up algorithms for context-free languages. The algorithms of Cheatham, Domolki, and others.

6. Languages for syntactic analysis and compilation such as COGENT and TMG. A simple syntactic compiler written in one of these languages.

7. Reductive grammars, Floyd productions, and semantics for arithmetic expressions. The work of Evans, Feldman, and others.

8. Theory of context-free grammars, normal forms for context-free grammars, elimination of productions of length zero and one. Chomsky and Greibach normal forms. Ambiguous and inherently ambiguous grammars. The characteristic sequence of a grammar. Strong equivalence, weak equivalence and equivalence with preservation of ambiguity. Asymptotic time and space requirements for context-free language recognition.

9. Combinatorial theorems for context-free grammars. Proof that $a^nb^nc^n$ cannot be represented by a context-free grammar. Linear and semilinear sets. Parikh's theorem.

10. Grammars and mechanical devices. Turing machines, linear bounded automata, pushdown automata, finite-state automata, and the corresponding grammars.

11. Properties of pushdown automata. Deterministic and non-deterministic automata and languages. Stack automata. Programming languages and pushdown automata.

ANNOTATED BIBLIOGRAPHY

1. BAR-HILLEL, Y., PERLES, M., AND SHAMIR, E. On formal properties of simple phrase structure grammars. *Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung* 14 (1961), 143-172. (Reprinted in Y. Bar-Hillel (Ed.), *Languages and Information, Selected Essays*. Addison-Wesley, Reading, Mass., 1964. CR-6562-7178.)
A well-written paper containing the first statement of many of the principal results of context-free languages.
2. BAUER, F. L., AND SAMELSON, K. Sequential formula translation. *Comm. ACM* 3, 2 (Feb. 1960), 76-83. CR-6015-0219.
The first systematic paper on the translation of programming languages from left to right using precedence techniques.
3. BROOKER, R. A., AND MORRIS, D. A general translation program for phrase structure grammars. *J. ACM* 9, 1 (Jan. 1962), 1-10.
Summarizes the design and machine-oriented characteristics of a syntax-directed compiler and describes both its syntactic and semantic features.
4. CHEATHAM, T. E., JR., AND SATTLEY, K. Syntax directed compiling. Proc. AFIPS 1964 Spring Joint Comput. Conf., Vol. 25, Spartan Books, New York, pp. 31-57. CR-6455-6304.
A description of one of the earliest operational top-down syntax-directed compilers.
5. CHOMSKY, N. Formal properties of grammars. In R. R. Bush, E. H. Galanter, and R. D. Luce (Eds.), *Handbook of Mathematical Psychology, Vol. 2*, Wiley, New York, 1962, pp. 323-418. CR-6676-10,731.
An excellent review of the work of both Chomsky and others in this field. Contains a good bibliography. See also the two companion chapters in this volume written by Chomsky and G. A. Miller.
6. EVANS, A. An ALGOL 60 compiler. In R. Goodman (Ed.), *Annual Review in Automatic Programming, Vol. 4*, Pergamon Press, New York, 1964, pp. 87-124. CR-6564-7905.
The first compiler design application of a reductive syntax with labelled productions.
7. FELDMAN, J. A. A formal semantics for computer languages and its application in a compiler-compiler. *Comm. ACM* 9, 1 (Jan. 1966), 3-9. CR-6674-10,080.
A description of a formal semantic language which can be used in conjunction with a language for describing syntax to specify a syntax-directed compiler.
8. FLOYD, R. W. A descriptive language for symbol manipulation. *J. ACM* 8, 4 (Oct. 1961), 579-584. CR-6234-2140.
The first discussion of reductive grammars, including a reductive grammar from which the first example in Ref. 9 was derived. The association of semantics with syntactic recognition is directly illustrated.
9. FLOYD, R. W. Syntactic analysis and operator precedence. *J. ACM* 10, 3 (July 1963), 316-333.
Defines the notions of operator grammars, precedence grammars (here called "operator precedence grammars"), precedence functions, and a number of other concepts. Examples of precedence grammars and nonprecedence grammars.
10. FLOYD, R. W. Bounded context syntactic analysis. *Comm. ACM* 7, 2 (Feb. 1964), 62-67. CR-6454-6074.
Introduces the basic concepts of bounded context grammars and gives a set of conditions for testing whether a given grammar is of bounded context (m, n).
11. FLOYD, R. W. The syntax of programming languages—a survey. *IEEE Trans. EC-13*, 4 (Aug. 1964), 346-353.
An expository paper which defines and explains such concepts as phrase-structure grammars, context-free languages, and syntax-directed analysis. Extensive bibliography.
12. GINSBURG, S. *The Mathematical Theory of Context-Free Languages*. McGraw-Hill, New York, 1966, 232 pp. CR-6783-12,074.
Presents a rigorous discussion of the theory of context-free languages and pushdown automata.
13. GREIBACH, S. A. A new normal-form theorem for context-free phrase structure grammars. *J. ACM* 12, 1 (Jan. 1965), 42-52. CR-6564-7830.
Shows that every grammar is equivalent with preservation of ambiguities to a grammar in the "Greibach Normal Form."
14. GRIFFITHS, T. V., AND PETRICK, S. R. On the relative efficiencies of context-free grammar recognizers. *Comm. ACM* 8, 5 (May 1965), 289-300. CR-6564-7999.

A comparative discussion of a number of syntactic analysis algorithms.

15. IEEE Computer Group, Switching and Automata Theory Committee. Conf. Rec. 1967 8th Ann. Symposium on Switching and Automata Theory. Special Publication 16 C 56, Institute of Electrical and Electronic Engineers, New York, 1967.
Contains a number of papers relevant to this course including those by: Rosenkrantz, pp. 14-20; Aho, pp. 21-31; Hopcroft and Ullman, pp. 37-44; Ginsburg and Greibach, pp. 128-139.
16. IRONS, E. T. A syntax directed compiler for ALGOL 60. *Comm. ACM* 4, 1 (Jan. 1961), 51-55.
The first paper on syntactic compilation. It discusses both a top-down implementation of a syntactic compiler and the way in which semantics is associated with the generation steps of such a compiler.
17. IRONS, E. T. Structural connections in formal languages. *Comm. ACM* 7, 2 (Feb. 1964), 67-72. CR-6455-6212.
The concept of structural connectedness defined is essentially the same as that of a bounded context grammar.
18. KNUTH, D. E. On the translation of languages from left to right. *Inf. Contr.* 8, 6 (Dec. 1965), 607-639. CR-6674-10,162.
The concept of a grammar which permits translation from left to right with forward context k (LR(k) grammar) is developed and analyzed.
19. McCLURE, R. M. TMG—a syntax directed compiler. Proc. ACM 20th Nat. Conf., 1965, pp. 262-274.
A description of a compiler in which the syntax is specified by an ordered sequence of labeled productions and in which semantics can be explicitly associated with productions.
20. NAUR, P. (Ed.) Revised report on the algorithmic language, ALGOL 60. *Comm. ACM* 6, 1 (Jan. 1963), 1-17. CR-6345-4540.
The first systematic application of context-free languages to the description of actual programming languages.
21. REYNOLDS, J. C. An introduction to the COGENT programming system. Proc. ACM 20th Nat. Conf., 1965, pp. 422-436.
Describes the structure and major facilities of a compiler-compiler system which couples the notion of syntax-directed compiling with that of recursive list processing.
22. WIRTH, N., AND WEBER, H. EULER: a generalization of ALGOL, and its formal definition, Parts I & II. *Comm. ACM* 9, 1 (Jan. 1966), 13-23, and 2 (Feb. 1966), 89-99.
The first discussion of pure precedence grammars and their use in ALGOL-like languages. Many of the concepts introduced by other authors are discussed in an illuminating way.

Course A2. Advanced Computer Organization (3-0-3)

APPROACH

This title could label either a course in the organization of advanced computers or an advanced course in computer organization. It is meant to be primarily the latter, with some material on novel computer organizations included. The approach is that of "comparative anatomy": first, each of several organization and system design problems should be identified; then, a comparison of the solutions to the problem should be made for several different computers; next, the rationale of each solution should be discussed; and finally, an attempt should be made to identify the best solution. Students should prepare papers on designs used in other machines for several of the problem areas. These papers should include discussions of the circuit technology available at the time the machine was designed and the intended market for the machine, and they should compare the machine design with other designs.

CONTENT

The computer system design problem areas which should be covered in this course are listed below followed by a list of some of

the machines which might be used to illustrate various solutions to these problems. Each major problem area should be discussed in general and at least three actual machine designs should be used to illustrate the widest possible range of solutions. A brief overall system description of each machine should be given before it is used to illustrate a particular design area.

Computer System Design Problem Areas

1. *Arithmetic Processing.* Integer and floating point representation, round-off and truncation, word and register lengths. Number and types of arithmetic units, malfunction detection and reaction, arithmetic abort detection, and reaction (overflow, etc.).
2. *Nonarithmetic Processing.* Addressable quanta and operation codes. Compatibility and interaction of nonarithmetic and arithmetic operands. Types of additional processing units.
3. *Memory Utilization.* Relationship between memory width and addressable quanta, memory block autonomy and phasing, memory access priorities (operands, instructions, input-output, etc.). Factoring of command-fetch processes (look-ahead).
4. *Storage Management.* Relocation, paging, and renaming. Storage protection. Hierarchy storage provisions and transfer mechanisms.
5. *Addressing.* Absolute addressing, indexing, indirect addressing, relative addressing, and base addressing.
6. *Control.* Clocking, interrupt processing, privileged mode operations, and autonomy of control functions.
7. *Input-Output.* Buffer facilities, channels (autonomy, interaction, interrupts), processing options (editing, formatting), input-output byte size versus memory width versus addressing quanta. Rate-matching (especially for input-output devices with inertia) and channel-sharing.
8. *Special System Designs.* Array or cellular computers, variable structure computers, and other advanced designs.

Illustrative Computers

ATLAS	one of the first machines to use paging
Bendix G-15	a bit serial machine with drum store
Burroughs B5000	a zero-address machine with stacks and Polish processing
CDC 6600	a very high-speed computer using look-ahead, instruction stacking and multiple peripheral processors
GIER (Denmark)	a machine which uses a small core storage with an auxiliary drum
IBM 701	a classic Von Neumann binary machine
IBM 1401	a serial character machine with peculiar addressing
IBM STRETCH	a machine which was intended to incorporate all state-of-the-art knowledge
IBM 360/ij	a series of machines which combine character and word handling capabilities
KDF-9	a computer using a nesting store
PB 440	a microprogrammable computer
SDS Sigma 7	a real-time time-sharing computer
UNIVAC I	a classic decimal machine

ANNOTATED BIBLIOGRAPHY

In addition to the references cited below, it is important that a collection of reference manuals for the actual computers be available to the student. These can be obtained from the computer manufacturers in most cases or from various reports such as the *Auerbach Standard EDP Reports*. In some cases where the title of the citation is self-explanatory, no annotation is given.

General references

1. AMDAHL, G. M., BLAAUW, G. A., AND BROOKS, F. P., JR. Architecture of the IBM System/360. *IBM J. Res. Develop.* 8, 2 (Apr. 1964), 87-101. CR-6465-8374.

Although not a technical paper, it does give some insight into

- the decisions which determined many of the features of this family of computers.
2. BLAAUW, G. A., ET AL. The structure of System/360. *IBM Syst. J.* 3, 2 (1964), 119-195.
Describes the design considerations relating to the implementation, performance, and programming of the System/360 family of computers.
 3. BOUTWELL, E. O., JR.; AND HOSKINSON, E. A. The logical organization of the PB 440 microprogrammable computer. Proc. AFIPS 1963 Fall Joint Comput. Conf., Vol. 24, Spartan Books, New York, pp. 201-213.
Describes the use of a fast-read, slow-write memory for microprograms. The bus structure connecting the processing registers allows data transfers under control of the microprogram.
 4. BUCHHOLZ, W. The system design of the IBM 701 computer. *Proc. IRE* 41, 10 (Oct. 1953), 1262-1274.
Describes one of the first commercial binary computers.
 5. BUCHHOLZ, W. (Ed.) *Planning A Computer System*. McGraw-Hill, New York, 1962, 336 pp. CR-6346-4786.
Describes in reasonable detail a design philosophy and its bearing on specific design decisions for an entire computer system—in this case, STRETCH.
 6. DEVONALD, C. H., AND FOTHERINGHAM, J. A. The ATLAS computer. *Datamation* 7, 5 (May 1961), 23-27. CR-6231-1405.
 7. Digital Computer Laboratory, University of Illinois. *On the Design of a Very High Speed Computer*. Rep. No. 80, U. of Illinois, Urbana, Ill., 1957.
A description of the first design for ILLIAC II.
 8. ECKERT, J. P., ET AL. The UNIVAC system. *Rev. of Elec. Dig. Comput.*, Proc. Joint IRE-AIEE Conf., Philadelphia, Pa., 10-12 Dec. 1951, pp. 6-14.
A description of the UNIVAC I computer.
 9. Engineering Summer Conference, University of Michigan. Theory of computing machine design (course notes). U. of Michigan, Ann Arbor, Mich., 1960-1962. (Distribution of these notes was limited to participants.)
Cover many aspects of design—from the applications of automata theory to the system design of parallel computers.
 10. FERNBACH, S. Very high-speed computers, 1964—the manufacturers' point of view. Proc. AFIPS 1964 Fall Joint Comput. Conf., Vol. 26, Pt. II, Spartan Books, New York, 1965, pp. 33-141.
Contains detailed reports on the CDC 6600, the IBM System/360 Model 92, and a Philco multiprocessing system.
 11. GRAM, C., ET AL. GIER—a Danish computer of medium size. *IEEE Trans. EC-12*, 6 (Dec. 1963), 629-650.
Gives an evaluation of the order structure and the hardware organization and describes the operating system and the ALGOL 60 system.
 12. GRAY, H. J. *Digital Computer Engineering*. McGraw-Hill, New York, 1963, 381 pp. CR-6341-3654.
Chap. 1 discusses ENIAC and EDVAC as examples of parallel and serial organization. Chaps. 2-4 deal with organization problems.
 13. GSCHWIND, H. W. *Design of Digital Computers*. Springer-Verlag, New York, 1967, 530 pp.
A general reference.
 14. HELLERMAN, H. *Digital Computer System Principles*. McGraw-Hill, New York, 1967, 424 pp.
A possible text. A good reference.
 15. HOLLANDER, G. L. (Ch.) The best approach to large computing capacity—a debate. Proc. AFIPS 1967 Spring Joint Comput. Conf., Vol. 30, Thompson Book Co., Washington, D. C., pp. 463-485.
Presents four approaches to achieving large computing capacity through: aggregation of conventional system elements (G. P. West); associative parallel processing (R. H. Fuller); an array computer (D. L. Slotnick); and the single-processor approach (G. M. Amdahl).
 16. MENDELSON, M. J., AND ENGLAND, A. W. The SDS Sigma 7: a real-time time-sharing system. Proc. AFIPS 1966 Fall Joint Comput. Conf., Vol. 29, Spartan Books, New York, pp. 51-64. CR-6700-0752.
Discusses seven critical design problems—including interrupt processing, memory protection, space sharing, and recursive processing—and their solutions.
 17. RICHARDS, R. K. *Electronic Digital Systems*. Wiley, New York, 1966, 637 pp. CR-6676-10,649.
Contains a good discussion of reliability and design automation.
 18. WALZ, R. F. Digital computers—general purpose and DDA. *Instrum. and Automat.* 28, 9 (Sept. 1955), 1516-1522.
Describes the G-15 computer and the use of a digital differential analyzer (DDA) in general purpose digital systems.
 19. WARE, W. H. *Digital Computer Technology and Design: Vol. I, Mathematical Topics, Principles of Operation and Programming; Vol. II, Circuits and Machine Design*. Wiley, New York, 1963, 237 pp. and 521 pp. CR-6562-7103 and 7104.
Useful because of its coverage of early design techniques. Contains extensive bibliographies (at the end of each chapter), which refer to papers presenting the design features of numerous machines.
- References on arithmetic and control*
20. BLAAUW, G. A. Indexing and control-word techniques. *IBM J. Res. Develop.* 3, 3 (July 1959), 288-301.
Describes some of the control techniques used in the STRETCH computer.
 21. BECKMAN, F. S., BROOKS, F. P., JR., AND LAWLESS, W. J., JR. Developments in the logical organization of computer arithmetic and control units. *Proc. IRE* 49, 1 (Jan. 1961), 53-66. CR-6232-1680.
Summarizes the developments in logical design and in arithmetic and control units through 1960. Contains a good bibliography.
 22. BROOKS, F. P., JR., BLAAUW, G. A., AND BUCHHOLZ, W. Processing data in bits and pieces. *IEEE Trans. EC-8*, 3 (June 1959), 118-124. CR-6012-0035.
Describes a data-handling unit which permits variable length binary or decimal arithmetic.
 23. SUMNER, F. H. The central control unit of the ATLAS computer. Proc. IFIP Congress, Munich, 1962, North-Holland Pub. Co., Amsterdam, pp. 292-296. CR-6342-3961.
- References on storage management*
24. ARDEN, B. W., GALLER, B. A., O'BRIEN, T. C., AND WESTERVELT, F. H. Program and addressing structure in a time-sharing environment. *J. ACM* 13, 1 (Jan. 1966), 1-16. CR-6781-11,210.
Describes the hardware and software devices used to facilitate program switching and efficient use of storage in a time-sharing computer system.
 25. BELADY, L. A. A study of replacement algorithms for a virtual memory computer. *IBM Syst. J.* 5, 2 (1966), 78-101.
Discusses several algorithms for automatic memory allocation and compares them using the results of several simulation runs.
 26. COCKE, J., AND KOLSKY, H. G. The virtual memory in the STRETCH computer. Proc. AFIPS 1959 Eastern Joint Comput. Conf., Vol. 16, Spartan Books, New York, pp. 82-93.
 27. EVANS, D. C., AND LECLERK, J. Y. Address mapping and the control of access in an interactive computer. Proc. AFIPS 1967 Spring Joint Comput. Conf., Vol. 30, Thompson Book Co., Washington, D. C., pp. 23-30.
Describes the hardware implementation of a design based on separate program and data entities.
 28. GIBSON, D. H. Considerations in block-oriented systems design. Proc. AFIPS 1967 Spring Joint Comput. Conf., Vol. 30, Thompson Book Co., Washington, D. C., pp. 75-80.
Analyzes block size, high-speed storage requirements, and job mix as they affect system design.

Reference on stack computers

29. ALLMARK, R. H., AND LUCKING, J. R. Design of an arithmetic unit incorporating a nesting store. Proc. IFIP Congress, Munich, 1962, North-Holland Pub. Co., Amsterdam, pp. 694-698. CR-6455-6460.
Describes the arithmetic unit for the KDF-9 computer.
30. HALEY, A. C. D. The KDF-9 computer system. Proc. AFIPS 1962 Fall Joint Comput. Conf., Vol. 22, Spartan Books, New York, pp. 108-120. CR-6452-5466.
Describes the stack register concept, the means used to communicate with it, and the use of zero-address instructions of variable length.
31. BARTON, R. S. A new approach to the functional design of a digital computer. Proc. AFIPS 1961 Western Joint Comput. Conf., Vol. 19, Spartan Books, New York, pp. 393-396. CR-6234-2158.
The earliest published description of a Polish string processor—the B5000.

Parallel and variable structure organizations

32. ESTRIN, G. Organization of computer systems: the fixed plus variable structure computer. Proc. AFIPS 1960 Western Joint Comput. Conf., Vol. 17, Spartan Books, New York, pp. 33-40. CR-6235-2643.
33. ESTRIN, G., AND VISWANATHAN, C. R. Organization of a fixed plus variable structure computer for computation of eigenvalues and eigenvectors of real symmetric matrices. *J. ACM* 9, 1 (Jan. 1962), 41-60.
34. ESTRIN, G., AND TURN, R. Automatic assignment of computations in a variable structure computer system. *IEEE Trans. EC-12*, 6 (Dec. 1963), 755-773.
35. GREGORY, J., AND McREYNOLDS, R. The SOLOMON computer. *IEEE Trans. EC-12*, 6 (Dec. 1963), 774-781.
Presents the system organization, functional description, and circuit design from a total systems viewpoint.
36. HOLLAND, J. H. A universal computer capable of executing an arbitrary number of subprograms simultaneously. Proc. AFIPS 1959 Eastern Joint Comput. Conf., Vol. 16, Spartan Books, New York, pp. 108-113.
37. HOLLAND, J. H. Iterative circuit computers. Proc. AFIPS 1960 Western Joint Comput. Conf., Vol. 17, Spartan Books, New York, pp. 259-265.
38. SLOTNICK, D. L., BORCK, W. C., AND McREYNOLDS, R. C. The SOLOMON computer. Proc. AFIPS 1962 Fall Joint Comput. Conf., Vol. 22, Spartan Books, New York, pp. 97-107.
A general description of the philosophy and organization of a highly parallel computer design which is a predecessor of ILLIAC IV.
39. SCHWARTZ, J. Large parallel computers. *J. ACM* 13, 1 (Jan. 1966), 25-32.
Considers various classes of machines incorporating parallelism, outlines a general class of large-scale multiprocessors, and discusses the problems of hardware and software implementation.

Course A3. Analog and Hybrid Computing (2-2-3)

APPROACH

This course is concerned with analog, hybrid, and related digital techniques for solving systems of ordinary and partial differential equations, both linear and nonlinear. A portion of the course should be devoted to digital languages for the simulation of continuous or hybrid systems (MIDAS, PACTOLUS, DSL/90, etc.). The course will have both lecture and laboratory sessions. The laboratory will allow the students to solve some problems on analog and/or hybrid computers and other problems through digital simulation of analog or hybrid computers. (Digital simulators of analog computers are

now available for digital machines of almost any size [13-22]. Some simulators are written in problem oriented languages such as FORTRAN and may be adapted to almost any computer.)

CONTENT

1. *Basic Analog Components.* Addition, multiplication by a constant, integration, function generation, multiplication, division, square roots, noise generation, and other operations. Laboratory assignments are used to familiarize the student with the operation of the components. (15%)
2. *Solution of Differential Equations.* The block-oriented approach to the solution of linear, nonlinear, and partial differential equations. Magnitude and time scaling. Estimation of maximum values. Equations with forcing functions and variable coefficients. Simultaneous equations. Statistical problems. (20%)
3. *Analog Computer Hardware.* Description of various analog computers. Amplifiers, potentiometers, and other linear and nonlinear components. The patch board and the control panel. Recording and display equipment. Slow and repetitive operation. Several laboratory assignments to give the student "hands-on" experience with the available machines. (15%)
4. *Hybrid Computer Systems.* Different types of hybrid systems. Patchable logic and mode control. Comparators, switches, and different types of analog memories. Control of initial conditions or parameters. (15%)
5. *Analog and Digital Conversion.* Brief treatment of analog-to-digital and digital-to-analog conversion. Methods of conversion, sampling, interpolation, smoothing. Accuracy and speed considerations. Multiplexing of analog-to-digital converters. (10%)
6. *Digital Simulation of Analog and Hybrid Systems.* Comparison of available languages. (One language should be presented in detail. The students should compare some of the previous analog solutions to those obtained by simulation.) (20%)
7. *Exams.* (5%)

ANNOTATED BIBLIOGRAPHY

In the citations which follow, applicable chapters are sometimes indicated in parentheses after the annotation.

General textbooks or references for the major part of the course

1. ASHLEY, J. R., *Introduction to Analog Computation.* Wiley, New York, 1963, 294 pp.
A compact textbook which stresses the use rather than the design aspects of analog computing. The text could be used for an undergraduate course, but hybrid computers would have to be covered from separate sources. (All chapters)
2. CARLSON, A., HANNAUER, G., CAREY, T., AND HOLSBERG, P. In *Handbook of Analog Computation.* Electronics Associates, Inc., Princeton, N. J., 1965.
Although this manual is oriented to EAI equipment, it is a good basic reference and has an extensive bibliography on selected applications.
3. FIFER, S. *Analog Computation, Volumes I-IV.* McGraw-Hill, New York, 1961, 1,331 pp. CR-6126-1122.
This series of four volumes contains a complete coverage of analog computers, including hardware and applications. It is a good source of problems and references up to 1961.
4. HUSKEY, H. D., AND KORN, G. A. (Eds.) *Computer Handbook.* McGraw-Hill, New York, 1962, 1,225 pp. CR-6234-2179.
This comprehensive volume on both analog and digital computers is somewhat hardware-oriented although applications are also included.
5. JACKSON, A. S. *Analog Computation.* McGraw-Hill, New York, 1960, 652 pp. CR-6015-0190.
Although now somewhat out-of-date, this is an excellent text for serious students in engineering, especially those with an interest in feedback-control theory. The text has many references and an appendix with problem sets. (Chaps. 2-5, 7, 8, 11, 14)

6. JENNESS, R. R. *Analog Computation and Simulation: Laboratory Approach*. Allyn and Bacon, Boston, 1955, 298 pp.

Two parts: the first introduces the analog computer; the second gives the solutions of 21 problems in great detail.

7. JOHNSON, C. L. *Analog Computer Techniques*, 2nd ed. McGraw-Hill, New York, 1963, 336 pp. CR-6451-5162.

The text assumes a knowledge of basic electrical and mathematical principles. Some parts require an understanding of servo-mechanism theory and the Laplace transform. Each chapter has references and problems. (Chaps. 1-3, 7, 10, 12)

8. KARPLUS, W. J. *Analog Simulation, Solution of Field Problems*. McGraw-Hill, New York, 1958, 434 pp. CR-6123-0729.

A reference on analog techniques for partial differential equations. Includes material on the mathematical background for analog study of field problems, a description of analog hardware, and a mathematical discussion of applications of analog techniques to different classes of differential equations. Problem-oriented. Extensive bibliographies.

9. KARPLUS, W. J., AND SOROKA, W. J. *Analog Methods*, 2nd ed. McGraw-Hill, New York, 1959, 496 pp.

Three parts: on indirect computing elements; on indirect computers; and on direct computers. Describes both electro-mechanical and electronic computers and covers applications comprehensively.

10. KORN, G. A., AND KORN, T. M. *Electronic Analog and Hybrid Computers*. McGraw-Hill, New York, 1963, 584 pp. CR-6562-7468.

This text covers the theory, design, and application of analog and hybrid computers and has one of the most complete bibliographies available. Its compactness makes it more suitable for an undergraduate text.

11. LEVINE, L. *Methods for Solving Engineering Problems Using Analog Computers*. McGraw-Hill, New York, 1964, 485 pp. CR-6455-6477.

One of the best available texts, but it needs supplementing on equipment. (Ref. 2 might be good for this purpose.) In addition to the usual topics, there are chapters on optimization techniques, estimation and testing of hypotheses, and applications in statistics. (Chaps. 1-7).

12. SMITH, G. W., AND WOOD, R. C. *Principles of Analog Computation*. McGraw-Hill, New York, 1959, 234 pp. CR-6121-0411.

Introduces analog computers and illustrates various programming techniques in simulation and computation.

Descriptions of digital simulators of continuous systems

13. BRENNAN, R. D., AND SANO, H. PACTOLUS—a digital analog simulator program for the IBM 1620. Proc. AFIPS 1964 Fall Joint Comput. Conf., Vol. 26, Spartan Books, New York, pp. 299-312. CR-6563-7630.

Well-written article describes a digital program for the simulation of an analog computer. The program is written in FORTRAN and may be adapted to most machines. It allows man-machine interaction.

14. FARRIS, G. J., AND BURKHART, L. E., The DIAN digital simulation program. *Simulation* 6, 5 (May 1966), 298-304.

Describes a digital computer program which has some of the features of a digital differential analyzer and which is particularly suitable for the solution of boundary value problems.

15. HARNETT, R. T., AND SANSOM, F. J. *MIDAS Programming Guide*. Report No. SEG-TDR-64-1, Analog Comput. Divn., Syst. Engng. Group, Res. and Techn. Divn., US Air Force Systems Command, Wright-Patterson Air Force Base, Ohio, Jan. 1964. CR-6672-9510.

This is the programming manual for the MIDAS simulation language. It is well-written and contains four examples complete with problem descriptions, block diagrams, coding sheets, and computed results.

16. JANOSKI, R. M., SCHAEFER, R. L., AND SKILES, J. J. COBLOC—a program for all-digital simulation of a hybrid computer. *IEEE Trans. EC-15*, 2 (Feb. 1966), 74-91.

COBLOC is a compiler which allows all-digital simulation of a hybrid computer having both analog and digital computation capability.

17. MORRIS, S. M., AND SCHIESSER, W. E. Undergraduate use of digital simulation. *Simulation* 7, 2 (Aug. 1966), 100-105. CR-6781-11,027.

Describes the LEANS (Lehigh Analog Simulator) program and shows the solution of a sample problem.

18. RIDEOUT, V. C., AND TAVERNINI, L. MADBLOC, a program for digital simulation of a hybrid computer. *Simulation* 4, 1 (Jan. 1965), 20-24.

Gives a brief description of the hybrid simulation language MADBLOC (one of the Wisconsin "BLOC" programs) which is written in the MAD language. Parameter optimization of a simple feedback system is given as an example.

19. STEIN, M. L., ROSE, J., AND PARKER, D. B. A compiler with an analog oriented input language. *Simulation* 4, 3 (Mar. 1965), 158-171.

Gives a description of a compiler program called "ASTRAL" which accepts analog oriented statements and produces FORTRAN statements. It is a reprint of the same paper from the Proc. of 1959 Western Joint Comput. Conf.

20. SYN, W. M., AND LINEBARGER, R. M. DSL/90—a digital simulation program for continuous system modeling. Proc. AFIPS 1966 Spring Joint Comput. Conf., Vol. 28, Spartan Books, New York, pp. 165-187. CR-6676-10,708.

A program which accepts block-oriented statements and compiles them into FORTRAN IV statements. Mixing of DSL/90 and FORTRAN statements is allowed. The program is available for the IBM 7090/94 and 7040/44.

Comparisons of digital simulators for continuous systems

21. LINEBARGER, R. N., AND BRENNAN, R. D. A survey of digital simulation: digital analog simulator programs. *Simulation* 3, 6 (Dec. 1964), 22-36. CR-6671-9009.

Gives a brief account of the following digital-analog simulation languages: SELFRIDE, DEPI, ASTRAL, DEPI4, DYSAC, PARTNER, DAS, JANIS, MIDAS, and PACTOLUS

22. LINEBARGER, R. N., AND BRENNAN, R. D. Digital simulation for control system design. *Instr. Contr. Syst.* 38, 10 (Oct. 1965), 147-152. CR-6675-10,425.

This paper has a very complete bibliography of digital simulators and a table classifying them.

Course A4. System Simulation (3-0-3)

APPROACH

This course can be taught from several different points of view: simulation can be treated as a tool of applied mathematics; it can be treated as a tool for optimization in operations research; or it can be treated as an example of the application of computer science techniques. Which orientation is used and the extent to which computer programs are an integral part of the course should depend upon the interests of the instructor and the students. Most instructors will find it useful to require several small programs and a term project. The availability of a simulation language for student use is desirable.

CONTENT

The numbers in square brackets refer to the items listed in the bibliography which follows.

1. *What is simulation?* (5%) [1, 2, 3]
 - a. Statistical sampling experiment. [20]
 - b. Comparison of simulation and other techniques. [7]
 - c. Comparison of discrete, continuous, and hybrid simulation. [12, 21]
2. *Discrete Change Models.* (20%)
 - a. Queueing models. [13]
 - b. Simulation models. [1, 2, 3]

3. *Simulation languages*. (20%) [16, 21, 22]
4. *Simulation Methodology*. (20%) [1, 2, 3]
 - a. Generation of random numbers and random variates. [14]
 - b. Design of experiments and optimization. [6, 8, 9, 17]
 - c. Analysis of data generated by simulation experiments. [10, 11, 15]
 - d. Validation of models and results. [8, 19]
5. *Selected Applications of Simulation*. (10%) [4, 5]
 - a. Business games.
 - b. Operations research. [18]
 - c. Artificial intelligence.
6. *Research Problems in Simulation Methodology*. (5%)
7. *Term Project*. (20%)

BIBLIOGRAPHY

Textbooks covering a number of the topics of this course

1. CHORAFAS, D. N. *Systems and Simulation*. Academic Press, New York, 1965, 503 pp.
2. NAYLOR, T. H., BALINTFY, J. L., BURDICK, D. S., AND CHU, K. *Computer Simulation Techniques*. Wiley, New York, 1966, 352 pp. CR-6781-11,103.
3. TOCHER, K. D. *The Art of Simulation*. D. Van Nostrand, Princeton, N. J., 1963, 184 pp. CR-6454-6091.

Bibliographies devoted to simulation

4. IBM Corporation. *Bibliography on Simulation*. Report 320-0924-0, 1966.
5. SHUBIK, M. Bibliography on simulation, gaming, artificial intelligence, and allied topics. *J. Amer. Statist. Assoc.* 55, 292 (Dec. 1960), 736-751. CR-6122-0581.

Other works on simulation (which also contain extensive bibliographies)

6. BURDICK, D. S., AND NAYLOR, T. Design of computer simulation experiments for industrial systems. *Comm. ACM* 9, 5 (May 1966), 329-338. CR-6783-11,714.
7. CONNORS, M. M., AND TEICHROEW, D. *Optimal Control of Dynamic Operations Research Models*. International Textbook Co., Scranton, Pa., 1967, 118 pp.
8. CONWAY, R. W. Some tactical problems in digital simulation. *Management* 10, 1 (Oct. 1963), 47-61.
9. EHRENFELD, S., AND BEN-TUVIA, S. The efficiency of statistical simulation procedures. *Technometrics* 4, 2 (May 1962), 257-276. CR-6341-3742.
10. FISHMAN, G. S. Problems in the statistical analysis of simulation experiments: The comparison of means and the length of sample records. *Comm. ACM* 10, 2 (Feb. 1967), 94-99. CR-6783-12,103.
11. FISHMAN, G. S., AND KIVIAT, P. J. The analysis of simulation-generated time series. *Mgmt. Sci.* 13, 7 (Mar. 1967), 525-557.
12. FORRESTER, J. W. *Industrial Dynamics*. M.I.T. Press, Cambridge, Mass., and Wiley, New York, 1961, 464 pp.
13. GALLIHER, H. Simulation of random processes. In *Notes on Operations Research*, Operations Research Center, M.I.T., Cambridge, Mass., 1959, pp. 231-250.
14. HULL, T. E., AND DOBELL, A. R. Random number generators. *SIAM Rev.* 4, 3 (July 1962), 230-254. CR-6341-3749.
15. JACOBY, J. E., AND HARRISON, S. Multivariable experimentation and simulation models. *Naval Res. Log. Quart.* 9, (1962), 121-136.
16. KRASNOW, H. S., AND MERIKALLIO, R. The past, present and future of general simulation languages. *Mgmt. Sci.* 11, 2 (Nov. 1964), 236-267. CR-6566-8521.
17. MCARTHUR, D. S. Strategy in research—alternative methods for design of experiments. *IRE Trans. Eng. Man.* EM-8, 1 (Jan. 1961), 34-40.
18. MORGENTHAUER, G. W. The theory and application of simulation in operations research. In Russel L. Ackoff (Ed.), *Progress in Operations Research*, Wiley, New York, 1961, pp. 363-419.
19. SCHENK, H., JR. Computing "AD ABSURDUM." *The Nation* 196, 12 (June 15, 1963), 505-507.

20. TEICHROEW, D. A history of distribution sampling prior to the era of the computer and its relevance to simulation. *J. Amer. Statist. Assoc.* 60, 309 (Mar. 1965), 27-49. CR-6673-9823.
21. TEICHROEW, D. Computer simulation—discussion of the technique and comparison of languages. *Comm. ACM* 9, 10 (Oct. 1966), 723-741. CR-6782-11,466.
22. TOCHER, K. D. Review of simulation languages. *Oper. Res. Quart.* 15, 2 (June 1965), 189-218.

Course A5. Information Organization and Retrieval (3-0-3)

APPROACH

This course is designed to introduce the student to information organization and retrieval of natural language data. Emphasis should be given to the development of computer techniques rather than philosophical discussions of the nature of information. The applicability of the techniques developed for both data and document systems should be stressed. The student should become familiar not only with the techniques of statistical, syntactic and logical analysis of natural language for retrieval, but also with the extent of success or failure of these techniques. The manner in which the techniques may be combined into a system for use in an operational environment should be explored. In the event that a computer is available together with natural language text in a computer-readable form, programming exercises applying some of the techniques should be assigned. If this is not possible, the student should present a critique and in-depth analysis of an article selected by the instructor.

CONTENT

1. *Information Structures*. Graph theory, document and term-document graphs, semantic road maps, trees and lists, thesaurus and hierarchy construction, and multilists.
2. *Dictionary Systems*. Thesaurus look-up, hierarchy expansion, and phrase dictionaries.
3. *Statistical Systems*. Frequency counts, term and document associations, clustering procedures, and automatic classification.
4. *Syntactic Systems*. Language structure, automatic syntactic analysis, graph matching, and automatic tree matching.
5. *Vector Matching and Search Strategies*. Keyword matching, direct and inverted files, combined file systems, correlation functions, vector merging and matching, matching of cluster vectors, and user feedback systems.
6. *Input Specifications and System Organization*. Input options. Supervisory systems, their general organization, and operating procedures.
7. *Output Systems*. Citation indexing and bibliographic coupling. Secondary outputs including concordances, abstracts, and indexes. Selective dissemination. Catalog systems.
8. *Evaluation*. Evaluation environment, recall and precision, presentation of results, and output comparison.
9. *Automatic Question Answering*. Structure and extension of data bases, deductive systems, and construction of answer statements.

ANNOTATED BIBLIOGRAPHY

There is no one book currently available that could be used as a text for this course, so most of the material must be obtained from the literature. References 2 and 6 provide excellent state-of-the-art surveys and guides to the literature.

1. BECKER, J., AND HAYES, R. M. *Introduction to Information Storage and Retrieval: Tools, Elements, Theories*. Wiley, New York, 1963, 448 pp.
A standard textbook, perhaps the best of those presently available.

2. CUADRA, C. (Ed.) *Annual Review of Information Science and Technology*. Interscience, New York, Vol. 1, 1966 and Vol. 2, 1967.
A survey and review publication.
3. HAYS, D. G. (Ed.) *Readings in Automatic Language Processing*. American Elsevier, New York, 1966.
Includes examples of text processing applications.
4. SALTON, G. Progress in automatic information retrieval. *IEEE Spectrum* 2, 8 (Aug. 1965), 90-103. CR-6675-10,409.
A survey of current capabilities in text processing.
5. SALTON, G. *Automatic Information Organization and Retrieval*. McGraw-Hill, New York, to be published in 1968.
A text concentrating on automatic computer-based information retrieval systems.
6. STEVENS, M. E. *Automatic Indexing: A State-of-the-Art Report*. Monograph 91, National Bureau of Standards, US Dept. of Commerce, Washington, D.C., March 30, 1965.
A survey article that covers the historical development of automatic indexing systems through 1964.
7. STEVENS, M. E., GIULIANO, V. E., AND HEILPRIN, L. B. (Eds.) *Statistical Association Methods for Mechanized Documentation*—Symposium Proceedings. Miscellaneous Publication 269, US Dept. of Commerce, Washington, D. C., 1964.
A collection of papers concerned with statistical association techniques.

Course A6. Computer Graphics (2-2-3)

APPROACH

Since this field is basically only a few years old, it is not surprising that no underlying theories are uniformly accepted by the researchers and implementers. Rather, the pertinent information exists as a number of loosely related project descriptions in conference proceedings and professional journals. This situation is similar to that in the information retrieval field, where those conducting courses at a number of major universities report on current accomplishments and research in an effort to coordinate and structure the mass of available information and to teach those techniques which have been found useful. Thus, for the present, this course probably should be taught as a seminar where the literature is read and perhaps reported by selected students. After some texts become available and after more experience has been gained, a more formal course atmosphere could be established.

Although the literature is plentiful, this course clearly assumes substantial value to the student only when it includes an intensive laboratory (hopefully using a display console) where the various algorithms can be tested, compared, and extended. The laboratory periods are meant to be used for explaining the details of algorithms or hardware and software that are not appropriate to a more formal lecture. A variety of programming projects in pattern recognition or display programming, for example, are within the scope of a one-semester course. The time spent on these projects would be in addition to the laboratory time.

CONTENT

The thread running through the topics listed below is the unit of information—the picture. The course should deal with common ways in which the picture is handled in a variety of hitherto largely unrelated disciplines. First hardware and then software topics should be considered, since the software today is still a function of present hardware.

The order and depth of coverage of the material suggested below is quite flexible—another sensible order of the material might be topics 1, 2, 6, 7, 8, and 3, 4, 5 optional, which would then constitute a course in displays. (In any case some material in Sections 6 and 7

would have to be covered briefly to prepare students for their projects.) Also an entire semester could be devoted to pattern recognition.

1. Motivation for graphical data processing and its history, particularly that of displays. (5%)

2. Brief introduction to psycho-physical photometry and display parameters such as resolution and brightness. Block diagram of display systems and delineation of the functions of their components: the computer subsystem; buffer or shared memory; command decoder; display generator; and producer(s) of points, lines, vectors, conic sections, and characters. Extended capabilities such as sub-routining, windowing, hardware matrix operations, and buffer manipulation. Comparison of various types of CRTs with other display producing techniques such as photochromics and electroluminescence. Brief discussion of passive graphics (output only) devices such as x-y plotters, and microfilm recorders. Interrupts, manual inputs and human interaction with active displays via light pens, voltage pens, function keys, tablets, wands, joy sticks, etc. Demonstration of equipment. (10%)

3. Contrast of information retrieval with document retrieval and definition of indexing and locating. Image recording parameters such as resolution, and their comparison with display parameters. Demonstration or discussion of microfilm and microfilm handling devices (manual and automated). Brief discussion of photochromics, thermoplastics and other nonconventional media. Brief discussion of electro-optical techniques for recording, modulating, and deflecting. (5%)

4. Scanning and digitizing of paper or film, and subsequent transmission of digitized information, including band-width /cost trade-offs. Brief review of digital storage techniques and parameters, and discussion of tradeoffs in bulk digital versus image storage for pictorial and digital data. Introduction of the notion of a combination of a digital and an image system. (5%)

5. Digitizing as an input process for pattern description and recognition and preprocessing of this input (cosmetology and normalization). Contrast of symbol manipulative, linguistic, and mathematical techniques, such as gestalt, caricatures, features, moments, random nets, decision functions, syntax-directed techniques and real-time tracking techniques using scopes and tablets. Electro-optical techniques such as optical Fourier transforms may also be covered briefly. (20%)

6. Picture models and data structures. Geometry, topology, syntax, and semantics of pictures, stressing picture /subpicture hierarchy. The differences between block diagram, wire frame, and surface representations. Use of tables, trees, lists, plexes, rings, associative memory, and hashing schemes for data structures, and the data structure (or list processing) languages which create and manipulate them. Mathematics of constraint satisfaction, windowing, three-dimensional transformations and projections, and hidden-line problems. (25%)

7. Display software (probably specific to a given installation). Creation and maintenance of the display file, translations between the data structure and the display file, interrupt handling, pen pointing and tracking, and correlation between light pen detects and the data structure. Use of macros or compiler level software for standard functions. Software for multiconsole time-shared graphics with real-time interaction. (20%)

8. Selected applications: (10%)

- a. Menu programming and debugging
- b. Flowchart and block diagram processing
- c. Computer assisted instruction
- d. Computer aided design
- e. Chemical modeling
- f. Business
- g. Animated movies

ANNOTATED BIBLIOGRAPHY

This bibliography is not complete in its coverage and consists

primarily of survey articles, as no textbooks exist. A number of the more detailed technical articles in the literature were omitted because they were concerned with specific situations or machines. Most of the survey articles listed have good bibliographies. A selection of topics from the outline above can thus be followed by a selection of appropriate research papers from the literature.

1. FETTER, W. A. *Computer Graphics in Communications*. McGraw-Hill, New York, 1965, 110 pp.
This volume is strong on engineering applications and illustrations.
2. GRAY, J. C. Compound data structures for computer-aided design: a survey. Proc. ACM 22nd Nat. Conf., 1967, Thompson Book Co., Washington, D. C., pp. 355-366.
A brief survey and comparison of various types of data structures currently in use.
3. GRUENBERGER, F. (Ed.) *Computer Graphics: Utility/Production/Art*. Thompson Book Co., Washington, D. C., 1967, 225 pp.
Collection of survey papers, useful for orientation.
4. NARASIMHAN, R. Syntax-directed interpretation of classes of pictures. *Comm. ACM* 9, 3 (Mar. 1966), 166-173.
An introduction to syntactic descriptive models for pictures, implemented using simulated parallel processing. This linguistic approach is also taken by Kirsch, Grenander, Miller, and others.
5. PARKER, D. B. Solving design problems in graphical dialogue. In W. J. Karplus (Ed.), *On-Line Computing*, McGraw-Hill, 1967, pp. 176-219.
A software-oriented survey of display console features.
6. ROSS, D. G., AND RODRIGUEZ, J. E. Theoretical foundations for the computer-aided design system. Proc. AFIPS 1963 Spring Joint Comput. Conf., Vol. 23, Spartan Books, New York, pp. 305-322.
Introduction of the "plex" as a compound list structure for both graphical and nongraphical entities. Outline of the AED philosophy and algorithmic theory of language.
7. SUTHERLAND, I. E. Sketchpad: a man-machine graphical communication system. Lincoln Lab. Tech. Rep. No. 296, M.I.T., Lexington, Mass., 1963, 91 pp.
Presents the pace-setting Sketchpad system: its capabilities, data structure, and some implementation details.
8. SUTHERLAND, W. R. The on-line graphical specification of computer procedures. Ph.D. Dissertation, M.I.T., Cambridge, Mass., Jan. 1966, and Lincoln Lab. Tech. Rep. No. 405, May 1966.
Describes a graphical language, executed interpretively, which avoids written labels and symbols by using data connections between procedure elements to determine both program flow and data flow.
9. VAN DAM, A. Bibliography on computer graphics. *ACM SIG-GRAPHICS Newsletter* 1, 1 (Apr. 1967), Association for Computing Machinery, New York.
This extensive bibliography is being kept up-to-date in successive issues of the *Newsletter*.
10. VAN DAM, A. Computer-driven displays and their uses in man/machine interaction. In F. L. Alt (Ed.), *Advances in Computers*, Vol. 7, Academic Press, New York, 1966, pp. 239-290.
A hardware-oriented description of CRT console functions.

Course A7. Theory of Computability (3-0-3)

APPROACH

This is a theoretical course which should be taught in a formal and precise manner, i.e. definitions, theorems, and proofs. The theory of recursive functions and computability should, however, be carefully motivated and illustrated with appropriate examples.

CONTENT

More material is listed than can easily be covered in a three-hour one-semester course. The first three topics should definitely be covered, but the instructor can select material from the remaining topics.

1. Introduction to Turing machines (TM's) and the invariance of general computability under alterations of the model. Wang machines, Shepherdson-Sturgis machines, machines with only 2 symbols, machines with only 2 states, machines with nonerasing tapes, machines with multiple heads and multidimensional tapes. (4 lectures)
2. Universal Turing machines. (2 lectures)
3. Gödel numbering and unsolvability results, the halting problem, and Post's correspondence problem. (3 lectures)
4. Relative uncomputability, many-one reducibility and Turing reducibility, and the Friedberg-Muchnik theorem. (6 lectures)
5. TM's with restricted memory access, machines with one counter, pushdown automata and their relation to context-free languages. Universality of machines with two counters. (3 lectures)
6. TM's with limited memory, linear bounded automata and their relation to context-sensitive languages, and the Stearns-Hartmanis-Lewis hierarchy. (5 lectures)
7. TM's with limited computing time and the Hartmanis-Stearns time hierarchy. (5 lectures)
8. Models for real-time computation, TM's with many tapes versus 1 or 2 tapes, and TM's with many heads per tape versus 1 head per tape. (4 lectures)
9. Random-access stored-program machines, iterative arrays, general bounded activity machines, n-counter real-time machines, and other computing devices. (8 lectures)
10. Complexity classification by functional definition, primitive recursive functions, the Grzegorzczuk hierarchy and its relation to ALGOL programming, real-time countability, and an algorithm for fast multiplication. (6 lectures)

ANNOTATED BIBLIOGRAPHY

1. AANDERAA, S., AND FISCHER, P. C. The solvability of the halting problem for 2-state Post machines. *J. ACM* 14, 4 (Oct. 1967), 677-682.
A problem unsolvable for quintuple Turing machines is shown to be solvable for the popular quadruple version of Post.
2. CAIANIELLO, E. R. (Ed.) *Automata Theory*. Academic Press, New York, 1966, 342 pp. CR-6676-10,935.
A collection of research and tutorial papers on automata, formal languages, graph theory, logic, algorithms, recursive function theory, and neural nets. Because of varying interest and difficulty, the papers might be useful for supplementary reading by ambitious students.
3. DAVIS, M. *Computability and Unsolvability*. McGraw-Hill, New York, 1958, 210 pp.
Contains an introduction to the theory of recursive functions, most of Kleene's and Post's contributions to the field and some more recent work.
4. DAVIS, M. (Ed.) *The Undecidable—Basic Papers on Undecidable Propositions, Unsolvability Problems and Computable Functions*. Raven Press, Hewlett, New York, 1965, 440 pp. CR-6673-9790.
An anthology of the fundamental papers of Church, Gödel, Kleene, Post, Rosser, and Turing on undecidability and unsolvability.
5. FISCHER, P. C. Multitape and infinite-state automata—a survey. *Comm. ACM* 8, 12 (Dec. 1965), 799-805. CR-6675-10,561.
A survey of machines which are more powerful than finite automata and less powerful than Turing machines. Extensive bibliography.
6. FISCHER, P. C. On formalisms for Turing machines. *J. ACM* 12, 4 (Oct. 1965), 570-580. CR-6675-10,558.

- Variants of one-tape Turing machines are compared and transformations from one formalism to another are analyzed.
7. FRIEDBERG, R. M. Two recursively enumerable sets of incomparable degrees of unsolvability (Solution of Post's Problem, 1944). *Proc. Nat. Acad. Sci.* 43, (1957), 236-238.
The "priority" method for generating recursively enumerable sets is introduced and used to solve this famous problem.
 8. GINSBURG, S. *Mathematical Theory of Context-Free Languages*. McGraw-Hill, New York, 1966, 243 pp.
The first textbook on the theory of context-free languages. It gives a detailed mathematical treatment of pushdown automata, ambiguity, and solvability.
 9. HARTMANIS, J., AND STEARNS, R. E. On the computational complexity of algorithms. *Trans. AMS* 117, 5 (May 1965), 285-306.
Turing computable sequences are classified in terms of the rate with which a multitape Turing machine can output the terms of the sequence, i.e. the "Hartmanis-Stearns time hierarchy."
 10. HERMES, H. *Enumerability, Decidability, Computability*. Academic Press, New York, 1965, 245 pp. CR-6673-9781.
A systematic introduction to the theory of recursive functions, using Turing machines as a base.
 11. KLEENE, S. C. *Mathematical Logic*. Wiley, New York, 1967, 398 pp.
A thorough yet elementary treatment of first-order mathematical logic for undergraduates. Contains much of the material of the author's graduate text, *Introduction to Metamathematics*, (D. Van Nostrand, Princeton, N. J., 1952, 550 pp.). The material has been updated and reorganized to be more suitable for the beginning student.
 12. McNAUGHTON, R. The theory of automata, a survey. In F. L. Alt (Ed.), *Advances in Computers*, Vol. 2. Academic Press, New York, 1961, pp. 379-421. CR-6342-3920.
Most of the areas of automata theory are included with the exception of switching theory and other engineering topics. A list of 119 references.
 13. MINSKY, M. L. *Computation: Finite and Infinite Machines*. Prentice-Hall, Englewood Cliffs, N. J., 1967, 317 pp.
The concept of an "effective procedure" is developed. Also treats algorithms, Post productions, regular expressions, computability, infinite and finite-state models of digital computers, and computer languages.
 14. MYHILL, J. Linear bounded automata. *WADD Tech. Note 60-165*, Wright-Patterson Air Force Base, Ohio, 1960.
The paper which first defined a new class of automata whose power lies between those of finite automata and Turing machines.
 15. POST, E. L. Recursive unsolvability of a problem of Thue. *J. Symbol. Logic* 11, (1947), 1-11.
Contains results on one variant of the "word problem" for semi-groups using Turing machine methods.
 16. ROGERS, H., JR. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, New York, 1967, 482 pp.
A current and comprehensive account of recursive function theory. Proceeds in an intuitive semiformal manner, beginning with the recursively enumerable sets and ending with the analytical hierarchy.
 17. SHANNON, C. E., AND MCCARTHY, J. (Eds.) *Automata Studies*. Princeton University Press, Princeton, N. J., 1956, 285 pp. CR-6565-8330.
A collection of many of the early papers on finite automata, Turing machines, and synthesis of automata which stimulated the development of automata theory. Philosophical papers, in addition to mathematical papers, are included since the aim of the collection is to help explain the workings of the human mind.
 18. SHEPHERDSON, J. C., AND STURGIS, H. E. Computability of recursive functions. *J. ACM* 10, 2 (Apr. 1963), 217-255. CR-6451-5105.
A class of machines which is adequate to compute all partial recursive functions is obtained by relaxing the definition of a Turing machine. Such machines can be easily designed to carry out some specific intuitively effective procedure.
 19. STEARNS, R. E., HARTMANIS, J., AND LEWIS, P. M. Hierarchies of memory limited computations. *1965 IEEE Conference Record on Switching Circuit Theory and Logic Design*, Special Publication 16 C 13, Institute of Electrical and Electronic Engineers, New York, Oct. 1965, pp. 179-190.
Turing computable functions are classified according to the relationship of the amount of storage required for a computation to the length of the input to the computation, i.e. the "Stearns-Hartmanis-Lewis hierarchy."
 20. TRAKHTENBROT, B. A. *Algorithms and Automatic Computing Machines*, transl. by J. Kristian, J. D. McCawley, and S. A. Schmitt. D. C. Heath, Boston, 1963, 101 pp.
A translation and adaptation from the second Russian edition (1960) of the author's elementary booklet on solvability and Turing machines.
 21. TURING, A. M. On computable numbers, with an application to the Entscheidungsproblem. *Proc. London Math. Soc.*, Ser. 2, 42, (1936-1937), pp. 230-265.
The famous memoir on decision problems which initiated the theory of automata.
 22. VON NEUMANN, J. *Theory of Self-Reproducing Automata*. (Edited and completed by A. W. Burks.) University of Illinois Press, Urbana, Illinois, 1966, 388 pp. CR-6700-0670.
Consists of all previously unpublished sections of Von Neumann's general theory of automata. Part I includes the kinematic model of self-reproduction. Part II, which is much longer, treats the logical design of a self-reproducing cellular automaton.
 23. WANG, H. A variant to Turing's theory of computing machines. *J. ACM* 4, 1 (Jan. 1957), 61-92.
An abstract machine is defined which is capable of carrying out any computation and which uses only four basic types of instructions in its programs.

Course A8. Large-scale Information Processing Systems (3-0-3)

APPROACH

This course is intended to give the student some appreciation of how computers fit into information systems and how information systems fit into a "large organization framework." As this field is evolving rapidly, the most interesting and relevant material appears in articles; moreover, the field is so large that not all the relevant material can be covered. The course may be conducted as a lecture course, but assignment of individual readings in a seminar-type situation might be more suitable.

Many information processing systems are so large that they require a number of computer programs to be run on a continuing basis using large quantities of stored data. The process of establishing such a large system involves a number of steps: (1) the determination of the processing requirements; (2) the statement of those requirements in a complete and unambiguous form suitable for the next steps; (3) the design of the system, i.e. the specification of computer programs, hardware devices, and procedures which together can "best" accomplish the required processing; (4) the construction of the programs and procedures, and the acquisition of the hardware devices; and (5) the testing and operation of the assembled components in an integrated system. This course is designed to help prepare the student to participate in the development of such systems.

CONTENT

The numbers in square brackets after each topic listed below refer to the items listed in the bibliography which follows.

1. Examples of large-scale information systems. (10%) [12, 28, 30]

- a. Computer centers. [20, 27, 34]
- b. Information retrieval. [2]
- c. Real-time and time-sharing. [15, 16, 33]
- d. Business data processing. [14, 18, 19, 31, 32]
2. Data structures and file management systems. (20%) [1, 4, 5, 9, 10, 13, 17, 29, 38]
3. Systems design methodology. (40%) [22, 32]
 - a. "Nonprocedural" languages. [8, 26, 38, 40]
 - b. Systems design. [3, 11, 21, 23, 24, 25, 36, 37]
 - c. Evaluation. [7]
4. Implementation problems. (10%) [6, 35]
5. Term project. (20%)

BIBLIOGRAPHY

1. BAUM, C., AND GORSUCH, L. (Eds.) Proceedings of the second symposium on computer-centered data base systems. TM-2624/100/00, System Development Corporation, Santa Monica, Calif., 1 Dec. 1965.
2. BERUL, L. Information storage and retrieval, a state-of-the-art report. Report AD-630-089, Auerbach Corporation, Philadelphia, Pa., 14 Sept. 1964.
3. BRIGGS, R. A mathematical model for the design of information management systems. M.S. Thesis, U. of Pittsburgh, Pittsburgh, Pa., 1966.
4. BROOKS, F. P., JR., AND IVERSON, K. E. *Automatic Data Processing*. Wiley, New York, 1963, 494 pp.
5. BRYANT, J. H., AND SEMPLE, P., JR. GIS and file management. Proc. ACM 21st Nat. Conf., 1966, Thompson Book Co., Washington, D. C., pp. 97-107.
6. BUCHHOLZ, W. (Ed.) *Planning a Computer System*. McGraw-Hill, New York, 1962, 322 pp. CR-6346-4786.
7. CALINGAERT, P. System evaluation: survey and appraisal. *Comm. ACM* 10, 1 (Jan. 1967), 12-18. CR-6782-11,661.
8. Codasyl Development Committee, Language Structure Group. An information algebra, phase I report. *Comm. ACM* 5, 4 (Apr. 1962), 190-201. CR-6235-2621.
9. CONNORS, T. L. ADAM—generalized data management system. Proc. AFIPS 1966 Spring Joint Comput. Conf., Vol. 28, Spartan Books, New York, pp. 193-203. CR-6676-10,822.
10. Control Data Corporation. *3600/3800 INFOL Reference Manual*. Publ. No. 60170300, CDC, Palo Alto, Calif., July, 1966.
11. DAY, R. H. On optimal extracting from a multiple file data storage system: an application of integer programming. *J. ORSA* 13, 3 (May-June, 1965), 482-494.
12. DESMONDE, W. H. *Computers and Their Uses*. Prentice-Hall, Englewood Cliffs, N. J., 1964, 296 pp. CR-6561-6829.
13. DOBBS, G. H. State-of-the-art survey of data base systems. Proc. Second Symposium on Computer-Centered Data Base Systems, TM-2624/100/00, System Development Corporation, Santa Monica, Calif., 1 Dec. 1965, pp. 2-3 to 2-10.
14. ELLIOTT, C. O., AND WASLEY, R. S. *Business Information Processing Systems*. Richard D. Irwin, Homewood, Ill., 1965, 554 pp.
15. FIFE, D. W. An optimization model for time-sharing. Proc. AFIPS 1966 Spring Joint Comput. Conf., Vol. 28, Spartan Books, New York, pp. 97-104. CR-6676-10,869.
16. FRANKS, E. W. A data management system for time-shared file processing using a cross-index file and self-defining entries. Proc. AFIPS 1966 Spring Joint Comput. Conf., Vol. 28, Spartan Books, New York, pp. 79-86. CR-6676-10,754.
17. General Electric Company. *Integrated Data Store—A New Concept in Data Management*. Application Manual AS-CPB-483A, Revision of 7-67, GE Computer Division, Phoenix, Ariz., 1967.
18. GOTLIEB, C. C. General purpose programming for business applications. In F. L. Alt (Ed.), *Advances in Computers*, Vol. 1, Academic Press, New York, 1960, pp. 1-42. CR-6016-0206.
19. GREGORY, R. H., AND VAN HORN, R. L. *Automatic Data Processing Systems*, 2nd ed. Wadsworth Pub. Co., San Francisco, 1963, 816 pp. CR-6016-0301, of 1st ed.
20. HUTCHINSON, G. K. A computer center simulation project. *Comm. ACM* 8, 9 (Sept. 1965), 559-568. CR-6673-9617.
21. KATZ, J. H. Simulation of a multiprocessor computer system. Proc. AFIPS 1966 Spring Joint Comput. Conf., Vol. 28, Spartan Books, New York, pp. 127-139. CR-6676-10,870.
22. LADEN, H. N., AND GILDERSLEEVE, T. R. *System Design for Computer Application*. Wiley, New York, 1963, 330 pp.
23. LANGEFORS, B. Some approaches to the theory of information systems. *BIT* 3, 4 (1963), 229-254. CR-6455-6399.
24. LANGEFORS, B. Information system design computations using generalized matrix algebra. *BIT* 5, 2 (1965), 96-121.
25. LOMBARDI, L. Theory of files. Proc. 1960 Eastern Joint Comput. Conf., Vol. 18, Spartan Books, New York, pp. 137-141. CR-6236-3165.
26. LOMBARDI, L. A general business-oriented language based on decision expressions. *Comm. ACM* 7, 2 (Feb. 1964), 104-111. CR-6671-9013.
27. LYNCH, W. C. Description of a high capacity, fast turnaround university computer center. *Comm. ACM* 9, 2 (Feb. 1966), 117-123. CR-6673-9546.
28. MALEY, G. A., AND SKIKO, E. J. *Modern Digital Computers*. Prentice-Hall, Englewood Cliffs, N. J., 1964, 216 pp. CR-6561-7081.
29. McCABE, J. On serial files with relocatable records. *J. ORSA* 13, 4 (July-Aug. 1965), 609-618.
30. McCARTHY, E. J., McCARTHY, J., AND HUMES, D. *Integrated Data Processing Systems*. Wiley, New York, 1966, 565 pp.
31. McCracken, D. D., Weiss, H., AND LEE, T.-H. *Programming Business Computers*. Wiley, New York, 1959, 510 pp. CR-6013-0076.
32. McGEE, W. C. The formulation of data processing problems for computers. In F. L. Alt (Ed.) *Advances in Computers*, Vol. 4, Academic Press, New York, 1964, pp. 1-52.
33. NIELSON, N. R. The simulation of time sharing systems. *Comm. ACM* 10, 7 (July 1967), 397-412.
34. ROSIN, R. F. Determining a computer center environment. *Comm. ACM* 8, 7 (July 1965), 463-488.
35. SCHULTZ, G. P., AND WHISTER, T. L. (Eds.) *Management Organization and the Computer*. Free Press, Macmillan, New York, 1960, 310 pp.
36. SMITH, J. L. An analysis of time-sharing computer systems using Markov models. Proc. AFIPS 1966 Spring Joint Comput. Conf., Vol. 28, Spartan Books, New York, pp. 87-95. CR-6676-10,835.
37. TURNBURKE, V. P., JR. Sequential data processing design. *IBM Syst. J.* 2, (Mar. 1963), 37-48.
38. VER HOEF, E. W. Design of a multilevel file management system. Proc. ACM 21st Nat. Conf., 1966, Thompson Book Co., Washington, D. C., pp. 75-86. CR-6781-11,185.
39. YOUNG, J. W., JR. Nonprocedural languages—a tutorial. Paper 7th Ann. Tech. Symposium, Mar. 23, 1965. South Calif. Chapters of ACM. Copies may be obtained from the author, Electronics Division, MS 50, National Cash Register, 2815 W. El Segundo Blvd., Hawthorne, Calif. 90750.
40. YOUNG, J. W., JR., AND KENT, H. Abstract formulation of data processing problems. *J. Ind. Eng.* 9, 6 (Nov.-Dec. 1958), 471-479. (Also reprinted in *Ideas for Management*, 1959.)

Course A9. Artificial Intelligence and Heuristic Programming (3-0-3)

APPROACH

As this course is essentially descriptive, it might well be taught by surveying various cases of accomplishment in the areas under study. Each student should undertake some independent activity as part of his course work. This might take the form of a survey article on some aspect of the field: a program which simulates some of the rudimentary features of learning and forgetting; a program which plays some simple game like three-dimensional tic-tac-toe; or some other comparable activity. It would probably be best for the student to write any such programs in a list processing language.

The following outline is only a guide. Depending on the instructor's preferences and experience, variations will be introduced and new material will be added to the subject matter to be presented.

1. Definition of heuristic versus algorithmic methods using an example such as game playing. Description of cognitive processes taking place in deriving a new mathematical theorem. Outline of Polya's and Hadamard's approaches to mathematical invention. Discussion of the heuristic method as an exploratory and as an exclusive philosophy (cf. theorem proving à la Newell-Shaw-Simon, Robinson and Wang). Objectives, goals and purposes of work in areas under discussion. (3 lectures)
2. Game playing programs (chess, checkers, go, go-moku, bridge, poker, etc.). (3 lectures)
3. Theorem proving in logic and geometry. (3 lectures)
4. Formula manipulation on computers. (3 lectures)
5. Pattern recognition and picture processing. (3 lectures)
6. General problem solvers and advice takers. (4 lectures)
7. Question answering programs. (3 lectures)
8. Verbal and concept learning simulators. (3 lectures)
9. Decision making programs. (3 lectures)
10. Music composition by computers. (3 lectures)
11. Learning in random and structured nets. Neural networks. (3 lectures)
12. Adaptive systems. (3 lectures)
13. State-of-the-art in machine translation of languages and natural language processing. (4 lectures)
14. Questions of philosophical import: the mind-brain problem and the nature of intelligence, the relevance of operational definitions, and what is missing in present day "thinking machines." (2 lectures)

BIBLIOGRAPHY

The entries given below are grouped according to the items of the "Content" above to which they apply. This list serves only as a starting point and can be extended easily using the bibliographies listed below.

General reference

1. FEIGENBAUM, E. A., AND FELDMAN, J. (Eds.) *Computers and Thought*. McGraw-Hill, New York, 1966, 535 pp. CR-6563-7473.
Contains many of the articles listed below and a "Selected Descriptor-Indexed Bibliography" by Marvin Minsky.

Heuristic versus algorithmic methods [Item 1]

2. ARMER, P. Attitudes toward intelligent machines. In *Computers and Thought*, pp. 389-405. CR-6125-0977 and CR-6236-2900.
3. FINDLER, N. V. Some further thoughts on the controversy of thinking machines. *Cybernetica* 6, (1963), 47-52.
4. HADAMARD, J. *The psychology of invention in the mathematical field*. Dover Publications, New York, 1945, 145 pp. CR-6345-4614.
5. MINSKY, M. Steps toward artificial intelligence. In *Computers and Thought*, pp. 406-450. CR-6232-1528.
6. NAGEL, E. *The Structure of Science: Problems in the Logic of Scientific Explanation*. Harcourt, Brace & World, New York, 1961, 612 pp.
7. POLYA, G. *Mathematics and Plausible Reasoning: Vol. I, Induction and Analogy in Mathematics; Vol. II, Patterns of Plausible Inference*. Princeton University Press, Princeton, N. J., 1954, 280 and 190 pp.

Game playing programs [Item 2]

8. BERLEKAMP, E. R. Program for double-dummy bridge problems—a new strategy for mechanical game playing. *J. ACM* 10, 3 (July 1963), 357-364. CR-6452-5297.
9. FINDLER, N. V. Computer models in the learning process. In *Proc. Internat. Symposium on Mathematical and Computational Methods in the Social and Life Sciences, Rome, 1966*.

10. NEWELL, A., SHAW, J. C., AND SIMON, H. A. Chess playing programs and the problem of complexity. In *Computers and Thought*, pp. 39-70. CR-6012-0048.
 11. PERVIN, I. A. On algorithms and programming for playing at dominoes, transl. from Russian. *Automation Express* 1 (1959), 26-28. CR-6235-2328.
 12. REMUS, H. Simulation of a learning machine for playing Go. Proc. IFIP Congress, Munich, 1962, North-Holland Pub. Co., Amsterdam, pp. 192-194. CR-6341-3420.
 13. SAMUEL, A. L. Some studies in machine learning using the game of checkers. In *Computers and Thought*, pp. 71-105.
- Theorem proving in logic and geometry [Item 3]*
14. DAVIS, M., LOGEMANN, G., AND LOVELAND, D. A machine program for theorem-proving. *Comm. ACM* 5, 7 (July 1962), 394-397.
 15. GELERTNER, H., HANSEN, J. R., AND LOVELAND, D. W. Empirical exploration of the geometry-theorem proving machine. In *Computers and Thought*, pp. 134-152. CR-6233-1928.
 16. NEWELL, A., SHAW, J. C., AND SIMON, H. A. Empirical explorations with the logic theory machine: a case study in heuristics. In *Computers and Thought*, pp. 109-133.
 17. ROBINSON, J. A. Theorem proving on the computer, *J. ACM* 10, 2 (Apr. 1963), 163-174. CR-6452-5460.
 18. WANG, H. Proving theorems by pattern recognition. *Comm. ACM* 3, 4 (Apr. 1960), 220-234. CR-6016-0369.

Formula manipulation on computers [Item 4]

19. BOND, E., AUSLANDER, M., GRISOFF, S., KENNEY, R., MYSZEWSKI, M., SAMMET, J. E., TOBEY, R. G., AND ZILLES, S. FORMAC—an experimental FORMula MANipulation Compiler. Proc. ACM 19th Nat. Conf., 1964, Association for Computing Machinery, New York, pp. K2.1-1 to K2.1-19.
20. BROWN, W. S. The ALPAK system for nonnumerical algebra on a digital computer, I and II. *Bell Syst. Tech. J.* 42 (1963), 2081-2119, and 43 (1964), 785-804.
21. PERLIS, A. J., AND ITURRIAGA, R. An extension to ALGOL for manipulating formulae. *Comm. ACM* 7, 2 (Feb. 1964), 127-130.
22. SAMMET, J. E. An annotated descriptor based bibliography on the use of computers for nonnumerical mathematics. *Com. Rev.* 7, 4 (Jul.-Aug. 1966), B-1 to B-31.
23. SLAGLE, J. R. A heuristic program that solves symbolic integration problems in freshman calculus. In *Computers and Thought*, pp. 191-203. CR-6236-3068.

Pattern recognition and picture processing [Item 5]

24. MCCORMICK, B. H., RAY, S. R., SMITH, K. C., AND YAMADA, S. ILLIAC III: A processor of visual information. Proc. IFIP Congress, New York, 1965, Vol. 2, Spartan Books, New York, pp. 359-361.
25. TIPPETT, J. T., BERKOWITZ, D. A., CLAPP, L. C., KOESTER, C. J., AND VANDERBURGH, A., JR. (Eds.) *Optical and Electro-Optical Information Processing*, Proc. Symp. Optical and Electro-Optical Inf. Proc. Tech., Boston, Nov. 1964. M.I.T. Press, Cambridge, Mass., 1965, 780 pp. CR-6673-9829.
26. UHR, L. (Ed.) *Pattern Recognition*. Wiley, New York, 1966, 393 pp. CR-6674-10,028.

General problem solver and advice taker [Item 6]

27. MCCARTHY, J. Programs with common sense. In D. V. Blake and A. M. Uttley (Eds.), *Proc. Symp. on Mechanisation of Thought Processes*, Two volumes, National Physical Laboratory, Teddington, England. H.M. Stationery Office, London, 1959, pp. 75-84.
28. NEWELL, A., SHAW, J. C., AND SIMON, H. A. A variety of intelligent learning in a general problem solver. In M. Yovits and S. Cameron (Eds.), *Self-Organizing Systems*, Pergamon Press, New York, 1960, pp. 153-159. CR-6236-2908.
29. NEWELL, A., AND SIMON, H. A. Computer simulation of human thinking. *Science* 134, 3495 (22 Dec. 1960), 2011-2017. CR-6234-2062.

Question answering programs [Item 7]

30. BOBROW, D. G. A question answering system for high school algebra word problems. Proc. AFIPS 1964 Fall Joint Comput. Conf., Vol. 26, Spartan Books, New York, pp. 591-614. CR-6562-7183.
31. GREEN, B. F., WOLF, A. K., CHOMSKY, C., AND LAUGHERY, K. Baseball: an automatic question answerer. In *Computers and Thought*, pp. 207-216. CR-6341-3417.
32. LINDSAY, R. K. Inferential memory as the basis of machines which understand natural language. In *Computers and Thought*, pp. 217-233.
33. RAPHAEL, B. A computer program which "understands." Proc. AFIPS 1964 Fall Joint Comput. Conf., Vol. 26, Spartan Books, New York, pp. 577-589. CR-6562-7207.
34. SIMMONS, R. F. Answering English questions by computer—a survey. *Comm. ACM* 8, 1 (Jan. 1965), 53-70. CR-6563-7643.

Verbal and concept learning [Item 8]

35. FEIGENBAUM, E. A. The simulation of verbal learning behavior. In *Computers and Thought*, pp. 297-309. CR-6234-2060.
36. FEIGENBAUM, E. A., AND SIMON, H. A. Forgetting in an associative memory. Preprints of papers presented at the 16th Nat. Meeting of the ACM, Los Angeles, Sept. 5-8, 1961, Association for Computing Machinery, New York. CR-6232-1667.
37. HUNT, E. B. *Concept Learning: An Information Processing Problem*. Wiley, New York, 1962, 286 pp. CR-6561-6872.
38. MILLER, G. A., GALANTER, E., AND PRIBRAM, K. *Plans and the Structure of Behavior*. Holt, Rinehart and Winston, New York, 1960.

Decision making programs [Item 9]

39. CLARKSON, G. P. E. A model of the trust investment process. In *Computers and Thought*, pp. 347-371. CR-6563-7473.
40. FELDMAN, J. Simulation of behavior in the binary choice experiment. In *Computers and Thought*, pp. 329-346. CR-6342-3760.
41. FINDLER, N. V. Human decision making under uncertainty and risk: computer-based experiments and a heuristic simulation program. Proc. AFIPS 1965 Fall Joint Comput. Conf., Pt. I. Spartan Books, New York, pp. 737-752. CR-6673-9594.

Music composition [Item 10]

42. Computers in Music. Session 7, Tues. Nov. 8, at the AFIPS 1966 Fall Joint Computer Conf., San Francisco. (The papers for this session were not published in the conference proceedings.)
43. GILL, S. A technique for the composition of music in a computer. *Comput. J.* 6, 2 (July 1963), 129-133. CR-6451-4983.
44. HILLER, L. A., JR., AND ISAACSON, L. M. *Experimental Music*. McGraw-Hill, New York, 1959, 197 pp. CR-6012-0047.
45. MATHEWS, M. V. The digital computer as a musical instrument. *Science* 142, 3592 (1 Nov. 1963), 553-557.
46. REITMAN, W. R. *Cognition and Thought: An Information Processing Approach*. (Chap. 6). Wiley, New York, 1965, 312 pp.
47. SEAY, A. The composer of music and the computer. *Comput. Autom.* 13, 8 (Aug. 1964), 16-18. CR-6563-7548.

Learning nets and neural networks [Item 11]

48. ARBIB, M. *Brains, Machines and Mathematics*. McGraw-Hill, New York, 1964, 163 pp. CR-6455-6254.
49. BLOCK, H. D. Adaptive neural networks as brain models. *Experimental Arithmetic, High Speed Computing and Mathematics*, Proc. of Symposia in Appl. Math. 15, American Mathematical Society, Providence, R. I., 1963, pp. 59-72. CR-6453-5608.
50. LETTVIN, J. Y., MATURANA, H., MCCULLOCH, W. S., AND PITTS, W. What the frog's eye tells the frog's brain. *Proc. IRE* 47, (1959), 1940-1951.
51. ROSENBLATT, F. *Principles of Neurodynamics*. Cornell Aeronaut. Lab. Rep. 1196-G-8, Spartan Books, New York, 1962.
52. YOUNG, J. Z. *A Model of the Brain*. Clarendon Press, Oxford, England, 1964, 384 pp.

Adaptive systems [Item 12]

53. FOGEL, L. J., OWENS, A. J., AND WALSH, M. J. *Artificial Intelligence Through Simulated Evolution*. Wiley, New York, 1966, 170 pp.
54. NILSSON, N. J. *Learning Machines*. McGraw-Hill, New York, 1965, 137 pp. CR-6565-8177.
55. TOU, J. T., AND WILCOX, R. H. (Eds.) *Computer and Information Sciences*. Proc. of Symposium at Northwestern University, 1963, Spartan Books, New York, 1964, 544 pp.
56. VON FOERSTER, H., AND ZOPF, G. W., JR., (Eds.) *Principles of Self-Organization*. Pergamon Press, New York, 1962.
57. YOVITS, M. C., JACOBI, G. T., AND GOLDSTEIN, G. D. (Eds.) *Self-Organizing Systems*, 1962. Spartan Books, New York, 1962, 563 pp. CR-6456-6603.

Natural language processing [Item 13]

58. BAR-HILLEL, Y. *Language and Information: Selected Essays on Their Theory and Application*. Addison-Wesley, Reading, Mass., 1964, 388 pp. CR-6562-7178.
59. BOBROW, D. G. Syntactic analysis of English by computer—a survey. Proc. AFIPS 1963 Fall Joint Comput. Conf., Vol. 24, Spartan Books, New York, pp. 365-387. CR-6671-8838.
60. CHOMSKY, N. *Aspects of the Theory of Syntax*. M.I.T. Press, Cambridge, Mass., 1965, 251 pp. CR-6676-10,735.
61. GARVIN, P. L. (Ed.) *Natural Language and the Computer*. McGraw-Hill, New York, 1963, 398 pp. CR-6456-6569.
62. HAYS, D. (Ed.) *Readings in Automatic Language Processing*. American Elsevier, New York, 1966, 202 pp.

Questions of philosophical import [Item 14]

63. MACKAY, D. M. Mind-like behavior in artifacts. *Brit. J. Phil. Sci.* 2, (1951), 105-121.
64. SAYRE, K. M., AND CROSSON, F. J. (Eds.) *The Modeling of Mind: Computers and Intelligence*. University of Notre Dame Press, Notre Dame, Ind., 1963, 275 pp. CR-6455-6205.
65. SIMON, H. A. The architecture of complexity. *Proc. Am. Phil. Soc.* 106, (1962), 467-482.
66. TURING, A. M. Computing machinery and intelligence. In *Computers and Thought*, pp. 11-35.

A Report of the ACM Curriculum Committee on Computer Education for Management

R.L. Ashenurst
Editor

Curriculum Recommendations for Graduate Pro- fessional Programs in Information Systems

The need for education related to information systems in organizations is discussed, and a curriculum is proposed for graduate professional programs in universities, at the Master's level. Material necessary for such programs is identified, and courses incorporating it are specified. Detailed course descriptions are presented, program organization discussed, and implementation questions considered.

Key Words and Phrases: education, management systems, systems analysis, management information systems, information systems development, information analysis, system design

CR Categories: 1.52, 3.51

Copyright © 1972, Association for Computing Machinery, Inc.

General permission to republish, but not for profit, all or part of this material is granted, provided that reference is made to this publication, to its date of issue, and to the fact that reprinting privileges were granted by permission of the Association for Computing Machinery.

This work was supported by Grant GJ-356 from the National Science Foundation.

Contents

Preface

1. Introduction
 2. Information Systems Development
 - 2.1 Information Systems in Organizations
 - 2.2 The Development Process
 - 2.3 Information Systems Positions
 - 2.4 Educational Needs
 3. Curriculum Requirements
 - 3.1 Output—Characteristics of Graduates
 - 3.2 The Educational Process
 - 3.3 Input—Prerequisites
 4. Courses
 - 4.1 Course Group A: Analysis of Organizational Systems
 - 4.2 Course Group B: Background for Systems Development
 - 4.3 Course Group C: Computer and Information Technology
 - 4.4 Course Group D: Development of Information Systems
 5. Programs
 - 5.1 Schedule for a Two-year Program
 - 5.2 Schedule for a One-year Program
 - 5.3 Options in MBA Degree Programs
 - 5.4 Options in Computer Science Master's Degree Programs
 - 5.5 Options in Other Graduate Programs
 6. Implementation
 - 6.1 Institutional Considerations
 - 6.2 Course Interactions
 - 6.3 Instructional Materials
 7. Summary
- Appendices
- A. Detailed Descriptions and References for Course Group A
 - B. Detailed Descriptions and References for Course Group B
 - C. Detailed Descriptions and References for Course Group C
 - D. Detailed Descriptions and References for Course Group D

Preface

This report contains curriculum recommendations prepared by the ACM Curriculum Committee on Computer Education for Management.

After extensive discussion with representatives of industry and educational institutions, the Committee prepared a position paper presenting preliminary conclusions concerning requirements for education relevant to information systems in organizations and outlining its proposed future activities [1].¹ Two of these activities are listed as: (1) developing detailed course outlines for major new courses necessary for a professional program in systems design; (2) recommending new fields of specialization in existing programs. This report is the result of the Committee's efforts in these two areas. The Subcommittee on Curriculum for Information Analysis and System Design undertook basic responsibility for the report. The entire Committee, however, participated actively in its preparation.

Other activities of the Committee are organized around Subcommittees concerned with Course Materials for Business Schools, Nondegree Programs in Systems Design, Faculty Training, and Undergraduate Curricula. The Subcommittee on Course Materials for Business Schools conducted a number of regional meetings on the current status of computer education in business schools, and compiled a report of its findings [2].

The Committee is indebted to the definitive report of the ACM Curriculum Committee on Computer Science [3], which provided a model for presenting its recommendations in this different but related area.

Many people assisted the Committee in its work on curricula. A draft version of this report was circulated for review to members of the academic and professional community. Subsequently a meeting for industry and government representatives was held to discuss the report (at Arden House, Harriman Campus of Columbia University in New York, January 12-14, 1972). Many suggestions and constructive criticisms from both the review process and the meeting have been incorporated in the report. The Committee is extremely grateful to the reviewers and the meeting participants, and to others whose assistance was helpful. Their names are listed at the end of the main body of the report. The Committee, of course, undertakes full responsibility for the substance of the report and the conclusions and recommendations contained in it.

¹References cited within the body of the report are listed on page 383.

The Committee membership during the preparation of this report was:

Daniel Teichroew, University of Michigan, Chairman
Russell M. Armstrong, Weyerhaeuser Company
Robert L. Ashenhurst, University of Chicago
Robert I. Benjamin, Xerox Corporation
J. Daniel Couger, University of Colorado
Gordon B. Davis, University of Minnesota
John F. Lubin, University of Pennsylvania
James L. McKenney, Harvard University
Howard L. Morgan, California Institute of Technology
Frederic M. Tonge Jr., University of California, Irvine

The members of the Subcommittee on Curriculum for Information Analysis and System Design were: R.L. Ashenhurst, Chairman; J.L. McKenney, H.L. Morgan, and F.M. Tonge Jr.

1. Introduction

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 systems specialization options within existing Master's degree programs.

As documented in the position paper [1, Sec. 2.6], 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 systems development presented here represents an attempt to integrate this knowledge and add new definition and perspective to the field.

Career positions related to information systems are themselves only beginning to be standardized. Section 2 gives a model of the information systems environment and development process on which the present curriculum approach is based. Section 3 characterizes the students for whom the program is intended and outlines the capabilities they are expected to acquire.

Section 4 presents a set of 13 courses which encompass the material needed to achieve the program aims. Section 5 treats the framework in which these courses form an independent graduate program in information systems development and also indicates how the content may be adapted to business administration, computer science, and other graduate degree programs.

Section 6 discusses aspects of implementing effective

programs of this type. Faculty from diverse fields must be assembled. Courses must be developed and taught in a manner which leads the student to appreciate the relationships among topics and their cumulative implications. Program coordination and availability of appropriate instructional materials are of prime importance.

Section 7 gives conclusions and acknowledgments. Detailed course specifications are presented as appendices. The format is similar to that of the ACM Curriculum 68 [3] with an added section which keys bibliographical references to individual topics within courses.

2. Information Systems Development

Organizations today are becoming increasingly large and complex. Even in organizations of moderate size activities are becoming increasingly diverse. The monitoring of operations becomes more complicated, reporting requirements more extensive, and planning more difficult. Managers must be prepared to respond to an ever-broadening range of internal and external situations, subject to ever-narrowing time constraints. In large organizations there is the additional factor that the sheer volume of operations has increased enormously, and this during a time when labor has become more costly.

The information systems available to those in charge of operating and directing organizational activities must keep pace with these trends. Information systems have always existed in organizations, and with the advent of computer techniques their operation has become a highly technical subject. As a result, information processing activities are becoming institutionalized within organizations. This gives rise to the need for professional and technical personnel to staff these activities.

As information systems increase in importance, their ongoing development becomes a further specialized organizational function. The cost of modifying existing systems and introducing new ones becomes a significant portion of the organizational budget. Questions of competitive efficiency replace earlier questions of mere feasibility. More coordination with line management is required to insure that information needs are appropriately addressed.

As the relationship of information systems to managerial effectiveness becomes more obvious, organizations come to consider the capital needed for future development of information systems as an investment in their management capacity. This gives rise to an information systems planning activity. Information systems planners must be able to work closely with top management, interpreting the capabilities of information systems and setting forth the options available.

2.1 Information Systems in Organizations

Information systems in organizations can exist on several levels. On the operational level, an information system can be an integral part of actions and transactions which take place on a time scale measured in seconds, minutes, or hours. On the control level, an information system may function to give line managers a summary picture of an operating unit or group of units over intervals measured in days or weeks. On the planning level, an information system may serve to indicate trends over months or years, providing top management with the basis for determining major policies and directions. There is some controversy about the relative ultimate importance of information systems at these various levels, and different organizations have different priorities in their development activities.

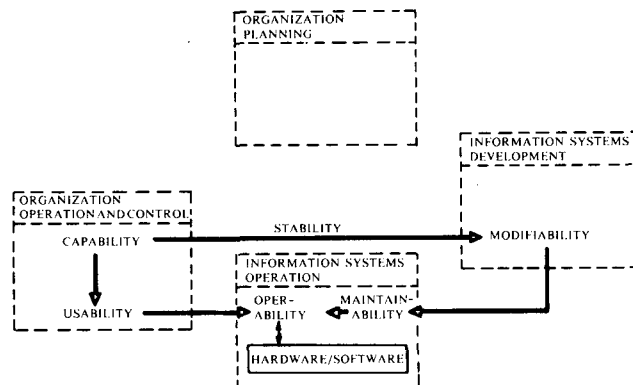
Information systems appropriate to the various levels and types of organizations have common aspects, which serve to focus the requirements for their successful development. A key quality by which any information system must be assessed is the extent to which it is consonant with the needs of people, those whom it affects directly or indirectly.

One aspect is the viewpoint of managers and others in the organization for whose benefit the system exists or of the customers or clients outside who interact with it. The system must have the *capability* to perform its intended functions in a manner suited to human action and decision making. In addition, the system must have a certain permanence or *stability*, which persists through changes in hardware, software, and development personnel. Contrasting with this, but equally important, the system must be responsive to inevitable changes in organizational requirements. Thus it must have an attribute of *modifiability*, whereby changes in its functioning and function can be accomplished in an orderly fashion by information systems professionals at the request of organization managers.

Other aspects of human interaction concern those who have day-to-day contact with the system—the direct users, operators, or maintenance personnel charged with incorporating the changes mentioned above. The attributes the system must have to meet their needs can be characterized respectively as *usability*, *operability*, and *maintainability*.

The organizational activities pertaining to information systems can be distinguished by the differing roles of those who pursue them, as indicated by the boxes in Figure 1. The “organization operation and control” box represents those whom the information system is designed to aid, together with the data administrators and information handlers needed for dealing effectively with the system. Customers and clients of the organization may interact with the information system in essentially the same way. The “information systems operation” box represents the operators and systems programmers, along with their supervisors and man-

Fig. 1. Information systems attributes and organizational functions.



agers, who have charge of the hardware/software on which the information system runs. This function often exists as an information processing department or center within the organization. The “information systems development” box represents those who must interact with the rest of the organization in regard to changing needs and translate these into systems changes for the operation group. It is assumed by the designation that the development group is also the one in charge of developing new and improved systems for the organization. The focal points of the “abilities” mentioned as attributes are shown in Figure 1 as they relate to these organizational functions.

Finally, the “organization planning” box represents those who formulate long-range plans for the organization and specify development efforts. An information systems perspective must be included here if planning is adequately to reflect technological possibilities for fulfilling organizational information needs.

The distinction between systems operation and systems development is not always recognized, and indeed, some of the difficulties encountered have come from entrusting the development function to the operating group. Increasingly, however, there are being established separate information systems development departments which are responsible for modifying existing systems and developing new ones. These departments contain analysts and designers—some perhaps specializing in configuration and conversion efforts—and project leaders who coordinate and direct them.

Although computer-based aspects of information systems have been stressed in this discussion, any information system involves manual operations and procedures which must be considered part of it. An important component of the development of an information system is determining which functions should be

in the "manual subsystem" and specifying these functions and their interaction with the "computer subsystem."

2.2 The Development Process

For purposes of this report the information systems development process is viewed as consisting of analysis, design and implementation phases, prior to the operation phase. Other essentially equivalent characterizations are cited in the position paper [1, Sec. 2.3] and elsewhere [4, Sec. 4]. These phases do not ordinarily take place strictly in the order given, but rather exist together in a continuing pattern of interaction. Analysis and design proceed in steps together, each affecting the other. An operation phase follows successful implementation, but analysis, design, and implementation activities generally continue as the system is modified and eventually supplanted. A special case is that of the conversion, in which the outward characteristics of the information system do not change radically but the hardware/software configuration on which it runs is replaced.

Operation involves the routine running of the system and is thereby appropriately the function of an information processing department, as indicated in Figure 1. This department also has responsibility for computer configuration planning and procurement in consultation with the information systems development group. Major conversion efforts require particularly close coordination between the two groups.

Implementation involves writing and debugging programs, gathering information for data bases, training personnel who will use, operate and maintain the system, and finally installing and checking out the system. Implementation is also necessarily a cooperative effort between development and operation groups.

The analysis and design functions, however, are pure developmental activities, and it is here that system inadequacies often have their origins. Such inadequacies may stem from failure to achieve a proper balance between organizational and technological factors, both of which are subject to continuing change. If technological aspects are given too much weight the resulting system may not be responsive to the needs of people, in both the capability-stability-modifiability aspects and the usability-operability-maintainability aspects. If organizational considerations are overemphasized, the resulting system although perhaps conceptually pleasing may not permit satisfactory implementation. To highlight the need for balance between these two sets of factors, the analysis and design phase of systems development is explicitly recognized as consisting of two activities: *information analysis* and *system design*.

The main emphasis in information analysis is on the determination of information needs and the patterns of information flow which will satisfy these needs. This requires interaction with organizational personnel and a

good understanding of how the organization functions. An important aspect of information analysis is willingness to consider that information problems may sometimes best be solved without resort to the computer. In fact, information analysis should start with a determination of what the problem is, and a decision whether the problem can be or should be subjected to an information systems approach at all.

The main emphasis in system design is the translation of specified information requirements into a detailed implementation plan which can be realized in hardware/software. This requires interaction with the information processing department and a good understanding of computer technology.

Information analysis and system design can be compared to product design and manufacturing system design in an industrial operation. The former looks to the question of purpose and function for potential users of the product, and the latter looks to the question of what machines are needed to manufacture the product and how they should be organized [1, Sec. 2.3]. In these terms, the function of the information processing department becomes analogous to that performed by the production control and manufacturing departments for the industrial operation.

The terms "logical system design" and "physical system design" are sometimes used to differentiate between the specification of the information system itself and its implementation in hardware/software. In a sense logical system design develops specifications for capability-stability-modifiability and physical system design develops specifications for usability-operability-maintainability. Both of these phases concentrate on the system, whereas information analysis concentrates on the organization. Two phases of information analysis may also be distinguished: analysis of information needs and analysis of how they may be satisfied in terms of requirements on an information system. These two phases are sometimes called "feasibility study" and "system specification" [4, Sec. 4].

The development of information systems then consists of an iterated process of information analysis, system design, and implementation. This "system life cycle," it has been pointed out [4, Sec. 3], applies to other kinds of development effort as well.²

Distinguishing between information analysis and system design does not imply that the two processes take place sequentially and not interactively, that there is no overlap between them, or that the same person may not often do both. Recognizing this distinction is

²Sections 2, 3, 4, 5, and 6 of this report document respectively the two phases of information analysis, the two phases of system design, and the prospective implementation phase of the educational program development recommended here.

Interestingly enough, this observation was only made after the report had reached essentially its final form. The Committee's experience in developing the report was certainly that of the iterative process characterizing the system life cycle. In particular, sections 2 and 3 were introduced rather than the rest, in response to review commentary. Thus the typical tendency to neglect information analysis in favor of system design was manifested.

appropriate because of the different styles which characterize the two processes, and because the pervasiveness of technological considerations frequently obscures the important function of information analysis in the development of effective and efficient systems.

A person functioning in either of these roles must have an understanding of systems relationships and human behavior. Information analysis requires that the organization be viewed in systematic terms to formulate means and ends effectively. It also requires understanding of the limitations imposed by human behavior on formal organizational functioning so that feasible information system specifications may be produced. System design requires an understanding not only of computer system technology but also of human behavior, since systems are used, operated, and maintained by people. If an information system is to interact with people outside the organization, it must also take their needs into account in both information analysis and system design. A further requirement for either role is the ability to model situations in quantitative terms in order to measure system performance.

For some organizational situations the foregoing model of the development process may seem too elaborate. A simpler version is one where an information processing center is run as a service by a group of highly capable technicians with "applications programmers" developing programs for this center which manipulate organizational information as required by the other departments. More and more it appears that this simpler model is inadequate even for smaller organizations because of the demands of the constantly changing organizational environment, by virtue of natural evolution of practices, and of the constantly changing information processing environment, by virtue of the dynamics of computer technology. Organizations and computer complexes are both systems undergoing constant transition, and information processing functions must be developed along similarly systematic lines to cope with the situation.

Throughout what follows, terminology is used consistently with this picture: information systems are developed to fulfill the needs of organization systems, and computer systems are the major (but not the sole) constituent in the operation of information systems. The developers of information systems must therefore be cognizant of the key aspects of both organization systems and computer systems. Information analysis must recognize organization system dynamics, and system design must recognize computer system dynamics.

2.3 Information Systems Positions

There are as yet no industry-wide standards for positions related to information systems in organizations. The larger organizations, however, are beginning to develop specializations, and their definitions seem generally consistent with the model introduced in the last section [1, Secs. 2.4, 2.5]. The following discussion

concerns functions, ignoring for the most part the distinction that employers must make regarding experience levels (trainee, junior, senior) and supervisory levels.

Programmers exist in all groups associated with information systems but are generally involved in implementation, not analysis and design. Programming positions are not within the scope of the present recommendations and are therefore not considered further.

The title Analyst or Systems Analyst was created when it was recognized that information systems development required more advanced skills than those of programmer. More and more, however, a further distinction is being made between Information Analyst and System Designer, particularly in larger organizations. Positions in the system development group carry these titles, or equivalents such as MIS Analyst for the former and System Developer or Computer Specialist for the latter. The Information Analyst is described as people-oriented or organization-oriented and is viewed by some as evolving out of the "methods and procedures" positions of less complicated times. The System Designer is described as computer-oriented or technology-oriented.

Entry-level positions in the development activity can thus be distinguished in two categories. Because of the intimate interaction required between information analysis and system design, however, the job requirements for either include a good understanding of the other. In smaller organizations there are positions involving the performance of both functions, for which the two sets of skills must be even more integrated.

An individual with either orientation can advance to Project Leader in the development group, and it is clear that a background combining information analysis and system design is a virtual necessity for promotion to this position and for successful functioning at the supervisory levels.

Positions in the information processing department tend to be more technology-oriented, but an appreciation of organizational considerations is still useful for them. Titles such as Computer Systems Analyst describe a person who is able to deal with the computer and its operating system—the hardware/software configuration—and tailor it to the efficient running of information systems. The departmental organization and its operating personnel must be taken into account in the process. More advanced positions are those of the specialist involved in planning and procurement of equipment, sometimes called a Configurator, and the specialist equipped to deal with systems conversion from one hardware/software system to another. For these positions a knowledge of organizational functions is again desirable.

Positions in departments that interact with the information systems actively tend to be more organization-oriented, but an appreciation of technological considerations is a practical necessity for them. Entry-level positions of this type may be as Assistants to vari-

ous line and staff managers, which would lead to a supervisory positions in these areas. A more advanced information systems project specialist can perform the valuable function of interfacing the two cultures of organization environment and computer environment. People in such positions are ordinarily oriented and motivated toward achieving high line managerial status, and an information systems planning specialty might well provide a route to top corporate positions.

Other advanced positions for information technology professionals are emerging as information systems become increasingly integrated into the organizational structure. Titles such as Data Base Administrator and Information Security Officer describe responsible positions associated with an ongoing information systems activity.

Manager of either the systems development group or of the information processing center is a line managerial position which may be attained after several years experience in one group or the other, perhaps after being an Assistant or Associate Manager. Combined technological and organizational qualifications are appropriate for these positions also.

Consulting positions are attractive opportunities for individuals with professional qualifications in information analysis and system design. These positions require a good perception of both organizational and technological considerations, with perhaps more emphasis on information analysis. In addition, knowledge is needed of auditing procedures, legal requirements, and other external aspects of information systems use. The demand for consulting services, already great, can only expand as information systems become increasingly complicated and more organizations have to seek outside help to deal with them.

Companies in the computer industry, which include hardware manufacturers and software developers, and in addition the newer facilities management and computer service companies, have specialized needs in the information systems area. Like consulting firms, these companies deal with other organizations representing a range of information systems applications. Entry-level positions here tend to emphasize the technological more than the organizational, although once again a combined orientation is useful. Advanced positions require abilities in technical marketing and supervision of diverse projects involving other organizations. Highly qualified professionals are needed to fill these positions.

Although the foregoing discussion has used "organization" in the commercial sense, information systems development is also necessary for government and other noncommercial organizations. People often enter such organizations motivated by societal preference rather than by management aspirations or a purely technical proficiency. The problems of information systems development are as difficult or more so for these or-

ganizations. Qualifications in both information analysis and system design, perhaps specialized to the type of organization involved, are required for entry-level positions.

2.4 Educational Needs

Many individuals currently being hired for entry-level positions of the type discussed above have an educational background inadequately suited to the job requirements, and those already filling such positions or more advanced ones often have only experience to qualify them. This will not be sufficient for the future, and the purpose of this report is to make recommendations for the educational programs appropriate for the entry-level positions which also provide support for later career advancement.

Adequate preparation for positions in information systems requires an intensive educational program, one which provides concentration on both organizational and computer systems, as well as on the development process itself. Earlier analysis [1, Sec. 3.1] has led to the conclusion that such programs do not now exist in American universities. Much existing education for management deals with making decisions on the basis of available data and does not prepare the student for clinically analyzing information needs in a systematic fashion. Similarly, existing computer science education, usually emphasizing algorithmic problem solving rather than system dynamics, does not prepare the student for the discipline of evolving system specifications. The problem then is to make up for these deficiencies on both the organizational and the technological sides and to offer an integrated approach to information analysis and system design.

The fact that the development activity requires working in communication with both these highly complex and changing environments at a time when the practice of the field itself is in a stage of evolution and rapid development imposes stringent requirements on the educational process. Professional programs in other areas are viewed as offering some basic tools and techniques which, when combined with practical apprenticeship, aid the graduate in becoming an experienced practitioner. This assumes, however, that professional practice is reasonably definitive and effective, which is more true for the established areas than for information systems development. This puts an even greater burden on the academic component of professional preparation.

The complexity of the field plus the requirement for specialization within it point to the need for a professional program at the graduate level. In fact, an undergraduate program providing the intense concentration needed to integrate the diverse aspects of this field presents formidable implementation difficulties.³ A graduate program has the added advantage of ena-

³ Recommendations for incorporating some aspects of the material of this report into undergraduate programs are now being evolved by the Subcommittee on Undergraduate Curricula.

bling the student to acquire at the undergraduate level the broad general education desirable for entering any professional field, particularly one that is changing at so rapid a pace.

The main focus of the present recommendations is thus an independent graduate professional program in information systems development, leading to a Master's degree [1, Sec. 3.3].

The program is based on a common core curriculum suitable for people who will take entry-level positions either as information analysts or system designers, or will combine the two specialties. Elective choices over and above the core curriculum can serve to reinforce a preference toward one specialty or the other.

In Section 4 a set of 13 courses is described which forms the basis for this curriculum. Section 5 describes the manner in which these courses are incorporated into the program, and how related variants can be derived from them. Section 6 deals with implementation questions for the program, which also apply to the variants in some degree.

Since this is a new curriculum, it could be offered as an independent program in either a school of business administration, a department of computer science, or some other appropriate academic unit.

The variants include options to be introduced into existing degree programs, such as Master's level programs in business administration or computer science, which cannot incorporate the recommended material in full because of their other requirements [1, Sec. 3.4]. These options would impart a flavor of the organizational-technological balance necessary for dealing with information systems effectively, as well as giving some exposure to information systems development methods. In these respects they would remedy a deficiency from which MBA programs and computer science MS programs suffer at present [2, 3]. These modifications of existing programs would also help satisfy the need for education geared to the entry-level positions in information processing centers and other organizational departments described in the previous section.

As a byproduct, courses in an MBA program designed to give a general acquaintance with the information systems field, of the "what every manager should know" variety, also can be developed out of the same basic material.

Further educational needs related to information systems, but not specifically addressed by these recommendations, are: (1) programs for individuals at an advanced stage of their careers, to allow them better to cope with the new technology of information systems; (2) programs for individuals in technical positions who seek to switch over to more managerially-oriented careers; and (3) programs for individuals in technical positions who need to keep up-to-date on the latest technological developments. Prescription for a program addressing need (1) is given in the position paper [1, Sec. 3.3]. Need (2) is currently being met by six- or

eight-week programs or evening offerings of graduate schools of business administration. Need (3) could possibly be met in a similar manner by computer science departments.

3. Curriculum Requirements

The need for graduate professional programs in information systems development having been established, the next question to be addressed is the educational requirements for students in such programs.

From analysis of the skills required for information systems development it is possible to formulate knowledge and abilities needed for entry-level positions in development groups. This may be regarded as a prescription for output of a degree program of the type under consideration.

Section 3.1 lists the knowledge and abilities students may be expected to achieve, along with the experiences they may be expected to have had, in a graduate professional program in information systems development. Section 3.2 gives a preliminary indication of how these aims may be achieved in the educational process. Finally, since a characterization of input to the process as well as output is needed, Section 3.3 rounds out the "information analysis" with a discussion of appropriate preparation for such a program, which in particular gives prerequisites for the courses described in Section 4.

3.1 Output—Characteristics of Graduates

The starting point of this discussion is that the graduate of a professional program in information systems development should be equipped to function in an entry-level position and also have a basis for continued career growth. The knowledge and abilities necessary to work effectively in this field may be characterized as obtainable by integrating concepts relating to people, models, and systems for the application of computer technology in the context of organizations and society.

Thus the requisite knowledge and abilities are conveniently grouped in six categories: (a) people; (b) models; (c) systems; (d) computers; (e) organizations; and (f) society. The first three categories are fundamental, and may be looked upon as providing tools for applications in the last three categories. A suggested list of needed knowledge and abilities is:

- (a) *people*
 - ability to hear others, as well as listen to them;
 - ability to describe individual and group behavior and to predict likely alternative future behavior in terms of commonly used variables of psychology and economics;
 - ability to describe and predict task-oriented, time-constrained behavior in an organizational setting.
- (b) *models*
 - ability to formulate and solve simple models of the operations research type;
 - ability to recognize in context the appropriate models for situations commonly encountered.

- (c) *systems*
 ability to view, describe, define any situation as a system—specifying components, boundaries, and so forth;
 ability to apply this “systems viewpoint” in depth to some class of organizations—manufacturing firms, government bureaus, universities, hospitals, service providers, etc.;
 ability to perform an economic analysis of proposed resource commitments (includes ability to specify needs for additional information and to make a set of conditional evaluations if information is unavailable);
 ability to present in writing a summary of a project for management action (suitable to serve as a basis for decision);
 ability to present in writing a detailed description of part of a project, for use in completing or maintaining same.
- (d) *computers*
 knowledge of basic hardware/software components of computer systems, and their patterns of configuration;
 ability to program in a higher-level language;
 ability to program a defined problem involving data files and communications structures;
 ability to develop several logical structures for a specified problem;
 ability to develop several different implementations of a specified logical structure;
 ability to develop specifications for a major programming project, in terms of functions, modules and interfaces;
 knowledge of sources for updating knowledge of technology;
 ability to develop the major alternatives (assuming current technology) in specifying an information processing system, including data files and communications structures, to the level of major system components;
 ability to make an economic analysis for selecting among alternatives above, including identification of necessary information for making that analysis, and also to identify noneconomic factors;
 ability to make “rough-cut” feasibility evaluations (in terms of economic and behavioral variables) of proposed new techniques or applications of current technology, identifying critical variables and making estimates and extrapolations;
 ability to develop specifications for the computer-based part of a major information system, with details of task management and data base management components.
- (e) *organizations*
 knowledge of the function of purposeful organizational structure, and of the major alternatives for that structure;
 knowledge of the functional areas of an organization—operations, finance, marketing, product specification and development;
 ability to identify in an ongoing organizational situation the key issues and problems of each functional area;
 knowledge of typical roles and role behavior in each functional area;
 ability to identify possible short-term and long-term effects of a specified action on organizational goals;
 ability to identify information needs appropriate to issues and roles above;
 knowledge of how information systems are superimposed on organizational patterns, on the operational, control, and planning levels;
 knowledge of techniques for gathering information;
 ability to gather information systematically within an organization, given specified information needs and/or specified information flows;
 ability to specify, given information needs and sources, several alternative sets of information transfers and processings to meet needs;
 ability to make “rough-cut” feasibility evaluations of such alternatives;
 ability to develop positive and negative impacts of a specified information system on specified parts of an organization;
 ability to develop specifications for a major information system, addressing a given organizational need, and determine the breakdown into manual and computer-based parts.
- (f) *society*
 ability to articulate and defend a personal position on some important issue of the impact of information technology and systems on society (important, as defined by Congressional interest, public press, semitechnical press, etc.);

ability to develop several positive and several negative impacts of a specified information system in a specified part of society; ability, given such specifications of impacts, to perform a “rough-cut” feasibility analysis of them in terms of behavioral and economic variables.

The last four abilities in both the “computers” and “organizations” categories, (d) and (e), are the key to the information systems development approach here. The ability to analyze alternatives and to make “rough-cut” designs is particularly critical in the changing information systems environment of today.

The knowledge and abilities listed are testable in the academic environment—by written or oral examinations, successfully operating computer programs, case discussions, judgment by a panel of experts and/or peers, and other commonly accepted means. Besides attaining knowledge and abilities, however, it is important for the student to have gained some experience in prototype work situations. A suggested list is:

- having gathered information in a “real” organization;
- having worked with an operations research specialist to model a complicated situation;
- having served as a member of a project team developing a specified programming system;
- having served as a member of a project team developing a specified information system;
- having participated in planning and conducting an oral presentation (and selling) of the results of a team project.

3.2 The Educational Process

Academic programs in universities have courses as major constituents, and the particular courses in a particular program represent one way of packaging its content. The courses presented in Section 4 represent a packaging of material and experience dictated by conditions of program flexibility and institutional acceptability.

The courses are intended collectively to impart to the student the knowledge, abilities, and experiences of the preceding section. The list may thus be viewed as a set of curriculum objectives. These objectives are not met automatically, however, and the list can serve to give guidelines for assessing the extent to which a given program implementation meets the requirements of entry-level positions.

Institutions adopting the general recommendations of this report may wish to modify the courses as dictated by existing circumstances. In particular, they may wish to expand some of the courses into two or three offerings. The list of knowledge, abilities, and experiences of the preceding section can also serve for assessing such modifications and for developing curriculum further.

Although the full list of knowledge, abilities, and experiences is aimed at preparation for entry-level positions in information systems development, selected parts can be used to characterize the entry-level requirements for the other more organizationally or more technologically oriented positions described earlier, and hence to assess program variants.

In considering the design, or redesign, of a curriculum it is important not to associate each entry in the list of knowledge and abilities only with the course for which it is most relevant. The principles of accumulation of knowledge and reinforcement of abilities are a desirable feature of any curriculum. Thus objectives of instilling these characteristics are best met through applying them as themes throughout a sequence of courses. For example, knowledge related to the behavior of humans in organizations should not be confined to a course on human and organizational behavior, but should be reflected in specific readings and activities in many courses. Integrational experiences such as games, cross-disciplinary projects (often cases), and seminars are further means beyond the formal course structure for achieving these objectives of a professional education. These possibilities are discussed in detail in Section 6.

There is merit in considering the value of some prior job experience for students in a program in information systems. It is certainly true that the course material will have more impact if measured against the real world in this way. In some other countries it may be possible to design programs which assume such experience, but it is difficult to envision large-scale changes in the pattern of education in this country, where graduate education normally follows directly after undergraduate education. So prior experience must be regarded as a desirable, but not necessary, factor in the process whereby these objectives are achieved.

A professional program might be envisioned, however, specifically tailored to the individual with a few years of job experience between undergraduate and graduate education. The material could be condensed; more motivation assumed, and the graduate could be prepared for returning to or entering the information systems field at a higher level than the person who had not had the intervening experience. Such a program would be particularly appropriate for a university in an urban environment.

3.3 Input—Prerequisites

A graduate professional program in information systems development should have a set of prerequisites which are easily attainable on the undergraduate level and which do not conflict with the aim of achieving a broad general education. The following list is consistent with the program of an undergraduate majoring in mathematics or engineering, and can presumably be worked into almost any program whose aims are known sufficiently early in the student's undergraduate career.

Prerequisite qualifications are identified in five subjects as follows:

- (i) finite mathematics, including the fundamentals of formal logic, sets and relations, and linear algebra;
- (ii) elementary statistics, including the fundamentals of probability, expected value, and construction of sample estimates;

- (iii) elementary computer programming, including problem analysis and algorithm synthesis, and competence in a higher-level language;
- (iv) elementary economics, including microeconomics and theory of the firm, and price theory;
- (v) elementary psychology, including fundamentals of personality formation, attitudes, and motivation.

These prerequisites can be satisfied in commonly available undergraduate courses. In particular, (iii) corresponds to the course B1 of Curriculum 68 [3], which is now available in most universities. A desirable additional prerequisite is some knowledge of elementary operations research (simple linear programming, optimization, queueing models), such as is often treated in finite mathematics courses.

These are the only prerequisites, other than a Bachelor's degree, assumed for the two-year graduate program presented in Section 5.1. A sufficiently motivated student could probably fit in much more on the undergraduate level and still not violate the aims of general education. Section 5.2 considers the possibility of a one-year program, assuming most of the first-year material of the full program is obtained by the student as an undergraduate.

A student aiming at a graduate program in business administration, computer science, or some other area, with one of the information systems options discussed in Sections 5.3, 5.4 and 5.5, would of course have to satisfy the normal undergraduate prerequisites for these programs. The list given above is generally suitable for an entering MBA candidate but less mathematical than required for the normal MS program in computer science. Such a program does not ordinarily require the economics and psychology prerequisites, so these would have to be looked upon as special preparation appropriate to the information systems orientation.

4. Courses

A set of 13 courses is presented as the basis for a program in information systems development. These are grouped into four categories: Analysis of Organizational Systems; Background for Systems Development; Computer and Information Technology; and Development of Information Systems. The course numbers and titles are given in Figure 2.

The method of grouping emphasizes the relationship of the course material to the description of the information systems development process given in Section 2 and to the curriculum requirements identified in Section 3. The course structure also lends itself to selective use in defining related programs other than the independent graduate program in information systems development recommended here. Few of the courses presented exist in universities, either in schools of business administration, departments of computer science, or elsewhere.

Fig. 2. The 13 courses.

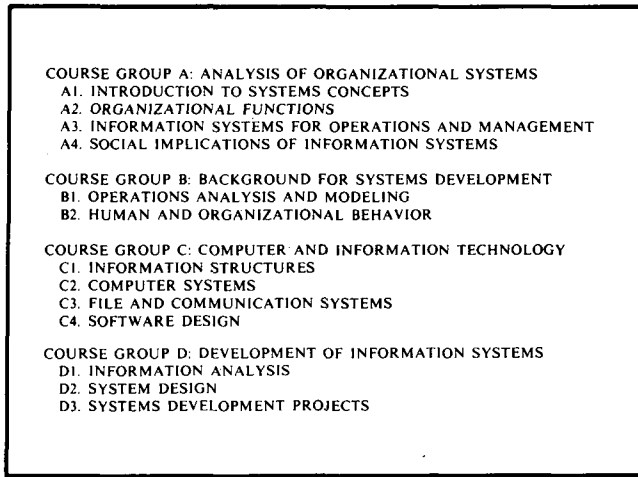
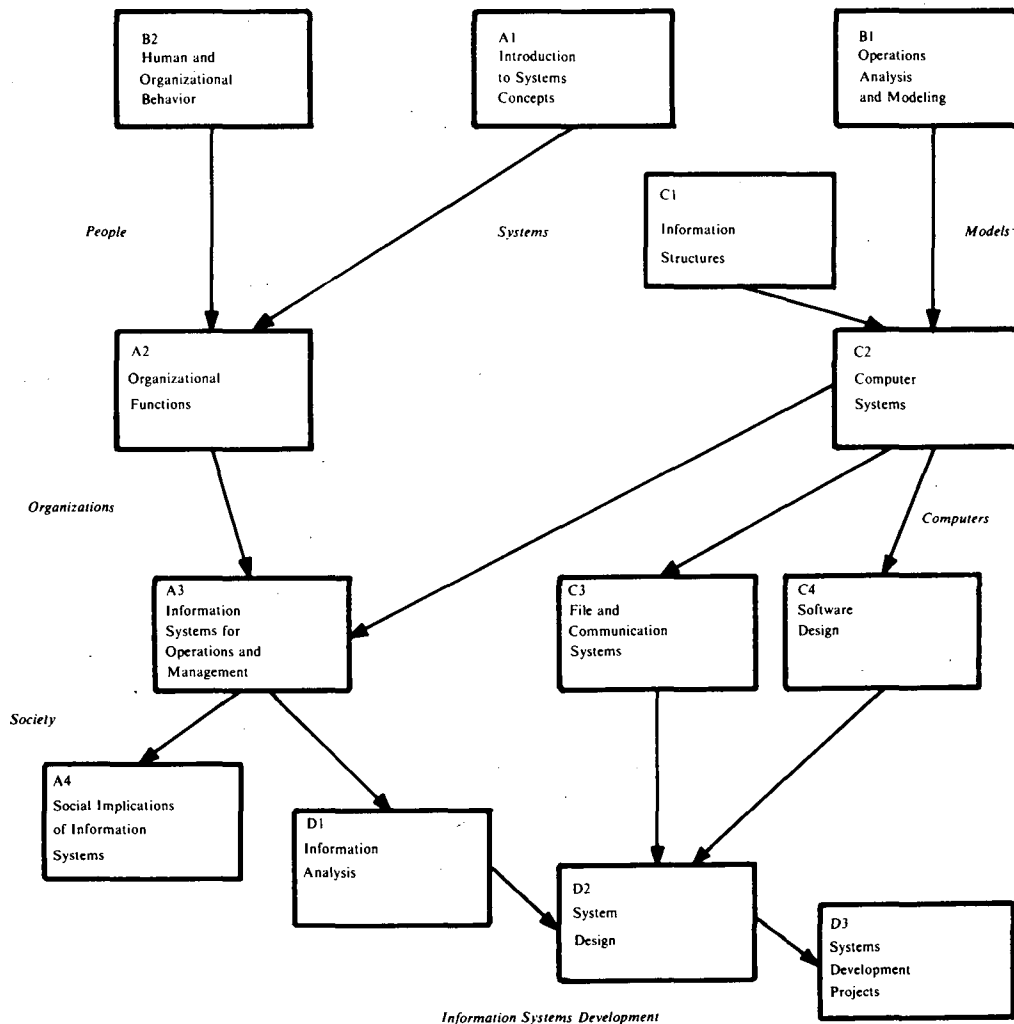


Figure 3 shows schematically the relationship of the courses and course groupings to each other. The arrows connecting the boxes containing the course names indicate the prerequisite structure. The labels outside the boxes indicate general areas of knowledge and abilities to which the proximate courses are relevant.

The material on behavior and formal modeling techniques given in courses B1 and B2 stands along with the introduction to systems concepts in course A1 as the basic conceptual framework for material on organizations (in the A courses) and computer technology (in the C courses). Part of the purpose of this structure is to give a balanced presentation of these two aspects of the information systems environment—distinguishing them but at the same time emphasizing their similarities. The unifying viewpoint is furnished by the systems concept, which underlies the whole approach on which the curriculum is based.

Thus the organizational situation modeled, the information system modeling it, and the computer system which implements the latter are all viewed as sys-

Fig. 3. Course relationships.



tems in a general sense. The systems concept is defined in terms of states, inputs and outputs, and a hierarchy of subsystem components interacting with each other over time. Although much remains to be accomplished for both the theoretical understanding of such general systems and the pragmatic means of dealing with their complexity, the unifying character of this underlying viewpoint is sufficiently powerful to give a good starting point for information systems development methods.

A second viewpoint which underlies the structuring of this curriculum is that it is desirable to emphasize the systematic features of systems development and to place the pragmatic aspects in a conceptual framework which will stand the student in good stead as new techniques replace current ones. Thus the material on information analysis and system design in courses D1 and D2 is closely related to the material on the organizational and technological environments in which the information system is embedded.

One pitfall in attempting to present such a balanced program is that the "hard" technological material tends to get attention at the expense of the "soft" organizational material, even though knowledge and abilities in the latter area are what information systems practitioners most often lack. To overcome this it is desirable to give an experiential flavor wherever possible in order to instill in the student a real feeling of the systems development situation. This is particularly the function of the projects course D3, but it runs as a theme throughout the A and C courses also.

Finally, it is assumed desirable to create a total environment for the student, to reinforce the integration of the material studied in separate courses. The techniques for doing this are further considered in Section 6.

There follows a discussion of overall aspects of each course group, along with brief course descriptions and prerequisites. The courses are all designed to carry three semester-hours of credit. Only immediate prerequisites and corequisites are cited, other prerequisites being implied in these. Undergraduate prerequisites, where given, are stated in terms of the general list included in Section 3.3.

The key (*x-y-z*) following each course title gives the recommended number of (*x*) lecture hours, (*y*) other hours (recitation, discussion section, computer laboratory etc.), and (*z*) credit hours, respectively.

Detailed course descriptions, including expanded topic outlines, references, and bibliographies, are given in Appendices A, B, C, and D following the body of this report.

4.1 Course Group A: Analysis of Organizational Systems

The four A courses lay the groundwork for the systems approach, develop this approach for organizations and their functions, demonstrate how information systems grow and serve to support organizational manage-

ment, and provide a perspective on how modern information systems technology influences and interacts with society.

As mentioned earlier, course A1, Introduction to Systems Concepts, has implications for more than organizational systems, but its inclusion among the A courses indicates its major emphasis. This course employs the example of basic financial and cost accounting systems as a concrete case. Since such systems appear in virtually every organizational enterprise, their use in this general context seems appropriate. Course A2, Organizational Functions, covers other components of organizational systems. Course A3, Information Systems for Operations and Management, deals with the information systems superimposed on these organizational components to support them. Course A4, Social Implications of Information Systems, treats the broader aspects of information systems technology, its effect on the work force and organizations generally, and impact on the individual. This course is intended to give the student some ability to make judgments in the larger context in which information systems exist.

The material and examples of the A courses relate specifically to commercial organizations. Emphasis could be shifted to governmental and/or other noncommercial organizations by substituting appropriately in course A2, and to a lesser extent in courses A3 and A4.

Course A1. Introduction to Systems Concepts (3-0-3)

Objectives. To introduce the student to the information analysis and system design curriculum. To identify the basic concepts that subsequent courses will draw upon: the systems point of view, the organization as a system, its information flows, and the nature of management information systems. To teach the rudiments of accounting as a system.

Description. The systems concept. Defining a system. Systems analysis. Management systems. Management information systems. Financial and cost accounting systems. *Prerequisite:* elementary economics.

Course A2. Organizational Functions (3-0-3)

Objectives. To introduce the process of managing an enterprise as an operational system. To provide the student with an understanding of the operations of production, finance, and marketing in an industrial firm. To identify the comprehensive information aspects of a production system, the institutional character of the information systems of finance, and the open-ended character of marketing information. To demonstrate the integrating role information systems have in the operation, control, and planning of an enterprise.

Description. Introduction to business systems. Elements of a production system. Controlling a production system. Financial systems. Identifying internal needs and external sources of funds. The marketing function. Managing a market. Integration of functions through information systems. *Prerequisite:* A1. *Corequisite:* B2.

Course A3. Information Systems for Operations and Management (3-1-3)

Objectives. To identify the decision requirements for the management of an organization, considering both traditional and novel organizational structure. To analyze the design of an information gathering and processing system intended to facilitate decision making and long-range planning. To show the time relationships of planning and control. To integrate A1 and A2 materials, emphasizing that one of the major systems of any organization is its information system.

Description. Information requirements for an organization. Operational level systems. Tactical level systems. Strategic level systems. Styles of interaction. Planning for a comprehensive information system. Measuring the effectiveness of an information system.

Prerequisites: A2, C2.

Course A4. Social Implications of Information Systems (3-0-3)

Objectives. To present the perspective of possible social effects of the information systems environment. To explore the current and projected social and economic effects of information systems in organizations. To indicate the problems which come from too narrowly defining the boundaries of a system. To understand issues, implications, and possible remedies.

Description. Historical perspective. The computer industry. Implications for the work force. Effects on organizational practice. Privacy and the quality of life. The individual and the social system. *Prerequisite:* A3.

4.2 Course Group B: Background for Systems Development

The two B courses provide basic tools and concepts relevant to the remainder of the curriculum. They are a continuation of the mathematical, statistical, economic, and psychological studies embodied in the undergraduate prerequisites. Course B1, Operations Analysis and Modeling, covers techniques of mathematical and statistical modeling. Course B2, Human and Organizational Behavior, deals with the human environment in which systems function.

Courses on operational models and organizational behavior are ordinarily not easily available within undergraduate programs. The presence of the material here in the curriculum affords the additional opportunity of keying it specifically to the information systems context.

Operational models are applicable both to organizational systems, as dealt with in the A courses, and to computer systems, as dealt with in the C courses. Similarly, the principles of organizational behavior govern both the population which information systems serve, for which the A courses are relevant, and the population of their operating environment, for which the C courses are relevant.

Course B1. Operations Analysis and Modeling (3-1-3)

Objectives. To introduce and exercise a range of analytical and simulation modeling techniques useful in decision making in the system design environment. To consider the function of such models as guides for data collection, structures for data manipulation, and as systems for testing assumptions and generating a variety of alternatives. To identify the problems of data collection, maintenance, and accuracy when using models to assist decision making activities.

Description. Characterization of scheduling situations. Analysis of allocation problems with mathematical programming. Queuing models. Inventory models. Use of simulation models. *Prerequisites:* finite mathematics, elementary statistics, elementary computer programming.

Course B2. Human and Organizational Behavior (3-1-3)

Objectives. To introduce the student to the principles governing human behavior, particularly as they relate to organizations and to the introduction and continued operation in organizations of computer-based information systems.

Description. Individual behavior. Interpersonal and group behavior. Organizational structure and behavior. The process of organizational change. The implementation and introduction of information systems. *Prerequisite:* elementary psychology.

4.3 Course Group C: Computer and Information Technology

The four C courses develop aspects of information processing hardware/software configurations, with special account taken of the technological and organizational factors which affect information systems development. Therefore in these courses hardware and software are not of interest for their own sake, as would be the case in a computer science curriculum, but for their application to information system building. For this reason human factors are stressed, relating to those who use, operate, and maintain computer systems.

Course C1, Information Structures, presents basic material on the systematic constructs used to represent information, and their organization into larger units for manipulation. Course C2, Computer Systems, covers the structure of hardware/software complexes in an integrated manner. Course C3, File and Communication Systems, follows with a unified treatment of two important aspects of large-scale computer systems technology. The approach is not for the developer of file and communication systems but rather emphasizes the manner in which they are incorporated as components in information systems. The other major components of information systems are collections of programs, and methods for efficiently organizing and operating such collections are treated in course C4, Software Design. Here again, however, the emphasis

is on understanding and incrementally improving software systems, not developing them from the ground up. Aspects of planning, testing, and project control are emphasized in both courses C3 and C4. Although it is thus basically practice-oriented, the presentation is general enough to equip the student to absorb future technological developments.

Course C1. Information Structures (2-2-3)

Objectives. To introduce the student to structures for representing logical relationships among elements of information, whether program or data, and to techniques for operating upon information structures. To examine the methods by which higher-level programming languages implement such structures and facilitate such techniques.

Description. Basic concepts of information. Modeling structures—linear lists. Modeling structures—multilinked structures. Machine-level implementation structures. Storage management. Programming language implementation structures. Sorting and searching. Examples of the use of information structures. *Prerequisite:* elementary computer programming.

Course C2. Computer Systems (2-2-3)

Objectives. To provide a working view of hardware/software configurations as integrated systems, with (possibly) concurrently functioning components.

Description. Hardware modules. Execution software. Operation software. Data and program handling software. Multiprogramming and multiprocessing environments. *Prerequisites:* B1, C1.

Course C3. File and Communication Systems (2-2-3)

Objectives. To introduce the basic functions of file and communication systems and current realizations of these systems. To analyze such realizations in terms of the tradeoffs between cost, capacity, responsiveness. To examine some systems integrating file and communication functions, such as the organizational data base system or the computer utility.

Description. Functions of file and communication systems. File system hardware. File system organization and structure. Analysis of file systems. Data management systems. Communication system hardware. Communication system organization and structure. Analysis of communication systems. Examples of integrated systems. *Prerequisite:* C2.

Course C4. Software Design (2-2-3)

Objectives. To examine how a complex computer programming task can be subdivided for maximum clarity, efficiency, and ease of maintenance and modification, giving special attention to available programming and linking structures for some frequently used interface programs such as file and communication modules. To introduce a sense of programming style into the program design process.

Description. Run-time structures in programming languages. Communication, linking, and sharing of programs and data. Interface design. Program docu-

mentation. Program debugging and testing. Programming style and aesthetics. Selected examples. *Prerequisite:* C2.

4.4 Course Group D: Development of Information Systems

The three D courses tie together the major objectives of the proposed curriculum. They use the material from the other areas in a systematic study of the process of information systems development. They emphasize how to determine what is technically possible, what is economical and practical, and what is operationally feasible in a real organizational environment. Course D1, Information Analysis, reviews methods for analyzing information needs and specifying system requirements. The life cycle concept is considered from economic, technological, and organizational perspectives. Course D2, System Design, develops a sense of the design process, incorporating the proper balance of the organizational and technological considerations. Particular attention is given to a realistic evaluation of available analytical techniques and to the importance of long-range planning. Course D3, Systems Development Projects, is a project course designed to give prototype experience in the development of moderately complex information systems.

Despite the titles of the courses, the division of material between D1 and D2 is not exactly the same as the division between the phases of information analysis and system design discussed in Section 2.3. This is because information analysis is not as formalized as system design, and consequently there is more material to be covered specifically related to the latter. Part of the information analysis process is covered in course A3, which then allows space in D1 for coverage of logical system design—the interface between information analysis and system design.

Course D1. Information Analysis (3-1-3)

Objectives. To analyze the concept of an information system. To review the approaches and techniques available to evaluate existing systems. To examine the concept of common data base for all functional modules, the factors necessary to determine the feasibility of computerization, and the organizational behavior effects of implementing a comprehensive system.

Description. Introduction to the system life cycle. System life cycle management. Basic analysis tools. Determining system alternatives. Determining system economics. Defining logical system requirements. Summary and introduction to physical system design. *Co-requisite:* A3.

Course D2. System Design (3-1-3)

Objectives. To provide the knowledge and tools necessary to develop a physical design and an operational system from the logical design. To estimate and evaluate the performance of such a system, emphasizing the value of computer and mathematical modeling techniques. Both technological and managerial aspects

of system design and implementation are considered.

Description. Basic design tools and objectives. Hardware/software selection and evaluation. Design and engineering of software. Data base development. Program development. System implementation. Post implementation analyses. Long-range planning. *Prerequisites:* C3, D1. *Corequisite:* C4.

Course D3. Systems Development Projects (1-4-3)

Objectives. To provide students with supervised and structured practical experience in the development of computer-based systems.

Description. (alternatives) Development of a system for a local firm. Development of a system for a university/college. Development of a system for a hypothetical application. *Corequisite:* D2.

5. Programs

The material in the courses described in the previous section embodies knowledge and develops abilities appropriate to entry-level positions in the information systems field. This section recommends ways in which the courses and variations on them can be incorporated into graduate degree programs in universities.

The major alternatives considered are those discussed briefly in Section 2.4: (1) a Master's degree program in information systems development, offered either as a two-year program (Section 5.1) or a one-year program (Section 5.2), depending upon the prerequisites assumed; (2) options in existing MBA programs (Section 5.3), computer science Master's degree programs (Section 5.4), and other graduate programs (Section 5.5).

It should be emphasized again that the Master's degree program in information systems development recommended here is something which does not now exist in universities. The justification for a new program is the need for putting the student into a total environment of information systems in order to achieve professional competence. This objective can be achieved only partially in existing graduate degree programs, due to the effort necessarily devoted to satisfying general requirements.

5.1 Schedule for a Two-year Program

A four-semester schedule containing the 13 courses of Section 4 is shown in Figure 4. The schedule accommodates the courses in a way consistent with their listed prerequisite structure.

If a two-year degree program is offered under the semester system, with a normal course load of four courses per semester, this leaves room for three additional courses, one in each of the first, second, and third semesters. The openings in the first and second could be used by the student for meeting program requirements not previously satisfied, or for electives. The opening in the third semester could be filled by making D3 a two-semester course, perhaps for double credit.

If a normal load is five courses per semester there is room for one additional elective course per semester. This presents a significant opportunity for the program to be oriented toward specific information system environments of interest to the student, such as hospital, library, or university. Alternatively, additional technical electives could be chosen, such as courses in system simulation, information retrieval or graphics (see Curriculum 68 courses A4, A5, and A6, respectively [3]). The character of the options which thus emerge are very dependent on the circumstances at the particular institution offering the program, so the specific possibilities are not discussed here. Another alternative is to offer the same courses in a three-semester, five-course format. This, however, would require modification of the course content and prerequisite structure.

5.2 Schedule for a One-year Program

In some institutions it may be possible to require that the more fundamental material covered in courses A1, A2, B1, B2, C1, and C2 be taken by the student as an undergraduate, and to offer the remaining material as a one-year graduate program, as shown in Figure 5.

Ordinarily the prior material would not be available in six existing course offerings, which would mean the student would have to take more than six courses in preparation. This might be remedied if most or all of A1, A2, B1, and B2 could be offered, perhaps by a faculty of business administration, as essentially self-

Fig. 4. Two-year program schedule.

SEMESTER				
1ST	A1 INTRODUCTION TO SYSTEMS CONCEPTS	B1 OPERATIONS ANALYSIS AND MODELING	C1 INFORMATION STRUCTURES	
2ND	A2 ORGANIZATIONAL FUNCTIONS	B2 HUMAN AND ORGANIZATIONAL BEHAVIOR	C2 COMPUTER SYSTEMS	
3RD	A3 INFORMATION SYSTEMS FOR OPERATIONS AND MANAGEMENT	D1 INFORMATION ANALYSIS	C3 FILE AND COMMUNICATION SYSTEMS	
4TH	A4 SOCIAL IMPLICATIONS OF INFORMATION SYSTEMS	D2 SYSTEM DESIGN	C4 SOFTWARE DESIGN	D3 SYSTEMS DEVELOPMENT PROJECTS

Fig. 5. One-year program schedule.

SEMESTER 1ST	A3 INFORMATION SYSTEMS FOR OPERATIONS AND MANAGEMENT	D1 INFORMATION ANALYSIS	C3 FILE AND COMMUNICATION SYSTEMS	
2ND	A4 SOCIAL IMPLICATIONS OF INFORMATION SYSTEMS	D2 SYSTEM DESIGN	C4 SOFTWARE DESIGN	D3 SYSTEMS DEVELOPMENT PROJECTS

contained undergraduate courses, since courses equivalent to C1 and C2 are often available to undergraduates among existing computer science offerings. Each of these would be useful courses for students other than information systems specialists.

This method of scheduling the program obviously does not relieve the institution of the necessity of coordinating all the 13 courses as a group. A possible plan might be to offer the one-year program as a fifth year of a combined Bachelor's-Master's professional program. In any case, it should not be assumed that students would achieve the necessary undergraduate preparation on their own. Counseling of undergraduates, particularly at the beginning of their third year, would be necessary to enable them to tailor their programs accordingly.

5.3 Options in MBA Degree Programs

Most programs leading to the Master of Business Administration degree do not have sufficient time available for electives to permit incorporation of the full curriculum of 13 courses and still satisfy mandatory general requirements. A maximum of five or six course openings is all that is traditionally available for an elective concentration, although this situation is changing as some schools are redesigning their curriculum for more flexibility. To make available a six-course concentration, the recommendation is as follows: offer four of the six courses, specifically A3, D1, D2, and D3, just as defined in the proposed curriculum and offer two additional courses covering the material from the C group in condensed form. Brief descriptions are given for two such courses at the end of the section. Course C10, Introduction to Computer Systems and Programming, draws material from C2 and C4, while course C11, Information Structures and Files, draws material from C1 and C3. No separate detailed descriptions of these courses are given in Appendix C since the brief descriptions relate to material in the C course group in an obvious way.

A prerequisite of elementary computer programming and introduction to algorithms is assumed. The MBA student normally has this preparation as part of regular program requirements.

Although the business student usually gets much of the substance of A1, A2, B1, and B2 as part of the

regular MBA program, the approach is typically not systems-oriented. Thus for those wishing to enter information systems development groups, as opposed to functioning in more management-oriented positions, this option is less preferable than the full two-year program or its one-year variant.

If there is room for only five elective courses for concentration, it is recommended that D3 be omitted, so that only A3, C10, C11, D1, and D2 are taken. In this case significant project work should be included in D1 and D2 since the project experience is essential to the understanding of the systems development process and should not be eliminated entirely.

Often an MBA candidate is not aiming to interact directly with information systems professionals but wants exposure to aspects of information systems technology as they affect general management functions. For this purpose it would be appropriate to elect courses A3, C10, and C11 as a minor concentration sequence.

Course C10. Introduction to Computer Systems and Programming (3-1-3)

Objectives. To provide experience in programming. To give the student a working view of hardware/software configurations and operating systems. To examine the process of synthesizing complex programs.

Description. Hardware modules. Operating system software. Multiprogramming and multiprocessing systems. Communication, linking and sharing of programs and data. Program modularity. [Material for this course is selected from courses C2 and C4.] *Prerequisite:* elementary computer programming.

Course C11. Information Structures and Files (3-1-3)

Objectives. To introduce the student to structures for representing logical relationships among elements of information, and to techniques for operating upon information structures. To describe and analyze the basic functions of file systems and current realizations of such systems.

Description. Basic concepts of information. Modeling structures—linear lists and multilinked structures. Sorting and searching. File system functions, organization and structure. Analysis of file systems. Data management systems. [Material for this course is selected from courses C1 and C3.] *Prerequisite:* C10.

5.4 Options in Computer Science Master's Degree Programs

The student whose primary area of interest is computer science but who wishes to specialize in system design should be able to obtain an MS degree in computer science with a concentration in information systems. In the ACM Curriculum 68 [3, Sec. 5] a Master's degree program consists of distribution courses from three areas: Information Structures and Processes, Information Processing Systems, and Methodologies. Besides these, it is stated, "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." A recommended adaptation of the information systems curriculum to these requirements is as follows: for Information Structures, the student takes C1 and some other "theory" course; for Information Processing Systems, the student takes courses C2, C3, and C4; the methodologies requirement is fulfilled by courses B1, D1 and D2, the latter offered in a format which includes significant project experience; and finally, course B2 and a special course representing a reorganization and combination of A1 and A3 round out the program. A brief description of such a special course is given at the end of the section. Course A10. Organizational Systems and Their Information Requirements, draws material from A1 and A3. Again no detailed description is given in Appendix A, but reference can be made to relevant parts of the detailed descriptions of courses A1 and A3.

The student graduating from such a program would be highly skilled technically but would have a considerably broader perspective on the problems of information systems development than is gained from a typical Master's degree program in computer science. This training would be useful for functioning in a systems programming group within an information processing department or perhaps as a specialist dealing with conversion problems.

Course A10. Organizational Systems and Their Information Requirements (3-0-3)

Objectives. To set forth the basic concepts: the systems point of view, the organization as a system, and the nature of management information systems. To identify the decision requirements for the management of an organization. To analyze the design of an information gathering and processing system intended to facilitate decision making and long-range planning.

Description. The systems concept. Defining a system. Management systems. Information requirements for an organization. Operational, tactical, and strategic level systems. Measuring the effectiveness of an information system. [The material in this course is drawn from courses A1 and A3. Since most computer science students are familiar with the computing center as an organization, it might prove advantageous to draw examples from that environment.] Prerequisites: elementary economics, elementary statistics.

5.5 Options in Other Graduate Programs

Industrial engineering programs are concerned with the design and analysis of industrial systems. Many are sufficiently flexible to accommodate the one-year program of Section 5.2 as an MIE program. Also, the undergraduate in industrial engineering is often well enough prepared to enter the one-year program without further prerequisites. Thus options within industrial engineering curricula may be formed using Section 5.2 as a guide.

Industrial engineering programs are also an appropriate place to consider process control and other online interactive applications. An information systems option in an industrial engineering program might be envisioned which emphasizes the role of the minicomputer in such applications, both as a stand-alone processor and as part of a hierarchical system.

Students in related areas, such as operations research, may find courses such as A3, B2, D1, and D2 of particular value in increasing their understanding and awareness of the information systems environment and the problems of systems development, and of the ways in which they can contribute their skills to the development process.

6. Implementation

The implementation of a professional degree program in information systems development represents a significant task for an institution, even one with strong existing programs in both management and information sciences. If only one of the options described in Section 5 is implemented, there are still many aspects which require close attention. Section 6.1 takes up some of these from the point of view of the institutions offering the program or programs. It emphasizes that strong coordination of the entire program is a requirement for success. Given such coordination, many possibilities arise for the development of unifying themes across several courses, and these are discussed in Section 6.2. Section 6.3 addresses itself to the question of appropriate instructional materials for supporting a program effectively.

6.1 Institutional Considerations

The previous discussion of courses and programs has stressed the need for a balanced point of view integrating the organizational and technological factors and bringing these to bear on the systems development process. It cannot be assumed that students will somehow achieve this point of view on their own, and a conscious effort must be made to set it forth in the program. This will be easier to accomplish if students enter and stay together as a class, rather than existing as one group of management-oriented students taking some computer science courses, and another group of computer-oriented students taking management courses.

If students enter and progress through the program in this way, there will be much more opportunity to interrelate the course material in meaningful ways. This is a strong reason in support of the two-year program rather than one of the variants. If an option is chosen for implementation in the absence of the two-year program, even more effort is required to achieve unification of diverse subjects. A capability must be acquired for teaching material which is not customary for the department or school offering the program (e.g. the technological material for the business school, the organizational material for the computer science department).

For these reasons it is recommended that substantial effort be allocated to coordination, in addition to teaching and organizing individual courses. This coordination should make use of the listing of knowledge, abilities, and experience in Section 3.1 to insure that the objectives of the program are being fulfilled. Through coordination, the possibilities for course interactions discussed in the next section can be more successfully exploited. It is of course desirable to achieve consistency in successive offerings of the same course, as well as among different courses, and an explicit coordination effort can also help in this regard.

Coordination cannot succeed, however, without a measure of agreement among the faculty teaching in the program. The development of an information system looks deceptively easy to one whose expertise has been gained from existing management and/or computer science curricula. The importance of countering this notion, both by word and by example, cannot be overemphasized. The complex nature of information systems, and the difficulties this poses for their development, must be continually addressed throughout the course program. Communication among faculty is necessary to insure that the various perspectives on information systems they represent are appropriately consistent.

This strongly suggests that the faculty be attached in some way to a single academic unit. While the program is clearly interdisciplinary, and the faculty may have to be drawn from different academic units, they should be able to identify themselves as a group insofar as they are concerned with the program. An important consideration is of course the availability of faculty. A clearly delineated academic structure surrounding a program is important for obtaining faculty from both inside and outside the institution.

If the two-year program (or its one-year variant) is offered, it is to some extent immaterial in what school or department it is housed, as long as the foregoing considerations are taken into account. Whether it is more appropriately offered in a business school, a computer science department, or jointly, depends on institutional circumstances. The option programs are naturally offered within existing academic units, but the need for group identification within these units may still be an important factor.

The project course D3 is intended to embody the

application of the material developed in the other courses. Careful planning and preparation is extremely important if the project experience is to be profitable. Some institutions are able to make arrangements with local firms to take on students and let them work on real projects. Successful field programs can establish continuing relationships over a three- or four-years period with firms, thereby providing an organizational association for students. Report requirements can be specified by the firm, and formal presentations can be made to management at the conclusion of each year's study. These projects may be initiated at the beginning of the academic year, with preliminary contacts being made and plans outlined prior to the second semester in order to utilize expeditiously the available time. Periodic reviews during the semester help to maintain project standards. Monitoring such arrangements puts a heavy load on those in charge of administering them, but there is general agreement as to the great value of such exposure to the real world.

Obviously a strong institutional commitment of any of the programs described is needed to achieve these unifying aims. Obtaining such a commitment, in organizational and budgetary terms, should be the first order of business for those organizing programs in information systems within universities.

6.2 Course Interactions

The coordination activity discussed in the last section can help to exploit some common themes which pervade the course structure. Three of these themes are: (a) the use of computer techniques; (b) the use of formal models; and (c) a pragmatic approach to human behavior.

(a) *The use of computer techniques.* The experience of using a computer as a simple problem-solving tool, with own-designed or packaged programs, can be incorporated into nearly all the courses. In course C1 the computer is the vehicle for the fundamental manipulation of formal data structures. In courses A1, A2, and B1 the problems are most naturally those of the organizational world. In C2, C3, C4, and D2 they are those involved with the operation of computer systems. Finally, in A3, D1, and D3 they are those concerned with information systems proper, combining aspects of both the foregoing.

The computer system as a component of a larger system has aspects over and above its use as a problem-solving tool, and no opportunity should be lost to emphasize this. Courses C2, C3, and C4, as well as the design courses D2 and D3, are the natural places to study the computer system from this point of view.

It is also important to consider the question of modes of computer system use. The student should be given access to both batch (centralized or remote) and interactive systems. The relative advantages and disadvantages of these two modes of use are subject to a good deal of controversy in the computer profession, and it

is important to put forth the notion that one should match the mode of use to the need rather than being "for" one type of system or the other.

The reliance upon the computer as a pervasive educational aid assumes proficiency in a range of programming languages and acquaintance with a range of special-purpose packages. Because of the variety of circumstances obtaining at different institutions, no explicit prescription for achieving this has been made in the course descriptions.

It is recommended that a set of programming language modules be available for those students with little prior computer experience. The programming skill required for most courses can be met by an individual taking three modules covering: (1) an algebraic procedure-oriented language; (2) a data processing procedure-oriented language; and (3) a symbolic assembly machine-oriented language. Proficiency in (1) is nominally taken care of in the preparation of entering students, but it is well to have the module available for remedial purposes. FORTRAN and COBOL, the most common in applications today, are suggested as the two languages for (1) and (2). A number of good elementary texts also exist for these languages. If appropriate software and texts are available, PL/I can substitute for either or both of these.

The programming proficiency desired can be attained by the coding and debugging of a few representative programs. The number as well as the complexity of the programs depends on the level of sophistication of the courses using the computer as a tool. Courses B2, C1, and C2 could serve as the testing ground for proficiency in algebraic, data processing, and symbolic assembly languages, respectively.

Such modules are presently offered in some institutions as one-week courses prior to registration or as intensive laboratory sessions early in the term. In any case they can generally be developed as units, with a standard set of programming projects included, so that they can be offered on a relatively routine basis by teaching assistants. It is of course important that the programming proficiency attained in this way be used in courses in a systematic fashion. Other types of programming languages can be introduced with relative ease in courses where they are appropriate. In particular, a simulation language could be introduced to support any or all of courses A2, A3, C2, C3, D1, and D2.

Standard packages for data and file management and the like are being increasingly employed as information system components. Furthermore, simulation, configuration, and performance monitoring packages are becoming available to aid in the design process. It is important that the student be exposed to the use of packages in both these ways.

(b) *The use of formal models.* The use of formal models is also a pervasive notion running throughout the courses. The introduction of the concepts of opera-

tions research in course B1 should be designed to prepare the students for using these techniques elsewhere in the program. This implies more than an "appreciation" approach and puts what is perhaps the most significant requirement on the preparation of entering students, the ability to employ the requisite mathematics. Once acquainted with these techniques, the student can be asked to use them to analyze organizational situations in courses A1, A2, and perhaps A4, the operation of computer systems in courses C2, C3, C4, and D2, and the functioning of information systems in courses A3, D1, and D3.

Course B1 therefore serves a dual purpose. On the one hand, the student is expected to gain from it some general modeling proficiency. On the other, it is desirable that the emphasis be on specific applications to information systems, not general organizational situations. The dual objectives can be best achieved if the information systems applications are framed in terms of the information processing department as a production and distribution operation. The extension to other applications then follows naturally and can be taken care of in the courses where these applications are introduced.

A less mathematical but perhaps equally important aspect of modeling is embodied in the general notion of information structure, introduced in the first part of course C1. Students should become accustomed in later courses to an information structure formulation of system problems. This serves both to help clarify the problem and to reflect the techniques actually used in implementing system design.

(c) *A pragmatic approach to human behavior.* Perhaps the most difficult task in program coordination is that of integrating the material on human behavior given in course B2 with the rest of the curriculum material. This is because B2 can give only a very general feeling for the application of behavioral techniques, and therefore it cannot be expected that students will learn to use them in any formal sense. Nevertheless, it is of crucial importance that this material be introduced informally in other courses, particularly in A3, D1, D2, and D3. It is also appropriate to emphasize the human interaction aspects of computer systems in courses C2, C3, and C4. If the program turns out practitioners who ignore or only pay lip service to the "people problems," which most experts agree pervade the information systems milieu of today, it will not have realized its aims.

6.3 Instructional Materials

Successful implementation of an information systems program requires careful attention to the question of instructional materials. Unfortunately no single textbook is currently available for most of the courses. It is often the case, however, that books plus readings will suffice, and the bibliographies in the appendices are referenced so as to help in this selection. The lack of

unified texts, however, makes it even more important to have other supporting materials available.

The rest of this section deals with three categories of such materials in more detail: (a) computer packages; (b) case studies; and (c) projects and games.

(a) *Computer packages.* The function of computer packages as information system components and as system design tools has already been mentioned. Computer packages have obvious pedagogical uses also, which increases their utility for a curriculum such as this. Introduction of packaged programs is desirable for many of the advanced courses where aspects of computer systems are studied since the questions here are usually too involved to attack by simple programming. Although advanced computer science courses may teach the pragmatics of computer systems without requiring hands-on involvement, it is to be remembered that they are designed for a student whose objectives are focused more on computer techniques and programming than on the general problems of information systems.

The references for topic sections 3, 4, and 5 of course C3 in Appendix C indicate some of the possibilities for packages and generalized file and data management concepts which could be introduced in class.

(b) *Case studies.* Case studies are a standard part of business education but have not yet been developed extensively for the information systems areas. Where available, cases can function to draw together the organizational and technological factors, the apposition of which is so characteristic of information systems problems. Techniques for using case studies have been developed for business administration curricula over the past years. These techniques would be appropriate here not only in connection with the management-oriented courses A1, A2, and A3 but also with the analysis and design courses D1 and D2.

The A courses, particularly A2, are framed in terms of a management system in an industrial context. The approach proposed seems generally applicable to the design of information systems in most operating enterprises. Other organizational systems, however, could be utilized as the basis for study. The possible use of alternative versions of course A2 based on government or institutional systems, such as those for managing hospitals, libraries, or other nonprofit enterprises, has already been mentioned. Case studies relevant to such alternatives, however, seem to be much less readily available than those for industrial management systems. Therefore an intensive development activity is necessary in order to implement the alternative points of view effectively.

In a coordinated program there is also the opportunity to develop cross-course cases, for emphasizing relationships between the material in different courses. This could be particularly effective in showing the relevance of the behavioral material of course B2 to the other courses.

(c) *Projects and games.* The recommended curriculum calls for an experiential as well as a lecture-discussion involvement in the course subject matter. One way of achieving this is through supervised projects, in which the student learns to participate in group efforts.

The possibility of organizing the major project course D3 with the cooperation of local firms has already been mentioned. No opportunity should be lost, however, to incorporate minor projects in other courses as appropriate.

Such projects can be participation in the implementation of a simple administrative control system in course A3, or of a simple computer operating system in course C2. Important aspects of file and communication systems and program libraries can be brought out by requiring projects in connection with courses C3 and C4. It is important that such projects include a requirement for testing in a prototype context, with users other than the developers, to give the appropriate emphasis to implementation difficulties.

Games are also useful for giving valuable experience in a class situation. These seem particularly appropriate to support the behavioral material of course B2 and its extension to other courses. Although computerized games can be used to advantage, the possibilities inherent in setting up role-playing situations independent of computer interaction should not be overlooked.

7. Summary

Information systems in an organization tend to be large and complex and interact with people in the organization in diverse ways. Although standard definitions for positions in the field of information systems development and other related areas are not yet fully agreed upon, it is clear that they require entry-level knowledge and abilities which combine organizational and technological factors, and they also require experience in both information analysis and system design as the basis for successfully combining the two.

For the preparation of individuals to fill such positions a graduate professional program is required, one which is coordinated and integrated in all its aspects. Common themes must be introduced and expanded throughout a series of courses, and experiences must be provided to relate the educational process to reality.

The courses presented here are intended to satisfy these criteria and hence form a basis for the implementation of programs in information systems development in universities.

Coordination of effort and development of instructional materials presents a challenge for institutions interested in such programs. Programs in information systems should not be created by merely identifying or renaming existing courses, thereby missing opportunities for curriculum enhancement. It is hoped that a number of innovative programs will be developed,

improving on the prototype given here. The members of the Committee presenting this report (listed in the Preface) would welcome opportunities to comment on proposed programs, as well as specific queries about, and continuing feedback from, these recommendations.

Acknowledgments. The following people assisted the Committee in one or another of the draft stages which preceded the final preparation of the report. The list which follows includes: those who supplied written reviews at the Committee's invitation; those who attended the meeting at Arden House mentioned in the Preface; and some of those who supplied useful commentary through other channels. The help of all is most gratefully acknowledged.

John J. Alexander Jr.
Julius Aronofsky
Robert Beard
Selwyn W. Becker
W. Lyle Brewer
Richard W. Conway
Robert B. Curry
Bruce G. Curry
Justin Davidson
Dennis Delavara
William R. Dill
A.S. Douglas
James C. Emery
Gary D. Eppen
James Fischer
Harvey Golub
C.C. Gottlieb
C. Jackson Grayson Jr.
Charles B. Greene
Richard W. Hamming
A.D. Hestenes
Richard Hughs
Kenneth Hunter
M.M. Irvine
Felix Kaufmann
W.P. Keating
Kenneth King
Charles H. Kriebel

Frank Land
Louise S. Lanzetta
Arie Y. Lewin
N.L. Lindabury
Peter Lykos
A.L. Mathios
Thomas W. McKeown
Alan G. Merten
Charles Meadow
Richard G. Mills
Sjir Nijssen
Jay F. Nunamaker Jr.
T. William Olle
David L. Parnas
Emmett K. Platt
William Pounds
Roy Saltman
Gerard Salton
Michael Samek
Edward Schefer
William F. Sharpe
Rowena W. Swanson
Peter Tas
Carl Vorder Bruegge
William J. Wenker
Warren Welsh
Kenneth Whipple
Kenneth Zearfoss

References

1. Teichroew, D. (Ed.) Education related to the use of computers in organizations (Position Paper—ACM Curriculum Committee on Computer Education for Management). *Comm. ACM* 14, 9 (Sept. 1971), 573-588. Reprinted in *IAGJ*, 4 (1971), 220-252.
2. McKenney, J.L., and Tonge, F.M. The state of computer oriented curricula in business schools 1970. *Comm. ACM* 14, 7 (July 1971), 443-448.
3. ACM Curriculum Committee on Computer Science. Curriculum 68: recommendations for academic programs in computer science. *Comm. ACM* 11, 3 (Mar. 1968), 151-197.
4. Benjamin, R.I. *Control of the Information System Development Cycle*. Wiley Communigraph Series on Business Data Processing, Wiley-Interscience, New York, 1971.

Appendices

Appendices A, B, C, and D give detailed descriptions and references for the courses in the corresponding course groups, whose brief descriptions appear in Section 4 of the main body of the report. For each course the title, hours (x - y - z) and prerequisites are given, followed by a short statement of pedagogic approach and a listing of content. The content outline is organized by topic headings, each of which corresponds to an item in the earlier brief description of Section 4. A percentage figure is given with each topic heading, indicating a suggested proportion of the "lecture hours" (x) of the entire course to be devoted to that topic. The "other hours" (y) are intended to cover recitation, which can be discussion or problem sessions, and/or laboratory, which can be computer labs with an instructor or self-help sessions at terminals. The "credit hours" (z) are uniformly 3 for all courses.

Following the content section for each course there is a reference section containing a set of explicit citations of bibliographic entries, organized by topic heading. Citations are in the form "author (year)," which indexes the corresponding item in the accompanying bibliography.

The bibliographies are by courses for groups A and B, and follow the corresponding reference sections. For groups C and D a combined bibliography for all the courses in the group is given, following the last course description.

Bibliographic entries are selectively annotated and in many cases accompanied by a citation to the review of the book or article in *Computing Reviews*. This is of the form "CR volume, number (year) review number."

Appendix A. Detailed Descriptions and References for Course Group A

A1. Introduction to Systems Concepts (3-0-3)

Prerequisite: elementary economics.

Approach: This course lays the groundwork for the curriculum by presenting the systems approach to understanding of both organizational and technological functions.

The method is a combination of a series of lectures, a few cases on particular systems, and an accounting simulation project. Out-of-classroom study can be spent in analyzing the behavior of simulated systems to obtain an acquaintance with a variety of structures. A project for the accounting portion might be the development of a cash flow simulation model by the student. This model could rely upon a case study as the foundation for understanding what is being modeled and to provide a basis for discussion.

Content:

1. *The systems concept* (20%)
States, transformations, inputs, outputs, hierarchical structure. System objectives. Systems with complex/conflicting/multiple objectives: methods of resolution. System boundaries. Open and closed systems. Open systems: properties, negative entropy, adaption. Elements (subsystems) (subunits) (components). Interfaces. Subsystems: independence/dependence, methods of decoupling. Suboptimization, side-effects. Deterministic systems; probabilistic systems. The feedback concept: maximizing, "satisficing," "adaptivizing," adjusting to change. Control in a system: standards as predicted output, feedback (open-loop, closed-loop), cost of control. General Systems Theory.
Examples of systems: ecological, medical service, transportation, manufacturing, logistics, etc.
2. *Defining a system* (10%)
Models as representations of systems. Complex versus simple models. Formal and informal systems; interaction between them. System structure: alternative structures. The identification problem. Tools: block diagrams, flow graphs, decision tables. Degrees of aggregation for systems.
3. *Systems analysis* (10%)
Selection of a "best" course of action from many possible alternatives: advantage versus disadvantage, benefit versus cost. Determination of appropriate elements, connections, and processes to achieve objectives. Objectives, alternatives, cost-benefits, criteria. Modeling. System design: improving an existing system, developing a new sys-

tem. The process of system design: problem identification and definition, alternative solutions, selection of a "solution," synthesis of the proposed system, testing of the system, refining the system. System optimization.

4. Management systems (15%)

Hierarchical structure. Interaction among subunits and within subunits. Human beings as elements in a system. "The Management System": the operations system, the decision system, the control system, time relationships and information flows. Information systems for the management system (as subsystems of the management system). Information system elements: managers, computer hardware and software, communication network, data bases, etc. Functional systems: accounting, procurement, inventory, etc.

5. Management information systems (15%)

Role of information systems in an organization. Delineating informational needs from traditional organizational structure; from non-traditional structure. Interface between man and system: man-machine systems. Information systems as operational (or production) processes. Distinction between logical and physical systems. Planning information systems. Approaches to the development of information systems.

6. Financial and cost accounting systems (30%)

Basic accounting concepts. Coding structures: the chart of accounts, reporting hierarchies. Statements. Transactions. Financial flow. Capital budgeting; inventories; depreciation; working capital; cash management. Short and long term sources of funds. Financial planning and control. Cost accounting concepts. Classification of costs. Volume relationships. The cost accounting cycle. Cost centers. Responsibility accounting. Absorption costing. Standards and standard costs. Inventory costing. Variance analysis. Cost control. Control of nonmanufacturing costs. Marginal income analysis. Interface with financial accounting.

References: No single text is available for this course, but there are extensive references from varied sources.

1. The systems concept

Ackoff (1970); Ackoff (1971); Anthony (1965); Anthony (1970); von Bertalanffy (1950); Boulding (1956); Buckley (1968); Cleland and King (1968); Ellis and Ludwig (1962); F.E. Emery (1969); J.C. Emery (1969); Forrester (1961); Goode and Machol (1957); Hitch (1953); Katz and Kahn (1966); Schoderbek (1967); Simon (1969).

2. Defining a system

von Bertalanffy (1968); Blumenthal (1969); Churchman et al. (1957); Cleland and King (1968); Goode and Machol (1957); Hitch (1953); Hitch (1961); Katz and Kahn (1966).

3. Systems analysis

von Bertalanffy (1968); Boulding (1956); Churchman et al. (1957); Churchman (1968); DeGreene (1970); J.C. Emery (1969); Forrester (1961); Hitch (1953); Hitch (1961); Katz and Kahn (1966); McKean (1958).

4. Management systems

Ackoff (1967); Ackoff (1970); Anthony (1965); Blumenthal (1969); Boulding (1956b); Buckley (1968); Churchman et al. (1957); Good and Machol (1957); Hitch (1953); Schoderbek (1967); Starr (1971).

5. Management information systems

Ackoff (1967); Benjamin (1971); Blumenthal (1969); Dearden (1972); J.C. Emery (1969); Krauss (1970); O'Brien (1970); Simon (1965).

6. Financial and cost accounting systems

Anthony (1970); Cleland and King (1968); Fremgen (1966); Horn-gren (1970); Ijiri (1968); Lindsay and Sametz (1967); Meigs (1970); Seiler (1971); Thomas (1968).

Bibliography:

Ackoff, R.L. (1967) Management misinformation systems. *Management Science* 14, 4, B- 147-56.

A paper outlining the systems concept and the problems resulting when the systems approach is ignored in the development of information systems.

Ackoff, R.L. (1970) *A Concept of Corporate Planning*. Wiley, New York.

Note especially Ch. 6, "Control," in which Ackoff argues that the Management Information System is but a subsystem of the Management System.

Ackoff, R.L. (1971) Towards a system of systems concepts. *Management Science* 17, 11, 661-71.

An exposition of the concepts and terms "used to talk about systems," with particular attention given to organizations.

Anthony, R.N. (1965) *Planning and Control Systems: A Framework for Analysis*. Graduate School of Business Administration, Harvard U., Boston.

Provides a framework structured on the notions of strategic planning, management controls and operational control.

Anthony, R.N. (1970) *Management Accounting Text and Cases*. Irwin, Homewood, Ill.

An introduction to financial and cost accounting systems.

Benjamin, R.I. (1971) *Control of the Information System Development Cycle*. Wiley, New York.

An introduction to the system life cycle and its possible evolutions. See also annotation in bibliography for Course Group D.

von Bertalanffy, L. (1950) The theory of open systems in physics and biology. *Science* 111, 23-29.

Generally accepted as the key paper drawing attention to the importance of the "open-system" concept.

von Bertalanffy, L. (1968) *General System Theory: Foundations, Development, Applications*. George Braziller, New York.

See especially Ch. 1 "Introduction" and Ch. 2 "The Meaning of General System Theory."

Blumenthal, S.C. (1969) *Management Information Systems: A Framework for Planning and Development*. Prentice-Hall, Englewood Cliffs, N.J., CR10, 10 (69) 17,647.

A highly individual and idiosyncratic attempt to apply "the systems planning" approach to the development of management information systems. Note especially Ch. 3 "The Systems Taxonomy of an Industrial Corporation." Also see annotation in bibliography for Course Group D.

Boulding, K.E. (1956) *The Image*. U. of Michigan Press, Ann Arbor, Mich.

Note especially Boulding's discussion of the levels in the hierarchy of organizations in Ch. 2 "The Image in the Theory of Organization."

Buckley, W. (1968) *Modern Systems Research for the Behavioral Scientist: A Sourcebook*. Aldine, Chicago.

The use of systems ideas in the behavioral sciences; good introduction to definitions, especially of "open systems."

Churchman, C.W., Ackoff, R.L., and Arnoff, E.L. (1957) *Introduction to Operations Research*. Wiley, New York.

Note especially Ch. 2 "An Operations Research Study of a System as a Whole" and Ch. 7 "Construction and Solution of the Models."

Churchman, C.W. (1968) *The Systems Approach*. Dell Books, New York.

Note especially "Supplement II" in which Churchman suggests additional readings and comments on the history of the systems approach, beginning with the statement that "Plato's *Republic* is a famous systems-science book."

Cleland, D.I., and King, W.R. (1968) *Systems Analysis and Project Management*. McGraw-Hill, New York. CR 10, 4 (69) 16, 532.

Note especially Ch. 6 "Planning-Programming-Budgeting and Systems Analysis."

Dearden, J. (1972) MIS is a mirage. *Harvard Bus. Rev.* (Jan.-Feb.), 90-99.

An attack on the concept of "The Management Information System," arguing that a single, integrated information system cannot be devised.

DeGreene, K.B. (Ed.) (1970) *Systems Psychology*. McGraw-Hill, New York.

"This book is written for the new interdisciplinary breed of student and professional who works with complex technological problems involving people... the book is a psychology book, a human factors book, and a management book, but above all it is a systems book."

Eckman, D.P. (Ed.) (1961). *Systems: Research and Design*. Wiley, New York.

Ellis, D.O., and Ludwig, F.J. (1962) *Systems Philosophy*. Prentice-Hall, Englewood Cliffs, N.J.

The systems concept from the engineering point of view.

Emery, F.E. (Ed.) (1969) *Systems Thinking: Selected Readings*. Penguin Books, New York.

Emphasizes systems thinking as developed from theorizing about biological systems to social systems rather than that which came from the design of complex engineering systems. Concentrates on "open systems" (open to exchange with an environment) and adaptive behavior.

- Emery, J.C. (1969) *Organizational Planning and Control Systems: Theory and Technology*. Crowell Collier and Macmillan, New York.
An analysis of multilevel planning and control and the development of a supporting information system. Note especially Ch. 1 "The Systems Concept," Ch. 2 "The Organization as a System," and Ch. 3 "The Technology of Information Systems." Also see annotation in bibliography for Course Group D.
- Forrester, J.W. (1961) *Industrial Dynamics*. MIT Press, Cambridge, Mass.
- Fremgen, J.M. (1966) *Managerial Cost Analysis*. Irwin, Homewood, Ill.
A reference text for the topics on financial and cost accounting systems.
- Goode, H.H., and Machol, R.E. (1957) *System Engineering: An Introduction to the Design of Large-Scale Systems*. McGraw-Hill, New York.
- Hitch, C. (1953) Sub-optimization in operations problems. *J. Oper. Res. Soc. Amer.* (now *Operations Research*) 1, 3, 87-99.
A landmark paper on the concept and problems of suboptimization.
- Hitch, C. (1961) On the choice of objectives in systems studies. In Eckman (1961).
- Horngren, C.T. (1970) *Accounting for Management Control*. Prentice-Hall, Englewood Cliffs, N. J.
A possible text for the topics on financial and cost accounting systems.
- Ijiri, Y. (1968) An application of input-output analysis to some problems in cost accounting. *Management Accounting* 4, 8, 49-61.
- Intercollegiate Bibliography* (1970) Selected Cases, Business Administration, Vol. 11-13, Intercollegiate Case Clearing House, Harvard U., Soldiers Field, Boston, MA 02163
- Intercollegiate Bibliography* (1971) Collected Bibliography of Cases, Vol. 14, Intercollegiate Case Clearing House, Harvard U., Soldiers Field, Boston, MA 02163
- Katz, D., and Kahn, R.L. (1966) *The Social Psychology of Organizations*. Wiley, New York.
Ch. 2 "Common Characteristics of Open Systems" is an elementary exposition of the "open system" concept and its implications for thinking about social systems. Contrasts this approach with older approaches of viewing organizations as closed systems.
- Krauss, L.I. (1970) *Computer-Based Management Information Systems*. American Management Assoc., New York.
An exposition of the basic ideas of "MIS."
- Lindsay, R., and Sametz, A.W. (1967) *Financial Management: An Analytical Approach* (rev. ed.). Irwin, Homewood, Ill.
An introductory text in financial management.
- McKean, R.N. (1958) *Efficiency in Government Through Systems Analysis*. Wiley, New York.
See especially the discussion of suboptimization and conflict among system objectives in Pt. 2 "Some General Problems of Analysis."
- Meigs, W.B. (1970) *Financial Accounting*. McGraw-Hill, New York.
A reference text for the topic on financial accounting.
- O'Brien, J.J. (1970) *Management Information Systems: Concepts, Techniques and Applications*. Van Nostrand Reinhold, New York.
An example of an elementary textbook presenting the essentials of the so-called "Management Information Systems" concept.
- Schoderbek, P.P. (1967) *Management Systems*. Wiley, New York.
A book of readings intended to be used as a textbook in management courses to help in the "understanding of the total systems concept as well as developing insight into some of the problems besetting management." Note especially the criticism of the total system concept by W.M.A. Brooker, "The Total System Myth."
- Seiler, R. (1971) *Financial Model of the Firm (FIMOF)*. Dept. of Accounting, College of Business Administration, U. of Houston, Houston, Texas.
A computer-based financial model for use in introductory accounting courses.
- Simon, H.A. (1965) *The Shape of Automation for Men and Management*. Harper & Row, New York. CR 7, 1(66)8773.
- Simon, H.A. (1969) *The Sciences of the Artificial*. MIT Press, Cambridge, Mass. CR 11, 1(70) 18,222
The entire work is recommended but note especially Ch. 4 "The Architecture of Complexity."
- Starr, M.K. (1971) *Management: A Modern Approach*. Harcourt Brace and Jovanovich, New York.
Uses systems thinking in a novel management textbook. Note especially Ch. 2 "Building Management Models," Ch. 3 "Using Models," Ch. 7 "Managing Systems with Complex Objectives," Ch. 12, "Communication and Information Control," and Ch. 13 "The Organization of Simple Systems and Aggregations."
- Thomas, W.E. (1968) *Readings in Cost Accounting, Budgeting and Control*. Southwestern U. Press, Georgetown, Texas.

A2. Organizational Functions (3-0-3)

Prerequisite: A1.

Corequisite: B2.

Approach: This course is structured on a balance of lecture and case material to describe and discuss organizational functions. (The business firm is emphasized in this outline, but alternatively the context could be other types of enterprise.)

Case discussions are used to develop understanding of the information needs necessary to manage particular functions. Simulation exercises are suggested for the logistics, inventory and cash flow sections. The financial systems and marketing management sections could use a variety of analytical models. A computerized simulation game might be used throughout the course to provide illustration of concepts, or at the end of the course to give students an understanding of the process of integration of management functions in the operation of the total enterprise.

Content:

- 1. Introduction to business systems (10%)**
The flow of product and its transformation from raw material to purchased goods. Identification of necessary personnel, capital, cash, equipment, facilities and structure. Consideration of the budget as the usual planning and control device. The process of management.
 - 2. Elements of a production system (20%)**
Objectives of production. Design of a production system. Standards. Product design and process planning, layout maintenance, inventory (logistics) systems, and work flow.
 - 3. Controlling a production system (10%)**
Forecasting, production planning and control, capacity planning, scheduling, dispatching, control of logistics. Production information systems.
 - 4. Financial systems (10%)**
Internal system capital structure. Cyclical nature of business. Sources of cash demands and flows: external systems banking; equity money markets; cash flow analysis; sources of funds. Information systems for financial management.
 - 5. Identifying internal needs and external sources of funds (15%)**
Capital needs. Change and growth. Uncertainty. Cyclical influences. Analysis of external sources of funds—capital structure, acquiring new capital. Information flow in financial institutions.
 - 6. The marketing function (10%)**
Overview of the marketing process: consumer analysis, market dynamics, competitive analysis. Nature of pricing decisions; demand and cost structures. Channels of distribution. Marketing information systems.
 - 7. Managing a market (10%)**
Product policy decisions—the product line. Maintaining a market. Advertising, selling, distribution. Product life cycle development and introduction of new products.
 - 8. Integration of functions through information systems (15%)**
Information interface between production-finance-marketing decision systems. Planning interaction and common files: the data base as an integrating system. The budgeting process: planning, control.
- References:**
- 1. Introduction to business systems**
Britt and Boyd (1963) Ch. 1, 5, 8; Buffa (1969) Ch. 2; Dearden et al. (1971) Ch. 1; Drucker (1954) Ch. 1 10; Haynes and Massie (1969) Ch. 1, 2; Prince (1970); Starr (1971a) Pt. 1; Starr (1971b) Ch. 1, 3; Van Horne (1968) Ch. 1, 2.
 - 2. Elements of a production system**
Buffa (1969) Ch. 1, 3, 9 15; Bowman and Fetter (1961); Eilon (1962); Prince (1970), Pt. 3; Starr (1971b) Ch. 2, 8 11.
 - 3. Controlling a production system**
Buffa (1969) Ch. 16; Starr (1971b) Ch. 4 7.

4. *Financial systems*
Anthony (1970); Horngren (1970); Meigs (1970); Van Horne (1968) Ch. 2, 7, 8, 25.
5. *Identifying internal needs and external sources of funds*
Van Horne (1968) Ch. 15-20.
6. *The marketing function*
Amstutz (1967) Ch. 7; Britt and Boyd (1963) Ch. 3, 4, 7, 9, 10, 38, 43, 49; Kotler (1967) Ch. 7, 9; Prince (1970) Pt. 3.
7. *Managing a market*
Britt and Boyd (1963) Ch. 17, 25, 28, 34, 35, 39; Kotler (1967) Ch. 14, 15, 16; Sheth (1967).
8. *Integration of functions through information systems.*
Blumenthal (1969); Dearden et al. (1971) Ch. 5; Emery (1969) Ch. 1-3; McFarlan et al. (1970); Morton (1971).

Bibliography:

- Amstutz, A.E. (1967) *Computer Simulation of Competitive Market Response*. M I T Press, Cambridge, Mass.
A well-described model of a market.
- Anthony, R.N. (1970) *Management Accounting: Text and Cases*. Irwin, Homewood, Ill.
- Blumenthal, S.C. (1969) *Management Information Systems; A Framework for Planning and Development*. Prentice-Hall, Englewood Cliffs, N. J. *CR 10*, 10(69) 17,647.
See annotation in bibliography for Course A1, also in bibliography for Course Group D.
- Bowman, E.H., and Fetter, R.B. (1961) *Analysis for Production Management*. Irwin, Homewood, Ill.
- Britt, S.H., and Boyd, H.W. (1963) *Marketing Management and Administrative Action*. McGraw-Hill, New York.
- Buffa, E.S. (1969) *Modern Production Management*. Wiley, New York.
A comprehensive introductory text.
- Dearden, J., McFarlan, F.W., and Zani, W.M. (1971) *Managing Computer-Based Information Systems*. Irwin, Homewood, Ill.
- Drucker, P.F. (1954) *The Practice of Management*. Harper & Row, New York.
- Eilon, S. (1962) *Elements of Production Planning and Control*. Crowell Collier and Macmillan, New York.
- Emery, J.C. (1969) *Organizational Planning and Control Systems*. Crowell Collier and Macmillan, New York.
See annotation in bibliography for Course A1, also in bibliography for Course Group D.
- Haynes, W.W., and Massie, J.L. (1969) *Management: Analysis, Concepts and Cases*. Prentice-Hall, Englewood Cliffs, N. J.
- Horngren, C.T. (1970) *Accounting for Management Control*. Prentice-Hall, Englewood Cliffs, N.J.
- Kotler, P. (1967) *Marketing Management: Analysis, Planning, and Control*. Prentice-Hall, Englewood Cliffs, N.J.
A well-articulated conceptually-oriented text.
- McFarlan, F.W., McKenney, J.L., and Seiler, J.A. (1970) *The Management Game*. Crowell Collier and Macmillan, New York.
A general business game programmed in FORTRAN IV.
- Meigs, W.B. (1970) *Financial Accounting*. McGraw-Hill, New York.
- Morton, M.S.S. (1971) *Management Decision Systems*. Graduate School of Business Administration, Harvard U., Boston. *CR 12*, 6(71)20,367.
Presents results of academic experimentation in manager/computer interactive terminal systems.
- Prince, T.R. (1970) *Information Systems for Management Planning and Control*. Irwin, Homewood, Ill.
Note especially Pt. 2 "Traditional Information Systems," Pt. 3 "Production and Operation Information Systems" and Pt. 4 "Marketing Information Systems."
- Sheth, J.N. (1967) A review of buyer behavior. *Management Science* 13, 12, B718-56.
- Starr, M.K. (1971a) *Management: A Modern Approach*. Harcourt Brace and Jovanovich, New York.
- Starr, M.K. (1971b) *Systems Management of Operations*. Prentice-Hall, Englewood Cliffs, N.J.
- Van Horne, J.C. (1968) *Financial Management and Policy*. Prentice-Hall, Englewood Cliffs, N.J.
A comprehensive, lucid discussion of financial management.

A3. Information Systems for Operations and Management (3-1-3)

Prerequisites: A2, C2.

Approach: This course deals with ways information systems are superimposed on organizational functions and may be thought of as an introduction to information analysis continued in course D1.

The method is primarily lecture and class discussion. Cases reinforce the planning and control concepts and emphasize the critical behavior aspects of decision systems. Of particular relevance are cases relying upon the analysis of the material, with interactive time shared models to improve the student's understanding of the manager/computer interaction. Research papers might require students to survey the literature on decision making in planning and control systems. A project to develop a system on a complex case study would be useful for integrating the course and preparing the student for the D courses.

Content:

1. *Information requirements for an organization* (30%)
Informal and formal channels of communication. Defining decisions. Decision criteria. Traditional decision making. Programmed decision making. Management-by-exception. Strategic, tactical and operational levels of decisions. External versus internal information sources and constraints. Integrating information systems. Cost and value of information. Management intelligence systems.
2. *Operational level systems* (10%)
Providing for information needs of operating level supervisors and their employees. Building on existing systems.
3. *Tactical level systems* (10%)
Providing for information needs of middle management. Planning and control systems. Varying needs of organizations: service-oriented versus product-oriented organizations.
4. *Strategic level systems* (20%)
Providing for information needs of executive level management. Effect of centralized versus decentralized organization structure. Strategic planning models. Analytical and simulation models in decision making.
5. *Styles of interaction* (15%)
Manager/computer interactive systems: technical and behavioral considerations. System outputs: printed, audio, graphic. Data base design considerations for fast response systems. Management control rooms.
6. *Planning for a comprehensive information system* (10%)
Project control for system development. Managing the information function. Top-down versus bottom-up approach to the overall management information system. Integrating systems.
7. *Measuring the effectiveness of an information system* (5%)
Setting up measures of effectiveness. Evaluating effectiveness: measuring system performance, calculating costs and performance, cost/performance trade-offs. Measuring user satisfaction.

References:

1. *Information requirements for an organization*
Ackoff (1970); Beckett (1971); Blumenthal (1969); Canning (1970a); Canning (1970b); Canning (1970c); Hodge and Johnson (1970); Kelly (1970); Krauss (1970); LeBreton (1969); Myers (1967); Morton (1971), Murdick and Ross (1971).
2. *Operational level systems*
Blumenthal (1969).
3. *Tactical level systems*
Anthony (1965); Canning (1968a); Emery (1969); Henderson and Dearden (1966); Lowry (1969).
4. *Strategic level systems*
Ackoff (1970); Anthony (1965); Canning (1968b); Emery (1969); Schrieber (1970); Zannetos (1964).
5. *Styles of interaction*
Canning (1970d); DeGreene (1970); Forrester (1961); Miller and Starr (1967); Morton (1971).
6. *Planning for a comprehensive information system*
Canning (1968c); Krauss (1970); McKinsey (1968); Raueo (1970); Shaw and Atkins (1970).
7. *Measuring the effectiveness of an information system*
Brandon (1963); Canning (1971); Dearden et al. (1971); Emery (1971); Orlicky (1969).

Bibliography:

- Ackoff, R.L. (1970) *A Concept of Corporate Planning*. Wiley, New York. See annotation in bibliography for Course A1.
- Anthony, R.N. (1965) *Planning and Control Systems: A Framework for Analysis*. Graduate School of Business Administration, Harvard U., Boston.
See annotation in bibliography for Course A1.
- Beckett, J. (1971) *Management Dynamics: The New Synthesis*. McGraw-Hill, New York.
- Blumenthal, S.C. (1969) *Management Information Systems: A Framework for Planning and Development*. Prentice-Hall, Englewood Cliffs, N.J. *CR 10*, 10(69)17,647.
See annotation in bibliography for Course Group A1, also in bibliography for Course Group D.
- Brandon, R. (1963) *Management Standards for Data Processing*. Van Nostrand Reinhold, New York. *CR 5*, 5(64)6162.
- Canning, R.G. (1968a) New dimensions in management control. *EDP Analyzer 6*, 2.
- Canning, R.G. (1968b) Systematic methods for business planning. *EDP Analyzer 6*, 3.
- Canning, R.G. (1968c) Overall guidance of data processing. *EDP Analyzer 6*, 8.
- Canning, R.G. (1970a) Creating the corporate data base. *EDP Analyzer 8*, 2.
- Canning, R.G. (1970b) Organizing the corporate data base. *EDP Analyzer 8*, 3.
- Canning, R.G. (1970c) Processing the corporate data base. *EDP Analyzer 8*, 4.
- Canning, R.G. (1970d) Progressive fast response systems. *EDP Analyzer 8*, 8.
- Canning, R.G. (1971) Get more computer efficiency. *EDP Analyzer 9*, 3.
- Dearden, J., McFarlan, F.W., and Zani, W.M. (1971) *Managing Computer-Based Information Systems*. Irwin, Homewood, Ill.
- DeGreene, K.B. (Ed.) (1970) *Systems Psychology*. McGraw-Hill, New York.
See annotation in bibliography for Course A1.
- Emery, J.C. (1969). *Organizational Planning and Control Systems: Theory and Technology*. Crowell Collier and Macmillan, New York.
See annotation in bibliography for Course A1, also in bibliography for Course Group D.
- Emery, J.C. (1971) *Cost/Benefit Analysis of Information Systems*. The Society for Management Information Systems, Chicago.
See annotation in bibliography for Course Group D.
- Forrester, J.W. (1961) *Industrial Dynamics*. MIT Press, Cambridge, Mass.
- Henderson, B.D., Dearden, J. (1966) New systems for divisional control. *Harvard Bus. Rev.* (Sept.-Oct.), 144-60.
- Hodge, B.J., Johnson, H.T. (1970) *Management and Organizational Behavior*. Wiley, New York.
- Intercollegiate Bibliography*. (1971) Collected Bibliography of Cases, Vol. 14, Intercollegiate Case Clearing House, Harvard U., Soldiers Field, Boston, MA 02163.
- Kelly, J.F. (1970) *Computerized Management Information Systems*. Crowell Collier and Macmillan, New York, *CR 11*, 10(70)19, 895.
- Krauss, L.I. (1970) *Computer-Based Management Information Systems*. American Management Assoc., New York.
- LeBreton, P.P. (1969) *Administrative Intelligence-Information Systems*. Houghton-Mifflin, Boston.
- Lowry, D.E. (1969) Computers in operational planning analysis and control. In *Computers and Management*, The 1967 Leatherbee Lectures, Graduate School of Business Administration, Harvard U., Boston.
- Malcolm, D.G., and Rowe, A. J. (Eds.) (1960) *Management Control Systems*. Wiley, New York.
- McKinsey & Co. (1968) *Unlocking the Computer's Profit Potential*. McKinsey & Co. New York. Reprinted in *Computers and Automation* (Apr. 1969), 24-33. *CR 10*, 11(69)17, 795.
- Miller, P.W., and Starr, M.K. (1967) *The Structure of Human Decisions*. Prentice-Hall, Englewood, Cliffs, N.J.
- Morton, M.S.S. (1971) *Management Decision Systems*. Graduate School of Business Administration, Harvard U., Boston, *CR 12*, 6(71)20, 367.
See annotation in bibliography for Course Group A2.
- Murdick, R.G., and Ross, J.E. (1971) *Information Systems for Mod-*

ern Management. Prentice-Hall, Englewood Cliffs, N.J. *CR 12*, 7(7)21, 519.

- Myers, C.A. (Ed) (1967) *The Impact of Computers on Management*. MIT Press, Cambridge, Mass. *CR 8*, 4(67)12, 265.
- Nolan, R.L. (1971) Systems analysis for computer based information systems design. *Data Base 3*, 4, 1-10.
Distinguishes organizational objectives analysis, decision analysis, and data-processing analysis as phases of information systems development. Includes over 100 references.
- Orlicky, J. (1969) *The Successful Computer System: Its Planning, Development and Management in a Business Enterprise*. McGraw-Hill, New York. *CR 10*, 11(69)17, 820.
Introduction to planning for the MIS.
- Rauseo, M.J. (1970) *Management Controls for Computer Processing*. American Management Assoc., New York. *CR 12*, 3(71)20,777.
- Schriber, A. (1970) *Corporate Simulation Models*. U. of Washington, Pullman, Wash.
- Shaw, J.C., and Atkins, W. (1970) *Managing Computer Systems Projects*. McGraw-Hill, New York. *CR 12*, 9(71)21, 832.
- Zannetos, Z. (1964) On the theory of divisional structures: some aspects of decentralization of control and decision-making. *Management Science 12*, 49-69.

A4. Social Implications of Information Systems (3-0-3)

Prerequisite: A3.

Approach: This course uses a variety of instructional methods (outside speakers would be particularly useful) and draws from a number of academic disciplines. Historical analogy is used to place the computer in historical perspective. Economic analysis is used in identifying the economic characteristics of the computer industry and in measuring the social costs of computer technology. Legal, philosophical, and moral concepts are applied to the questions of privacy and quality of life. Case studies are used to explore the impact of various applications. Emphasis is on discussion of issues, implications, and possible information system and societal remedies. Student participation in the selection of specific issues or facets to study seems appropriate.

Content:

1. *Historical perspective* (10%)
Technological change in the 19th and 20th centuries. Economic and social problems of technology. Historical analogies. Method of assessing social costs of technological change.
2. *The computer industry* (15%)
Sales and employment in the computer industry. Growth pattern. Competition in the computer industry. Standardization. Government regulation. Employment in information processing jobs. Problem of providing training.
3. *Implications for the work force* (15%)
Impact on industrial occupations. Impact on clerical occupations. Impact on managerial occupations.
4. *Effects on organizational practice* (20%)
Centralization versus decentralization. Patterns of obtaining and providing services. Legal requirements. Possibilities for individualization. Effect on capacity to and rate of change.
5. *Privacy and the quality of life* (20%)
Public and private data banks. Rights of privacy. Relation of the individual to organizational data systems. Consumer protection.
6. *The individual and the social system* (20%)
Influence on the educational process. Influence on the political process. Systems for administering justice, welfare, health care.

References: Some discussion of overall approaches may be found in Horowitz et al. (1972).

1. *Historical perspective*
Kelson et al. (1967); Rosenberg (1971); Taviss (1970); Viavant (1971); Westin (1971).
2. *The computer industry*
Desmonde (1971); Martin and Norman (1970); Sharpe (1969); Taviss (1970).
3. *Implications for the work force*
Gossman (1969); Myers (1967); Rosenberg (1971); Simon (1965); Taviss (1970).
4. *Effects on organizational practice*
Leavitt and Whistler (1958); Martin and Norman (1970); Myers (1967); Pylyshyn (1970); Simon (1965); Withington (1970).

5. *Privacy and the quality of life*
Greenberger (1971); Harrison (1967); Hoffman (1969); Martin and Norman (1970); Miller (1971); Pylyshyn (1970); Taviss (1970); Westin (1967).
6. *The individual and the social system*
Desmonde (1971); Martin and Norman (1970); Pylyshyn (1970); Taviss (1970).

Bibliography:

- Desmonde, W.H. (1971) *Computers and Their Uses*. Prentice-Hall, Englewood Cliffs, N.J.
A layman's view of how computers are used.
- Grossman, F.W. (1969) *The Impact of Technological Change on Man-Power and Skill Demand*. Department of Industrial Engineering, U. of California, Berkeley, Calif.
- Greenberger, M. (Ed.) (1971) *Computers, Communications and the Public Interest*. Johns Hopkins Press, Baltimore, Md. CR 12, 11(71)22, 096.
A series of lectures by knowledgeable and thoughtful people on the relations between computers and society.
- Harrison, A. (1967) *The Problems of Privacy in the Computer Age: An Annotated Bibliography*. Document RM-5495-PR, The RAND Corp., Santa Monica, Calif.
- Hoffman, L. (1969) Computers and privacy: a survey. *Computing Surveys* 1, 2, 85-103.
A survey of technical literature and a discussion of what the technology can do to assist in maintaining privacy of information.
- Horowitz, E., Morgan, H., and Shaw, A. (1972) Computers and society: a proposed course for computer scientists. *Comm. ACM* 15, 4, 257-261.
- Kelson, R.R., Peck, J., and Kalacheck, E. (1967) *Technology, Economic Growth, and Public Policy*. Brookings Institute, Washington, D.C.
- Leavitt, H.J., and Whisler, T.L. (1958) Management in the 1980's. *Harvard Bus. Rev.* (Nov.-Dec.), 41-48. CR 9, 4(68)13,985.
- Martin, J., and Norman, A. (1970) *The Computerized Society*. Prentice-Hall, Englewood Cliffs, N.J.
The first part of the book titled "Euphoria" is very interesting. Later sections deal more with the technology.
- Miller, A. (1971) *The Assault on Privacy: Computers, Data Banks, and Dossiers*. U. of Michigan Press, Ann Arbor, Mich. CR 12, 8(71)21, 631.
A valuable compendium of the legal and ethical problems attendant to the growing use and sharing of data banks.
- Myers, C.A. (Ed.) (1967) *The Impact of Computers on Management*. MIT Press, Cambridge, Mass. CR 8 4(67)12, 265.
- Pylyshyn, Z.W. (Ed.) (1970) *Perspectives on the Computer Revolution*. Prentice-Hall, Englewood Cliffs, N.J. CR 12, 6(71)21, 297.
Especially good on the educational and intellectual uses of computers, and the effects of such uses.
- Rosenberg, N. (Ed.) (1971) *The Economics of Technological Change*. Penguin Books, New York.
- Sharpe, W.F. (1969). *The Economics of Computers*. Columbia U. Press, New York.
Especially interesting are the sections on vendor behavior and selection of equipment. See also annotation in bibliography for Course Group D.
- Simon, H.A. (1965) *The Shape of Automation for Men and Management*. Harper & Row, New York. CR 7, 1(66)8773.
Three separate lectures and papers, all worth reading.
- Taviss, I. (Ed.) (1970) *The Computer Impact*. Prentice-Hall, Englewood Cliffs, N.J. The readings provide good, broad coverage of the entire area.
- Viavant, W. (Ed.) (1971) *Readings in Computers and Society*, Science Research Assoc., Palo Alto, Calif.
- Westin, A.F. (1967) *Privacy and Freedom*. Atheneum Press, New York.
A legal and philosophical look at the problem of privacy.
- Westin, A.F. (Ed.) (1971) *Information Technology in a Democracy*. Harvard U. Press, Cambridge, Mass.
- Withington, F. (1970) *The Real Computer: Its Influences, Uses and Effects*. Addison-Wesley, Reading, Mass.

Appendix B. Detailed Descriptions and References for Course Group B.

B1. Operations Analysis and Modeling (3-1-3)

Prerequisites: finite mathematics, elementary statistics, elementary computer programming.

Approach: This course is based on the use of analytical models as aids in the formulation and resolution of system alternatives. Emphasis is on problem formulation and resolution relying upon available analysis packages. The discussion of projects should focus on the decision itself and on the use of models to consider alternatives and test assumptions. Problems of data acquisition, preparation, and maintenance should be stressed.

Projects should be drawn from the information system design area. The course might conclude with each student participating in the formulation of a simulation project that includes several of the analytical models introduced early in the course.

Content:

1. *Characterization of scheduling situations (20%)*
Characterization of a set of interlocking activities as a network. Popular algorithms for formulating and solving critical path models. Problems of manipulating estimates and range of accuracy measurements. Job scheduling and dispatching rules. Use of network models for control of projects. Scheduling in operating systems.
 2. *Analysis of allocation problems with mathematical programming (20%)*
Methods of formulating and solving linear programming problems using packaged computer programs. Linear programming as an aid to planning the allocation of interdependent resources. Value of models in the sensitivity testing of formulations. Evolutionary nature of large models as a decision making aid. Applications to scheduling and computer network design. Optimization of computer networks. Note: particular attention should be paid to the data management requirements of LP models allowing examination of the general notions of constraints, objective functions, and optimization in modeling.
 3. *Queueing models (20%)*
Concept of queueing models and their general applicability to a broad range of situations. Considerations of the many queueing processes within computer systems.
 4. *Inventory models (10%)*
Inventory models ranging from simple, single product to multiple product under uncertainty. The data base as an inventory. Possible application of LP or dynamic programming analyses to inventory.
 5. *Use of simulation models (30%)*
Examples and class projects to explore the need for problem definition and reliance upon tailoring standard concepts to new situations, especially through dynamic models. Note: the analysis of the user and operating system parts of a time-sharing system might serve as class projects to integrate this topic with prior ones.
- References:** Several good textbooks (see Bibliography) contain teaching materials on the range of decision models covered in this course. The *Journal of the ACM* has published many specific papers on the use of operations analysis techniques in the design of computer systems.
1. *Characterization of scheduling situations*
Conway et al. (1967); Denning (1967).
 2. *Analysis of allocation problems with mathematical programming*
Aho et al. (1971); Day (1965); Ramamoorthy and Chandy (1970); Theiss (1965).
 3. *Queueing models*
Abate et al. (1968); Coffman (1969); Frank (1969); Gaver (1966).
 4. *Inventory models*
Gaver and Lewis (1971); Martin (1967); Sharpe (1969); Woodrum (1970).
 5. *Use of simulation models*
Lum et al. (1970); Senko et al. (1969); Sutherland (1971).

Bibliography:

- Abate, J., Dubner, H., and Weinberg, S.B. (1968). Queueing analysis of the IBM 2314 disk storage facility. *J. ACM* 15, 4, 577-89. CR 10, 9(69)17, 499.
- Ackoff, R., and Sasieni, M. (1968) *Fundamentals of Operations Research*. Wiley, New York.
A good basic text for the not too mathematically inclined. Easy to read.

- Aho, A., Denning, P.J., and Ullman, J.D. (1971) Principles of optimal page replacement. *J. ACM* 18, 1, 80-93. *CR* 12, 7(71)21, 554.
A dynamic programming model for optimizing paging.
- Coffman, E.G.Jr. (1969) Analysis of a drum input/output queue under scheduled operation in a paged computer system. *J. ACM* 16, 1, 73-90.
- Conway, R., Maxwell, W., and Miller, L. (1967) *Theory of Scheduling*. Addison-Wesley, Reading, Mass.
Comprehensive treatment of scheduling problems and the techniques for solving them, including simulation.
- Day, R.H. (1965) On optimal extracting from a multiple file data storage system: an application of integer programming. *Operations Research* 13, 3, 482-94.
- Denning, P.J. (1967) Effects of scheduling on file memory operations. *Proc. AFIPS SJCC*, Vol. 30, AFIPS Press, Montvale, N.J. 9-21. *CR* 8, 6(67)13, 301.
- Frank, H. (1969) Analysis and optimization of disk storage devices for time-sharing systems. *J. ACM* 16, 4, 602-20. *CR* 11, 2(70)18, 503.
- Gaver, D. (1966) *Probability Models for Multiprogramming Computer Systems*. Doc. AD 640-706, Carnegie-Mellon U., Pittsburgh, Pa. *CR* 9, 1(68)13, 459.
- Gaver, D.P., and Lewis, P.A.W. (1971) Probability models for buffer storage allocation problems. *J. ACM* 18, 4, 186-98. *CR* 12, 9(71) 21, 870.
- Hillier, F., and Lieberman, G. (1967) *Introduction to Operations Research*. Holden-Day, San Francisco.
An excellent textbook, with good coverage of probabilistic models.
- Lum, V., Ling, H., and Senko, M. (1970) Analysis of a complex data management access method by simulation modeling. *Proc. AFIPS FJCC* Vol. 37, AFIPS Press, Montvale, N.J. 211-22.
- Martin, J. (1967) *Design of Real-Time Computer Systems*. Prentice-Hall, Englewood Cliffs, N.J. *CR* 9, 2(68)13,607.
This well-written volume stresses the use of quantitative analyses at all stages in the design of information systems and gives examples of the techniques.
- Ramamoorthy, C.V., and Chandy, K.M. (1970) Optimization of memory hierarchies in multiprogrammed systems. *J. ACM* 17, 3, 426-45.
Shows the use of both linear and integer programming models.
- Senko, M., Lum, V., and Owens, P. (1969) A file organization and evaluation model (FOREM). *Proc. IFIP Congress 68*. *CR* 11, 4(70)18, 813.
A description of a generalized simulation model for file systems.
- Sharpe, W.F. (1969) *The Economics of Computers*. Columbia U. Press, New York.
Quantitative analyses from an economist's perspective.
- Sutherland, J.W. (1971) The configurator: today and tomorrow (Pt. 1) and Tackle systems selection systematically (Pt. 2). *Computer Decisions* (Feb., Apr.), 38-43; 14-19. *CR* 12, 7(71)21,521.
A two-part article on the use of simulation and analytical methods in the selection of a computer configuration.
- Teichrow, D. (1964) *An Introduction to Management Science*. Wiley, New York.
Broad coverage of operations research/management science techniques.
- Theiss, H.E. (1965) Mathematical programming techniques for optimal computer use. *Proc. 1965 ACM National Conference*, 501-12. *CR* 7, 1(66)8864.
- Veinott, A. (1965) *Mathematical Studies in Management Science*. Crowell Collier and Macmillan, New York.
Deals mainly with the probabilistic techniques—especially inventory theory.
- Wagner, H. (1970) *Principles of Management Science*. Prentice-Hall, Englewood Cliffs, N.J. *CR* 12, 2(71)20, 616.
Truly comprehensive, well written, and usable. Emphasizes deterministic models, but provides good coverage of the other areas.
- Woodrum, L.J. (1970) A model of floating buffering. *IBM Systems J.* 9, 2, 118-44. *CR* 11, 11(70)20, 149.
Uses ideas of Markov and semi-Markov processes.

B2. Human and Organizational Behavior (3-1-3)

Prerequisite: elementary psychology.

Approach: This course examines the principles of human behavior in individuals, groups, and organizations in the contexts relevant to information systems.

Behaviorally-oriented reference material relating specifically to information systems is sparse, and particularly so for the final section on implementation. The cited references frequently have a management or engineering orientation, leaving the behavioral implications to be supplied by the instructor or by the class.

An appropriate computer game or interactive laboratory experiment could be used as an effective tool to demonstrate aspects of individual, interpersonal, and group behavior, with the student population itself as subject.

Content:

1. *Individual behavior* (20%)
Human sensing and processing functions. Visual, auditory, motor, and linguistic mechanisms. Perception, cognition, and learning. Human factors engineering in information systems.
 2. *Interpersonal and group behavior* (20%)
Personality and role. Motivation, participation, and communication. Influence and effectiveness. Authority and leadership. Mechanisms for group action. The impact of information systems on interpersonal and group behavior.
 3. *Organizational structure and behavior* (25%)
Organization theory. Impact of information systems on organizational structures and behavior. Implications for management.
 4. *The process of organizational change* (25%)
Resistance to and acceptance of change. The management of change. Problems of adjustment to the information systems environment.
 5. *The implementation and introduction of information systems* (10%)
Interaction between information analysis and system design groups and the remainder of the organization. Information system project teams and their management. Preparation for installation and operation. Note: this section is background for material covered more extensively in courses DI and D2.
- References:** No one book covers the full scope of the course. Fogel (1967), Katz and Kahn (1966), and Likert (1967) are books on individual and organizational behavior written from the systems point of view. Wadia (1968) is a book of readings which cover the behavioral sciences aspects of the course fairly well. Bennis (1968) is an excellent treatment of organizational change, of which Toffler (1970) is a popular treatment. Tomeski (1970) and Withington (1969) give insight into the impact of the computer on organizations and people.
1. *Individual behavior*
Berelson and Steiner (1964) Ch. 5; Chapanis (1965); David (1967); Davies and Tune (1969); DeGreene (1970); Fogel (1967); Gregory (1966); Meister and Rabideau (1965); Miller (1967).
 2. *Interpersonal and group behavior*
Argyris (1957); Argyris (1971); Berelson and Steiner (1964) Ch. 6, 8; Cartwright and Lippitt (1957); DeGreene (1970); Haines et al. (1961); Likert (1953); MacKinnon (1962); Schein (1971); Shostrom (1967); Zalkind and Costello (1962).
 3. *Organizational structure and behavior*
Argyris (1957); Bavelas (1960); Beckett (1967); Berelson and Steiner (1964) Ch. 9; Cyert and March (1963); DeCarlo (1967); Hage (1965); Katz and Kahn (1966); Klahr and Leavitt (1967); Likert (1967); March and Simon (1958); McGregor (1960); Simon (1964); Steiner (1964); Whisler (1967); Woodward (1965).
 4. *The process of organizational change*
Bennis (1966); Burns and Stalker (1961); DeCarlo (1967); Ginzburg and Reilley (1957); Fuller (1969); Lippitt (1969); Lippitt et al. (1958); Morison (1966); Tannenbaum (1968); Toffler (1970); Whisler (1970).
 5. *The implementation and introduction of information systems*
Canning (1957) Ch. 6-9; Canning and Sisson (1967) Ch. 3, 4; Head (1964); Johnson et al. (1967) Ch. 10, 14; Meadow (1970) Ch. 12; Orden (1960); Orlicky (1969) Ch. 5-8; Postley (1960); Sackman (1967); Simon and Newell (1960); Sisson and Canning (1967); Tomeski (1970) Ch. 13, 14; Withington (1966) Ch. 8, 9.

Bibliography:

- Argyris, C. (1971) Management information systems: the challenge to rationality and emotionality. *Management Science* 17, 6, B275-92.

- Argyris, C. (1957) The individual and organization: some problems of mutual adjustment. *Administrative Science Quarterly* 2, 1-24.
- Bavelas, A. Communication and organization. In Shultz and Whisler (1960).
- Beckett, J. A. The total-systems concept: its implications for management. In Myers (1967).
- Bennis, W.G. (1968) *Changing Organizations*. McGraw-Hill, New York.
- Berelson, B., and Steiner, G.A. (1964) *Human Behavior: An Inventory of Scientific Findings*. Harcourt Brace and Jovanovich, New York.
A compendium of behavioral sciences accomplishments. Ch. 5 on learning and thinking. Ch. 6 on motivation. Ch. 8 on small group relationships, and Ch. 9 on organizations are relevant to the course.
- Burns, T., and Stalker, G.M. (1961) *The Management of Innovation*. Tavistock Publications, London.
A treatment of the external and internal constraints affecting organizational change.
- Canning, R.G. (1957) *Installing Electronic Data Processing Systems*. Wiley, New York.
Ch. 6-10 cover the programming, installation, and operation phases of information systems development. Appendices discuss human factors.
- Canning, R.G., and Sisson, R.L. (1967) *The Management of Data Processing*. Wiley, New York. CR 9, 4(68)13, 939.
Ch. 3, 5, and 6 treat the organization and staffing of a data processing operation.
- Cartwright, D., and Lippitt, R. (1957) Group dynamics and the individual. *Internat. J. of Group Psychotherapy* 7, 86-102. In Wadia (1968).
- Chapanis, A. (1965) *Man-Machine Engineering*. Wadsworth, Belmont, Calif.
- Cyert, R., and March, J.G. (1963) *Behavioral Theory of the Firm*. Prentice-Hall, Englewood Cliffs, N.J.
A behavioral view of the functioning of organizations.
- David, E.E. Jr. Physiological and psychological considerations. In Karplus (1967).
- Davies, D.R., and Tune, G.S. (1969) *Human Vigilance Performance*. American Elsevier, New York.
A study of human factors in operations with high attention requirements.
- DeCarlo, C.R. (1967) Changes in management environment and their effect upon values. In Myers (1967).
- DeGreene, K.B. (Ed.). (1970) *Systems Psychology*. McGraw-Hill, New York.
- Fogel, L.J. (1967) *Human Information Processing*. Prentice-Hall, Englewood Cliffs, N.J.
Looks at the human as an input, decision-making, output processor.
- Fuller, R.B. (1969) *Operating Manual for Spaceship Earth*. Southern Illinois U. Press, Carbondale, Ill.
A treatise on the need for human adaptation to changed environmental circumstances, by one of the more innovative thinkers of our time.
- Ginzberg, E., and Reilley, E.W. (1957) *Effecting Change in Large Organizations*. Columbia U. Press, New York.
A step-by-step analysis for managing organizational change.
- Gregory, R.L. (1966) *Eye and Brain: The Psychology of Seeing*. McGraw-Hill, New York.
- Hage, J. (1965) An axiomatic theory of organizations. *Administrative Science Quarterly* 10, 289-320.
- Haines, G., Heider, F., and Remington, D. (1961) The computer as a small group member. *Administrative Science Quarterly* 6, 3, 360-74.
- Head, R.V. (1964) *Real-Time Business Systems*. Holt, Rinehart and Winston, New York.
- Karplus, W.J. (Ed.) (1967) *On-Line Computing: Time-Shared Man-Computer Systems*. McGraw-Hill, New York. CR 8 3(67)11,952.
- Katz, D., and Kahn, R.L. (1966) *The Social Psychology of Organizations*. Wiley, New York.
A particular point of view on organizational behavior. See also annotation in bibliography for Course A1.
- Klahr, D., and Leavitt, H.J. (1967) Tasks, organization structures, and computer programs. In Myers (1967).
- Likert, R. (1953) Motivation: the core of management. *Personnel Series* No. 155, American Management Assoc., 3-21. In Wadia (1968).
- Likert, R. (1967) *The Human Organization: Its Management and Value*. McGraw-Hill, New York.
A systems approach to organizational behavior.
- Lippitt, G.L. (1969) *Organization Renewal: Achieving Viability in a Changing World*. Appleton-Century-Crofts, New York.
An analysis of the dynamics of organizational change.
- Lippitt, R., Watson, J., and Westley, B. (1958) *The Dynamics of Planned Change*. Harcourt Brace and Jovanovich, New York.
- MacKinnon, D.W. (1962) What makes a person creative? *The Saturday Review* (Feb. 10), 15-69. In Wadia (1968).
- March, J.G., and Simon, H.A. (1958) *Organizations*. Wiley, New York.
- McGregor, D. (1960) The role of staff in modern industry. In Shultz and Whisler (1960).
- Meadow, C.T. (1970) *Man-Machine Communication*. Wiley, New York. CR 12, 4(71)20, 918.
- Meister, D., and Rabideau, G.F. (1965) *Human Factors Evaluation in System Development*. Wiley, New York.
An analysis of human interaction with dynamic systems.
- Miller, G.A. (1967) *The Psychology of Communication*. Basic Books, New York.
A collection of perceptive articles on human communication, including the author's well-known "The magical number seven, plus or minus two" paper.
- Morison, E.E. (1966) *Men, Machines, and Modern Times*. MIT Press, Cambridge, Mass. CR 8, 2(67)11, 356.
A set of anecdotal case studies, bearing on the position of man pitted against technology.
- Myers, C.A. (Ed.) (1967) *The Impact of Computers on Management*. MIT Press, Cambridge, Mass. CR 8, 4(67)12, 265.
- Orden, A. (1960) Man-machine computer systems. In Shultz and Whisler (1960).
- Orlicky, J. (1969) *The Successful Computer System: Its Planning, Development and Management in a Business Enterprise*. McGraw-Hill, New York. CR 10, 11(69)17, 820.
Introduction to planning for the MIS.
- Postley, J.A. (1960) *Computers and People*. McGraw-Hill, New York. CR 1, 5(60)300.
- Sackman, H. (1967) *Computers, System Science and Evolving Society: The Challenge of Man-Machine Digital Systems*. Wiley, New York. CR 9, 5(68)14, 154.
Ch. 9, 11, and 12 are relevant to behavioral considerations.
- Schein, E.H. (1961) Management development as a process of influence. *Industrial Management Review* (May), 59-77. In Wadia (1968).
- Shostrom, E.L. (1967) *Man, the Manipulator: The Inner Journey from Manipulation to Actualization*. Abingdon Press, Nashville, Tenn.
- Shultz, G.P., and Whisler, T.L. (Eds.) (1960) *Management Organization and the Computer*. The Free Press, Glencoe, Ill.
- Simon, H. A., and Newell, A. (1960) What have computers to do with management?
In Shultz and Whisler (1960).
- Simon, H.A. (1964) On the concept of organizational goal. *Administrative Science Quarterly* 9, 1-22. In Wadia (1968).
- Sisson, R.L., and Canning, R.G. (1967) *A Manager's Guide to Computer Processing*. Wiley, New York.
- Steiner, G.A. (1964) The creative organization. *Stanford U. Graduate School of Business Bulletin* 33, 12-16. In Wadia (1968).
- Toffler, A. (1970) *Future Shock*. Random House, New York.
A much talked-about analysis of the impact of rapid external change on human behavior.
- Tomeski, E.A. (1970) *The Computer Revolution: The Executive and the New Information Technology*. Crowell Collier and Macmillan, New York.
Covers both new patterns of administration brought about by information technology and the administration of that new technology itself.
- Wadia, M.S. (Ed.) (1968) *Management and the Behavioral Sciences*. Allyn and Bacon, Boston.
- Whisler, T.L. (1967) The impact of information technology on organizational control. In Myers (1967).
- Whisler, T.L. (1970) *Information Technology and Operational Change*. Wadsworth, Belmont, Calif.

- Withington, F.C. (1969) *The Real Computer: Its Influences, Uses and Effects*. Addison-Wesley, Reading, Mass.
 Insightful discussion of the myth and the reality of the impact of the computer on people.
- Woodward, J. (1965) *Industrial Organization, Theory and Practice*. Oxford U. Press, Oxford, England.
- Zalkind, S.S., and Costello, T.W. (1962) Perception: implications for administration. *Administrative Science Quarterly* 7, 218-35. In Wadia (1968).

Appendix C. Detailed Descriptions and References for Course Group C

C1. Information Structures (2-2-3)

Prerequisite: elementary computer programming.

Approach: The structures which may be used to represent the information involved in solving problems are presented. Both modeling structures and implementation (storage) structures are covered. Emphasis is placed on treating these structures independently of particular applications. Examples, however, particularly from information system design, should be used wherever possible. The interrelationship between problem solving procedure, modeling structure, and implementation structure is stressed. Alternative implementations of a particular model are explored. Implementations in higher-level languages of several modeling structures are presented.

Students should apply the techniques presented to a number of problems. Care should be taken to separate development of modeling structures from implementation; and in many instances the student's analysis of a problem can stop at the modeling structure level. For at least some of the problems, however, students should implement and test their proposed representations.

Content:

1. *Basic concepts of information* (10%)
 Representation of information outside and inside the computer. Bits, bytes, fields, items, records, files. Numbers and characters. Characteristics of computer arithmetics—conversion, truncation and roundoff, overflow and underflow. Names, values, environments, and binding of data. Use of pointers or linkage variables to represent structure. Identifying entities about which data is to be maintained, and selecting data nodes and structures which are to be used in problem solution.
2. *Modeling structures—linear lists* (10%)
 Linear lists, stacks, queues, deques. Single, double, circular linkage. Strings, insertion, deletion, and accessing of list elements.
3. *Modeling structures—multilinked structures* (20%)
 Trees and forests. Free, oriented, and ordered trees. Binary tree representation of general trees. Traversal methods—preorder, post-order, endorder. Threading trees. Examples of tree structures as algebraic formulas, dictionaries, and other hierarchical information structures. Arrays and tables. Storage mapping functions. Linked representation of sparse arrays. Multilinked structures with heterogeneous fields and/or nodes (plexes).
4. *Machine-level implementation structures* (15%)
 Word packing and part-word addressing. Sequential allocation. Linked allocation and pointer manipulation. Scatter storage; hash table formats, hashing functions, methods of resolving collisions. Direct and indirect address calculation. Implementation of linked structures in hardware.
5. *Storage management* (5%)
 Static versus dynamic allocation. Stacks and available space lists. Explicit release of available space, reference counts, and garbage collection. Coalescing adjacent free space. Variable block size—stratified available space lists, the buddy system.
6. *Programming language implementation structures* (15%)
 Examples of the implementation of modeling structures in higher-level languages. FORTRAN, PL/I and ALGOL arrays. SNOBOL and PL/I strings. Lists in PL/I, GPSS, SIMSCRIPT, IDS. Tables and records in PL/I, COBOL.
7. *Sorting and searching* (10%)
 Radix sort, merge sort, bubble sort, address table sort, tree sort, and other sorting methods. Comparative efficiency of sorting methods. Use of sort packages. Linear search, binary search, indexed search, and other searching methods. Tradeoffs between sorting effort and

searching effort. Effect of information structures on sorting and searching techniques.

8. *Examples of the use of information structures* (15%)

Representation of information by translators. Representation of information during execution; activation records. Implementation of higher-level language data structures. Organization of an inverted file for document retrieval. Examples in graphic manipulation systems, simulation packages, data management systems.

References: Berztiss (1971) or Knuth (1968) are most suitable candidates for use as texts in this course, supplemented by additional readings in a few topics.

1. *Basic concepts of information*
 Berztiss (1971); Codd (1970); Iverson (1962); Johnson (1970); Knuth (1968); Mealy (1967); Wegner (1968).
2. *Modeling structures—linear lists*
 Berztiss (1971); Dodd (1969); Flores (1970); Hopgood (1969); Iverson (1962); Johnson (1970); Knuth (1968); Mealy (1967); Williams (1971).
3. *Modeling structures—multilinked structures*
 Berztiss (1971); Dodd (1969); Flores (1970); Hopgood (1969); Iverson (1962); Johnson (1970) Ch. 1-3; Knuth (1968); Mealy (1967); Ross (1967); Williams (1971).
4. *Machine-level implementation structures*
 Berztiss (1971); Dodd (1969); Flores (1970); Gauthier and Ponto (1970); Hopgood (1969); Iliffe (1968); Johnson (1970); Knuth (1968); Madnick (1969); Morris (1968); Ross (1967); Wegner (1968); Williams (1971).
5. *Storage management*
 Berztiss (1971); Flores (1970); Iliffe (1968); Johnson (1970); Knuth (1968); Madnick (1969); Ross (1967); Shorr and Waite (1967).
6. *Programming language implementation structures*
 Berztiss (1971); CODASYL (1971); Gordon (1969); Griswold et al. (1968); Iverson (1962); Lawson (1967); Rosen (1967) Pt. 3; Sammet (1968); Wegner (1968); Williams (1971).
7. *Sorting and searching*
 Berztiss (1971); Flores (1969a); Gauthier and Ponto (1970); Hopgood (1969); Iverson (1962); Johnson (1970) Ch. 4, 5; Wegner (1968).
8. *Examples of use of information structures*
 Berztiss (1971); Codd (1970); Dodd (1969); Knuth (1968); Wegner (1968); Williams (1971).

Bibliography: The references are to the combined bibliography given at the end of Appendix C.

C2. Computer Systems (2-2-3)

Prerequisites: B1, C1

Approach: Computer systems, their hardware and basic operating software, are studied, with attention to the human factors involved in computer system operation and maintenance. Types of modules and types of system function mode (batch, interactive, online, etc.) should be carefully distinguished.

Block diagrams, flowcharts, and some kind of formal descriptive language should be used to set forth the systems aspects discussed. A suitable choice for the latter would be an assembly language with macro capability. Problem assignments should involve proposing system or subsystem attributes and parameters for given performance specifications and testing the proposals by simulation. Simulation packages for evaluating subsystem configurations should be available.

Content:

1. *Hardware modules* (20%)
 Processor, memory, input/output, mass storage, remote transmission modules; function and possible realization of each. Microprogramming. Styles of buffering, interfacing, communication and interrupt handling. Memory management, virtual memory. Channel management, virtual configurations. Network and multiprocessor configurations. Note: the approach of this section is conceptual, to point up the need for a comprehensive hardware/software viewpoint—the concepts are then elaborated in programming and operational terms in subsequent sections.
2. *Execution software* (20%)
 General interpretive modules for execution support, e.g. list processors. Modules for memory management in real and virtual memory systems. Processor and channel modules for support of input/output, mass storage and remote transmission units in real and virtual configurations. Concepts of multiply-reentrant programs and cooperating processes.

3. *Operation software* (20%)

Loading, interrupt monitoring, diagnostic modules. Scheduling, resource allocation, performance monitoring packages. Concepts of state resurrection and interprocess protection.

4. *Data and program handling software* (20%)

Media and format conversion modules. File handling facilities. Control specifications for datasets. Translating, compiling, generating modules. Macro facilities. Editing and debugging facilities. Linkage and job control specifications for subroutines, coroutines and standard packages. Problems of identification and security. Note: this topic is background for courses C3 and C4.

5. *Multiprogramming and multiprocessing environments* (20%)

Levels of multiaccessing and multiplexing. Batch and interactive modes. Requirements for effective usability, operability, maintainability of operating systems. Performance monitoring and management of complex hardware/software configurations.

References: There is no single introductory text for a combined hardware/software course at this level. An advanced text which embodies this kind of approach is Beizer (1971). The references given for Curriculum 68 courses I3 and I4 are in general relevant. Text materials may also be drawn from the operating manuals of whatever large-scale computer system is available for use in the course.

1. *Hardware modules*

Beizer (1971) Vol. 1; Bell and Newell (1971); Buchholz (1962); Flores (1969b); Gear (1969); Hellerman (1967); Husson (1970); Iliffe (1968); Martin (1967); Tucker and Flynn (1971).

2. *Execution software*

Beizer (1971) Vol. 1; Daley and Dennis (1968); Denning (1970); Dijkstra (1968c).

3. *Operation software*

Barron (1969); Beizer (1971) Vol. 1; Rosin (1969).

4. *Data and program handling software*

Barron (1969); Rosen (1967).

5. *Multiprogramming and multiprocessing environments*

Beizer (1971) Vol. 2; Coffman and Kleinrock (1968); Conway (1963); Daley and Dennis (1968); Hellerman (1969); Lampson (1968); Rosen (1967) Pt. 5; Stimler (1969); Watson (1970); Wilkes (1967); Wilkes and Needham (1968); Wirth (1969).

Bibliography: The references are to the combined bibliography given at the end of Appendix C.

C3. File and Communication Systems (2-2-3)

Prerequisite: C2.

Approach: The basic components of file and communication systems are presented and analyzed. The functioning of these systems as integral components of an information system is stressed.

The instructional approach is primarily lecture and problem discussion. This is neither a project nor a programming course as such; most student assignments concern design or analysis of carefully specified and limited subprograms or subsystems. When possible, if a file management language is available, a small file system design and implementation project would be desirable.

Content:

1. *Functions of file and communication systems* (5%)

Role of information acquisition, storage and transmission in an organization. Typical operations in file systems: file creation, maintenance, interrogation. Typical operations using communication systems. Issues of information availability, privacy, security.

2. *File system hardware* (5%)

Characteristics of auxiliary storage devices. Capacity, access, cost. Device types: tape, disk, mass storage.

3. *File system organization and structure* (25%)

Data fields, records, files, hierarchies of files. Directories and indices, inverted files. Structure and access: sequential, direct indexed sequential, randomized, randomized with buckets. Storage allocation and control techniques.

4. *Analysis of file systems* (10%)

Estimating capacity and timing requirements. Tradeoffs between access time, capacity and density of use, cost. Tradeoffs between file creation/maintenance activity and access activity. Relevant formulas and tables.

5. *Data management systems* (10%)

Generalized data management systems. Directory maintenance, query languages, data description, job management. Characteristics of available systems.

6. *Communication system hardware* (15%)

Theoretical concepts: channels and channel capacity, noise, error detection and correction. Existing communication facilities: line types, exchanges; utilities, regulatory agencies, and tariffs. Pulse techniques. Transmission codes. Transmission modes. Line termination and terminal devices.

7. *Communication system organization and structure* (10%)

Single line: point-to-point, multidrop. Networks: centralized, decentralized, distributed. Control and protocol: acknowledgment, wait-requests, contention, polling. Switched, store-and-forward. Data concentrators and distributors.

8. *Analysis of communication systems* (5%)

Estimating line and terminal requirements: volume and message length, speed and timing, cost implications. Bottlenecks and queues, queuing analysis, simulation.

9. *Examples of integrated systems* (15%)

The data base concept: integrated data approach, coordination, control, multiple use of data. The data administrator; the computer utility. System resiliency and integrity, privacy, and security considerations.

References: While no textbook available at present is organized to match the scope of this course, Martin (1967) covers much of the material here.

1. *Function of file and communications systems*

Gruenberger (1968a); Gruenberger (1969); Martin (1967); Martin and Norman (1970); Meadow (1967); Minker and Sable (1967); Salton (1968); Senko (1969).

2. *File system hardware*

Lefkovitz (1967); Martin (1967).

3. *File system organization and structure*

CODASYL (1969); CODASYL (1971); Dodd (1969); IBM (1969); IFIP (1969); Lefkovitz (1967); Lowe (1968); McGee (1968); Martin (1967); Meadow (1967); Minker and Sable (1967); Salton (1968); Senko (1969); Watson (1970).

4. *Analysis of file systems*

CODASYL (1971); IBM (1969); IFIP (1969); Lefkovitz (1967); Lowe (1968); Martin (1967); Salton (1968).

5. *Data management systems*

CODASYL (1969); CODASYL (1971); Gruenberger (1969); IFIP (1969); Lefkovitz (1967); McGee (1968); Martin (1967); Minker and Sable (1967); Senko (1969).

6. *Communication system hardware*

Davenport (1971); Gentle (1965); Martin (1967); Martin (1969a); Martin (1969b).

7. *Communication system organization and structure*

Davenport (1971); Gentle (1965); Martin (1967); Martin (1969a); Martin (1969b).

8. *Analysis of communication systems*

Martin (1967); Martin (1969a); Martin (1969b).

9. *Examples of integrated systems*

Gruenberger (1968a); Martin (1967); Martin, Gruenberger, and Norman (1970); Parkhill (1966); Watson (1970).

Bibliography: The references are to the combined bibliography given at the end of Appendix C.

C4. Software Design (2-2-3)

Prerequisite: C2

Approach: This course brings the student to grips with the actual problems encountered in designing, implementing and modifying systems of computer programs. The concept of programming style should permeate most of the material presented, although it appears as a specific lecture topic toward the end of the course. Careful verification of program operation and documentation of programs should be emphasized. Much of the course, particularly in the laboratory sessions, may be devoted to the actual implementation of the programs. It would be useful to have an exercise in which each student must modify a program written by someone else.

Content:

1. *Run-time structures in programming languages* (10%)

Textual versus execution semantics in languages. Binding of names. Examples from FORTRAN, ALGOL, PL/I, and some data management system. Run-time stacks.

2. *Communication, linking, and sharing of programs and data* (30%)

Separation of program and data segments. Common (global)

versus local data. Block structures—static and dynamic nesting, internal and external procedures. Subroutines and coroutines as linkage structures. Sharing of code—reentrancy, recursion, pure procedures. Static and dynamic linking and loading, relocatability, self-relocating programs. Table driven programs. Asynchronous versus synchronous control, cooperating processes, multitasking.

3. *Interface design* (10%)

Parameters, work space, automating, and documenting interfaces. Control blocks.

4. *Program documentation* (10%)

Self-documenting programs. Levels of detail in documentation. Automatic flowcharting methods. Motivation for documentation—maintenance and modification of programs.

5. *Program debugging and testing* (15%)

Automating the debugging process. Symbolic debugging aids. Automatic generation of test data and expected results. Analysis of testing procedures. Hierarchical testing. Exhaustive testing versus random sampling. Testing of communications programs. Simulation as a tool. Abnormal condition handling.

6. *Programming style and aesthetics* (10%)

Modular programming—functional modules, breaking up of large functional modules. Central versus distributed control structures. Macro and micro modularity. Interlanguage and intralanguage communication. Clarity and documentation—block diagramming.

7. *Selected examples* (15%)

File handling modules. Error retry, request queueing. Communication interface modules. Polling versus contention, interrupt handling. Selected materials such as graphics programming, programming for realtime sensing devices, process control systems. Man-machine interactions.

References:

1. *Run-time structures in programming languages*
Galler (1970); Randell and Russell (1964); Rosen (1967) Pt. 2, 3; Wegner (1968).
2. *Communication, linking, and sharing of programs and data*
Daley and Dennis (1968); Dijkstra (1960); Gear (1969); Knuth (1968); Martin (1967); Morgan (1970); Wegner (1968).
3. *Interface design*
ben-Aaron (1969); Dijkstra (1968a); Martin (1967).
4. *Program documentation*
Applied Data Research (1970); Billups (1969); Walsh (1969).
5. *Program debugging and testing*
Billups (1969); Dijkstra (1968a); Gruenberger (1968b), Hassitt (1967); Martin (1967); Van Horn (1968).
6. *Programming style and aesthetics*
Dijkstra (1960); Dijkstra (1968a); Dijkstra (1968b); Knuth (1968); Knuth and Floyd (1971); Morgan (1970); Wirth and Hoare (1966); Wirth (1971).
7. *Selected examples*
Head (1971); Martin (1967).

Bibliography: The references are to the combined bibliography given at the end of Appendix C.

Combined Bibliography—Course Group C

Applied Data Research. (1970) *Autoflow Reference Manual*. Applied Data Research Corp., Princeton, N. J.

Barron, D.W. (1969) *Assemblers and Loaders*. American Elsevier, New York. CR 11, 5(70)19,077.

Beizer, B. (1971) *The Architecture and Engineering of Digital Computer Complexes*, Vols. 1 and 2. Plenum Press, New York.
A comprehensive text on hardware/software architecture and design methods. Volume 1 covers basic hardware units (processors, memories, controllers, peripherals) and software units (loaders, assemblers, compilers, interpreters), and introduces some optimization and evaluation techniques. Volume 2 deals with aspects of complex computer configurations.

Bell, C.G., and Newell, A. (1971) *Computer Structures: Readings and Examples*. McGraw-Hill, New York. CR 12, 5(71)21,279; also CR 12, 7(71)21,618.
A systematic treatment of computer architecture, with examples drawn from existing computer families. A definitive work, containing notational and pedagogical innovations which should set the style for future works in this area.

ben-Aaron, M. (1969) Programming systems standardisation. *Proc. Fourth Australian Computer Conference, Vol. 1.*, 309-12. CR 12, 10(71)22,030.

Bertziss, A.T. (1971) *Data Structures: Theory and Practice*. Academic Press, New York. CR 12, 11(71)22,086.
Combines an introduction to discrete structures with a treatment of modeling and storage structures. Example program segments use FORTRAN; last chapter describes several other programming languages for information structures. Contains an extensive bibliography.

Billups, R., and Louis, G. (1969) X-raying a programming system. *Software Age* (May), 8-17.
A discussion of documentation relating to diagnostics and program design.

Buchholz, W. (Ed.). (1962) *Planning a Computer System*. McGraw-Hill, New York. CR 4, 6(63)4786.
An earlier but still useful discussion of how a computer hardware system is designed. Based on the development of the IBM Stretch computer.

CODASYL Data Base Task Group. (1969) *October 69 Report*. Report to the CODASYL Programming Language Committee, available through ACM. CR 11, 5(70)19,080.
Contains a proposal for a Data Description Language for describing a data base and a Data Manipulation Language which, when associated with the facilities of a host language, allows manipulation of data bases described in the Data Description Language.

CODASYL Systems Committee (1971) *Feature Analysis of Generalized Data Base Management Systems*. Technical report, available from ACM, New York.
Gives the reader a good feel for the state of the art in some widely-used generalized data base management systems. The introduction to this report also appears in *Comm. ACM* 14, 5, 308-318.

Codd, E.F. (1970) A relational model of data for large shared data banks. *Comm. ACM* 13, 6, 377-87. CR 12, 3(71)20,780.
Presents a model for data base relations based on *n*-ary relations, and a normal form for such relations.

Coffman, E.G. Jr., and Kleinrock, L. (1968) Computer scheduling methods and their countermeasures. *Proc. AFIPS 1968 SJCC*, Vol. 32, AFIPS Press, Montvale, N.J., 11-21. CR 10, 2(69)16,222.

Conway, M. (1963) A multiprocessor system design. *Proc. AFIPS 1963 FJCC*, Vol. 24, Spartan Books, New York, 139-46. CR 5, 3(64)5700.

Cuadra, C.A. (Ed.) (1966-1971) *Annual Review of Information Science and Technology*, Vols. 1-6. CR 8, 1(67)11,128 (Vol. 1); CR 10, 4(69)16, 550 (Vol. 3); CR 11, 7(70)19, 391 (Vol. 4).
An excellent survey and review publication, covering mainly information storage and retrieval systems.

Daley, R.C., and Dennis, J.B. (1968) Virtual memory, processes, and sharing in MULTICS. *Comm. ACM* 11, 5, 306-12. CR 9, 8(68)14,978.
Discusses the concepts of dynamic linking and loading, and the sharing of procedures and data in virtual memory.

Davenport, W.P. (1971) *Modern Data Communications*. Hayden, New York.
An introductory textbook covering many topics in data communications in an elementary way.

Denning, P.J. (1970) Virtual memory. *Computing Surveys* 2, 3, 153-89. CR 12, 4(71)21,031
A thorough treatment of this fundamental concept for design of multiprogramming systems.

Dijkstra, E.W. (1960) Recursive programming. *Numerische Mathematik* 2, 312-18. In Rosen (1967). CR 10, 8(69)17, 275.

Dijkstra, E.W. (1968a) The structure of "THE"-multiprogramming system. *Comm. ACM* 11, 5, 341-46.
The techniques described here for program design, implementation, and verification are quite useful in illustrating program aesthetics.

Dijkstra, E.W. (1968b) Go to statement considered harmful. *Comm. ACM* 11, 3, 147-48.

Dijkstra, E.W. (1968c) Co-operating sequential processes. In Genuys (1968).
A definitive article which sets forth the basic aspects of concurrently running processes in computer systems.

Dodd, G.G. (1969) Elements of data management systems. *Computing Surveys* 1, 2, 117-33. CR 10, 11(69)17,780.
Title is misleading. Actually a presentation of information structures in the context of data management systems.

Flores, I. (1969a) *Computer Sorting*. Prentice-Hall, Englewood Cliffs, N.J. CR 10, 1(69)16,053.
Presents various sorting techniques, with assembly language level procedures.

Flores, I. (1969b) *Computer Organization*. Prentice-Hall, Englewood Cliffs, N.J. CR 10, 11(69)17, 921.

- Flores, I. (1970) *Data Structure and Management*. Prentice-Hall, Englewood Cliffs, N.J. CR 12, 4(71)20,916.
An elementary treatment of data structure and management, using IBM software and hardware for examples.
- Galler, B., and Perlis, A. (1970) *A View of Programming Languages*. Addison-Wesley, Reading, Mass.
- Gauthier, R., and Ponto, S. (1970) *Designing Systems Programs*. Prentice-Hall, Englewood Cliffs, N.J. CR 12, 3(71)20, 829.
See particularly Ch. 7 (Data Representation) and Ch. 8 (Search Structures).
- Gear, C.W. (1969) *Computer Organization and Programming*. McGraw-Hill, New York. CR 10, 9(69)17,372.
A good text in assembly-level programming, which treats both hardware and basic software in this context.
- Gentle, E.C. Jr. (Ed.) (1965) *Data Communications in Business: An Introduction*. Publishers Service Co. New York. CR 7, 6(66) 10,837.
An introduction to the role and uses of data communications in business.
- Genuys, F. (Ed.) (1968) *Programming Languages*. Academic Press, New York.
- Gordon, G. (1969) *System Simulation*. Prentice-Hall, Englewood Cliffs, N.J. CR 11, 3(70)18,682.
- Griswold, R.E., Poage, J.F., and Polonsky, I.P. (1968) *The SNOBOL 4 Programming Language*. Prentice-Hall, Englewood Cliffs, N.J. CR 10, 11(69)17,858.
- Gruenberger, F. (Ed.) (1968a) *Computers and Communications—Toward a Computer Utility*. Prentice-Hall, Englewood Cliffs, N.J. CR 9, 6(68)14,439.
A collection of symposium papers.
- Gruenberger, F. (1968b) Program testing and validating. *Datamation* (July) 39 47.
- Gruenberger, F. (Ed.) (1969) *Critical Factors in Data Management*. Prentice-Hall, Englewood Cliffs, N.J. CR 11, 2(70)18,384.
A collection of symposium papers.
- Hassitt, A. (1967) *Computer Programming and Computer Systems*. Academic Press, New York. CR 8, 4(67)12,355.
Ch. 8 on debugging philosophy and Ch. 9 on the dynamic use of storage are most useful.
- Head, R.V. (1971) *A Guide to Packaged Systems*. Wiley, New York.
- Hellerman, H. (1967) *Digital Computer System Principles*. McGraw-Hill, New York. CR 9, 1(68)13,313.
Principles of digital computer systems, presented in such a way as to show common aspects of programming, machine design, and problem description.
- Hellerman, H. (1969) Some principles of time-sharing scheduler strategies. *IBM Systems J.* 8, 2, 94 117. CR 10, 9(69)17,501.
- Hopgood, F.R.A. (1969) *Compiling Techniques*. American Elsevier, New York. CR 10, 11(69)17,773.
Distinguishes between abstract data structures and internal storage structures. Ch. 2 (Data Structures), Ch. 3 (Data Structure Mappings), and Ch. 4 (Tables) are particularly relevant here.
- Husson, S. (1970) *Microprogramming: Principles and Practices*. Prentice-Hall, Englewood Cliffs, N.J.
Contemporary treatment of the implementation of computer control, using several existing processor designs as extended examples.
- IBM Corporation. (1969) *File Design Handbook*. Information Sciences Dept., IBM Research, San Jose, Calif.
A prototype file design handbook with special emphasis on providing equations, guidelines, and simulation data for use by the file designer in meeting user constraints on cost, storage capacity, response time, etc.
- IFIP. (1969) *File Organization*. Selected papers from File 68—an I.A.G. Conference. Swets and Zeitlinger N.V., Amsterdam, The Netherlands.
Papers ranging from the nature of management and information systems, and details of file structure design and programming support systems, through specific case studies.
- Iliffe, J.K. (1968) *Basic Machine Principles*. American Elsevier, New York.
Presents a computer system design in which many data handling and storage management functions typically "software" are defined in the hardware.
- Iverson, K.E. (1962) *A Programming Language*. Wiley, New York.
Contains considerable material on data structures, graphs, trees, and sorting, as well as descriptions of these in APL.
- Johnson, L.R. (1970) *System Structure in Data, Programs and Computers*. Prentice-Hall, Englewood Cliffs, N.J. CR 12, 1(71)20,504.
Systematic treatment of much of computer science, taking the tree as a basic structural element.
- Knuth, D.E. (1968) *The Art of Computer Programming, Vol. 1: Fundamental Algorithms*. Addison-Wesley, Reading, Mass. CR 9, 6(68)14,505.
An extensive compendium (Ch. 2, Information Structures) of information and techniques on data structures and storage management. Does not distinguish between modeling and implementation structures. Section 1.4 on subroutines, coroutines, and linking is must reading. Other related topics are thoroughly covered. Many good examples and exercises.
- Knuth, D.E., and Floyd, R.F. (1971) Notes on avoiding "go to" statements. *Information Processing Letters* 1, 1, 23 31.
- Lampson, B.W. (1968) A scheduling philosophy for multiprocessing systems. *Comm. ACM* 11, 5, 347-60. CR 9, 8(68)14, 980.
- Lawson, H.W. Jr. (1967) PL/I list processing. *Comm. ACM* 10, 6, 358 67.
- Lefkowitz, D. (1967) *File Structures for On-Line Systems*. Spartan Books, New York. CR 10, 7(69)17,049.
Presents alternative methods for file structure design and access, primarily in the context of information storage and retrieval systems.
- Lowe, T.C. (1968) The influence of data base characteristics and usage on direct access file organization. *J. ACM* 15, 4, 535-48.
A model for describing memory utilization and retrieval time for direct access inverted files as a function of characteristics of the data base and the usage pattern.
- McGee, W.C. (1968) File structures for generalized data management. *Proc. IFIP Congress 68*. CR 10, 4(69)16, 557.
Presents techniques for representing complex data structures as directed graphs and for making explicit declarations of graph-structured files.
- Madnick, S.E. (1967) String processing techniques. *Comm. ACM* 10, 7, 420-24.
Presents and evaluates six techniques for implementing strings in storage.
- Martin, J. (1967) *Design of Real-Time Computer Systems*. Prentice-Hall, Englewood Cliffs, N.J. CR 9, 2(68)13, 607.
A thorough treatment of analysis and design methodology for implementing real-time computer systems. Contains much material on both file and communications subsystems. Ch. 2, 3, and 10 on basic techniques and building blocks for these programs are quite good. Ch. 9 on the dynamic use of memory is also relevant.
- Martin, J. (1969a) *Telecommunications and the Computer*. Prentice-Hall, Englewood Cliffs, N.J. CR 11, 8(70)19, 602-19, 603.
See annotation in bibliography for Course Group D.
- Martin, J. (1969b) *Teleprocessing Network Organization*. Prentice-Hall, Englewood Cliffs, N.J. CR 11, 11(70)20, 210.
- Martin, J., and Norman, A.R.D. (1970) *The Computerized Society*. Prentice-Hall, Englewood Cliffs, N.J.
An imaginative presentation of present and prospective impacts of computer technology on many aspects of society.
- Meadow, C.T. (1967) *The Analysis of Information Systems*. Wiley, New York. CR 9, 8(68)14, 939.
Subtitled "A Programmer's Introduction to Information Retrieval." A thoughtful presentation from the standpoints of both library science and computer science.
- Mealy, G.H. (1967) Another look at data. *Proc. AFIPS 1967 FJCC*, Vol. 31, AFIPS Press, Montvale, N.J., 525-34.
Sketches a theory of data based on relations.
- Minker, J., and Sable, J. (1967) File organization and data management. In *Cuadra* (1967), Vol. 2, 123-60.
A report of then-recent developments in file organization and data management, organized in a tutorial and expository framework, with an extensive bibliography.
- Morgan, H.L. (1970) An interrupt based organization for management information systems. *Comm. ACM* 13, 12, 734-38. CR 12, 6(71)21, 403.
Describes a means of linking subprograms together by means of interrupts.

- Morris, R. (1968) Scatter storage techniques. *Comm. ACM* 11, 1, 38-44.
Surveys hashing schemes for symbol table algorithms. Presents analytic formulations of processing requirements.
- Parkhill, D. (1966) *The Challenge of the Computer Utility*. Addison-Wesley, Reading, Mass. CR 8 1(67)11, 053.
Discusses the history, technology, economic, and legal aspects of computer utilities.
- Randell, B., and Russell, L. (1964) *ALGOL 60 Implementation*. Academic Press, New York. CR 6, 5(65)8246.
- Rosen, S. (Ed.) (1967) *Programming Systems and Languages*. McGraw-Hill, New York. CR 10, 1(69)15,975.
Pt. 2 of this collection contains reprint articles on the major programming languages, and Pt. 3 articles on compiling and assembling. Pt. 4 contains papers of historical interest on various string and list processing languages. Pt. 5 is devoted to operating systems.
- Rosin, R.F. (1969) Supervisory and monitor systems. *Computing Surveys* 1, 1, 37-54. CR 10, 8(69)17, 284.
A survey of operating system development, tracing the evolution from the earliest crude monitor systems to the present.
- Ross, D.T. (1967) The AED free storage package. *Comm. ACM* 10, 8, 481-91. CR 9, 1(68)13,437.
Describes a storage allocation and management system for multilinked structures with heterogeneous fields and/or nodes (plexes).
- Salton, G.J. (1968) *Automatic Information Organization and Retrieval*. McGraw-Hill, New York. CR 10, 6(69)16,841.
Concentrates on automatic computer-based information retrieval systems. Includes a selective bibliography in information storage and retrieval and related topics.
- Sammet, J.E. (1969) *Programming Languages: History and Fundamentals*. Prentice-Hall, Englewood Cliffs, N.J. CR 10, 11(69)17, 854.
Ch. 4-9 describe a number of different programming languages and their data structures. In particular, Ch. 6 concerns string and list processing languages.
- Schorr, H., and Waite, W.M. (1967) An efficient machine-independent procedure for garbage collection in various list structures. *Comm. ACM* 10, 8, 501-06. CR 8, 6(67)13,179.
Reviews and compares past garbage collection algorithms and presents a new algorithm.
- Senko, M.E. (1969) File organization and management information systems. In Cuadra (1969), Vol. 4, 111-43.
A review of management information systems applications and structure, viewed from the standpoints of both the information scientist and the systems programmer. Includes an extensive bibliography.
- Stimler, S. (1969) *Real-Time Data-Processing Systems*. McGraw-Hill, New York. CR 10, 9(69)17,391.
A text on hardware configuration design, emphasizing file and communications subsystems. Develops and illustrates many evaluation and optimization techniques.
- Tucker, A.B., and Flynn, M.J. (1971) Dynamic microprogramming: processor organization and programming. *Comm. ACM* 14, 4, 240-50. CR 12, 8(71)21,784.
- Van Horn, E.C. (1968) Three criteria for designing computer systems to facilitate debugging. *Comm. ACM* 11, 5, 360-64. CR 9, 11(68)15,580.
- Walsh, D. (1969) *A Guide for Software Documentation*. Inter-ACT, McGraw-Hill, New York. CR 11, 7(70)19, 392.
Contains a number of models for the design of documentation procedures.
- Watson, R.W. (1970) *Time Sharing System Design Concepts*. McGraw-Hill, New York. CR 12, 12(71)22, 310.
A detailed exposition of the functional units of timesharing systems, with alternative design approaches.
- Wegner, P. (1968) *Programming Languages, Information Structures, and Machine Organization*. McGraw-Hill, New York. CR 10, 2(69)6,228.
There is a good section on coroutines, tasks, and asynchronous processing (4.10). Includes an extensive bibliography.
- Wilkes, M.V. (1967) The design of multiple-access computer systems. *Computer J.* 10, 1, 1-9. CR 9, 10(68)15,414.
- Wilkes, M.V., and Needham, R.M. (1968) The design of multiple-access computer systems: Pt. 2. *Computer J.* 10, 4, 315-320. CR 9, 10(68)15,415.
- Williams, R. (1971) A survey of data structures for computer graphics systems. *Computing Surveys* 3, 1, 1-21. CR 12, 7(71)21,621.
Includes an extensive bibliography.
- Wirth, N. and Hoare, C.A.R. (1966) A contribution to the development of ALGOL. *Comm. ACM* 9, 6, 413-32. CR 10, 1(69)15, 980.
Section 3.2.1 on the case construction provides some interesting thoughts on program design.
- Wirth, N. (1969) On multiprogramming, machine coding, and computer organization. *Comm. ACM* 12, 9, 489-98.
- Wirth, N. (1971) Program development by stepwise refinement. *Comm. ACM* 14, 4, 321-27. CR 12, 8(71)21, 630.
An interesting exposition of the program design process.

Appendix D. Detailed Descriptions and References for Course Group D

D1. Information Analysis (3-1-3)

Corequisite: A3.

Approach: This is the first course in the sequence of two that covers the system life cycle. This course emphasizes the information analysis and the logical design of the system, while course D2 covers the physical design. Emphasis should be placed on the iterative nature of the analysis and design process.

Exercises and case studies are used to give students proficiency in information analysis techniques; however, the projects course D3 which parallels this course is the vehicle for providing practical application in systems development and implementation. Field trips to organizations with sophisticated information systems are useful in reinforcing concepts.

Content:

- Introduction to the system life cycle (5%)**
Overview of the phases of system development and their interrelationships. Conception, information analysis, system design, programming, documentation, installation, reevaluation.
- System life cycle management (15%)**
Project control for system development. Levels of sophistication in system design. Responsibilities of system analysts, system designers, programmers, operators, and data processing management. Organizational behavior effects of system design and implementation approaches.
- Basic analysis tools (20%)**
Steps in analysis: preliminary investigation, general feasibility study, general system proposal, detailed analysis. Techniques for analysis, such as ARDI (Philips), BISAD (Honeywell), SOP (IBM). Event-oriented organizational flowcharts, decision tables, precedence network analysis.
- Determining system alternatives (15%)**
Manual systems. Manual versus automated parts of systems. Manager computer interaction requirements; response, performance, language—including "natural" language requirements. Generalized versus tailored output; graphic versus textual, and audio; inquiry versus automatic exception reporting; information retrieval versus specified analytical treatment of data. Disaggregation versus aggregation of data and hardware. Determining elements for common data bases. Data management alternatives. Response needs versus economic hardware/software and organizational constraints. Programmed decision making.
- Determining system economics (20%)**
Cost and value of information. Establishing measures of system performance: cost, response, accuracy, reliability, flexibility, security, capacity, quality, efficiency. Identifying and quantifying costs of system: personnel costs, equipment costs, conversion costs, installation costs. Identifying, quantifying, and measuring system advantages: direct and indirect benefits. Analyzing the improved quality of information. Allocation of costs and pricing of computer services.
- Defining logical system requirements (20%)**
Format of the system requirements statement. Distinction of logical design (of system) from physical design (of files, programs, and procedures). System output requirements: operating level, first level supervision, middle management, executive management. Information for strategic versus tactical planning and decision making. Specification of output methods and formats. System documentation requirements. System specification techniques: manual tech-

niques; semiautomated techniques: ADS (NCR), ISDOS (U. of Michigan), TAG (IBM), Young-Kent approach.

7. *Summary and introduction to physical system design* (5%)

References:

1. *Introduction to the system life cycle*
Benjamin (1971); Glans et al. (1968); Hartman et al. (1968); Nolan (1971); Rubin (1970a).
2. *System life cycle management*
Rubin (1970b); Shaw and Atkins (1970).
3. *Basic analysis tools*
Chapin (1971); Couger (1972); Farina (1970); Gatto (1964); Gildersleeve (1970); Gleim (1970); Hare (1967); Hartman et al. (1968); Honeywell (1968); McDaniel (1970a); Pollack et al. (1971).
4. *Determining system alternatives*
Amstutz (1967); Gildersleeve (1970); Martin (1965); Martin (1969); Olle (1970); Optner (1968); Rubin (1970c); Rubin (1970d).
5. *Determining system economics*
Emery (1971); Fisher (1971); Gregory (1963); Hurst (1969); Joslin (1971); Langefors (1970); Marschak (1971); Seiler (1969); Sharpe (1969).
6. *Defining logical system requirements*
Clifton (1970); Gray (1969); Hartman et al. (1968); IBM (1963); Laden and Gildersleeve (1963); Lyon (1971); National Cash Register (1967); Teichroew and Sayani (1971); Young and Kent (1958).
7. *Summary and introduction to physical system design*
Benjamin (1971); Blumenthal (1969); Hartman et al. (1968); Rubin (1970b).

Bibliography: The references are to the combined bibliography given at the end of Appendix D.

D2. System Design (3-1-3)

Prerequisites: C3, D1.

Corequisite: C4.

Approach: This course is the second covering the system life cycle, thus continuing the thrust of course D1. The lectures focus on underlying principles of system design as well as techniques. The techniques are utilized in the projects course D3. A theme to be carried throughout the course is the iterative nature of the analysis and design process. Implementation and conversion problems are also considered.

Case studies should be used as appropriate. Laboratory exercises should include the use of computer-assisted methods for system design. The human engineering aspects of system design should be emphasized.

Content:

1. *Basic design tools and objectives* (10%)
Review of the system life cycle. Documentation of various levels of design. Objectives of system design—system integrity. Types of system design: batch, interactive. Budgeting and project management.
2. *Hardware/software selection and evaluation* (15%)
Equipment selection—evaluation of hardware and software requirements. Automated evaluation techniques—simulation, analytical models. Detailed cost analyses: personnel, hardware, software. Competitive bidding.
3. *Design and engineering of software* (20%)
Design modularity. Design of user interfaces with automated procedures. Standardization of subsystem designs—data collection editing, processing, and retrieval. Design of subsystem interfaces. Data and production controls. Internal and external accounting within the system. Conversion subsystems. Human engineering.
4. *Data base development* (15%)
Data base construction—creation, structure, maintenance, and interrogation of data bases. Integrity of the data base. Review and use of C3 course material on data base management systems.
5. *Program development* (10%)
Selection of languages. Use of standard building blocks and common programs. Error recovery; robustness of programs. Programming standards and documentation. Review and use of C4 course material.
6. *System implementation* (10%)
Levels of testing and debugging; planning and executing conversion; management of programming, testing, and installation. Coordination of manual and automated procedures. Techniques for cutover (parallel operation, etc.); implementation schedules.
7. *Post implementation analyses* (10%)
Auditing system performance; costing of system development

effort and system performance. Evaluating hardware/software performance; "tuning" systems. Redesign cycle; modifying the system. Integrating changes into a running system.

8. *Long-range planning* (10%)

Trends in information system design. Integrating several systems into a corporate MIS. Planning for commonality and transferability of programs and data. Long-range forecasting of information requirements.

References: In addition to the references in the bibliography for the D courses, the references listed for courses C3 and C4 are particularly relevant for topics 4 and 5, respectively, of this course.

1. *Basic design tools and objectives*
Benjamin (1971) Sec. 4; Glans et al. (1968); Hartman et al. (1968); IBM (1966); Matthews (1971) Ch. 2, 3; Myers (1963); Shaw and Atkins (1970).
2. *Hardware/software selection and evaluation*
Couger (1972); Emery (1971); Fisher (1971); Gregory and Van Horn (1963); Head (1971); Joslin (1971); Martin (1965); Martin (1967); Martin (1969); Seiler (1969); Sharpe (1969) Sec. 4.
3. *Design and engineering of software*
Benjamin (1971); Martin (1965); Martin (1967) Sec. 6; Matthews (1971) Ch. 5 7; McDaniel (1970b); Rubin (1970a); Rubin (1970b); Teichroew and Sayani (1971).
4. *Data base development*
CODASYL (1971); Flores (1970); GUIDE/SHARE (1970); Gildersleeve (1971); Lyon (1971); Martin (1967) Ch. 22; Rubin (1970b).
5. *Program development*
Chapin (1971); Gray (1969); Martin (1965); Matthews (1971); Rosen (1967); Rubin (1970a); Rubin (1970b); Walsh (1969).
6. *System implementation*
Benjamin (1971) Sec. 9; Hartman et al. (1968); Martin (1965); Matthews (1971) Ch. 13; Shaw (1970).
7. *Post implementation analyses*
Benjamin (1971); Hartman et al. (1968); Martin (1965); Martin (1967); Matthews (1971) Ch. 11; Sutherland (1971).
8. *Long-range planning*
Emery (1969); Miller (1971); Morton (1971); Orlicky (1969).

Bibliography: The references are to the combined bibliography given at the end of Appendix D.

D3. Systems Development Projects (1-4-3)

Corequisite: D2

Approach: Students are assigned one or more system development projects. The projects involve the complete system development cycle: analysis, design, programming, and implementation. Students work in teams to acquire practical experience in such projects, especially regarding the behavioral considerations in systems development. They work with users to define system requirements and to prepare implementation plans and procedures.

The work parallels other courses in the final year of the degree program. If possible, it should be extended over two semesters, perhaps for additional credit. The information analysis portion of the project should begin in the first semester, as soon as the students are armed with sufficient capability to begin applying information analysis techniques. Projects should be completed and documented in the final month of the second semester. Thus, even if three hours of credit are granted only for the second semester, some of the work is done during the first.

Once a class has completed its project, the next class can expand on it, obtaining experience in the revision and sophistication of existing computer-based systems. The following are suggested alternatives for projects, the exact description of which will be dictated by circumstances.

Content:

1. *Development of a system for a local firm*
Under supervision of the systems analysis staff, students could develop a subsystem for one of the major modules of a computer-based management information system of a local firm. Students might also work as members of established client companies' teams.
2. *Development of a system for a university/college*
Under the supervision of the university administrative data processing unit, students could develop a system which would provide

them experience and at the same time benefit the university. Examples are: alumni record and follow-up system, bookstore ordering/accounting, classroom scheduling system.

3. *Development of a system for a hypothetical application.*

As an example, a case (SRA) currently available provides students with experience in each phase of system development for a hypothetical electronics firm. The material is organized into 13 assignments: orientation, documentation, written procedure, system flowcharts, gathering information, classification and coding, printed output source documents and punched cards, records design, data controls, run controls, audit trails, and file organization.

References: The entire set of references for the D courses are appropriate for this course. While many excellent cases exist for analysis of subtopics in information systems, there is a lack of in-depth cases. At considerable effort, an instructor can build a case by assembling a number of cases from the Intercollegiate Case Clearing House. The Intercollegiate Bibliography may be obtained for \$10.00 from I.C.C.H., Soldiers Field, Boston, MA 02163.

1. *Development of a system for a local firm*
Blumenthal (1969) Ch. 3-4.
2. *Development of a system for a university/college*
Johnson and Katzenmeyer (1969), Pt. 3.
3. *Development of a system for a hypothetical application*
Science Research Associates (1970).

Bibliography: The references are to the combined bibliography given at the end of Appendix D.

Combined Bibliography—Course Group D

Ackoff, R.L. (1970) *A Concept of Corporate Planning*. Wiley, New York.

Amstutz, A. (1967) *Computer Simulation of Competitive Market Response*. MIT Press, Cambridge, Mass. CR 9, 5(68)14,248.

Proposes an approach to policy management based on the use of microanalytic computer simulation. Describes how simulation-based computer systems can provide realistic artificial environments in which managers evaluate strategies.

Ansoff, H.I. (1965) *Corporate Strategy*. McGraw-Hill, New York.

Benjamin, R.I. (1971) *Control of the Information System Development Cycle*. Wiley, New York.

Develops a conceptual methodology for controlling the development of information systems. See also annotation in bibliography for Course A1.

Benton, W.K. (1971) *The Use of the Computer in Planning*. Addison-Wesley, Reading, Mass.

Blumenthal, S.C. (1969) *Management Information Systems: A Framework for Planning and Development*. Prentice-Hall, Englewood Cliffs, N.J. CR 10, 10(69)17, 647.

Bridges the gap between theory and practice by detailing the specific technical and organizational steps necessary to synthesize a comprehensive, integrated system plan for the enterprise. See also annotation in bibliography for Course A1.

Chapin, N. (1971) *Flowcharts*. Auerbach, Princeton, N.J. CR 12, 12(71)22, 295.

Covers program flowcharts, system flowcharts, computer-produced flowcharts, ANSI Standard flowcharts.

Clifton, D.H. (1970) *Systems Analysis for Business Data Processing*. Auerbach, Princeton, N.J. CR 12, 4(71) 20, 952.

An introductory book on system analysis and design.

Couger, J.D. (1972) *Systems Analysis Techniques*. Wiley, New York.

A collection of articles on system analysis techniques, describing approaches which concentrate on concepts and principles of system analysis.

CODASYL Systems Committee (1971) *Feature Analysis of Generalized Data Base Management Systems*. Technical report, available from ACM, New York.

See annotation in bibliography for Course Group C.

Emery, J.C. (1969) *Organizational Planning and Control Systems: Theory and Technology*. Crowell Collier and Macmillan, New York.

A technical approach, covering the system concept, the organization as a system, the technology of information systems, the economics of information, and planning and control.

Emery, J.C. (1971) *Cost/Benefit Analysis of Information Systems*.

The Society for Management Information Systems, Chicago. Provides foundation covering determination of value of information, cost of information, characteristics that govern value and cost, and techniques for making cost/benefit analyses.

Farina, M.V. (1970) *Flowcharting*. Prentice-Hall, Englewood Cliffs, N.J. CR 11, 7(70)19,469.

Introduces flowcharting by relating it to application in programming the BASIC language.

Fisher, G.H. (1971) *Cost Considerations in Systems Analysis*. American Elsevier, New York.

One of the first books to concentrate on the cost analysis aspects of system analysis. Uses examples in the Department of Defense. Written, however, so the reader can see the transferability to problems in transportation, health, public housing, and environmental resource planning.

Flores, I. (1970) *Data Structure and Management*. Prentice-Hall, Englewood Cliffs, N.J. CR 12, 4(71)20,916.

See annotation in bibliography for Course Group C.

Gatto, O.T. (1964) *Autosate*. *Comm. ACM* 7, 7, 425-32.

Describes an automated method for performing data gathering and the first stages of information analysis.

Gildersleeve, T.R. (1970) *Decision Tables and Their Practical Application*. Prentice-Hall, Englewood Cliffs, N.J. CR 12, 2(71)20, 655.

A programmed instruction introduction to decision tables.

Gildersleeve, T.R. (1971) *Design of Sequential File Systems*. Wiley, New York.

Covers the design of files and strategies for sequential storage media.

Glans, T.B., Grad, B., Holstein, D., Meyers, W.E., and Schmidt, R.N. (1968) *Management Systems*. Holt, Rinehart and Winston, New York.

A detailed treatment of the initial stages of the system life cycle—analysis and design of the system. Includes concepts first published by IBM under the name "Study Organization Plan."

GUIDE/SHARE. (1970) *Guide/Share Data Base Management System Requirements*. Technical report.

A well-written statement of requirements, emphasizing the importance and functions of the people in the system.

Gleim, G. (1970) *Program Flowcharting*. Holt, Rinehart and Winston, New York.

Covers USASI symbols and two levels of flowcharting: systems flowcharts, program flowcharts.

Gray, M., and London, K.R. (1969) *Documentation Standards*. Brandon/Systems Press, Princeton, N.J. CR 10, 9(68)17,373.

The first book developed exclusively for this subject: covers all the salient facts concerning documentation.

Gregory, R.H., and Van Horn, R.L. (1963) *Automatic Data Processing Systems*. Wadsworth, Belmont, Calif.

An introduction to data processing which includes chapters on system analysis, system design, and the value and cost of information.

Hare, V. (1967) *Systems Analysis: A Diagnostic Approach*. Harcourt Brace and Jovanovich, New York. CR 8, 5(67)12, 554.

A technical treatment of system analysis, useful for graduate level courses.

Hartman, W., Matthes, H., and Proeme, A. (1968) *Management Information Systems Handbook*. McGraw-Hill, New York.

A comprehensive coverage of the steps in system development, developed by the Netherlands-based Philips Corporation.

Head, R.V. (1971) *A Guide to Packaged Systems*. Wiley, New York.

Honeywell Corporation (1968) *Business Information System Analysis and Design*. Technical Report No. 144.0000.0000.0-954.

The system analysis and design approach advocated by Honeywell.

Hurst, E.G. Jr. (1969) Analysis for management decisions. *Wharton Quarterly* (winter).

IBM Corporation. (1963) *Study Organization Plan Documentation Techniques*. Technical Report No. C20-8075-0.

The system analysis and design approach advocated by IBM.

IBM Corporation (1966) *The Time Automated Grid System*. Technical Report No. Y20-0358-0.

Johnson, C.B. and Katzenmeyer, W.G. (Eds.) (1969) *Management Information Systems in Higher Education: The State of the Art*. Duke U. Press, Durham, N.C.

Joslin, E. (Ed.) (1971) *Analysis, Design and Selection of Computer Systems*. College Reading Inc., Arlington, Va.

A book of readings, from earlier published articles.

- Laden, H.N., and Gildersleeve, T.R. (1963) *Systems Design for Computer Applications*. Wiley, New York.
Covers techniques for design of sequential file systems.
- Langefors, B. (1970) *Theoretical Analysis of Information Systems, Vol. 1, 2*. Studentlitteratur Lund. Available from Barnes & Noble, New York, CR 11, 4(70)18, 831 (Vol. 1); 18, 832 (Vol. 2).
A technical treatment of systems analysis, useful for graduate level courses.
- Lyon, J.K. (1971) *An Introduction to Data Base Design*. Wiley, New York.
Concentrates on techniques in the design of online files.
- Martin, J. (1965) *Programming Real-Time Computer Systems*. Prentice-Hall, Englewood Cliffs, N.J.
Concentrates more on system design than programming aspects of online systems.
- Martin, J. (1967) *Design of Real-Time Computer Systems*. Prentice-Hall, Englewood Cliffs, N.J. CR 9, 2(68)13,607.
Continuation of his prior book, listed above. See also annotation in bibliography for Course Group C.
- Martin, J. (1969) *Telecommunications and the Computer*. Prentice-Hall, Englewood Cliffs, N.J. CR 11, 8(70)19,602, 19,603.
Technical aspects of the design of communication networks.
- Marschak, J. (1971) Economics of information systems. *J. American Statistical Association* 66, 192-219.
An important contribution towards formalizing, in a precise statistical manner, the concepts of cost and benefit of information.
- Matthews, D.Q. (1971) *The Design of the Management Information System*. Auerbach, Princeton, N.J. CR 12, 8(71)21, 668.
An introductory book on the MIS approach.
- McDaniel, H. (1970a) *Applications of Decision Tables—A Reader*. Brandon/Systems Press, Princeton, N.J. CR 12, 2(71)20, 613.
Examples of the use of decision tables, at the introductory level.
- McDaniel, H. (1970b) *Decision Table Software—A Handbook*. Brandon/Systems Press, Princeton, N.J., CR 12, 2(71)20, 613.
Examples of the use of decision tables at the introductory level.
- Miller, E.C. (1971) *Advanced Techniques for Strategic Planning*. AMA Research Study 104, American Management Assoc., New York.
- Morton, M.S.S. (1971) *Management Decision Systems*. Graduate School of Business Administration, Harvard U., Boston. CR 12, 6(71)21,367.
Presents results of academic experimentation in manager/computer interactive terminal systems.
- Myers, D.H. (1963) *A Time-Grid Techniques for the Design of Information Systems*. IBM Systems Research Institute, New York.
A description of this technique is also given in Kelly (1970), pp. 367-403, listed in the bibliography for Course A1.
- National Cash Register Company. (1967) *Accurately Defined Systems*. Dayton, Ohio.
The system analysis and design approach advocated by NCR.
- Nolan, R. L. (1971) Systems analysis for computer based information systems design. *Data Base* 3, 4, 1-10.
See annotation in bibliography for Course A3.
- Olle, T.W. (1970) MIS: data bases. *Datamation* (Nov.).
An excellent classification and characterization of file management systems, and how they fit into the world of management information systems.
- Optner, S. (1968) *Systems Analysis for Business Management*. Prentice-Hall, Englewood Cliffs, N.J.
Should be titled "Analysis of Systems for Management of Business." Includes cases.
- Orlicky, J. (1969) *The Successful Computer System: Its Planning, Development and Management in a Business Enterprise*. McGraw-Hill, New York. CR 10, 11(69)17,820.
Introduction to planning for the MIS.
- Pollack, S.L., Hicks, H.T. Jr., and Harrison, W.J. (1971) *Decision Tables: Theory and Practice*. Wiley, New York.
The theory and theorems of the decision table technique. Includes examples.
- Rosen, S. (Ed.) (1967) *Programming Languages and Systems*. McGraw-Hill, New York. CR 10, 1(69)15,975.
Readings which provide valuable historical perspective in the area of systems programming and the design of large scale operating systems.
- Rubin, M. (1970a) *Introduction to the System Life Cycle, (Vol. 1)*. Auerbach, Princeton, N.J.
Provides introductory level description of eight steps in the system life cycle.
- Rubin, M. (1970b) *System Life Cycle Standards, (Vol. 2)*. Auerbach, Princeton, N.J.
Provides standards, procedures and forms for system development.
- Rubin, M. (1970c) *Advanced Technology: Input and Output*. Auerbach, Princeton, N.J.
A reference for I/O approaches and design considerations.
- Rubin, M. (1970d) *Advanced Technology: Systems Concepts*. Auerbach, Princeton, N.J.
Introduction to systems analysis concepts.
- Science Research Associates (1970) *Case Study on Business Systems Design*. College Division, Palo Alto, Calif.
A laboratory manual providing thirteen assignments in developing an EDP system for a hypothetical electronics firm.
- Seiler, K. (1969) *Introduction to Systems Cost-Effectiveness*. Wiley, New York.
Provides general concepts; not directed specifically toward computerized systems.
- Shaw, J.C., and Atkins, W. (1970) *Managing Computer Systems Projects*. McGraw-Hill, New York, CR 12, 9(71)21,832.
Suggests approaches for managing the systems development effort.
- Sharpe, W.F. (1969) *The Economics of Computers*. Columbia U. Press, New York.
A theoretical treatment for decisions on selection, financing and/or use of computers. See also annotation in bibliography for Course A4.
- Sutherland, J.W. (1971) The configurator: today and tomorrow (Pt. 1); Tackle systems selection systematically (Pt. 2). *Computer Decisions*, (Feb., Apr.), 38-43, 14-19. CR 12, 7(71)21,521.
A two-part article on the use of simulation and analytical methods in the selection of a computer configuration.
- Teichroew, D., and Sayani, H. (1971) Automation of system building. *Datamation* (Aug. 15), 25-30. CR 12, 12(71)22,264.
- Young, J.W., and Kent, H. (1958) Abstract formulation of data processing problems. *J. Industrial Engineering* (Nov.-Dec.).
- Walsh, D. (1969) *A Guide for Software Documentation*. McGraw-Hill, New York. CR 11, 7(70)19,392.
Provides forms and procedures to follow when documenting the design and coding of a software system.

A Computer Science Course Program for Small Colleges

Richard H. Austing
University of Maryland
and
Gerald L. Engel
The Pennsylvania State University

The ACM Subcommittee on Small College Programs of the Committee on Curriculum in Computer Science (C³S) was appointed in 1969 to consider the unique problems of small colleges and universities, and to make recommendations regarding computer science programs at such schools. This report, authorized by both the subcommittee and C³S, supplies a set of recommendations for courses and necessary resources.

Implementation problems are discussed, specifically within the constraints of limited faculty and for the purposes of satisfying a wide variety of objectives. Detailed descriptions of four courses are given; suggestions are made for more advanced work; and an extensive library list is included.

Key Words and Phrases: computer science education, course proposals, small colleges, programming course, social implications course, computer organization course, file organization course, bibliographies

CR Categories: 1.52

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 the resources necessary for a major program in this field should refer to the existing reports of ACM's Committee on Curriculum in Computer Science (C³S) [87a] and other curriculum studies. Institutions which desire to complement computer science course offerings with a set of courses in computational mathematics should consider the report of the Committee on the Undergraduate Program in Mathematics [86d].

The Program

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

March 1972 and subsequently appeared in the IAG Journal under the title Computer Science Education in Small Colleges—A Report with Recommendations. Authors' addresses: Richard H. Austing, Computer Science Center, University of Maryland, College Park, MD. 20742; Gerald L. Engel, Computer Science Department, The Pennsylvania State University, University Park, PA. 16802; on leave from Hampden-Sydney College, Hampden-Sydney, VA 23368.

Copyright © 1973, Association for Computing Machinery, Inc. General permission to republish, but not for profit, all or part of this material is granted, provided that reference is made to this publication, to its date of issue, and to the fact that reprinting privileges were granted by permission of the Association for Computing Machinery.

The work reported here was supported in part by National Science Foundation grant GJ-1177 to the Association for Computing Machinery. A preliminary version was presented for discussion at the second annual SIGCSE Symposium, St. Louis, Missouri in

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.

Course 1 is the introduction, which in most cases gives a student his first experience in computer science. This is accomplished primarily by the presentation of a higher level programming language. Course 2 expands on Course 1 by giving the student further programming experience. In addition the student is introduced to a variety of applications of computers and the effects that these applications will have on the individual and on society. In Course 3, the student gains familiarity with various aspects of computer systems and how the parts of such systems interact. Finally, in Course 4 the concepts and applications of data representation and organization are considered.

Three of the courses (Courses 1, 3, and 4) correspond in basic content to courses in "Curriculum 68" [87a]. However, there is a good deal of difference in structure and emphasis in these courses from the way in which they are commonly taught. In order to allow as many students as possible to take the courses, the prerequisite structure is held to a minimum. Also, in order to provide a more general background, the courses (especially Courses 3 and 4) are more concerned with concepts of a particular system than with details or extensive programming exercises. For example, in Course 3 no particular assembler would be studied, but rather the general concept and vocabulary of computer systems would be presented. In this way a student, anticipating a career in business management, could equip himself with the tools to select a computer system without having to bury himself in the details of a particular system.

Course 2 does not have an equivalent in "Curriculum 68." This course in applications would, in most cases, be the natural sequel to the introductory programming course. It combines further experience in programming with a limited survey of application areas. Though programming would be an integral part of the course, something of the overall descriptive nature of the program would be involved. Where possible and appropriate, the students would be expected to use programs and data bases that are available. For example, if the class was studying simulation, it would be appropriate for the student to gain experience by using a computer-based traffic flow simulation to study the I/O problems involved, and possibly by studying some of the techniques involved in writing appropriate programs, but not necessarily by writing the program itself.

Implementation of this program would make instruction available to all students on campus at least at the level of being able to communicate intelligently with a computer. In addition advanced instruction would be readily accessible. For the student anticipating a career in computing or considering application for graduate work in computer science, several approaches are possible. Independent study courses can provide intro-

ductions to certain topics (courses in assembly language programming, programming languages, or even some large scale programming project would be appropriate). Also, since we are dealing with small schools, cooperation with other departments can be anticipated. Through this interdepartmental cooperation, certain courses can be modified to serve the student anticipating graduate work in computer science. Such a student should be advised to follow a mathematics curriculum, and could anticipate taking at least a computer oriented course offered by the mathematics department in probability and statistics, or a numerical analysis course, or a course in abstract algebra that would emphasize computer applications, or any of the computational mathematics courses recommended in the CUPM report [87b]. Finally, with the general introduction of computers in the undergraduate curricula as documented in the proceedings of the Conferences on Computers in the Undergraduate Curricula [86e], it seems reasonable to anticipate that an interested student can select several courses from various disciplines that make significant use of computers.

Implementation

One of the purposes of this program is to ensure its implementation with a minimal staff. Obviously, computing equipment must also be considered, and since most small schools are working under a small budget for computer services, the course structure reflects the fact that extensive computer power will probably not be available on campus. The courses recommended require that the students have access to a computer which has a higher level programming language for student use. Only one higher level language is required inasmuch as every computer installation satisfies that requirement. If additional languages are available, their use might be appropriate in one or more courses. Whether the computer is a small stand-alone or has one or more terminals makes little difference. The important requirement is that the students have easy access to the equipment and to student oriented software.

As important as the computer science course structure is, the most important area of computing at a small school is the service area. The cost of computing on campus, in terms of both equipment and personnel, can only be justified if computing services are used on a campus-wide basis. To achieve this, the development of a community of computer users on campus, as well as the excellent development of Course 1, is necessary and for schools that are not already involved in such programs, the first effort of the faculty member in charge of the development of computing must be made in these directions. The introduction of the additional course work should take place after these aspects of the program are completed.

The program requires one full-time instructor. In

most cases, Course 1 and Course 2 would be offered each semester, while Course 3 and Course 4 would be offered once each year. It is common practice in small schools to have the computer science faculty and computer center staff one and the same. It is clear that the demands of this program (at least nine hours per semester) plus the desirability of offering additional special courses at the more advanced level make this situation impossible. Thus the instructional staff and computer center staff should be separate. There should be, of course, a close relationship between the instructor and the center staff, but the instructor should have no administrative responsibilities in the center.

Another common practice in small schools is to take a faculty member from a department that is a computer user, and assign him the responsibility for computer science instruction. Such a practice often leads to the courses being not in computer science but rather in applications of computers. Whenever possible this should be avoided, but if it is necessary, the instructional material should be clearly separated from any other department of the school.

It is well to note that the present market situation places a small school in an excellent position to hire a computer scientist. Where possible this should be done, at least to the extent of bringing in the individual responsible for the implementation of the program. Where this cannot be done, a commitment should be made to allow an existing faculty member to develop himself in computer science education. Summer programs for this purpose are not plentiful, and doing such work in the normal environment of teaching and other responsibilities at the small school is close to impossible. Thus, where an existing faculty member is asked to be responsible for the program, it is strongly recommended that this faculty member be granted a year's leave of absence to work and gain experience in a computer science department. It is also recommended that universities with the facilities to do so develop programs that will help these faculty members to achieve their objective.

As with any program, the usual supporting facilities of the college are necessary. Though no great amount of specialized material is expected, it should be recognized that there will be a need for a rather large initial expenditure in the area of library materials, both books and periodicals. To provide a starting point for the development of a collection, a library list has been included.

Courses

There is much evidence that some exposure to computers should be an essential part of every college student's education. Many students will become users in their chosen occupations. Included in this group would be teachers, managers, researchers, and programmers who will need the computer as a tool. Other students will become directly involved in computer education and the computer industry. All students will be affected by the use of computers in our society.

As a minimum, students should acquire some understanding of the implications of the computer impact on individuals, organizations, and society. One way in which an academic institution can do this is to offer a survey type course in computers and society. However, there are some inherent difficulties with such an approach, particularly in schools which have no more than one or two faculty members in the computer science area. The breadth and amount of knowledge needed to give a worthwhile course of this type almost precludes its being offered by any one person. Developments and applications span such a wide range of areas that faculty from a variety of fields would need to be used. The course then might take on the flavor of a lecture series in which students would be presented a great deal of information but almost no feeling about what a computer is or how it should be used.

A better approach, as well as a more practical one in terms of faculty utilization, would consist of teaching fundamentals of computer science in a first course and allowing students the option of acquiring additional knowledge through their own reading, on-the-job training, or further course work in computer science or other disciplines. The first course described below follows this approach. It plays the role of a beginning course and the prerequisite course to each of the other three courses described. The latter three courses are designed not to be sequential. However, the most desirable path through them for students taking all of them would be in the order presented.

There is an intended overlap in the material of the four courses. Some ideas are worth repeating at different levels. Also, the same problem or concept can be enhanced by looking at it from different points of view or by bringing different material to bear on it.

Very few matters related to courses or curriculum are generally agreed upon among computer scientists. The question of what language to teach in a first course is no exception. Although there appears to be general agreement that a higher level language should be presented before an assembly language, there is a substantial difference of opinion regarding the specific language to use. APL, BASIC, FORTRAN, and PL/I, to name a few, each has a band of advocates. FORTRAN is still the most widely used general purpose language and is the most easily transferable from computer to computer. Despite its shortcomings, FORTRAN would seem to be the most

useful for the greatest number of students and is the language recommended for the first course. PL/I, if it is available, could be chosen in place of FORTRAN, particularly because its capabilities for nonnumeric applications make it useful in Courses 2 and 4. If strong reasons compel a different choice of language, some modifications might be necessary in course topics or approach. The introduction of and programming in a second language (e.g. ALGOL, APL, SNOBOL) is not recommended; it greatly decreases the programming experience and competence the student acquires in the first course. However, if time permits, a short discussion of a different kind of language and a demonstration program could be added at the end of the course.

Course 1 (3 Credits)

Introduction. This is a first course which emphasizes good programming techniques in a higher level language. No programming background is assumed. Upon completion of this course, the student: (a) should have practical experience in programming, including segmentation of both a problem and a program for its solution, debugging, implementation of basic data structures such as lists, and use of "canned" programs; (b) should know basic characterization of computer organization; (c) should be able to distinguish among program assembly, loading, compilation and execution, including some of the kinds of programming errors that can occur at each stage; and (d) should know the details of the language and have a basic idea of the relation of its statements to machine code.

The topics listed for this course do not differ substantially from the topics included in the outline of course B1 in "Curriculum 68"; however, a shift in emphasis is recommended. Course B1 stressed the notion of algorithm, problem analysis, and the formulation of algorithms for problem solution. Learning a language, practice in its use, and concepts of computer organization were also emphasized, but mainly as the means to obtain the actual solution of the problem. Unfortunately, no texts have appeared which have achieved the goal of presenting the subject of problem solving in an effective way (several books by Polya might be considered exceptions to this statement but they are not of the algorithmic orientation specified in course B1). Judging from the great variety found in introductory computing courses, it would seem that few, if any, teachers have been able to achieve the goal. It is not an easy problem to solve, but it is worth working toward a solution.

On the other hand, it is possible to teach programming techniques with the aid of a language manual and, possibly, one of the existing texts. The textbook could be used as a source of problems, at least, and in some cases, to supplement discussions of appropriate programming techniques applied to specific classes of problems. By concentrating on programming, the instructor is better able to teach a language, put it in

proper perspective with computer organizations and systems, develop good programming practices (including coding, debugging, and documentation), and motivate the need for algorithms in the solution process. Students should be required to use subprograms extensively (both their own and ones that are provided); this, in turn, would encourage at least one good problem solving technique—breaking up a problem into solvable parts.

An important benefit to the general approach suggested here is that the course is more easily defensible as a service course. Students could be urged to find problems in their own field of interest which they would program as course projects. Duplication of first courses for different groups of students could be minimized and, possibly, avoided entirely. For the first few semesters it might be difficult to obtain reasonable problems from a variety of areas, but as more faculty members become users, their fields of interest will become a source of good problems. In addition, a collection of (possibly large) data bases and subprograms can be accumulated and used as files to be referenced by student programs. The degree of success achieved by the computer center in developing a community of computer users has a significant influence here. As a result, some very interesting and nontrivial problems can be considered both in this course and in Course 2.

Though laboratory-like sessions for small groups of students may be desirable, they are not essential. If these sessions are used, an instructor may want to scatter them throughout the semester or bunch them at the beginning of the course and let the students program on a more individual basis toward the end of the course. Whether or not the laboratory sessions should be scheduled is a matter that is best decided by the instructor and/or the department.

Catalog Description. A first course in programming, using the FORTRAN language. Introductory concepts of computer organization and systems. Programming projects, including at least one from the student's field of interest.

Outline. Even though topics are listed sequentially, some topics (e.g. computer organization) should be distributed throughout the course with increasing degrees of detail. Problem analysis should be emphasized.

1. Overview of a computer. Basic computer modules, organization, and program execution. (5%)
2. Overview of problem solving process, beginning with the problem statement and ending with verification of the correct computer solution. (5%)
3. Introduction to the specific computer environment in which the student will work. Information needed by the student to interact with the computer in this course. (5%)
4. Language details. Components and types of assignment, control, and specification statements; data repre-

sentation and structures; storage allocation; i/o; subprograms; local and global variables; common and equivalence statements. (30%)

5. Programming techniques. Segmentation of problems and programs; comments and other documentation; debugging; library subroutines. (15%)

6. Simple data structures and list processing. Pointers; structures such as strings, stacks, linear and circular lists. (10%)

7. Limitations of FORTRAN. Nonnumeric programming; recursion. (5%)

8. Computer organization and systems. More detailed presentation of hardware and systems software, including registers, instruction codes, addressing, assembler, loader, compiler, and characteristics of components; peripheral units; past, present, and future developments. (20%)

9. Examinations. (5%)

Texts. A language manual, either the manufacturer's or one of the numerous manuals and primers that are available, should be used. Also, any local documentation concerning the installation's computer and/or systems should be readily available. No current book covers the material as presented in the outline, but parts of many books could be used as source material or student reference. For example, the following references are pertinent: 1, 3, 4, 11, 12, 15, 17, 22, 24, 28, 32, 34, 42, 59, 65, 66, 73, 77-80, 86a, 86d, and 88a-d.

Course 2

Introduction. This course emphasizes the use of computers in a variety of problem areas. It is an applications oriented course which should give the student concrete experience in solving representative problems of a practical nature. As in Course 1, large data bases can be established as experience in teaching the course is gained. Discussion of problems and problem areas should include algorithms, application techniques from Course 1, and social implications. New concepts and tools (e.g. complex data structures, tree search techniques, sorting methods) can be introduced as required in the context of specific problems, and the need for additional tools, including different kinds of languages, can be motivated. Occasionally, it might be feasible to invite a faculty member from another department or university or a local businessman to supplement material on a topic. Student assignments should vary, both in depth and in subject areas. In particular, a student who has completed Course 3 or 4 should be expected to use different techniques and solve larger or more difficult problems than a student who has completed only Course 1. Students should be encouraged to discover and solve problems in their own areas of interest.

Because students in this course have completed a programming course, no discussion should be necessary on such topics as what a computer is and how it works, number representation, flowcharts, and other elemen-

tary matters included in a computer appreciation-type course. However, a discussion of various systems (time-sharing, batch, etc.) should be included so that students are aware of the kinds of computer environments in which problems are solved.

The instructor should pose a suitably difficult problem in a real context, indicate possible approaches to its solution, break it up into smaller problems, discuss appropriate algorithms, introduce whatever new topics pertain to the problem, and let the student write a program to obtain the solution. If an entire problem is too difficult to solve in this way, one or more subproblems can be identified and handled as described. More advanced methods can be indicated when appropriate, and the student can be directed to appropriate references. Social and historical implications can be discussed at various stages of the solution process. As the course progresses, students should be expected to do more analysis and algorithm writing than specified above. The desired effects are that the student becomes acquainted with the computer's impact in a number of areas, is exposed to concepts and methods applicable to different kinds of problems, and gains practical experience in solving problems.

Catalog Description. Prerequisite, Course 1. Survey of computer applications in areas such as file management, gaming, CAI, process control, simulation, and modeling. Impact of computers on individuals and society. Problem solving using computers with emphasis on analysis. Formulation of algorithms, and programming. Projects chosen from various application areas including student's area of interest.

Outline. The selection and ordering of topics are highly dependent on the local situation. The topics are listed separately but should be combined as much as possible during discussion of problems. Problems and projects should have a practical flavor and should use a variety of computer oriented techniques and concepts. Attention should be given to the kind of technique that applies to a particular class of problems but not to other classes of problems. Each problem should be discussed in such a way that the student is aware of its relation to a real world context and sees the computer as a natural tool in the solution process.

1. Computer systems. Batch and interactive; real-time; information management; networks. Description of each system, how it differs from the others, and kinds of applications for which each system is best suited. (15%)

2. Large data bases. Establishment and use; data definition and structures. (10%)

3. Errors. Types; effects; handling. (5%)

4. Social implications. Human-machine interface, privacy; moral and legal issues. (15%)

5. Future social impact. Checkless society; CAI; national data bank. (10%)

6. **Languages.** Business oriented; list processing; simulation; string and symbol manipulation. Brief exposition of characteristics which make these languages appropriate for particular classes of problems. (10%)
7. Concepts and techniques used in solving problems from applications areas such as CAI, data management, gaming, information retrieval, and simulation. (25%)
8. Discussion of completed projects and/or examinations. (10%)

Texts. The italicized references cited below could serve as basic texts for this course. Many books and magazine articles could provide useful supplementary material either for class use or for student or teacher reference. Only a sampling of the available material is included in the Library List: 2, 3, 8, 9, 14, 15, 16, 19, 34, 36, 44, 56-59, 61, 63, 64, 68, 70, 72, 74, 76, 77, 85a, 85b, and 86a-e.

Course 3

Introduction. This course emphasizes the relationships between computer organization (hardware) and software. Each module's organization should be discussed, and its features should be related to the implementation of programming language features and assembly language instructions. Whenever possible, explanations should be included about why specific hardware features are better suited than others to certain types of problems or environments (e.g. real-time computing, interactive systems, data processing, scientific applications), and how this could affect selection of components. The effects of adding or changing modules should be viewed with respect to costs, capabilities, and software. Minicomputers should be discussed both as stand-alone computers and as components of larger systems.

Programming in assembly language should not be taught as such. However, students should be exposed to the use of macros and microprogramming. They should acquire a basic understanding of monitors, interrupts, addressing, program control, as well as implementation of arrays, stacks, and hash tables. In short, they should become familiar with assembly language concepts but in relation to their use in the total computer environment rather than through extensive programming. The need for assembly language programming experience is no longer great enough to argue that most students should have it. For those students who become interested in it, a special study course can be provided. With the background acquired in Course 3, a student should be able to gain programming experience without much additional guidance.

Catalog Description. Prerequisite, Course 1. Relationships among computer components, structures, and systems. Hardware features, costs, capabilities, and selection. Assembly language concepts and implementation.

Outline. Because this course is, at least to some extent, dependent on the specific computer available, the selection, ordering, and depth of coverage of topics will vary from institution to institution.

1. Processor. Arithmetic and control functions; relationships of features to language features; data handling; addressing. (20%)
2. Memory. Various types; cost, capabilities, and functions of each type; direct, random and sequential access; implementation of arrays, stacks, and hash tables. (20%)
3. I/O. Types, costs, and capabilities of units and media; control; channels; interrupts. (20%)
4. Communication among components. Effects of changing configurations; interactive and real-time systems. (5%)
5. Minicomputers. Capabilities as stand-alone computers; components of larger systems; costs. (10%)
6. Assembly language concepts. Instructions and their relations to components included above; macros, micro-programming. (20%)
7. Examinations. (5%)

Texts. No available text is suitable for this course. Material can be drawn from the following references and from manufacturers' manuals: 5, 6, 7, 13, 16, 26, 27, 29, 30, 31, 33, 35, 38-41, 43, 45-48, 53, 56, 60, 67, 69, 71, 74, 75, 81, 84, 86a, and 87b.

Course 4

Introduction. This is a course in file organization and manipulation. It stresses concepts, data structures, and algorithms used in the solution of non-numerical problems. Proper motivation for each should be given; an encyclopedia approach is not intended. Whenever several methods for achieving the same result are discussed (e.g. sorting or searching algorithms) comparative evaluations should be included. Differences between using core only and core plus auxiliary memory for various applications should be pointed out. If appropriate hardware is available, students should be assigned programming projects that require performing operations on large data bases and that require manipulating records on auxiliary memory devices. Immediate sources of problems are in the areas of mailing lists, registration, scheduling, student records, and library automation. If a suitable language for list processing applications is available, it could be taught and used in part of the course. Otherwise, characteristics of languages for this purpose should be given.

Catalog Description. Prerequisite, Course 1. Data structures, concepts and algorithms used in the solution of non-numerical problems. Applications to data management systems, file organization, information retrieval, list processing, and programming languages.

Outline. Neither mathematical applications nor mathematical properties of structures is included in this outline. They could become part of the course if students have sufficient background. Although some of the topics are discussed in Courses 1, 2 and 3, only the material in Course 1 is assumed.

1. Stacks, queues, arrays, lists. Structures; algorithms for manipulating, storage allocation and maintenance; applications. (25%)
2. Languages for list processing. Features of one or more languages (e.g. LISP, L⁶, PL/I). (5%)
3. Trees. Binary; threaded; traversal schemes; storage representation; applications. (15%)
4. Hash coding. Addressing; collisions; applications of symbol tables; storage allocation. (15%)
5. Searching and sorting. Comparison and evaluation of methods; techniques for use with auxiliary memory devices; applications. (15%)
6. Complex structures. Hierarchical; indexed sequential; inverted list; multilinked; applications to large information systems including case studies with illustrations of why they might not work. (20%)
7. Examinations. (5%)

Texts. A text for this course could be chosen from the italicized items included in the following list. However, the text would have to be supplemented with material from other references. *IO*, 20, 21, 24, 26, 28, 31, 37, 41, 44, 47, 49, 51, 54, 75, and 81a.

Additional Recommended Courses

The four courses described above are designed to service a broad segment of the undergraduate student body with an extremely limited number of faculty members, possibly one. Students should also have the opportunity to take computer-oriented courses in their own departments. The number of possible courses in this category is too great to try to list. Instead, we will recommend additional courses for the student who is seriously interested in computer science whether or not that student intends to pursue a graduate degree program in the field.

Each of the following specific courses could be given for special study to one or a few students or as a regular course if the demand is great enough and an instructor is available. Other topics could be included but might not be possible to implement in a practical way unless access to a large computer were available.

Assembly Language Programming. This course would enable a student interested in software to apply the concepts learned in Course 3; it provides a means to become experienced in assembly language programming and an introduction to systems programming. Desirable goals for this course include proficiency in assembly language programming, particularly using the system on hand; knowledge of basic principles of sys-

tems programming; and implementation of specific segments of systems programs (e.g. I/O routines). Manufacturer's manuals would initially serve as texts. The COSINE Committee's report, "An Undergraduate Course on Operating Systems Principles" (June 1971) provides a number of ideas for possible topics and references after the student acquires some programming experience.

Structure of Programming Languages. This course would include an introduction to grammars, languages they generate, scanners, recognizers, and other topics as time allows. Reference material for this course might include portions of *Compiler Construction for Digital Computers* by David Gries, *Ten Mini-Languages* by H.F. Ledgard or *A Comparative Study of Programming Languages* by E. Higman. Also the features of languages such as ALGOL and SNOBOL4 could be studied.

Programming Languages. If any language other than those included in courses is available, a special-study programming course may be appropriate. As part of this course, a student might be required to design and implement a major software project of some benefit either to the center or to the user community. Such a course might carry only one credit and it might be best given as a month-long course in schools on 4-1-4 system.

Library List

The following list is not exhaustive. No attempt was made to compile a list of all books on any specific topic. Certain areas are omitted entirely; namely, programming language manuals, books directed toward specific computers, and books primarily oriented toward use in other disciplines (such as numerical methods, computers and music, and programming for the behavioral sciences).

1. Arden, B.W. *An Introduction to Digital Computing*. Addison-Wesley, Reading, Mass., 1963.
2. Baer, R.M. *The Digital Villain*. Addison-Wesley, Reading, Mass., 1972.
3. Barrrodale, I., Roberts, F., and Ehle, B. *Elementary Computer Applications*. Wiley, New York, 1971.
4. Barron, D.W. *Recursive Techniques in Programming*. American Elsevier, New York, 1968.
5. Barron, D.W., *Assemblers and Loaders*. American Elsevier, New York, 1969.
6. Beizer, B. *The Architecture and Engineering of Digital Computer Complexes*. Plenum Press, New York, 1971.
7. Bell, C.G., and Newell, A. *Computer Structures: Readings and Examples*. McGraw-Hill, New York, 1971.
8. Bemer, R.M. (Ed.) *Computers and Crisis*. ACM, New York, 1971.
9. Benice, D.D. (Ed.) *Computer Selections*. McGraw-Hill, New York, 1971.
10. Bertiss, A.T. *Data Structures: Theory and Practice*. Academic Press, New York, 1971.
11. Brooks, F., and Iverson, K. *Automatic Data Processing*. Wiley, New York, 1969.
12. Cole, R.W. *Introduction to Computing*. McGraw-Hill, New York, 1969.
13. Cuttle, G., and Robinson, P.B. (Eds.) *Executive Programs and Operating Systems*. American Elsevier, New York, 1970.
14. Davenport, W.P. *Modern Data Communications*. Hayden, New York, 1971.
15. Desmonde, W.H. *Computers and Their Uses*. Prentice-Hall, Englewood Cliffs, N.J., 1971.
16. Dippel, G., and House, W.C. *Information Systems*. Scott, Foresman, Chicago, 1969.

17. Dorf, R.C. *Introduction to Computers and Computer Science*. Boyd and Fraser, San Francisco, 1972.
18. Elson, M. *Concepts of Programming Languages*. Science Research Associates, New York. In press.
19. Feigenbaum, E.A., and Feldman, J. (Eds.) *Computers and Thought*. McGraw-Hill, New York, 1963.
20. Flores, I. *Sorting*. Prentice-Hall, Englewood Cliffs, N.J. 1969.
21. Flores, I. *Data Structures and Management*. Prentice-Hall, Englewood Cliffs, N.J., 1970.
22. Forsythe, A.I., Keenan, T.A., Organick, E.I., and Stenborg, W. *Computer Science: A First Course*. Wiley, New York, 1969.
23. Foster, J.M. *List Processing*. American Elsevier, New York, 1967.
24. Galler, B.A. *The Language of Computers*. McGraw-Hill, New York, 1962.
25. Galler, B.A., and Perlis, A.J. *A View of Programming Languages*. Addison-Wesley, Reading, Mass. 1970.
26. Gauthier, R., and Ponto, S. *Designing Systems Programs*. Prentice-Hall, Englewood-Cliffs, N.J., 1970.
27. Gear, C.W. *Computer Organization and Programming*. McGraw-Hill, New York, 1969.
28. Gear, C.W. *Introduction to Computer Science*. Science Research Associates, New York, In press.
29. Genuys, F. (Ed.) *Programming Languages*. Academic Press, New York, 1968.
30. Gordon, G. *System Simulation*. Prentice-Hall, Englewood-Cliffs, N.J. 1969.
31. Gries, D. *Compiler Construction for Digital Computers*. Wiley, New York, 1971.
32. Gruenberger, F. *Computing: An Introduction*. Harcourt Brace and Jovanovich, New York, 1969.
33. Gruenberger, F. *Computing: A Second Course*. Canfield Press, Cleveland, Ohio, 1971.
34. Gruenberger, F., and Jaffray, G. *Problems for Computer Solution*. Wiley, New York, 1965.
35. Gschwind, H.W. *Design of Digital Computers, An Introduction*. Springer-Verlag, New York, 1970.
36. Hamming, R. W. *Computers and Society*. McGraw-Hill, New York, 1972.
37. Harrison, M.C. *Data Structures and Programming*. Courant Institute of Mathematical Sciences, New York U., New York, 1970.
38. Hassitt, A. *Computer Programming and Computer Systems*. Academic Press, New York, 1967.
39. Hellerman, H. *Digital Computer System Principles*. McGraw-Hill, New York, 1967.
40. Higman, B. *A Comparative Study of Programming Languages*. American Elsevier, New York, 1967.
41. Hopgood, F.R.A. *Compiling Techniques*. American Elsevier, New York, 1969.
42. Hull, T.E., and Day, D.D.F. *Computers and Problem Solving*. Addison-Wesley, Don Mills, Ontario, Canada, 1970.
43. Husson, S. *Microprogramming: Principles and Practice*. Prentice-Hall, Englewood Cliffs, N.J., 1970.
44. IFIP. *File Organization*, selected papers from File 68—an I.A.G. Conference. Swets and Zeitinger N.V., Amsterdam, 1969.
45. Iliffe, J. K. *Basic Machine Principles*. American Elsevier, New York, 1968.
46. Iverson, K. *A Programming Language*. Wiley, New York, 1962.
47. Johnson, L.R. *System Structure in Data, Programs and Computers*. Prentice-Hall, Englewood Cliffs, N.J., 1970.
48. Katzan Jr., H. *Computer Organization and the System/370*. Van Nostrand Rheinhold, New York, 1971.
49. Knuth, D. *The Art of Computer Programming, Vol. 1, Fundamental Algorithms*. Addison-Wesley, Reading, Mass., 1969.
50. Knuth, D. *The Art of Computer Programming, Vol. 2, Seminumerical Algorithms*. Addison-Wesley, Reading, Mass. 1969.
51. Knuth, D. *The Art of Computer Programming, Vol. 3, Sorting and Searching*. Addison-Wesley, Reading, Mass., In press.
52. Korfhage, R. *Logic and Algorithms with Applications to the Computer and Information Sciences*. Wiley, New York, 1966.
53. Laurie, E. J. *Modern Computing Concepts—The IBM 360 Series*. Southwestern, Cincinnati, Ohio, 1970.
54. Lefkowitz, D. *File Structures for On-Line Systems*. Wiley, New York, 1967.
55. Martin, J. *Design of Real-Time Computer Systems*. Prentice-Hall, Englewood Cliffs, N.J., 1967.
56. Martin, J. *Telecommunications and the Computer*. Prentice-Hall, Englewood Cliffs, N.J., 1969.
57. Martin, J. *Introduction to Teleprocessing*. Prentice-Hall, Englewood Cliffs, N.J., 1972.
58. Martin, J., and Norman, A.R.D. *The Computerized Society*. Prentice-Hall, Englewood Cliffs, N.J., 1970.
59. Maurer, H.A., and Williams, M.R. *A Collection of Programming Problems and Techniques*. Prentice-Hall, Englewood Cliffs, N.J., 1972.
60. Maurer, W.D. *Programming: An Introduction to Computer Languages and Techniques*. Holden-Day, San Francisco, 1972.
61. Meadow, C. *The Analysis of Information Systems*. Wiley, New York, 1967.
62. Minsky, M. *Computation: Finite and Infinite Machines*. Prentice-Hall, Englewood Cliffs, N.J., 1967.
63. Oettinger, A.G., and Marks, S. *Run Computer Run*. Harvard U. Press, Boston, 1969.
64. Parkhill, D. *The Challenge of the Computer Utility*. Addison-Wesley, Reading, Mass., 1966.
65. Ralston, A. *Introduction to Programming and Computer Science*. McGraw-Hill, New York, 1971.
66. Rice, J. K., and Rice, J. R. *Introduction to Computer Science: Problems, Algorithms, Languages, Information and Computers*. Holt, Rinehart and Winston, New York, 1969.
67. Rosen, S. (Ed.) *Programming Languages and Systems*. McGraw-Hill, New York, 1967.
68. Rothman, S., and Mosmann, C. *Computers and Society*. Science Research Associates, New York, 1972.
69. Sammet, J. E. *Programming Languages: History and Fundamentals*. Prentice-Hall, Englewood Cliffs, N.J., 1969.
70. Sanders, D. *Computers in Society: An Introduction to Information Processing*. McGraw-Hill, New York, In press.
71. Sayers, A.P. (Ed.) *Operating Systems Survey*. Auerbach Corp., Princeton, N.J., 1971.
72. Sprague, R.E. *Information Utilities*. Prentice-Hall, Englewood Cliffs, N.J., 1970.
73. Sterling, T.D., and Pollack, S.V. *Computing and Computer Science*. Macmillan, New York, 1970.
74. Stimler, S. *Real-Time Data-Processing Systems*. McGraw-Hill, New York, 1969.
75. Stone, H.S. *Introduction to Computer Organization and Data Structures*. McGraw-Hill, New York, 1972.
76. Taviss, I. *The Computer Impact*. Prentice-Hall, Englewood Cliffs, N.J., 1970.
77. Teague, R. *Computing Problems for FORTRAN Solution*. Canfield Press, Cleveland, Ohio, 1972.
78. Trakhtenbrot, B. A. *Algorithms and Automatic Computing Machines*. D.C. Heath, Boston, 1963.
79. Walker, T. *Introduction to Computer Science: An Interdisciplinary Approach*. Allyn and Bacon, Boston, 1972.
80. Walker, T., and Cotterman, W.W. *An Introduction to Computer Science and Algorithmic Processes*. Allyn and Bacon, Boston, 1970.
81. Watson, R.W. *Timesharing System Design Concepts*. McGraw-Hill, New York, 1970.
82. Wegner, P. *Programming Languages, Information Structures and Machine Organization*. McGraw-Hill, New York, 1968.
83. Weingarten, F. *Translation of Computer Languages*. Holden-Day, San Francisco. In press.
84. Wilkes, M.V. *Time-Sharing Computer Systems*. American Elsevier, New York, 1968.
85. In addition to the above list, several collections of articles originally appearing in *Scientific American* have been published in book form by W.H. Freeman, San Francisco. Specifically, they are
 - a. *Information*, 1966.
 - b. *Computers and Computation*, 1971.
86. Various conference proceedings, journals, bulletins, and the like, should also be maintained in a library collection. The following are of special interest:
 - a. *Communications of the ACM* (monthly); *Computing Reviews* (monthly); *Computing Surveys* (quarterly); *Proceedings, ACM National Conference* (yearly); *SIGCSE Bulletin* (ACM's Special Interest Group-Computer Science Education); *SIGCUE Bulletin* (ACM's Special Interest Group-Computer Uses in Education); *SIGUCC Bulletin* (ACM's Special Interest Group-University Computing Centers). (Information on these publications may be obtained from

- ACM Headquarters Office, 1133 Avenue of the Americas, New York, NY 10036.)
- b.) Proceedings, AFIPS Fall Joint Computer Conference (yearly); Proceedings, AFIPS Spring Joint Computer Conference (yearly). (Available from AFIPS Press, 210 Summit Avenue, Montvale, NJ 07645.)
 - c. Proceedings, IFIP Congress (every three years). (Available through North-Holland, P.O. Box 3489, Amsterdam.)
 - d. Proceedings, IFIP World Conference on Computer Education. (Distributed by Science Associates/International, New York.)
 - e. Proceedings, Conference on Computers in the Undergraduate Curriculum, 1970-1-2. (Available through Southern Regional Education Board, Atlanta, GA 30313.)
87. The following curriculum reports are relevant to computer science education:
- a. Curriculum 68—Recommendations for academic programs in computer science. *Comm. ACM 11* (Mar. 1968), 151-197.
 - b. An undergraduate course on operating systems principles. COSINE Committee Report, June 1971. (Available from Commission on Education, National Academy of Engineering, 2101 Constitution Avenue, N.W., Washington, DC 20418.)
 - c. Curriculum recommendations for graduate programs in information systems. Report of the ACM Curriculum Committee on Computer Education for Management, *Comm. ACM 15* (May 1972), 363-398.
 - d. Recommendations for an undergraduate program in computational mathematics. Committee on the Undergraduate Program in Mathematics May 1971. (Available from CUPM, P.O. Box 1024, Berkeley, CA 94701.)
88. The following statistical reports provide information on the status of computing as obtained from recent surveys:
- a. Hamblen, J.W. *Computers in Higher Education: Expenditures, Sources of Funds and Utilization for Research and Instruction: 1964-65 with Projections for 1968-69*. (1967) 325 pp. (Available through Southern Regional Education Board, Atlanta, GA 30313.)
 - b. Hamblen, J.W. *Inventory of Computers in U.S. Higher Education 1966-67: Utilization and Related Degree Programs*. (1970) 400 pp. (Available through Superintendent of Documents, U.S. Government Printing Office, Washington, D.C.)
 - c. Hamblen, J.W. *Inventory of Computers in U.S. Higher Education 1969-70: Utilization and Related Degree Programs*. (1972) 400 pp. (Available through Superintendent of Documents, U.S. Government Printing Office, Washington, D.C.)
 - d. Engel, G.L. Computer science instruction in small colleges—An initial report. *SIGCSE Bull.* 3, 2 (June 1971), 8-18.

A Report of the ACM Curriculum Committee on Computer Education for Management

Curriculum Recommendations for Undergraduate Programs in Information Systems

J. Daniel Couger
Editor

The need for education related to information systems in organizations is discussed, and a curriculum is proposed for an undergraduate program. Material necessary for such programs is identified, and courses incorporating it are specified. Detailed course descriptions are presented. Program organization and problems of implementation are discussed.

Key Words and Phrases: education, undergraduate curricula, management systems, information systems, information analysis, system design, systems analysis

CR Categories: 1.52, 3.51

Preface

This report contains recommendations for undergraduate curriculum in information systems, prepared by the ACM Curriculum Committee on Computer Education for Management (C³EM).

The need for degree programs in information systems was documented in the Committee's position paper [1]. Comprehensive curriculum recommendations for a graduate-level program in information systems development were presented in a further report [2]. The present report gives recommendations for undergraduate-level programs, based on the same

Copyright © 1973, Association for Computing Machinery, Inc. General permission to republish, but not for profit, all or part of this material is granted, provided that ACM's copyright notice is given and that reference is made to the publication, to its date of issue, and to the fact that reprinting privileges were granted by permission of the Association for Computing Machinery.

The work of the Committee was supported in part by National Science Foundation Grant GJ-356.

general concept of the information systems speciality in organizations. Two subspecialities for undergraduate concentration are presented, the emphasis being characterized either as organizational or as technological.

A draft version of this report was circulated for review to members of the academic and professional community. Representatives of government, industry, and educational institutions participated in a two-day review of the curriculum at the University of Colorado, Colorado Springs, November 16-17, 1972.

Many suggestions and constructive criticisms from both the review process and the meeting have been incorporated in the report. The Committee is extremely grateful to the reviewers and the meeting participants, and to others whose assistance was helpful. Their names are listed at the end of the main body of the report. The Committee, of course, undertakes full responsibility for the substance of the report and the conclusions and recommendations contained in it.

Chairman of the Subcommittee on Undergraduate Curricula was J. Daniel Couger (University of Colorado). Other members of the C³EM Committee who participated in the preparation of this report are: Daniel Teichroew (University of Michigan), Chairman, Russell M. Armstrong (HBR-Singer Co.), Robert L. Ashenhurst (University of Chicago), Robert I. Benjamin (Xerox Corporation), Gordon B. Davis (University of Minnesota), John F. Lubin (University of Pennsylvania), James L. McKenney (Harvard University), Howard L. Morgan (University of Pennsylvania), and Frederic M. Tonge Jr. (University of California, Irvine).

1. Introduction

The basic recommendations for a master's level program in information systems development were formulated in terms of a single program comprehending both information analysis and system design [2, Sec. 2.4]. The recommendations for undergraduate programs for information systems specialists, however, are presented in terms of two concentration options.

1.1. Organizational Concentration

This option is designed to prepare a person to be an effective computer user. The undergraduate student, therefore, combines information systems course work with the academic area of emphasis in a field of application, such as business or government. With the five-course option in information systems, the student essentially has a double major with, for example, marketing or political science or hospital administration. Upon entering a career field, the graduate will be able to participate effectively on a systems development team of users and practitioners. Until recently, an additional option would have extended a student's program beyond the normal four-year requirement. Today most programs allow enough electives to accommodate easily a second area of emphasis.

1.2. Technological Concentration

This option is designed to prepare a person for an entry-level job in an information processing department. The graduate would typically begin as a programmer and, through practical experience and advanced education, qualify to move into the area of logical and physical system design.

1.3. Structure of Presentation

Section 2 of this report provides a brief description of the information systems field. Section 3 lists the output characteristics of graduates of the program and also prerequisites incorporated into the program. Section 4 provides course descriptions which are integrated into the specific program concentrations in Section 5. Section 6 discusses implementation of the program, in either or both of the concentration options. The Appendix provides detailed course descriptions and supporting bibliographies.

2. Background on the Information Systems Specialty

The previously cited reports of the Committee provide the background for degree programs in information systems. The position paper [1] gives the justification for the degree programs, citing surveys on demand for such persons and the kinds of capabilities required of practitioners in the field. The curriculum report [2] provides detailed course descriptions for

graduate-level professional programs. The reader is encouraged to obtain copies of those papers as a background for the undergraduate curriculum recommendations.

2.1. The Information Systems Field

For purposes of this report, as in [2], the information systems development process is viewed as consisting of analysis, design, and implementation phases, prior to the operation phase. These phases do not ordinarily take place strictly in the order given but rather exist together in a continuing pattern of interaction.

Analysis and design proceed in steps together, each affecting the other. An operation phase follows successful implementation, but analysis, design, and implementation activities generally continue as the system is modified and eventually supplanted.

Implementation involves writing and debugging programs, gathering information for data bases, training personnel who will use, operate, and maintain the system, and finally installation and checkout.

Operation involves the routine running of the system and is thereby appropriately the function of an information processing department.

The analysis and design functions, however, are pure developmental activities, and it is here that system inadequacies often have their roots. Such inadequacies may stem from failure to achieve a proper balance between organizational and technological factors, each of which is subject to continuing change.

To highlight the need for balance between these two sets of factors, the analysis and design phase of systems development is explicitly recognized as consisting of two activities: information analysis and system design.

The main emphasis in information analysis is the determination of information needs and patterns of information flow which will satisfy these needs. This requires interaction with organizational personnel and a good understanding of how the organization functions.

The main emphasis in system design is the translation of specified information requirements into a detailed implementation plan which can be realized in hardware/software. This requires interaction with the information processing department and a good understanding of computer technology.

The terms logical system design and physical system design are sometimes used to differentiate between the specification of the information system itself and its implementation in hardware/software.

Both of these phases concentrate on the system, whereas information analysis concentrates on the organization. Two phases of information analysis may also be distinguished—analysis of information needs and analysis of how they may be satisfied in terms of requirements on an information system. The two phases of information analysis are sometimes called "feasibility study" and "system specification."

The development of information systems then con-

sists of an iterated process of information analysis, system design, and implementation. This "system life cycle," it has been pointed out, applies to other kinds of development effort as well.

2.2. Growth of the Information Systems Field

In its position paper [1] the Committee summarized information on education, employment, and future needs of personnel in information systems. Since then a number of papers and studies have been published, and it is worthwhile at this point to summarize recent results, particularly their effect on the question of whether undergraduate programs are needed, and if so, what their content should be.

Considerable attention has been paid recently to job classifications and career paths. Part of the concern arises from the status of programmers and systems analysts under the Fair Labor Standards Act. Amendments to the Act were published in the Federal Register, Vol. 36, No. 232, Thursday, December 2, 1971, following an open hearing held February 2-11, 1971. A new society, The Association of Computer Programmers and Analysts, has been formed and is involved in and concerned with standardizing position qualifications. An attempt to relate data processing training to job requirements is given by Hammond [3].

A university degree may not have been necessary for a position in information systems in the past, but informal surveys show that a college degree is an implicit if not explicit requirement for information systems positions in medium to large size companies using third generation equipment.

A few years ago the shortage of qualified applicants for data processing positions was commonly accepted. Manpower estimates by the Bureau of Labor Statistics [4, 5] show annual average openings for system analysts at 27,000 and programmers at 23,000 for the period 1968-80. In the last few years the shortage of personnel is much less in evidence, and some of the urgency seems to have gone out of the need for academic programs. Hamblen [6], for example, concludes: "When the figures of supply are compared with estimates of demand we see that there is no longer a need to encourage a crash effort to start new degree programs at any level. However, if we examine the course offerings of the associate and bachelor's degree programs, in particular, as I have had occasion to do in the two NSF Inventories, there is definitely a need to strengthen these programs both in facilities available and course offerings."

Estimates of the number of openings in various occupations based on the Bureau of Labor reports are summarized in Table I. These figures show that the number of programmers and systems analysts needed annually is approximately the same as that for physicians, engineers, and accountants—for which established college programs have existed for some time.

From the above survey results it is evident that the requirement for analysts and programmers is far from

Table I. Growth in Selected Occupations 1968-1980*

Occupational group	1968 Employment	Percent Growth Forecast 1970-1980	Net increase in occupation	Average annual openings not including transfers
All occupational groups	75,920,000	25.3	19,100,000	
All professional and technical	10,325,000	50.1	5,175,000	
Programmers	175,000	129	200,000	23,000
Engineers	1,100,000	40.2	400,000	73,400
Accountants	500,000	43.4	220,000	33,200
Systems Analysis	150,000	183	275,000	27,000
Physicians	295,000	53.1	155,000	20,000

* Source: Bureau of Labor Statistics Reports [4, 5]

being satisfied by graduates of available academic programs. It is unnecessary, perhaps, to start a number of new programs. However, the committee fully agrees with Hamblen that many of the existing ones warrant improvement. This report is an attempt to provide a framework for improving programs and courses. The analysis supports the conclusion that different-level education will be required for the various occupations. In particular, many more graduates with BS degrees will be needed than graduates with MS degrees.

2.3. Need for Differing Educational Levels

Basically, two levels of education concerning the computer are required.

1. A minimal level is required for persons who are *users* of the results of computer processing. This level could be provided through one to four courses, depending upon the using activity. For example, an introduction to information systems might be sufficient for a manager who is communicating with the computer via a terminal. More depth of understanding is required of the user representative on the team which designs the reservations system.

2. At the other end of the educational spectrum is the person who is preparing for a career as an *information systems specialist*. Again, the education required for the professional depends upon the position. The educational level required of a person preparing computer programs is less than that of the system designer.

1. **Education for Users.** Earlier work of the Committee reported the status of education for users of the results of computer processing. The report "The State of Computer Oriented Curricula in Business Schools in the 70's" [8] indicated a disparity in curriculum content: "Various positions were expressed ranging from a passing acquaintance with computer information systems, in order to avoid being taken in by experts and enthusiasts, to the view that future managers would be their own specialists and therefore need much of the training appropriate for specialists."

A four-phase curriculum is in existence in the majority of member schools of the American Association

of Collegiate Schools of Business, according to a survey by the *Computing Newsletter for Schools of Business* [9].

1. Coverage of computer fundamentals, systems analysis, and design and programming through a course required of all students in their academic program.

2. Coverage of the applications of computers through incorporation of this material into the functional area courses, e.g. computer applications in finance in the finance courses, and computer applications in marketing in the marketing courses.

3. Coverage of computer capabilities for abetting decision making in a dynamic business environment through computer-oriented business games.

4. Coverage of integration and optimization of computer applications through a course on design and implementation of a sophisticated, computer-based management information system.

Such an approach should adequately prepare computer users who received their education through schools of business. A similar approach is needed in other user areas, such as political science and hospital administration.

2. Education for Information System Specialists.

Formal education for computer operators and applications programmers is provided through a two-year program, such as those offered by the community colleges. Other positions such as the information analyst and the system designer required advanced education. This view is consistent with those expressed in the recommendations of the National Advisory Committee for Computer Curriculum of the American Association of Junior Colleges in its report "The Computer and the Junior College: Curriculum" [10].

The master's level curriculum [2] provides the depth of education necessary for the systems development function. This report recommends curricula to provide entry-level qualifications.

From a practical standpoint, the ratio of bachelor's to master's candidates dictates the need for undergraduate-level education in information systems. With experience and advanced education, the BS graduate may qualify for the system designer position.

2.4. Approach to the Specifications for Undergraduate Curricula

The basic framework of the recommendations for graduate professional programs is used as a context for specifying undergraduate programs. There are, however, differences not only in level but in program objectives and in envisioned program implementation environments. The approach to the graduate curriculum was to present a single "standalone" program based on 13 specified courses. Using this course material as a base, modified programs were presented for adoption by business schools, computer science, or other departments which have a less comprehensive emphasis in the development area. The modified programs involved the

definition of three additional courses, containing certain pairs of the 13.

Entry-level positions for people with undergraduate degrees are expected to be less demanding, in terms of the knowledge and abilities required. On the other hand, the fact that an undergraduate program is necessarily less concentrated on the "major" subject makes it more difficult to be comprehensive in the coverage of the information systems field, even in more condensed form. Accordingly, instead of a single program of information systems courses, two concentration options are distinguished, labeled "organizational" and "technological." These two terms correspond to the terms "information analysis" and "system design." Each concentration option is specified in terms of a set of core courses (seven in the case of organizational, eight in the case of technological). The two sets of core requirements share four courses in common, so both concentrations can be offered by implementing a total of 11 courses. These courses are described in detail in Section 4 and in the Appendix.

The contents of the two sets of core courses seem to fit most naturally into two different undergraduate schools—organizational into business and technological into engineering. Each concentration is regarded as essentially based on the idea of a double major covering a field of application. Thus, "typical" programs for each of the concentration options are presented in Section 5. A university desiring to have both concentration options available for undergraduates could achieve this by having the programs separately available in the undergraduate business and engineering schools, but this would require duplication of the common core courses. A collaborative effort would obviously be more satisfactory. It might be further argued that a most desirable solution would be to make the combined program, with both concentration options available, in a school of arts and science or equivalent, thus removing the more narrow emphasis in "business" or "engineering." Although this would undeniably make a broader set of combined application fields possible, the fact that an information systems speciality has an ultimate aim which is practical rather than intellectual should not be disregarded; and for a particular university, the compatibility of this with the rest of the arts and science curricula should be carefully considered. These matters are considered further in Section 6 on implementation.

The question also arises how the undergraduate program will relate to the master's level program. After the undergraduate program has been completed, only one year is required to complete the master's program. This is discussed explicitly in Section 5.

The content of the 11 undergraduate courses is drawn from that of the 13 graduate courses proposed earlier. Although the present report can be read independently, it is desirable to allow readers familiar with the earlier report [2] to make comparisons; and the

nomenclature for the 11 courses has been selected accordingly. A key to the correspondence is given at the beginning of Section 4.

The program objectives and prerequisites can also be compared to the earlier ones, and this is elaborated in Section 3.

3. Requirements

The graduate curriculum recommendations provided a list of output qualifications: knowledge, abilities, and experience [2, Sec. 3.1]. This list serves as a set of program requirements against which specific implementations can be checked.

A similar list is provided for the undergraduate curriculum. It takes into account the difference between the two concentration options (organizational and technological) and recognizes the less comprehensive coverage of the undergraduate curriculum.

3.1. Output Qualifications

The following list is based on the earlier one in which knowledge and abilities are grouped in six categories: (a) people, (b) models, (c) systems, (d) computers, (e) organizations, and (f) society. In addition to modifying or omitting certain of the "knowledge" and "ability" entries of the earlier list, some characterizations in terms of "understanding of the process" and "general knowledge" have been included to reflect the necessarily more modest objectives of an undergraduate program. For example, acquiring the ability to develop specifications requires considerably more course time than merely understanding of the process of developing specifications. The capabilities listed are testable in the academic environment—by written or oral examinations, successfully operating computer programs, and other commonly accepted means.

A suggested list of objectives common to the organizational and technological concentrations is:

- (a) *people*
ability to interact verbally with others, to listen and understand the views of others, to articulate and explain complex ideas.
- (b) *models*
ability to formulate and solve simple models of the operations research type, and to recognize the kind of situations in which they apply.
- (c) *systems*
ability to view, describe, define any situation as a system—specifying components, boundaries, and so forth;
ability to present in writing a summary of a project for management action (suitable to serve as a basis for decision);
ability to present in writing a detailed description of part of a project, for use in completing or maintaining same.
- (d) *computers*
general knowledge of basic hardware/software components of computer systems, and their patterns of configuration;
ability to program in a higher-level language;
ability to program a defined problem involving data files and communications structures;
general knowledge of sources for updating knowledge of technology;

ability to discuss the major alternatives (assuming current technology) in specifying an information processing system, including data files and communications structures, to the level of major system components;

ability to sketch "rough-cut" feasibility evaluations (in terms of economic and behavioral variables) of proposed new techniques or applications of current technology, identifying critical variables and making estimates and extrapolations;

ability to sketch an economic analysis for selecting among alternatives above, including identification of necessary information for making that analysis, and also to identify noneconomic factors;

understanding of the process of developing specifications for the computer-based part of a major information system, with details of task management and data base management components.

- (e) *organizations*
general knowledge of the function of purposeful organizational structure, and of the major alternative for that structure;
knowledge of how information systems are superimposed on organizational patterns, on the operational, control, and planning levels;
general knowledge of techniques for gathering information;
ability to gather information systematically within an organization, given specified information needs and/or specified information flows;
ability to outline, given information needs and sources, several alternative sets of information transfers and processing to meet needs;
ability to sketch "rough-cut" feasibility evaluations of such alternatives;
understanding of the process of developing specifications for a major information system, addressing a given organizational need, and determining the breakdown into manual and computer-based parts.
- (f) *society*
ability to articulate and defend a personal position on some important issue of the impact of information technology and systems on society (important, as defined by Congressional interest, public press, semitechnical press, etc.).

The above should be achieved, in more or less similar degree, by information systems undergraduates in either the organizational or technological concentration. The former group should have in addition the following, originally listed under the "organizations" heading:

- knowledge of the functional areas of an organization—operations, finance, marketing, product specification and development;
- knowledge of typical roles and role behavior in each functional area;
- ability to suggest possible short-term and long-term effects of a specified action on organizational goals;
- ability to discuss information needs appropriate to issues and roles above;
- understanding of the process of developing positive and negative impacts of a specified information system on specified parts of an organization.

The undergraduate in the technological concentration, however, is not expected to be supplied the background in either organizational functions or organizational behavior except to attain a very general notion of the foregoing. The same can be said of the following entry originally listed under the "people" heading:

- ability to describe individual and group behavior and to predict likely alternative and future behavior in terms of commonly used variables of psychology and economics.

On the other hand, the technological option gives the student considerably more in-depth exposure to

computer and programming techniques. Therefore, the following three abilities, originally listed under the "computers" heading, are appropriate for the technological option graduate:

- ability to develop several logical structures for a specified problem;
- ability to develop several different implementations of a specified logical structure;
- ability to develop specifications for a major programming project, in terms of functions, modules, and interfaces.

The student in the organizational concentration is only expected to have some general acquaintance with the process of performing these tasks.

Fig. 1. The 11 courses required for both undergraduate programs.

-
- UB1. OPERATIONS ANALYSIS AND MODELING
 - UB2. HUMAN AND ORGANIZATIONAL BEHAVIOR
 - UC1. INFORMATION STRUCTURES
 - UC2. COMPUTER SYSTEMS
 - UC3. FILE AND COMMUNICATION SYSTEMS
 - UC4. SOFTWARE DESIGN
 - UC8. PROGRAMMING STRUCTURES AND TECHNIQUES
 - UC9. COMPUTERWARE
 - UA8. SYSTEMS CONCEPTS AND IMPLICATIONS
 - UD8. INFORMATION SYSTEMS ANALYSIS
 - UD9. SYSTEM DESIGN AND IMPLEMENTATION
-

One of the primary limitations an undergraduate program has that a concentrated graduate program does not have is in the amount of exposure to prototype "real world" situations that can be included. Nevertheless, it is desirable that the undergraduate have at least some of the experience listed as desirable for the graduate, in particular:

- having gathered information in "real" or hypothetical organization;
- having served as a member of a project team outlining an information system, then programming a module of that system;
- having participated in planning and conducting an oral presentation of the results of a team project.

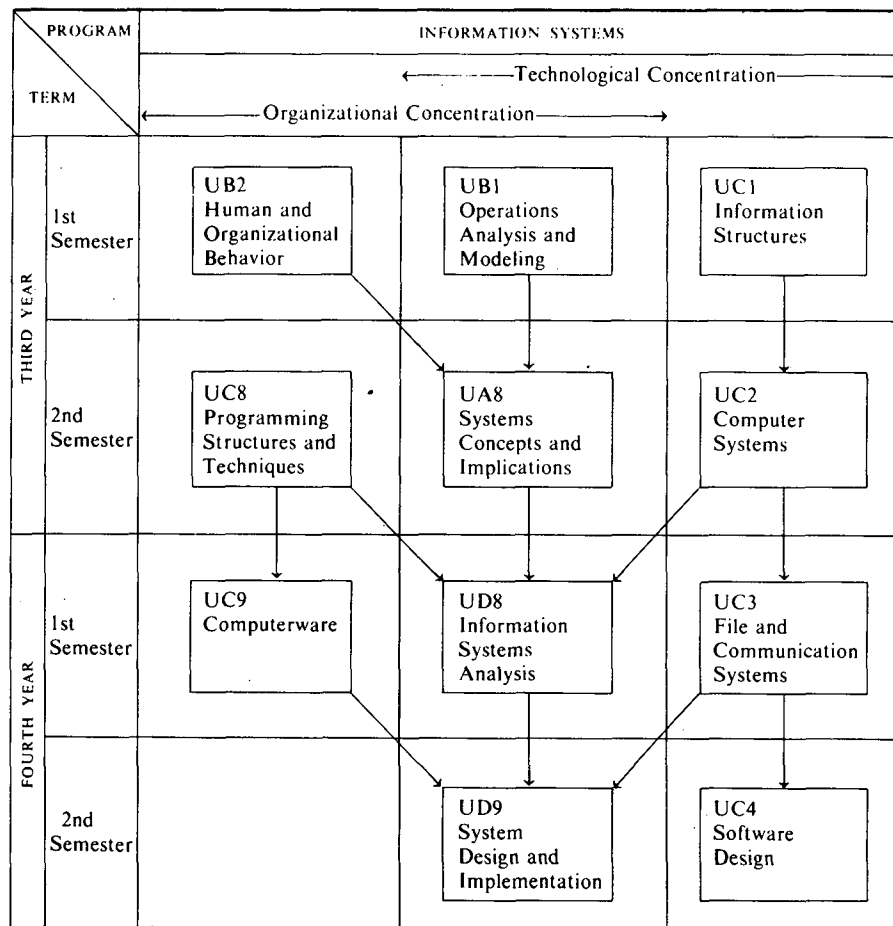
An individual meeting these output qualifications should be well prepared for an entry-level position. The two concentrations are designed to meet the above list of objectives.

3.2. Prerequisite Qualifications

The areas of prerequisite qualifications for the graduate program are listed in terms of five undergraduate course subjects [1, Sec. 3.3]:

- (i) finite mathematics, including the fundamentals of formal logic, sets and relations, and linear algebra;

Fig. 2. Core course sequences for information systems programs.



- (ii) elementary statistics, including the fundamentals of probability, expected value, and construction of sample estimates;
- (iii) elementary computer programming, including problem analysis and algorithm synthesis, and competence in a higher-level language;
- (iv) elementary economics, including microeconomics and theory of the firm, and price theory;
- (v) elementary psychology, including fundamentals of personality formation, attitudes, and motivation.

Essentially the same prerequisites, in terms of subject matter, are assumed for the undergraduate information systems program, in either concentration. Since the prerequisite courses in this case would necessarily have to be taken in the first and second undergraduate years, a somewhat less comprehensive coverage of the subjects may have to be assumed. Typical schedules incorporating these prerequisite courses for both the organizational and technological options are given in Section 5.

4. Course Descriptions

The 11 courses prepared for the two undergraduate concentration options are listed in Figure 1. Of these, six are undergraduate versions of courses specified for the graduate program: UB1, UB2, UC1, UC2, UC3, UC4. In each case the number is like that used for the original, prefixed by a "U."¹ The course titles are identical. Comparison with the descriptions of these with the earlier courses B1, B2 shows that the material is comparable but the coverage is less comprehensive, appropriate for undergraduate offerings.

The remaining five courses are combinations of pairs of earlier courses. In particular, UA8 combines material from earlier courses A1 and A4, with major emphasis on A1. Courses UD8 and UD9 combine material from A3 and D1, D2 and D3. Specifically, UD8 combines A3 and D1 material, with major emphasis on D1; and UD9 combines D2 and D3 material, with major emphasis on D2.

Finally, UC8 and UC9 are combinations of C courses, specifically C1 and C4 for UC8; and C2 and C3 for UC9. These two combined courses substitute in the organizational concentration for the more expanded offerings of UC1, UC2, UC3 and UC4 in the technological option. In contrast, the other combined

¹The 13 courses specified for the graduate program are: (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 Systems development projects.

courses, UA8, UD8 and UD9 are common to both concentration options. The roles and prerequisite structures of these courses are shown schematically in Figure 2, which also gives the sequencing of the course through the last two undergraduate years. Integration of these options into four-year undergraduate programs is discussed further in Section 5.

The remainder of this section gives brief descriptions for each of the 11 courses, which can be compared with the earlier descriptions of 13 courses [2, Sec. 4]. Detailed outlines and references are presented in the Appendix.

Course Group UB

Course UB1. Operations Analysis and Modeling

Objectives. To introduce and exercise a range of analytical and simulation modeling techniques useful in decision making in the system design environment. To consider the function of such models as guides for data collection, structures for data manipulation, and as systems for testing assumptions and generating a variety of alternatives. To identify the problems of data collection, maintenance, and accuracy when using models to assist decision-making activities.

Description. Characterization of scheduling situations. Analysis of allocation problems with mathematical programming. Queuing models. Inventory models. Use of simulation models. *Prerequisites:* finite mathematics, elementary statistics, elementary computer programming.

Course UB2. Human and Organizational Behavior

Objectives. To introduce the student to the principles governing human behavior, particularly as they relate to organizations and to the introduction and continued operation in organizations of computer-based information systems.

Description. Individual behavior. Interpersonal and group behavior. Organizational structure and behavior. The process of organizational change. The implementation and introduction of information systems. *Prerequisite:* elementary psychology.

Course Group UC

Approach to the UC Courses. In the undergraduate program more emphasis is placed on programming skills than in the graduate-level program. In particular, since it is likely that graduates will take entry-level programming positions, the student should have written programs to:

- perform data entry, editing, and validation
- update master files (sequential and random)
- generate reports
- perform error checking and handling
- search and sort

As part of course UC4, the student should participate in a team programming project, with emphasis on how to decide on the subdivision of the programming task.

Course UC1. Information Structures

Objectives. To introduce the student to structures for representing the logical relationship between elements of information, whether program or data, and to techniques for operating upon information structures. To examine the methods by which higher-level programming languages implement such structures and facilitate such techniques.

Description. Basic concepts of information. Modeling structures—linear lists. Modeling structures—multi-linked structures. Machine-level implementation structures. Storage management. Programming language implementation structures. Sorting and searching. Examples of the use of information structures. *Prerequisite:* elementary computer programming.

Course UC2. Computer Systems

Objectives. To provide a working view of hardware/software configurations as integrated systems, with (possibly) concurrently functioning components.

Description. Hardware modules. Execution software. Operation software. Data and program handling software. Multiprogramming and multiprocessing environments. *Prerequisite:* UC1.

Course UC3. File and Communication Systems

Objectives. To introduce the basic functions of file and communication systems, and to current realizations of those systems. To analyze such realizations in terms of the tradeoffs among cost, capacity, responsiveness. To examine some systems integrating file and communication functions, such as the organizational data base system or the computer utility.

Description. Functions of file and communication systems. File system hardware. File system organization and structure. Analysis of file systems. Data management systems. Communication system hardware. Communication system organization and structure. Analysis of communication systems. Examples of integrated systems. *Prerequisite:* UC2.

Course UC4. Software Design

Objectives. To examine how a complex computer programming task can be subdivided for maximum clarity, efficiency, and ease of maintenance and modification, giving special attention to available programming and linking structures for some frequently used interface programs, such as file and communication modules. To introduce a sense of programming style into the program design process.

Description. Run-time structures in programming languages. Communication, linking, and sharing of programs and data. Interface design. Program documentation. Program debugging and testing. Programming style and aesthetics. Selected examples. *Prerequisite:* UC3.

Course UC8. Programming Structures and Techniques

Objectives. To introduce the student to structures for representing the logical relationship between elements of information, whether program or data, and to

techniques for operating upon information structures. To examine how a complex computer programming task can be subdivided for maximum clarity, efficiency, and ease of maintenance and modification.

Description. Basic concepts of information. Storage management. Programming language implementation structures. Examples of the use of information structures. Searching and sorting. Communication linking, and sharing of programs and data. Interface design. Program documentation, debugging and testing. *Prerequisite:* computer programming.

Course UC9. Computerware

Objectives. To provide a working view of hardware/software configurations as integrated systems. To introduce the basic functions of file and communication systems, in terms of the tradeoffs among cost, capacity, responsiveness. To examine some systems integrating file and communications functions, such as the organizational data base system or the computer utility.

Description. Hardware modules. Execution software, multiprogramming and multiprocessing. Operation software. Data and program handling software. Functions of file and communication systems. File systems. Review of data management systems and analysis. Review of communication systems. Examples. *Prerequisite:* UC8.

Course Group UA

Course UA8. Systems Concepts and Implications

Objectives. To introduce the student to the information analysis and system design curriculum. To identify the basic concepts that subsequent courses will draw upon: the systems point of view, the organization as a system, its information flows, and the nature of management information systems. To explore the current and projected social and economic effects of information systems in organizations.

Description. The systems concept. Defining a system. Systems analysis. Management systems. Management information systems. Historical perspective of the computer industry. Effects on organizational practice. Privacy and the quality of life. *Prerequisite:* UB1.

Course Group UD

Course UD8. Information Systems Analysis

Objectives. To identify the decision requirements for the management of an organization. To analyze the design of an information gathering and processing system intended to facilitate decision making and planning and control. To analyze the concept of an information system. To review the approaches and techniques available to evaluate existing systems. To examine the concept of common data base for all functional modules.

Description. Nature of the decision-making process. Operational, tactical, and strategic-level systems. System life cycle management. Basic analysis tools. Defining logical system requirements. Determining eco-

nomics of alternative systems. *Prerequisites:* UA8, and UC2 or UC8.

Course UD9. System Design and Implementation

Objectives. To provide the knowledge and tools necessary to develop a physical design and an operational system from the logical design. To provide students with supervised and structured practical experience in the development of computer-based systems.

Description. Basic design tools and objectives. Hardware/software selection and evaluation. Design and engineering of software. Data base development. System implementation. Post implementation analyses. Long-range system planning. System development projects. *Prerequisites:* UD8, and UC3 or UC9.

5. Programs and Scheduling

The courses described in Section 4 are the basis for undergraduate programs following the two concentration options: *organizational and technological*. This section discusses how these options fit into four-year undergraduate programs and how they may feed into graduate programs.

5.1 Undergraduate Program Scheduling

The sequence of core courses shown in Figure 2, for both organizational and technological options, implies that the core concentration for the information systems speciality is fitted into the third and fourth undergraduate years. The typical undergraduate program must include general education components and general field requirements imposed by the department or school under whose auspices the program is taken, referred to as "university requirements" or "school requirements." The latter reflects the fact that these programs will most likely be incorporated in an undergraduate professional school, such as business or engineering.

There are requirements to be satisfied for the students major, including an application field in addition to the information systems core courses appropriate to each of the concentration options. These are referred to as "program requirements" for the core, and "parallel program requirements" for the other component of the "double major."

Figure 3 gives a typical four-year program (assuming five courses per semester, two semesters per year) for the organizational concentration in a business school, with accounting as the parallel program field. Figure 4 gives a corresponding typical program for the technological option in an engineering school. To fit these requirements into arts and sciences programs is more complicated, in general, since there is much more variation in the "typical." Nevertheless, the programs given should be adaptable to such requirements in most university settings.

A further question which naturally arises is how the

undergraduate work prepares the student for a professional master's level program. The undergraduate preparation would be adequate to cover some but not all of the corresponding graduate components. In particular, the D group courses should be taken in their entirety at the graduate level, despite the condensed coverage already afforded by UD8 and UD9.

Figure 5 shows two one-year graduate programs, one for each of the undergraduate concentration options. The organizational concentration, Figure 5(a), enables the graduate program to be completed in one year and is consistent with the second year of the graduate program specified in the earlier report [2, Sec. 5.2.].

The technological concentration, Figure 5(b), involves five of the same courses, but also two additional graduate courses B2 and A2, reflecting the behavioral and organizational material not included in the undergraduate technological option.

Note that in the format specified, B2 and A2 are assumed to be offered in the first semester, which is at variance with their scheduling for the full two-year graduate program in the second semester [2, Sec. 5.1].

Also C3 and C4 are omitted for the student with an undergraduate technological concentration. Thus, in this case only is it assumed that the corresponding undergraduate courses, UC3 and UC4, are sufficient substitutes for second-year master's level work.

6. Implementation

The graduate report [2, Sec. 6] also covers implementation, and discusses institutional considerations, course interactions, and instructional materials related to a graduate professional two-year program, or the one-year program, or options in other programs. Many of the remarks made and conclusions reached there are also relevant to initiation of undergraduate programs. There are, however, some differences that warrant discussion and recommendation when undergraduate programs are being considered. These are made here in the same three categories: institutional considerations, course interactions, and instructional materials.

6.1. Institutional Considerations

In Section 5, a number of alternative programs have been presented. In this section, the questions how such programs are to be started, where the program is to be placed in the institution, what resources will be required and how they are to be acquired will be discussed.

According to the estimates made by Hamblen, 1,700 institutions of higher education had a computing center, and more than 500 had degree programs of some kind in 1970. Most institutions are therefore starting from some base. For those that have no degree program and offer only a few courses under the auspices of a computing center, the easiest approach is to grad-

Fig. 3. Typical course schedule for undergraduate program—organizational concentration (illustrated with accounting as parallel program).

TERM YEAR	1st Semester					2nd Semester				
First Year	UNIV. RQMTS.				Elective	UNIV. RQMTS.				Elective
	Finite Math*	Econ*	English	Science		Psych*	English	Science	Humanities	
Second Year	SCHOOL RQMTS.		UNIV. RQMTS.		Elective	SCHOOL RQMTS.		UNIV. RQMTS.		Elective
	Computer Programming*	Intro. Acct.	Social Science	Humanities		Statistics*	Intro. Acct.	Social Science	Humanities	
Third Year	SCHOOL RQMTS.			Elective	Elective	SCHOOL RQMTS.			Elective	
	UB1 Human & Organizational Behavior	UB2 Operations Analysis & Modeling	Marketing			UA8 Systems Concepts & Implications	UC8 Programming Structures & Techn.	Money & Banking		Finance
Fourth Year	UD8 System & Information Analysis	UC9 Computerware	PARALLEL PROGRAM RQMTS.		Elective	PARALLEL PROGRAM RQMTS.				
			(e.g.) Cost Acct.	(e.g.) Intermed Acct.		UD9 System Design & Implementation	(e.g.) Auditing Theory	(e.g.) Acct. Theory	(e.g.) Acct. Prob. & Cases	

* Information systems prerequisite

Fig. 4. Typical course schedule for undergraduate program—technological concentration (assuming program is in an Engineering College and industrial engineering courses are used as electives).

TERM YEAR	1st Semester					2nd Semester				
First Year	UNIV. RQMTS.				Elem* Computer Progr.	UNIV. RQMTS.				Elective
	Finite* Math	Econ*	English	Science		Psych*	English	Science	Humanities	
Second Year	ENG. RQMTS.		UNIV. RQMTS.		Elective	ENG. RQMTS.		UNIV. RQMTS.		Elective
			Social Science	Humanities				Social Science	Humanities	
Third Year	I. E. RQMTS.			Elective	Elective	I. E. RQMTS.			Elective	
	UB1 Operations Analysis & Modeling	UC1 Info Structures	Probabil-ity			UA8 Systems Concepts & Implications	UC2 Computer Systems	Statistics		Stochastic Processes
Fourth Year	UD8 Info Systems Analysis	UC3 File & Communications Systems	I. E. RQMTS.		Elective	I. E. RQMTS.				
							UD9 System Design & Implementation			

* Information systems prerequisites

Fig. 5. One-year graduate programs for students who completed undergraduate options.

(a) Undergraduate program had organizational concentration.

1st Semester				2nd Semester			
A3 Information Systems for Operations and Management	D1 Information Analysis	C3 File and Communication Systems	Elective	A4 Social Implications of Information Systems	D2 System Design	D3 Systems Development Projects	C4 Software Design

(b) Undergraduate program had technological concentration.

1st Semester				2nd Semester			
A3 Information Systems for Operations and Management	D1 Information Analysis	B2 Human and Organizational Behavior	A2 Organizational Functions	A4 Social Implications of Information Systems	D2 System Design	D3 Systems Development Projects	Elective

ually expand this set of courses in line with the recommendation given in this report.

Other universities may be in a position to offer undergraduate or graduate programs or both in other academic areas: computer science, business, electrical or industrial engineering. For these institutions the problem is to move to a new program.

Usually the stimulus for the creation of an information systems program will come from one of the existing schools, and in the natural course of events, the program will be established there. Sometimes the initial interest will arise at the university level and some decision about where to house the program will have to be made. Occasionally there may be more than one school or college that claims jurisdiction over the program.

Universities differ widely in organization and objectives and in the number and strengths of faculty in various disciplines. Also, because the field of information systems is still relatively new and because the information systems program interacts with most of the disciplines and programs at a university, it is not possible to give a generally applicable recommendation about the organizational location of the information systems program. The curriculum may be housed in the school of business, computer science department, or industrial engineering department—depending on the particular academic structure and interests of faculty.

However, unless an institution is particularly rich in computer faculty, it is important to pull together all faculty talent through joint appointments with the department where the systems development program is to be housed. This approach enables implementation in the shortest time span. Later, when the program

grows to the point of multiple sections of courses, it may be advisable to begin hiring specialist faculty and to establish a separate department for the program.

It is possible, however, to list some considerations that should be included in the decision. Foremost, wherever the program is housed, its courses should be easily available to majors in other programs.

Implementation of a program requires resources in terms of faculty time and computer capabilities. Adoption of both options may be beyond the budget capability of some schools. The Committee assumes that most schools will decide to adopt one of the two options rather than both. The special circumstances and emphases of various schools will dictate a preference for one of the options.

Also, some schools will prefer to offer either the graduate or the undergraduate program, but not both programs. However, for a school where both programs are being considered, the approach shown in Figure 5 permits a student to obtain both a baccalaureate and a master's degree in five years.

Resources required for the programs include faculty to teach the courses, computer facilities, and facilities for students to obtain experience. The need for faculty to teach courses is covered in Section 6.2. Computer resources required depend on the particular alternatives chosen, but the amount and kind are not materially different from that required for other computer related courses. One way to provide opportunity for experience is through a cooperative program.

Other institutional considerations include gaining acceptance of a program. Students can be easily persuaded concerning the value of the program by showing its relevance to the needs of the society and to the

positions available to graduates. Faculty acceptance can be gained through the opportunity for association with a rapidly growing field. One area of concern is the question of accreditation by professional bodies. In most cases the implementation of the recommendations made here should not lead to conflict with these bodies.

6.2. Courses and Course Interactions

Implementation of the program will require development and offering of a number of courses which are not now generally available to undergraduates. Of these, three could be taught in schools of business:

- UB2. Human and Organizational Behavior
- UB1. Operations Analysis and Modeling
- UA8. Systems Concepts and Implications

Six of these could be taught in the computer science department or in Engineering:

- UC1. Information Structures
- UC2. Computer Systems
- UC3. File and Communications Systems
- UC4. Software Design
- UC8. Programming Structures and Techniques
- UC9. Computerware

and the remaining two could be taught in either:

- UD8. Information Systems Analysis
- UD9. System Design and Implementation

A major feature of the graduate program is the concept of the program as an integrated whole. Students would enter the program as a class and proceed through the same set of courses and the same experiences. The faculty would view the program with common objectives and consequently plan the material to be consistent, integrated, and supportive through the program.

Although this approach is possible, desirable, and efficient in a professional program—certainly as an ideal—such an approach is usually not practical in undergraduate programs. The information systems courses are only a small part of the student's learning experience. This person is concerned with obtaining a general education at least as much as with getting enough practical training to obtain a first position. Students may take a given course at different stages in their undergraduate life—even prerequisites cannot always be enforced consistently. Any given class will usually contain a mixture of students—some who regard the course as central to their interests; others who treat it as an elective for broadening their general education.

For all of these reasons it is not practical to expect much integration of the course material to be accomplished by the program. Flexibility is much more important at the undergraduate level. The person graduating from that program will therefore not be well prepared to assume immediate responsibility for information system development, but rather, he will be ready to enter an apprenticeship position. This is taken into

account in the list of qualifications for those who complete the undergraduate program (Section 3).

All of this is not to say that the interaction of courses or the integration of course material is not desirable when it can be accomplished.

6.3. Instructional Materials

The graduate curriculum report [2] contains extensive bibliographies for the topics covered in the recommended courses. It was assumed that instructors would be qualified to evaluate the various sources and make selections appropriate to their own situation. This assumption is less likely to hold for undergraduate courses where teaching loads tend to be greater and where the instructors do not have time to devote to research and to keeping up with the rapidly growing field.

An attempt has been made to recommend textbooks for courses. Unfortunately, few texts are available for the recommended undergraduate courses. The particular ones listed should be regarded as candidates, but instructors are encouraged to examine new books appearing on the market.

The number of films and computer-aided instructional material is increasing. While these generally are not substitutes for formal courses, lectures, and texts, they can be very useful as supplementary material. A survey and review of such material is published annually in the May issue of the *Computing Newsletter* [11].

7. Summary

The growth in size and complexity of computer-based systems necessitates more depth of knowledge on the part of the system design team if improved performance of the system is to be achieved. Users and practitioners alike need a broader understanding of both the managerial process and the technology in computerizing managerial systems.

Entry-level personnel for the information systems field may be properly prepared through an undergraduate education. With experience and advanced education, the individual can make a significant contribution to the system design processes.

The courses recommended for the undergraduate program may be housed in several academic units, permitting the degree program to be implemented with a minimum of additional resources. Many schools already have courses in a variety of academic units; analysis of these courses with the perspective of the recommended curriculum may permit an undergraduate program to be implemented merely by redesigning existing courses.

The committee welcomes feedback on its prototype proposal and is willing to comment on proposed implementations.

Acknowledgments. The following persons assisted

the Committee in one or another of the draft stages which preceded the final preparation of the report. The list includes those who supplied written reviews and those who participated in the meeting at Colorado Springs, mentioned in the Preface. The help of all is most gratefully appreciated.

Gerald St. Amand
Lloyd J. Buckwell
William K. Daugherty
Harold J. Highland
Hugh R. Howson
P. Jin
Boulton B. Miller
Andy S. Phillippakis
Ralph Sprague

William M. Taggart Jr.
Norman Taylor
Gerald Wagner
Lawrence Wergin
Theodore C. Willoughby
Howard Wilson
William Windham
Philip Wolitzer
Leon Youssef

References

1. Teichroew, D. (Ed.) Education related to the use of computers in organizations (Position Paper—ACM Curriculum Committee on Computer Education for Management). *Comm. ACM* 14, 9 (Sept. 1971), 573–588. Reprinted in *IAGJ*, 4 (1971), 220–252.
2. Ashenhurst, R.L. (Ed.) Curriculum recommendations for graduate professional programs in information systems. *Comm. ACM* 15, 5 (May 1972), 365–398.
3. Hammond, J.O. Planning data processing education to meet job requirements. Proc. AFIPS 1972 SJCC, Vol. 40, AFIPS Press, Montvale, N.J., pp. 59–67. *CR* 23, 10(71)874.
4. Tomorrow's manpower needs. Bureau of Labor Statistics, Bull. 1606, U.S. Dep. of Labor, Feb. 1969, Vol. I, App. A.
5. Occupational manpower and training needs. Bureau of Labor Statistics, Bull. 1701, U.S. Dep. of Labor, 1971.
6. Hamblen, J.W. Production and utilization of computer manpower in U.S. higher education. Proc. AFIPS 1972 SJCC, Vol. 40, AFIPS Press, Montvale, N.J., pp. 637–632.
7. Gilchrist, B., and Weber, R.W. Sources of trained computer personnel—A quantitative survey. Proc. AFIPS 1972 SJCC, Vol. 40, AFIPS Press, Montvale, N.J., pp. 633–647.
8. McKenney, J.L., and Tonge, F.M. The state of computer oriented curricula in business schools 1970. *Comm. ACM* 14, 7 (July 1971), 443–448.
9. Couger, J.D. Updating the survey on computer uses and computer curriculum. *Computing Newsletter for Schools of Business*, Colorado Springs, Colo., Oct. 1970, p. 1.
10. Brightman, R.W. (Ed.) *The Computer and the Junior College: Curriculum*, American Association of Junior Colleges. One Dupont Circle, N.W., Washington, DC 20036, 1970.
11. Couger, J.D. Guide to audio/visual instruction in data processing. *Computing Newsletter for Schools of Business*, Colorado Springs, Colo. (May issues, 1971, 1972, 1973).

Appendix

The Appendix gives detailed descriptions of and references for the courses whose brief descriptions appear in Section 4 of the main body of the report. For each course, the title, hours (x - y - z), and prerequisites are given, followed by a short statement of pedagogic approach and a listing of content. The content outline is organized by topic headings, each of which corresponds to an item in the earlier brief description of Section 4. A percentage figure is given with each topic heading, indicating a suggested proportion of the "lecture hours" (x) of the entire course to be devoted to that topic. The "other hours" (y) are intended to cover recitation, which can be discussion or problem sessions, and/or laboratory, which can be computer labs with an instructor or self-help sessions at terminals. The "credit hours" (z) are uniformly 3 for all courses.

Following the content section for each course there is a reference section containing a set of explicit citations of bibliographic entries, organized by topic heading. Citations are in the form "author (year)," which indexes the corresponding item in the accompanying bibliography.

Bibliographic entries are selectively annotated and in many cases accompanied by a citation to the review of the book or article in *Computing Reviews*. This is of the form "CR volume, number (year) review number."

UA8. Systems Concepts and Implications (3-0-3)

Prerequisite: UBI.

Approach: This course lays the groundwork for the curriculum by presenting the systems approach to understanding of both organizational and technological functions.

The method proposed is a combination of a series of lectures, a few cases on particular systems, and an accounting simulation project. Out-of-classroom study can be spent in analyzing the behavior of simulated systems to obtain an acquaintance with a variety of structures. Emphasis is on discussion of issues, implications, and possible information system and societal remedies. Student participation in the selection of specific issues is appropriate.

Content:

1. *The systems concept* (20%)
States, transformations, inputs, outputs, hierarchical structure. System objectives. Systems with complex/conflicting/multiple objectives: methods of resolution. System boundaries. Open and closed systems. Open systems: properties, adaptation. Elements (subsystems) (subunits) (components). Interfaces. Subsystems: independence/dependence, methods of decoupling. Suboptimization, side effects. Deterministic systems: probabilistic systems. The feedback concept: maximizing, "satisficing," "adaptivizing," adjusting to change. Control in a system: standards as predicted output, feedback (open-loop, closed-loop), cost of control. General Systems Theory. Examples of systems: ecological, medical service, transportation, manufacturing, logistics, etc.
2. *Defining a system* (10%)
Models as representations of systems. Complex versus simple models. Formal and informal systems: interaction between them. System structure: alternative structures. The identification problem. Tools: block diagrams, flow graphs, decision tables. Degrees of aggregation for systems. Modularity.
3. *System analysis* (10%)
Selection of a "best" course of action from many possible alternatives: advantage versus disadvantage, benefit versus cost. Determination of appropriate elements, connections, and processes to achieve objectives. Objectives, alternatives, cost-benefits, criteria. Modeling. System design: improving an existing system, developing a new system. The process of system design: problem identification and definition, alternative solutions, selection of a "solution," synthesis of the proposed system, testing of the system, refining the system. System optimization.
4. *Management systems* (20%)
Hierarchical structure. Interaction among subunits and within subunits. Human beings as elements in a system. "The Management System": the operations system, the decision system, the control system, time relationships and information flows. Information systems for the management system (as subsystems of the management system). Information system elements: managers, computer hardware and software, communication network, data bases, etc. Functional systems: accounting, procurement, inventory, etc.
5. *Management information systems* (15%)
Role of information systems in an organization. Delineating informational needs from traditional organizational structure; from nontraditional structure. Interface between man and system: man-machine systems. Information systems as operational (or production) processes. Distinction between logical and physical systems. Planning information systems. Approaches to the development of information systems. Examples of management information systems.
6. *Historical perspective of the computer industry* (5%)
Technological change in the 19th and 20th centuries. Economic and social problems of technology. Historical analogies. Method of assessing social costs of technological change. Sales and employment in the computer industry. Growth pattern. Competition in the computer industry. Standardization. Government regulation. Employment in information processing jobs. Problem of providing

training. Impact on industrial occupations, clerical occupations and on managerial occupations.

7. *Effects on organizational practice* (5%)

Centralization versus decentralization. Patterns of obtaining and providing services. Legal requirements. Possibilities for individualization. Effect on capacity to and rate of change.

8. *Privacy and the quality of life* (15%)

Public and private data banks, Rights of privacy. Relation of the individual to organizational data systems. Consumer protection. Influence on the educational process. Influence on the political process. Systems for administering justice, welfare, health care.

References: No single text is available for this course, but there are extensive references from varied sources.

1. *The systems concept:* Ackoff (1970); Ackoff (1971); Cleland and King (1968); F.E. Emery (1969); J.C. Emery (1969); Forrester (1961); Schoderbek (1967); Simon (1969).

2. *Defining a system:* Blumenthal (1969); Churchman et al. (1957); Cleland and King (1968).

3. *System analysis:* Churchman et al. (1957); Churchman (1968); J.C. Emery (1969); Forrester (1961).

4. *Management systems:* Ackoff (1967); Ackoff (1970); Blumenthal (1969); Churchman et al. (1957); Schoderbek (1967); Starr (1971).

5. *Management information systems:* Ackoff (1967); Benjamin (1971); Blumenthal (1969); Dearden (1972); J.C. Emery (1969); Krauss (1970).

6. *Historical perspective of the computer industry:* Kelson et al. (1967); Martin and Norman (1970); Rosenberg (1971); Sharpe (1969); Taviss (1970); Viavant (1971); Westin (1971).

7. *Effects on organizational practice:* Leavitt and Whistler (1958); Martin and Norman (1970); Pylyshyn (1970); Withington (1970).

8. *Privacy and the quality of life:* Greenberger (1971); Hoffman (1969); Martin and Norman (1970); Miller (1971); Pylyshyn (1970); Taviss (1970).

Bibliography: The references are to the combined bibliography given at the end of Course Groups UA and UD.

UD8. Information Systems Analysis (3-1-3)

Prerequisites: UA8, UC2 or UC8.

Approach: This is the first course in the sequence of two that covers systems development. The course begins with a study of the decision-making process and levels of decision making to provide a framework for the information system. This course emphasizes the information analysis and the logical design of the system, while course UD9 covers the physical design. Emphasis should be placed on the iterative nature of the analysis and design process.

Exercises and case studies (from the Intercollegiate Case Clearing House) are used to give students proficiency in information analysis techniques; however, the next course, UD9, provides practical application in system development and implementation. Of particular relevance are cases relying upon the analysis of the material, with interactive time-shared models to improve the student's understanding of the manager/computer interaction. Field trips to organizations with sophisticated information systems are useful in reinforcing concepts.

Content:

1. *Nature of the decision-making process* (25%)

Informal and formal channels of communication. Defining decisions. Decision criteria. Traditional decision making. Programmed decision making. Management-by-exception. External versus internal information sources and constraints. Manager/computer interactive systems: technical and behavioral considerations. System outputs: printed, audio, graphic.

2. *Operational, tactical and strategic level systems* (10%)

Providing for information needs of operating level supervisors and their employees; middle management; of executive level management. Effect of centralized versus decentralized organization structure. Planning and control models. Management information systems. Integrating systems.

3. *System life cycle management* (10%)

Overview of the phases of system development and their interrelationships. Conception, information analysis, system design, programming, documentation, installation, reevaluation.

Project control for system development. Levels of sophistication in system design. Responsibilities of system analysts, system

designers, programmers, operators, and data processing management.

4. *Basic analysis tools* (25%)

Steps in analysis: preliminary investigation, general feasibility study, general system proposal, detailed analysis. Techniques for analysis, such as: event-oriented organizational flowcharts, decision tables, precedence network analysis.

5. *Defining logical system requirements* (15%)

Format of the system requirements statement. Distinction of logical design (of system) from physical design (of files, programs, and procedures). System output requirements: specification of output methods and formats. System documentation requirements. System specification techniques: manual techniques; semi-automated techniques.

6. *Determining economics of alternative systems* (15%)

Manual versus automated parts of systems. Determining elements for common data bases. Data management alternatives. Response needs versus economic hardware/software and organizational constraints. Cost and value of information. Identifying and quantifying costs of system: personnel costs, equipment costs, conversion costs, installation costs. Identifying, quantifying, and measuring system advantages: direct and indirect benefits. Analyzing the improved quality of information. Allocation of costs and pricing of computer services.

References:

1. *Nature of the decision making process:* Ackoff (1970); Blumenthal (1969); Canning (1970d); Forrester (1961); LeBreton (1969); Miller and Starr (1967); Morton (1971).

2. *Operational, tactical and strategic level systems:* Ackoff (1970); Blumenthal (1969); Canning (1968b); Schrieber (1970).

3. *System life cycle management:* Benjamin (1971); Glans et al. (1968); Hartman et al. (1968).

4. *Basic analysis tools:* Chapin (1971); Couger (1973); Hartman et al. (1968); Pollack et al. (1971).

5. *Defining logical system requirements:* Clifton (1970); Couger (1973); Gray (1969); Glans et al. (1968); Hartman et al. (1968); Teichroew (1971).

6. *Determining economics of alternative systems:* Couger (1973); Emery (1971); Joslin (1971); Martin (1965); Martin (1969); Ollie (1970); Rubin (1970c); Sharpe (1969).

UD9. System Design and Implementation (3-1-3)

Prerequisites: UD8 and UC9 or UC3.

Approach: This course is the second covering the system life cycle, thus continuing the thrust of course UD8. The lectures focus on underlying principles of system design as well as on techniques. The techniques are utilized in the project. A theme to be carried throughout the course is the iterative nature of the analysis and design process. Implementation and conversion problems are also considered.

Case studies should be used as appropriate. Laboratory exercises should include the use of computer-assisted methods for system design.

Students are assigned a small project on a module of a large system development project. The projects involve the complete system development cycle: analysis, design, programming, and implementation. Students work in teams to acquire practical experience in such projects, especially regarding the behavioral considerations in systems development. They work with users to define system requirements and to prepare implementation plans and procedures.

Content:

1. *Basic design tools and objectives* (10%)

Review of the system life cycle. Documentation of various levels of design. Objectives: performance, internal control. Types of system design: batch, interactive. Budgeting and project management.

2. *Hardware/software selection and evaluation* (5%)

Equipment selection—evaluation of hardware and software requirements. Automated evaluation techniques—simulation, analytical models. Cost analyses. Competitive bidding.

3. *Design and engineering of software* (20%)

Design modularity. Design of user interfaces with automated procedures. Standardization of subsystem designs—data collection

editing, processing, and retrieval. Data and production controls. Audit trails. Internal and external accounting within the system. Conversion subsystems. Human engineering.

4. *Data base development* (15%)

Data base construction—creation, structure, maintenance, and interrogation of data bases. Integrity of the data base. Review and use of C3 course material on data base management systems.

5. *System implementation* (10%)

Levels of testing and debugging: planning and executing conversion; management of programming, testing, and installation. Coordination of manual and automated procedures. Techniques for cutover (parallel operation, etc.); implementation schedules.

6. *Post implementation analyses* (5%)

Auditing system performance: costing of system development effort and system performance.

7. *Long-range system planning* (5%)

Trends in information system design. Integrating several systems into a corporate MIS. Long-range forecasting of information requirements.

8. *System development projects* (30%)

Under supervision of the systems analysis staff, students could develop a subsystem for one of the major modules of a computer-based management information system of a local firm. Students might also work as members of established client companies' teams, or under the supervision of the university administrative data processing unit, students could develop a system which would provide them experience and at the same time benefit the university. Examples are: alumni record and follow-up system, bookstore ordering/accounting, classroom scheduling system. Or students could develop a system for a hypothetical application.

As an example, a case (SRA) currently available provides students with experience in each phase of system development for a hypothetical electronics firm. The material is organized into 13 assignments: orientation, documentation, written procedure, system flowcharts, gathering information, classification and coding, printed output source documents and punched cards, records design, data controls, run controls, audit trails, and file organization.

References: In addition to the references below, the references listed for courses UC3 and UC4 are particularly relevant for topics 2, 3, and 4.

1. *Basic design tools and objectives:* Benjamin (1971) Sec. 4; Hartman et al. (1968).
2. *Hardware/software selection and evaluation:* Couger (1973); Gregory and Van Horn (1963); Head (1971); Joslin (1971); Martin (1965); Martin (1967); Martin (1969); Sharpe (1969) Sec. 4; Sutherland (1971).
3. *Designing and engineering software:* Martin (1965); Martin (1967) Sec. 6; Matthews (1971) Ch. 5-7; Pollack et al. (1971); Rosen (1967); Rubin (1970c); Teichroew and Sayani (1971).
4. *Data base development:* CODASYL (1971); Flores (1970); GUIDE/SHARE (1970); Gildersleeve (1971); Lyon (1971); Martin (1967) Ch. 22.
5. *System implementation:* Benjamin (1971); Hartman et al. (1968); Martin (1965); Matthews (1971) Ch. 13.
6. *Post implementation analyses:* Benjamin (1971); Hartman et al. (1968); Matthews (1971) Ch. 11.
7. *Long-range system planning:* Blumenthal (1969).
8. *Development of a system for a local firm:* Blumenthal (1969) Ch. 3, 4; *Development of a system for a university/college:* Johnson and Katzenmeyer (1969) Pt. 3. *Development of a system for a hypothetical application:* Science Research Associates (1970).

Combined Bibliography—Courses UA–UD

Ackoff, R.L. (1967) Management misinformation systems. *Management Science* 14, 4, B-147-56.

A paper outlining the systems concept and the problems resulting when the systems approach is ignored in the development of information systems.

Ackoff, R.L. (1970) *A Concept of Corporate Planning*. Wiley, New York.

Note especially Ch. 6, "Control," in which Ackoff argues that the Management Information System is but a subsystem of the Management System.

Ackoff, R. L. (1971) Towards a system of systems concepts. *Management Science* 17, 11, 661-71.

An exposition of the concepts and terms "used to talk about systems," with particular attention given to organizations.

Ansoff, H.I. (1965) *Corporate Strategy*. McGraw-Hill, New York.

Benjamin, R.I. (1971) *Control of the Information System Development Cycle*. Wiley, New York.

Introduction to the system life cycle and its possible evolutions.

Blumenthal, S.C. (1969) *Management Information Systems: A Framework for Planning and Development*. Prentice-Hall, Englewood Cliffs, N.J., CR10, 10(69) 17,647.

A highly individual and idiosyncratic attempt to apply "the systems planning" approach to the development of management information systems. Note especially Ch. 3 "The Systems Taxonomy of an Industrial Corporation."

Brandon, R. (1963) *Management Standards for Data Processing*. Van Nostrand Reinhold, New York. CR 5, 5(64)6162.

Canning, R.G. (1968b) Systematic methods for business planning. *EDP Analyzer* 6, 3.

Canning, R.G. (1970d) Progressive fast response systems. *EDP Analyzer* 8, 8.

Chapin, N. (1971) *Flowcharts*. Auerbach, Princeton, N.J. CR 12, 12(71)22, 295.

Covers program flowcharts, system flowcharts, computer-produced flowcharts, ANSI Standard flowcharts.

Churchman, C.W., Ackoff, R.L., and Arnoff, E.L. (1957) *Introduction to Operations Research*. Wiley, New York.

Note especially Ch. 2 "An Operations Research Study of a System as a Whole" and Ch. 7 "Construction and Solution of the Models."

Churchman, C.W. (1968) *The Systems Approach*. Dell Books, New York.

Note especially "Supplement II" in which Churchman suggests additional readings and comments on the history of the systems approach, beginning with the statement that "Plato's Republic is a famous systems-science book."

Cleland, D.I., and King, W.R. (1968) *Systems Analysis and Project Management*. McGraw-Hill, New York. CR 10, 4 (69)16, 532.

Note especially Ch. 6 "Planning-Programming-Budgeting and Systems Analysis."

Clifton, D.H. (1970) *Systems Analysis for Business Data Processing*. Auerbach, Princeton, N.J. CR 12, 4(71) 20, 952.

An introductory book on system analysis and design.

CODASYL Systems Committee (1971) *Feature Analysis of Generalized Data Base Management Systems*. Technical report, available from ACM, New York.

See annotation in bibliography for Course Group C.

Couger, J.D. (1973) *System Analysis Techniques*. Wiley, New York.

A collection of articles on system analysis techniques, describing approaches which concentrate on concepts and principles of system analysis and cost/effectiveness analysis.

Davis, G. (1968) *Auditing and E.D.P.* Wiley, New York.

Dearden, J. (1972) MIS is a mirage. *Harvard Bus. Rev.* (Jan.-Feb.), 90-99.

An attack on the concept of "The Management Information System," arguing that a single, integrated information system cannot be devised.

Emery, F. E. (Ed.) (1969) *Systems Thinking: Selected Readings*. Penguin Books, New York.

Emphasizes systems thinking as developed from theorizing about biological systems to social systems rather than that which came from the design of complex engineering systems. Concentrates on "open systems" (open to exchange with an environment) and adaptive behavior.

Emery, J.C. (1969) *Organizational Planning and Control Systems: Theory and Technology*. Crowell Collier and Macmillan, New York.

An analysis of multilevel planning and control and the development of a supporting information system. Note especially Ch. 1 "The Systems Concept," Ch. 2 "The Organization as a System," and Ch. 3 "The Technology of Information Systems."

Emery, J.C. (1971) *Cost/Benefit Analysis of Information Systems*. The Society for Management Information Systems, Chicago.

Flores, I. (1970) *Data Structure and Management*. Prentice-Hall, Englewood Cliffs, N.J. CR 12, 4(71)20,916.

Forrester, J. W. (1961) *Industrial Dynamics*. MIT Press, Cambridge, Mass.

- Gildersleeve, T.R. (1971) *Design of Sequential File Systems*. Wiley, New York.
Covers the design of files and strategies for sequential storage media.
- Glans, T.B., Grad, B., Holstein, D., Meyers, W.E., and Schmidt, F.N. (1968) *Management Systems*. Holt, Rinehart and Winston. New York.
A detailed treatment of the initial stages of the system life cycle—analysis and design of the system. Includes concepts first published by IBM entitled "Study Organization Plan."
- Gray, M., and London, K.R. (1969) *Documentation Standards*. Brandon/Systems Press, Princeton, N.J. CR 10, 9(68)17,373.
The first book developed exclusively for this subject: covers all the salient facts concerning documentation.
- Greenberger, M. (Ed.) (1971) *Computers. Communications and the Public Interest*. Johns Hopkins Press, Baltimore, Md. CR 12, 11(71)22, 096.
A series of lectures by knowledgeable and thoughtful people on the relations between computers and society.
- Gregory, R.H., and Van Horn, R.L. *Automatic Data Processing Systems*. Wadsworth Pub. Co., San Francisco, 1963.
- GUIDE/SHARE. (1970) *Guide/Share Data Base Management System Requirements*. Technical report.
A well-written statement of requirements, emphasizing the importance and functions of the people in the system.
- Hartman, W., Matthes, H., and Proeme, A. (1968) *Management Information Systems Handbook*. McGraw-Hill, New York.
A comprehensive coverage of the steps in system development, developed by the Netherlands-based Philips Corporation.
- Head, R. V. (1971) *A Guide to Packaged Systems*. Wiley, New York.
- Hoffman, L. (1969) Computers and privacy: a survey. *Computing Surveys* 1, 2, 85-103.
A survey of technical literature and a discussion of what the technology can do to assist in maintaining privacy of information.
- Intercollegiate Bibliography* (1972) Collected Bibliography of Cases, Vol. 14, Intercollegiate Case Clearing House, Harvard U, Soldiers Field, Boston, MA 02163.
- Johnson, C.B. and Katzenmeyer, W.G. (Eds.) (1969) *Management Information Systems in Higher Education: The State of the Art*. Duke U. Press, Durham, N.C.
- Joslin, E. (Ed.) (1971) *Analysis, Design and Selection of Computer Systems*. College Reading Inc., Arlington, Va.
A book of readings, from earlier published articles.
- Kelson, R.R., Peck, J., and Kalacheck, E. (1967) *Technology, Economic Growth, and Public Policy*. Brookings Institute, Washington, D.C.
- Krauss, L.I. (1970) *Computer-Based Management Information Systems*. American Management Assoc., New York.
An exposition of the basic ideas of "MIS."
- Leavitt, H.J., and Whisler, T.L. (1958) Management in the 1980's. *Harvard Bus. Rev.* (Nov.-Dec.), 41-48. CR 9, 4(68)13,985.
- LeBreton, P.P. (1969) *Administrative Intelligence-Information Systems*. Houghton-Mifflin, Boston.
- Lyon, J.K. (1971) *An Introduction to Data Base Design*. Wiley, New York.
Concentrates on techniques in the design of online files.
- Martin, J. (1965) *Programming Real-Time Computer Systems*. Prentice-Hall, Englewood Cliffs, N.J.
Concentrates more on system design than programming aspects of online systems.
- Martin, J. (1967) *Design of Real-Time Computer Systems*. Prentice-Hall, Englewood Cliffs, N.J. CR 9, 2(68)13,607.
Continuation of his prior book, listed above. See also annotation in bibliography for Course Group C.
- Martin, J. (1969) *Telecommunications and the Computer*. Prentice-Hall, Englewood Cliffs, N.J. CR 11, 8(70)19,602, 19,603.
Technical aspects of the design of communication networks.
- Martin, J., and Norman, A. (1970) *The Computerized Society*. Prentice-Hall, Englewood Cliffs, N.J.
The first part of the book titled "Euphoria" is very interesting. Later sections deal more with the technology.
- Matthews, D.Q. (1971) *The Design of the Management Information System*. Auerbach, Princeton, N.J. CR 12, 8(71)21,668.
An introductory book on the MIS approach.
- McDaniel, H. (1970b) *Decision Table Software—A Handbook*. Brandon/Systems Press, Princeton, N.J. CR 12, 2(71)20,613.
Examples of the use of decision tables at the introductory level.
- Miller, A. (1971) *The Assault on Privacy: Computers, Data Banks, and Dossiers*. U. of Michigan Press, Ann Arbor, Mich. CR 12, 8(71)21,631.
A valuable compendium of the legal and ethical problems attendant to the growing use and sharing of data banks.
- Miller, P.W., and Starr, M.K. (1967) *The Structure of Human Decisions*. Prentice-Hall, Englewood, Cliffs, N.J.
- Morton, M.S.S. (1971) *Management Decision Systems*. Graduate School of Business Administration, Harvard U., Boston. CR 12, 6(71)20,367.
See annotation in bibliography for Course Group A2.
- National Cash Register Company. (1967) *Accurately Defined Systems*. Dayton, Ohio.
The system analysis and design approach advocated by NCR.
- Olle, T.W. (1970) MIS: data bases. *Datamation* (Nov.).
An excellent classification and characterization of file management systems, and how they fit into the world of management information systems.
- Orlicky, J. (1969) *The Successful Computer System: Its Planning, Development and Management in a Business Enterprise*. McGraw-Hill, New York. CR 10, 11(69)17,820.
Introduction to planning for the MIS.
- Pollack, S.L., Hicks, H.T. Jr., and Harrison, W.J. (1971) *Decision Tables: Theory and Practice*. Wiley, New York.
The theory and theorems of the decision table technique. Includes examples.
- Pylyshyn, Z.W. (Ed.) (1970) *Perspectives on the Computer Revolution*. Prentice-Hall, Englewood Cliffs, N.J. CR 12, 6(71)21,297.
Especially good on the educational and intellectual uses of computers, and the effects of such uses.
- Rosen, S. (Ed.) (1967) *Programming Languages and Systems*. McGraw-Hill, New York. CR 10, 1(69)15,975.
Readings which provide valuable historical perspective in the area of systems programming and the design of large scale operating systems.
- Rosenberg, N. (Ed.) (1971) *The Economics of Technological Change*. Penguin Books, New York.
- Rubin, M. (1970a) *Introduction to the System Life Cycle, (Vol. 1)*. Auerbach, Princeton, N.J.
Provides introductory level description of eight steps in the system life cycle.
- Rubin, M. (1970b) *System Life Cycle Standards, (Vol. 2)*. Auerbach, Princeton, N.J.
Provides standards, procedures and forms for system development.
- Rubin, M. (1970c) *Advanced Technology: Input and Output*. Auerbach, Princeton, N.J.
A reference for I/O approaches and design considerations.
- Rubin, M. (1970d) *Advanced Technology: Systems Concepts*. Auerbach, Princeton, N.J.
Introduction to systems analysis concepts.
- Schoderbek, P.P. (1967) *Management Systems*. Wiley, New York.
A book of readings intended to be used as a textbook in management courses to help in the "understanding of the total systems concept as well as developing insight into some of the problems besetting management." Note especially the criticism of the total system concept by W.M.A. Brooker, "The Total System Myth."
- Schrieber, A. (1970) *Corporate Simulation Models*. U. of Washington, Pullman, Wash.
- Science Research Associates (1970) *Case Study on Business Systems Design*. College Division, Palo Alto, Calif.
A laboratory manual providing thirteen assignments in developing an EDP system for a hypothetical electronics firm.
- Sharpe, W.F. (1969). *The Economics of Computers*. Columbia U. Press, New York.
Especially interesting are the sections on vendor behavior and selection of equipment.
- Shaw, J.C., and Atkins, W. (1970) *Managing Computer Systems Projects* McGraw-Hill. New York. CR 12, 9(71)21,832.
- Simon, H.A. (1965) *The Shape of Automation for Men and Management*. Harper & Row, New York. CR 7, 1(66)8773.
- Simon, H.A. (1969) *The Sciences of the Artificial*. MIT Press, Cambridge, Mass. CR 11, 1(70)18,222.
The entire work is recommended but note especially Ch. 4 "The Architecture of Complexity."

Starr, M.K. (1971) *Management: A Modern Approach*. Harcourt Brace and Jovanovich, New York.

Uses systems thinking in a novel management textbook. Note especially Ch. 2 "Building Management Models," Ch. 3 "Using Models," Ch. 7 "Managing Systems with Complex Objectives," Ch. 12, "Communication and Information Control," and Ch. 13 "The Organization of Simple Systems and Aggregations".

Sutherland, J.W. (1971) The configurator: today and tomorrow (Pt. 1); Tackle systems selection systematically (Pt. 2). *Computer Decisions*. (Feb., Apr.), 38-43, 14-19. *CR 12*, 7(71)21,521.

A two-part article on the use of simulation and analytical methods in the selection of a computer configuration.

Taviss, I. (Ed.) (1970) *The Computer Impact*. Prentice-Hall, Englewood Cliffs, N.J. The readings provide good, broad coverage of the entire area.

Teichroew, D., and Sayani, H. (1971) Automation of system building. *Datamation* (Aug. 15), 25-30. *CR 12*, 12(71)22,264.

Viavant, W. (Ed.) (1971) *Readings in Computers and Society*. Science Research Assoc., Palo Alto, Calif.

Walsh, D. (1969) *A Guide for Software Documentation*. McGraw-Hill, New York. *CR 11*, 7(70)19,392.

Provides forms and procedures to follow when documenting the design and coding of a software system.

Westin, A.F. (Ed.) (1971) *Information Technology in a Democracy*. Harvard U. Press, Cambridge, Mass.

Withington, F. (1970) *The Real Computer: Its Influences, Uses and Effects*. Addison-Wesley, Reading, Mass.

UB1. Operations Analysis and Modeling (3-1-3)

Prerequisites: finite mathematics, elementary statistics, elementary computer programming.

Approach: This course is based on the use of analytical models as aids in the formulation and resolution of system alternatives. Emphasis is on problem formulation and resolution relying upon available analysis packages. The discussion of projects should focus on the decision itself and on the use of models to consider alternatives and test assumptions. Problems of data acquisition, preparation, and maintenance should be stressed.

Projects should be drawn from the information system design area. The course might conclude with each student participating in the formulation of a simulation project that includes several of the analytical models introduced early in the course.

Content:

1. *Characterization of scheduling situations* (20%)

Characterization of a set of interlocking activities as a network. Popular algorithms for formulating and solving critical path models. Problems of manipulating estimates and range of accuracy measurements. Job scheduling and dispatching rules. Use of network models for control of projects. Scheduling in operating systems.

2. *Analysis of allocation problems with mathematical programming* (20%)

Methods of formulating and solving linear programming problems using packaged computer programs. Linear programming as an aid to planning the allocation of interdependent resources. Value of models in the sensitivity testing of formulations. Evolutionary nature of large models as a decision making aid. Applications to scheduling and computer network design. Optimization of computer networks. Note: particular attention should be paid to the data management requirements of LP models allowing examination of the general notions of constraints, objective functions, and optimization in modeling.

3. *Queueing models* (20%)

Concept of queueing models and their general applicability to a broad range of situations. Considerations of the many queueing processes within computer systems.

4. *Inventory models* (10%)

Inventory models ranging from simple, single product to multiple product under uncertainty. The data base as an inventory. Possible application of LP or dynamic programming analyses to inventory.

5. *Use of simulation models* (30%)

Examples and class projects to explore the need for problem

definition and reliance upon tailoring standard concepts to new situations, especially through dynamic models. Note: the analysis of the user and operating system parts of a time-sharing system might serve as class projects to integrate this topic with prior ones.

References: Several good textbooks (Ackoff and Sasieni, Hillier and Lieberman, Teichroew, and Wagner) contain teaching materials on the range of decision models covered in this course. The *Journal of the ACM* has published many specific papers on the use of operations analysis techniques in the design of computer systems.

1. *Characterization of scheduling situations:* Conway et al. (1967); Denning (1967).

2. *Analysis of allocation problems with mathematical programming:* Aho et al. (1971); Day (1965); Ramamoorthy and Chandy (1970); Theiss (1965).

3. *Queueing models:* Abate et al. (1968); Coffman (1969); Frank (1969); Gaver (1966).

4. *Inventory models:* Gaver and Lewis (1971); Martin (1967); Sharpe (1969); Woodrum (1970).

5. *Use of simulation models:* Lum et al. (1970); Senko et al. (1969); Sutherland (1971).

Bibliography for Course UB1:

Abate, J., Dubner, H., and Weinberg, S.B. (1968). Queueing analysis of the IBM 2314 disk storage facility. *J. ACM* 15, 4, 577-89. *CR 10*, 9(69)17,499.

Ackoff, R., and Sasieni, M. (1968) *Fundamentals of Operations Research*. Wiley, New York.

A good basic text for the not too mathematically inclined. Easy to read.

Aho, A., Denning, P.J., and Ullman, J.D. (1971) Principles of optimal page replacement. *J. ACM* 18, 1, 80-93. *CR 12*, 7(71)21,554.

A dynamic programming model for optimizing paging.

Coffman, E.G.Jr. (1969) Analysis of a drum input/output queue under scheduled operation in a paged computer system. *J. ACM* 16, 1, 73-90.

Conway, R., Maxwell, W., and Miller, L. (1967) *Theory of Scheduling*. Addison-Wesley, Reading, Mass.

Comprehensive treatment of scheduling problems and the techniques for solving them, including simulation.

Day, R.H. (1965) On optimal extracting from a multiple file data storage system: an application of integer programming. *Operations Research* 13, 3, 482-94.

Denning, P.J. (1967) Effects of scheduling on file memory operations. *Proc. AFIPS SJCC*, Vol. 30, AFIPS Press, Montvale, N.J. 9-21. *CR 8*, 6(67)13,301.

Frank, H. (1969) Analysis and optimization of disk storage devices for time-sharing systems. *J. ACM* 16, 4, 602-20. *CR 11*, 2(70)18,503.

Gaver, D. (1966) *Probability Models for Multiprogramming Computer Systems*. Doc. AD 640-706, Carnegie-Mellon U., Pittsburgh, Pa. *CR 9*, 1(68)13,459.

Gaver, D.P., and Lewis, P.A.W. (1971) Probability models for buffer storage allocation problems. *J. ACM* 18, 4, 186-98. *CR 12*, 9(71)21,870.

Hillier, F., and Lieberman, G. (1967) *Introduction to Operations Research*. Holden-Day, San Francisco.

An excellent textbook, with good coverage of probabilistic models.

Lum, V., Ling, H., and Senko, M. (1970) Analysis of a complex data management access method by simulation modeling. *Proc. AFIPS FJCC* Vol. 37, AFIPS Press, Montvale, N.J., 211-22.

Martin, J. (1967) *Design of Real Time Computer Systems*. Prentice-Hall, Englewood Cliffs, N.J. *CR 9*, 2(68)13,607.

This well-written volume stresses the use of quantitative analyses at all stages in the design of information systems and gives examples of the techniques.

Ramamoorthy, C.V., and Chandy, K.M. (1970) Optimization of memory hierarchies in multiprogrammed systems. *J. ACM* 17, 3, 426-45.

Shows the use of both linear and integer programming models.

Senko, M., Lum, V., and Owens, P. (1969) A file organization and evaluation model (FOREM). *Proc. IFIP Congress 68*. *CR 11*, 4(70)18,813.

A description of a generalized simulation model for file systems.

- Sharpe, W.F. (1969) *The Economics of Computers*. Columbia U. Press, New York.
Quantitative analyses from an economist's perspective.
- Sutherland, J.W. (1971) The configurator: today and tomorrow (Pt. 1) and Tackle systems selection systematically (Pt. 2). *Computer Decisions* (Feb., Apr.), 38-43: 14-19. *CR 12*, 7(71) 21,521.
A two-part article on the use of simulation and analytical methods in the selection of a computer configuration.
- Teichroew, D. (1964) *An Introduction to Management Science*. Wiley, New York.
Broad coverage of operations research/management science techniques.
- Theiss, H.E. (1965) Mathematical programming techniques for optimal computer use. *Proc. 1965 ACM National Conference*, 501-12. *CR 7*, 1(66)8864.
- Veinott, A. (1965) *Mathematical Studies in Management Science*. Crowell Collier and Macmillan, New York.
Deals mainly with the probabilistic techniques—especially inventory theory.
- Wagner, H. (1970) *Principles of Management Science*. Prentice-Hall, Englewood Cliffs, N.J. *CR 12*, 2(71)20, 616.
Truly comprehensive, well written, and usable. Emphasizes deterministic models, but provides good coverage of the other areas.
- Woodrum, L.J. (1970) A model of floating buffering. *IBM Systems J.* 9, 2, 118-44. *CR 11*, 11(70)20, 149.
Uses ideas of Markov and semi-Markov processes.

UB2. Human and Organizational Behavior (3-1-3)

Prerequisite: elementary psychology.

Approach: This course examines the principles of human behavior in individuals, groups, and organizations in the contexts relevant to information systems.

Behaviorally-oriented reference material relating specifically to information systems is sparse, and particularly so for the final section on implementation. The cited references frequently have a management or engineering orientation, leaving the behavioral implications to be supplied by the instructor or by the class.

An appropriate computer game or interactive laboratory experiment could be used as an effective tool to demonstrate aspects of individual, interpersonal, and group behavior, with the student population itself as subject.

Content:

- Individual behavior (20%)**
Human sensing and processing functions. Visual, auditory, motor, and linguistic mechanisms. Perception, cognition, and learning. Human factors engineering in information systems.
- Interpersonal and group behavior (20%)**
Personality and role. Motivation, participation, and communication. Influence and effectiveness. Authority and leadership. Mechanisms for group action. The impact of information systems on interpersonal and group behavior.
- Organizational structure and behavior (25%)**
Organization theory. Impact of information systems on organizational structures and behavior. Implications for management.
- The process of organizational change (25%)**
Resistance to and acceptance of change. The management of change. Problems of adjustment to the information systems environment.
- The implementation and introduction of information systems (10%)**
Interaction between information analysis and system design groups and the remainder of the organization. Information system project teams and their management. Preparation for installation and operation. Note: this section is background for material covered more extensively in courses D1 and D2.

References: No one book covers the full scope of the course. Fogel (1967), Katz and Kahn (1966), and Likert (1967) are books on individual and organizational behavior written from the systems point of view. Wadia (1968) is a book of readings which cover the behavioral sciences aspects of the course fairly well. Bennis (1968) is an excellent treatment of organizational change, of which Toffler (1970) is a popular treatment. Tomeski (1970) and Withington

(1969) give insight into the impact of the computer on organizations and people.

- Individual behavior:** Berelson and Steiner (1964) Ch. 5; Fogel (1967); Miller (1967).
- Interpersonal and group behavior:** Berelson and Steiner (1964) Ch. 6, 8; Cartwright and Lippit (1957); Likert (1953); MacKinnon (1962); Schein (1971); Zalkind and Costello (1962).
- Organizational structure and behavior:** Bavelas (1960); Beckett (1967); Berelson and Steiner (1964) Ch. 9; DeCarlo (1967); Katz and Kahn (1966); Klahr and Leavitt (1967); Likert (1967); McGregor (1960); Simon (1964); Steiner (1964); Whisler (1967).
- The process of organizational change:** Bennis (1966); DeCarlo (1967); Ginzburg and Reilley (1957); Fuller (1969); Morison (1966); Toffler (1970).
- The implementation and introduction of information systems:** Orden (1960); Orlicky (1969) Ch. 5-8; Sackman (1967); Simon and Newell (1960); Tomeski (1970) Ch. 13, 14; Withington (1966) Ch. 8, 9.

Bibliography for Course UB2

- Bavelas, A. Communication and organization. In Shultz and Whisler (1960).
- Beckett, J.A. The total-systems concept: its implications for management. In Myers (1967).
- Bennis, W.G. (1968) *Changing Organizations*. McGraw-Hill, New York.
- Berelson, B., and Steiner, G.A. (1964) *Human Behavior: An Inventory of Scientific Findings*. Harcourt Brace and Jovanovich, New York.
A compendium of behavioral sciences accomplishments. Ch. 5 on learning and thinking. Ch. 6 on motivation, Ch. 8 on small group relationships, and Ch. 9 on organizations are relevant to the course.
- Cartwright, D., and Lippit, R. (1957) Group dynamics and the individual. *Internat. J. of Group Psychotherapy* 7, 86-102. In Wadia (1968).
- DeCarlo, C.R. (1967) Changes in management environment and their effect upon values. In Myers (1967).
- DeGreene, K.B. (Ed.) (1970) *Systems Psychology*. McGraw-Hill, New York.
- Fogel, L.J. (1967) *Human Information Processing*. Prentice-Hall, Englewood Cliffs, N.J.
Looks at the human as an input, decision-making, output processor.
- Fuller, R.B. (1969) *Operating Manual for Spaceship Earth*. Southern Illinois U. Press, Carbondale, Ill.
A treatise on the need for human adaptation to changed environmental circumstances, by one of the more innovative thinkers of our time.
- Ginzberg, E., and Reilley, E.W. (1957) *Effecting Change in Large Organizations*. Columbia U. Press, New York.
A step-by-step analysis for managing organizational change.
- Karplus, W.J. (Ed.) (1967) *On-Line Computing. Time-Shared Man-Computer Systems*. McGraw-Hill, New York, *CR 8* 3(67)11, 952.
- Katz, D., and Kahn, R.L. (1966) *The Social Psychology of Organizations*. Wiley, New York.
A particular point of view on organizational behavior.
- Klahr, D., and Leavitt, H.J. (1967) Tasks, organization structures, and computer programs. In Myers (1967).
- Likert, R. (1953) Motivation: the core of management. *Personnel Series* No. 155. American Management Assoc., 3-21. In Wadia (1968).
- Likert, R. (1967) *The Human Organization: Its Management and Value*. McGraw-Hill, New York.
A systems approach to organizational behavior.
- MacKinnon, D.W. (1962) What makes a person creative? *The Saturday Review* (Feb. 10), 15-69. In Wadia (1968).
- McGregor, D. (1960) The role of staff in modern industry. In Shultz and Whisler (1960).
- Miller, G.A. (1967) *The Psychology of Communication*. Basic Books, New York.
A collection of perceptive articles on human communication, including the author's well-known "The magical number seven, plus or minus two" paper.
- Morison, E.E. (1966) *Men, Machines, and Modern Times*. MIT Press, Cambridge, Mass. *CR 8*, 2(67)11, 356.

- A set of anecdotal case studies, bearing on the position of man pitted against technology.
- Myers, C.A. (Ed.) (1967) *The Impact of Computers on Management*. MIT Press, Cambridge, Mass. CR 8, 4(67)12, 265.
- Orden, A. (1960) Man-machine computer systems. In Shultz and Whisler (1960).
- Orlicky, J. (1969) *The Successful Computer System: Its Planning, Development and Management in a Business Enterprise*. McGraw-Hill. New York. CR 10, 11(69)17, 820. Introduction to planning for the MIS.
- Sackman, H. (1967) *Computers, System Science and Evolving Society: The Challenge of Man-Machine Digital Systems*. Wiley, New York. CR 9, 5(68)14, 154. Ch. 9, 11, and 12 are relevant to behavioral considerations.
- Schein, E.H. (1961) Management development as a process of influence. *Industrial Management Review* (May), 59-77. In Wadia (1968).
- Shultz, G.P., and Whisler, Y.L. (Eds.) (1960) *Management Organization and the Computer*. The Free Press, Glencoe, Ill.
- Simon, H.A., and Newell, A. (1960) What have computers to do with management? In Shultz and Whisler (1960).
- Simon, H.A. (1964) On the concept of organizational goal. *Administrative Science Quarterly* 9, 1-22. In Wadia (1968).
- Sisson, R.L., and Canning, R.G. (1967) *A Manager's Guide to Computer Processing*. Wiley, New York.
- Steiner, G.A. (1964) The creative organization. *Stanford U. Graduate School of Business Bulletin* 33, 12-16. In Wadia (1968).
- Toffler, A. (1970) *Future Shock*. Random House, New York. A much talked-about analysis of the impact of rapid external change on human behavior.
- Tomeski, E.A. (1970) *The Computer Revolution: The Executive and the New Information Technology*. Crowell Collier and Macmillan, New York. Covers both new patterns of administration brought about by information technology and the administration of that new technology itself.
- Withington, F.C. (1969) *The Real Computer: Its Influences, Uses and Effects*. Addison-Wesley, Reading, Mass. Insightful discussion of the myth and the reality of the impact of the computer on people.
- Woodward, J. (1965) *Industrial Organization, Theory and Practice*. Oxford U. Press, Oxford, England.
- Zalkind, S.S., and Costello, T.W. (1962) Perception: implications for administration. *Administrative Science Quarterly* 7, 218-35. In Wadia (1968).

UC1. Information Structures (2-2-3)

Prerequisite: elementary computer programming.

Approach: The structures which may be used to represent the information involved in solving problems are presented. Both modeling structures and implementation (storage) structures are covered. Emphasis is placed on treating these structures independently of particular applications. Examples, however, particularly from information system design, should be used wherever possible. The interrelationship between problem solving procedure, modeling structure, and implementation structure is stressed. Alternative implementations of a particular model are explored. Implementations in higher-level languages of several modeling structures are presented.

Students should apply the techniques presented to a number of problems. Care should be taken to separate development of modeling structures from implementation; and in many instances the student's analysis of a problem can stop at the modeling structure level. For at least some of the problems, however, students should implement and test their proposed representations.

Content:

1. *Basic concepts of information* (10%)
Representation of information outside and inside the computer. Bits, bytes, fields, items, records, files. Numbers and characters. Characteristics of computer arithmetics—conversion, truncation and roundoff, overflow and underflow. Names, values, environments, and binding of data. Use of pointers or linkage variables to represent structure. Identifying entities about which data is to be maintained, and selecting data nodes and structures which are to be used in problem solution.

2. *Modeling structures—linear lists* (10%)
Linear lists, stacks, queues, deques. Single, double, circular linkage. Strings, insertion, deletion, and accessing of list elements.
 3. *Modeling structures—multilinked structures* (10%)
Trees and forests. Free, oriented, and ordered trees. Binary tree representation of general trees. Traversal methods—preorder, postorder, endorder. Threading trees. Examples of tree structures as algebraic formulas, dictionaries, and other hierarchical information structures. Arrays and tables. Storage mapping functions. Linked representation of sparse arrays. Multilinked structures with heterogeneous fields and/or nodes (plexes).
 4. *Machine-level implementation structures* (10%)
Word packing and part-word addressing. Sequential allocation. Linked allocation and pointer manipulation. Scatter storage; hash table formats, hashing functions, methods of resolving collisions. Direct and indirect address calculation. Implementation of linked structures in hardware.
 5. *Storage management* (5%)
Static versus dynamic allocation. Stacks and available space lists. Explicit release of available space, reference counts, and garbage collection. Coalescing adjacent free space. Variable block size—stratified available space lists, the buddy system.
 6. *Programming language implementation structures* (30%)
Examples of the implementation of modeling structures in higher-level languages. FORTRAN, PL/I and ALGOL arrays. SNOBOL and PL/I strings. Lists in PL/I, GPSS, SIMSCRIPT, IDS. Tables and records in PL/I, COBOL.
 7. *Sorting and searching* (10%)
Radix sort, merge sort, bubble sort, address table sort, tree sort, and other sorting methods. Comparative efficiency of sorting methods. Use of sort packages. Linear search, binary search, indexed search, and other searching methods. Tradeoffs between sorting effort and searching effort. Effect of information structure on sorting and searching techniques.
 8. *Examples of the use of information structures* (15%)
Representation of information by translators. Representation of information during execution; activation records. Implementation of higher-level language data structures. Organization of an inverted file for document retrieval. Examples in graphic manipulation systems, simulation packages, data management systems.
References: Bertziss (1971) or Knuth (1968) are most suitable candidates for use as texts in this course, supplemented by additional readings in a few topics.
1. *Basic concepts of information:* Bertziss (1971); Iverson (1962); Johnson (1970); Knuth (1968); Mealy (1967); Wegner (1968).
 2. *Modeling structures—linear lists:* Bertziss (1971); Dodd (1969); Hopgood (1969); Iverson (1962); Johnson (1970); Knuth (1968); Mealy (1967); Williams (1971).
 3. *Modeling structures—multilinked structures:* Bertziss (1971); Dodd (1969); Hopgood (1969); Iverson (1962); Johnson (1970) Ch. 1-3; Knuth (1968); Mealy (1967); Williams (1971).
 4. *Machine-level implementation structures:* Bertziss (1971); Dodd (1969); Gauthier and Ponto (1970); Hopgood (1969); Johnson (1970); Knuth (1968); Morris (1968); Wegner (1968); Williams (1971).
 5. *Storage management:* Bertziss (1971); Johnson (1970); Knuth (1968).
 6. *Programming language implementation structures:* Bertziss (1971); CODASYL (1971); Gordon (1969); Griswold et al. (1968); Iverson (1962); Rosen (1967) Pt. 3; Wegner (1968); Williams (1971).
 7. *Sorting and searching:* Bertziss (1971); Flores (1969); Gauthier and Ponto (1970); Hopgood (1969); Iverson (1962); Johnson (1970) Ch. 4, 5; Wegner (1968).
 8. *Examples of use of information structures:* Bertziss (1971); Dodd (1969); Knuth (1968); Wegner (1968); Williams (1971).
- Bibliography:** The references are to the combined bibliography given at the end of Course Group UC.

UC2. Computer Systems (2-2-3)

Prerequisites: UBI, UC1.

Approach: Computer systems, their hardware and basic operating software, are studied, with attention to the human factors involved in computer system operation and maintenance. Types

of modules and types of system function mode (batch, interactive, online, etc.) should be carefully distinguished.

Block diagrams, flowcharts, and some kind of formal descriptive language should be used to set forth the systems aspects discussed. A suitable choice for the latter would be an assembly language with macro capability. Problem assignments should involve proposing system or subsystem attributes and parameters for given performance specifications and testing the proposals by simulation. Simulation packages for evaluating subsystem configurations should be available.

Content:

1. *Hardware modules* (20%)

Processor, memory, input/output, mass storage, remote transmission modules; function and possible realization of each. Micro-programming. Styles of buffering, interfacing, communication and interrupt handling. Memory management, virtual memory. Channel management, virtual configurations. Network and multiprocessor configurations. Note: the approach of this section is conceptual, to point up the need for a comprehensive hardware/software viewpoint—the concepts are then elaborated in programming and operational terms in subsequent sections.

2. *Execution software* (15%)

General interpretive modules for execution support, e.g. list processors. Modules for memory management in real and virtual memory systems. Processor and channel modules for support of input/output, mass storage and remote transmission units in real and virtual configurations. Concepts of multiply-reentrant programs and cooperating processes.

3. *Operation software* (25%)

Loading, interrupt monitoring, diagnostic modules. Scheduling, resource allocation, performance monitoring packages. Concepts of state resurrection and interprocess protection.

4. *Data and program handling software* (30%)

Media and format conversion modules. File handling facilities. Control specifications for datasets. Translating, compiling, generating modules. Macro facilities. Editing and debugging facilities. Linkage and job control specifications for subroutines, coroutines and standard packages. Problems of identification and security. Note: this topic is background for courses C3 and C4.

5. *Multiprogramming and multiprocessing environments* (10%)

Levels of multiaccessing and multiplexing. Batch and interactive modes. Requirements for effective usability, operability, maintainability of operating systems. Performance monitoring and management of complex hardware/software configurations.

References: There is no single introductory text for a combined hardware/software course at this level. An advanced text which embodies this kind of approach is Beizer (1971). The references given for Curriculum 68 courses 13 and 14 are in general relevant. Text materials may also be drawn from the operating manuals of whatever large-scale computer system is available for use in the course.

1. *Hardware modules:* Bell and Newell (1971); Buchholz (1962); Gear (1969); Husson (1970); Martin (1967).

2. *Execution software:* Daley and Dennis (1968); Denning (1970); Dijkstra (1968b).

3. *Operation software:* Rosin (1969).

4. *Data and program handling software:* Rosen (1967).

5. *Multiprogramming and multiprocessing environments:* Daley and Dennis (1968); Rosen (1967) Pt. 5; Stimler (1969); Watson (1970); Wilkes (1967).

Bibliography: The references are to the combined bibliography given at the end of Course Group UC.

UC3. File and Communication Systems (2-2-3)

Prerequisite: UC2.

Approach: The basic components of file and communication systems are presented and analyzed. The functioning of these systems as integral components of an information system is stressed.

The instructional approach is primarily lecture and problem discussion. This is neither a project nor a programming course as such; most student assignments concern design or analysis of carefully specified and limited subprograms or subsystems. When possible, if a file management language is available, a small file system design and implementation project would be desirable.

Content:

1. *Functions of file and communication systems* (5%)

Role of information acquisition, storage and transmission in an organization. Typical operations in file systems: file creation, maintenance, interrogation. Typical operations using communication systems. Issues of information availability, privacy, security.

2. *File system hardware* (5%)

Characteristics of auxiliary storage devices. Capacity, access, cost. Device types: tape, disk, mass storage.

3. *File system organization and structure* (25%)

Data fields, records, files, hierarchies of files. Directories and indices, inverted files. Structure and access: sequential, direct indexed sequential, randomized, randomized with buckets. Storage allocation and control techniques.

4. *Analysis of file systems* (10%)

Estimating capacity and timing requirements. Tradeoffs between access time, capacity and density of use, cost. Tradeoffs between file creation/maintenance activity and access activity. Relevant formulas and tables.

5. *Data management systems* (10%)

Generalized data management systems. Directory maintenance, query languages, data description, job management. Characteristics of available systems.

6. *Communication system hardware* (15%)

Theoretical concepts. channels and channel capacity noise, error detection and correction. Existing communication facilities; line types, exchanges: utilities, regulatory agencies, and tariffs. Pulse techniques. Transmission codes. Transmission modes. Line termination and terminal devices.

7. *Communication system organization and structure* (10%)

Single line: point-to-point, multidrop. Networks: centralized, decentralized, distributed. Control and protocol: acknowledgment, wait-requests, contention, polling. Switched, store-and-forward. Data concentrators and distributors.

8. *Analysis of communication systems* (5%)

Estimating line and terminal requirements: volume and message length, speed and timing, cost implications. Bottlenecks and queues, queueing analysis, simulation.

9. *Examples of integrated systems* (15%)

The data base concept: integrated data approach, coordination, control, multiple use of data. The data administrator; the computer utility. System resiliency and integrity, privacy, and security considerations.

References: While no textbook available at present is organized to match the scope of this course, Martin (1967) covers much of the material here.

1. *Function of file and communications systems:* Gruenberger (1969); Martin (1967); Meadow (1967); Minker and Sable (1967); Salton (1968); Senko (1969).

2. *File system hardware:* Martin (1967).

3. *File system organization and structure:* CODASYL (1969); CODASYL (1971); Dodd (1969); IBM (1969) IFIP (1969); Martin (1967); Meadow (1967); Salton (1968); Senko (1969).

4. *Analysis of file systems:* CODASYL (1971); IBM (1969); Martin (1967); Salton (1968).

5. *Data management systems:* CODASYL (1969); CODASYL (1971); Gruenberger (1969); IFIP (1969); Martin (1967); Senko (1969).

6. *Communication system hardware:* Davenport (1971); Gentle (1965); Martin (1967); Martin (1969).

7. *Communication system organization and structure:* Davenport (1971); Martin (1967); Martin (1969).

8. *Analysis of communication systems:* Martin (1967); Martin (1969).

9. *Examples of integrated systems:* Martin (1967); Parkhill (1966).

Bibliography: The references are to the combined bibliography given at the end of Course Group UC.

UC4. Software Design (2-2-3)

Prerequisite: UC2.

Approach: This course brings the student to grips with the actual problems encountered in designing, implementing and modifying systems of computer programs. The concept of pro-

programming style should permeate most of the material presented, although it appears as a specific lecture topic toward the end of the course. Careful verification of program operation and documentation of programs should be emphasized. Much of the course, particularly in the laboratory sessions, may be devoted to the actual implementation of the programs. A useful exercise would be to have each student modify a program written by someone else.

Content:

1. *Run-time structures in programming languages (10%)*
Textual versus execution semantics in languages. Binding of names. Examples from FORTRAN, ALGOL, PL/I, and some data management system. Run-time stacks.
2. *Communication, linking, and sharing of programs and data (30%)*
Separation of program and data segments. Common (global) versus local data. Block structures—static and dynamic nesting, internal and external procedures. Subroutines and coroutines as linkage structures. Sharing of code—reentrancy, recursion, pure procedures. Static and dynamic linking and loading, relocatability, self-relocating programs. Table driven programs. Asynchronous versus synchronous control, cooperating processes, multitasking.
3. *Interface design (10%)*
Parameters, work space, automating, and documenting interfaces. Control blocks.
4. *Program documentation (10%)*
Self-documenting programs. Levels of detail in documentation. Automatic flowcharting methods. Motivation for documentation—maintenance and modification of programs.
5. *Program debugging and testing (15%)*
Automating the debugging process. Symbolic debugging aids. Automatic generation of test data and expected results. Analysis of testing procedures. Hierarchical testing. Exhaustive testing versus random sampling. Testing of communications programs. Simulation as a tool. Abnormal condition handling.
6. *Programming style and aesthetics (10%)*
Modular programming—functional modules, breaking up of large functional modules. Central versus distributed control structures. Macro and micro modularity. Inter- and intra-language communication. Clarity and documentation—block diagramming.
7. *Selected examples (15%)*
File handling modules. Error retry, request queueing. Communication interface modules. Polling versus contention, interrupt handling. Selected materials such as graphics programming, programming for realtime sensing devices, process control systems. Man-machine interactions.

References:

1. *Run-time structures in programming languages:* Galler (1970); Rosen (1967) Pt. 2, 3; Wegner (1968).
2. *Communication, linking, and sharing of programs and data:* Daley and Dennis (1968); Gear (1969); Knuth (1968); Martin (1967); Wegner (1968).
3. *Interface design:* Dijkstra (1968a); Martin (1967).
4. *Program documentation:* Walsh (1969).
5. *Program debugging and testing:* Dijkstra (1968a); Hassitt (1967); Martin (1967); Van Horn (1968).
6. *Programming style and aesthetics:* Dijkstra (1968a); Knuth (1968); Wirth (1971).
7. *Selected examples:* Head (1971); Martin (1967).

Bibliography: The references are to the combined bibliography given at the end of Course Group UC.

UC8. Programming Structures and Techniques (2-2-3)

Prerequisite: elementary computer programming.

Approach: The structures which may be used to represent the information involved in solving problems are presented.

This course brings the student to grips with the actual problems encountered in designing, implementing, and modifying systems of computer programs. The concept of programming style should permeate most of the material presented. Careful verification of program operation and documentation of programs should be emphasized. Much of the course, particularly in the laboratory sessions, may be devoted to the actual implementation of the programs. It would be useful to have an exercise in which each student must modify a program written by someone else.

Content:

1. *Basic concepts of information (20%)*
Representation of information outside and inside the computer. Bits, bytes, fields, items, records, files. Numbers and characters. Use of pointers or linkage variables to represent structure. Arrays and tables. Scatter storage: hash table formats, hashing functions.
 2. *Storage management (5%)*
Static versus dynamic allocation. Stacks and available space lists. Explicit release of available space, reference counts, and garbage collection. Coalescing adjacent free space. Variable block size—stratified available space lists, the buddy system.
 3. *Programming language implementation structures (10%)*
Examples of the implementation of modeling structures in higher-level languages. FORTRAN, PL/I arrays. SNOBOL and PL/I strings. Lists. Tables and records in PL/I, COBOL.
 4. *Sorting and searching (5%)*
Sorting methods. Comparative efficiency of sorting methods. Use of sort packages. Linear search, binary search, indexed search, and other searching methods. Tradeoffs between sorting effort and searching effort. Effect of information structures on sorting and searching techniques.
 5. *Examples of the use of information structures (20%)*
Representation of information during execution; implementation of higher-level language data structures. Organization of an inverted file for document retrieval. Examples in graphic manipulation systems, data management systems.
 6. *Communication, linking, and sharing of programs and data (5%)*
Separation of program and data segments. Common (global) versus local data. Block structures—internal and external procedures. Subroutines as linkage structures. Sharing of code—reentrancy, recursion, pure procedures. Table driven programs. Asynchronous versus synchronous control. Cooperating processes.
 7. *Interface design (10%)*
Parameters, work space, automating, and documenting interfaces. Control blocks.
 8. *Program documentation, debugging, and testing (25%)*
Self-documenting programs. Levels of detail in documentation. Motivation for documentation—maintenance and modification of programs. Automating the debugging process. Symbolic debugging aids. Automatic generation of test data and expected results. Analysis of testing procedures. Hierarchical testing. Exhaustive testing vs. random sampling. Testing of communications programs. Simulation as a tool. Abnormal condition handling.
- References:** Bertziss (1971) or Knuth (1968) are most suitable candidates for use as texts in this course, supplemented by additional readings in a few topics.
1. *Basic concepts of information:* Bertziss (1971); Iverson (1962); Johnson (1970); Knuth (1968); Mealy (1967); Wegner (1968).
 2. *Storage management:* Bertziss (1971); Johnson (1970); Knuth (1968).
 3. *Programming language implementation structures:* Bertziss (1971); CODASYL (1971); Gordon (1969); Griswold et al. (1968); Iverson (1962); Rosen (1967) Pt. 3; Wegner (1968); Williams (1971).
 4. *Sorting and searching:* Bertziss (1971); Flores (1969); Gauthier and Ponto (1970); Hopgood (1969); Iverson (1962); Johnson (1970) Ch. 4, 5; Wegner (1968).
 5. *Examples of use of information structures:* Bertziss (1971); Dodd (1969); Knuth (1968); Wegner (1968); Williams (1971).
 6. *Communication, linking, and sharing of programs and data:* Daley and Dennis (1968); Gear (1969); Knuth (1968); Martin (1967); Wegner (1968).
 7. *Interface design:* Dijkstra (1968a); Martin (1967).
 8. *Program documentation, debugging and testing:* Dijkstra (1968a); Gruenberger (1968b); Hassitt (1967); Martin (1967); Van Horn (1968); Walsh (1969).

UC9. Computerware

Prerequisite: UC8.

Approach: Computer systems, their hardware and basic operating software, are studied, with attention to the human

factors involved in computer system operation and maintenance. Types of modules and types of system function mode (batch, interactive, online, etc.) should be carefully distinguished. The basic components of file and communication systems are presented and analyzed. The functioning of these systems as integral components of an information system is stressed. When possible, if a file management language is available, a small file system design and implementation project would be desirable.

Content:

1. *Hardware modules (15%)*

Processor, memory, input/output, mass storage, remote transmission modules; function and possible realization of each. Micro-programming. Styles of buffering, interfacing, communication and interrupt handling. Channel management, virtual configurations. Network and multiprocessor configurations.

2. *Execution software, multiprogramming, and multiprocessing (10%)*

General interpretive modules for execution support, e.g. list processors. Modules for memory management in real and virtual memory systems. Processor and channel modules for support of input/output, mass storage and remote transmission units in real and virtual configurations. Concepts of multiply-reentrant programs and cooperating processes. Levels of multiaccessing and multiplexing. Batch and interactive modes.

3. *Operation software (10%)*

Loading, interrupt monitoring, diagnostic modules. Scheduling, resource allocation, performance monitoring packages. Concepts of state resurrection and interprocess protection.

4. *Data and program handling software (15%)*

Media and format conversion modules. File handling facilities. Control specifications for datasets. Translating, compiling, generating modules. Editing and debugging facilities. Linkage and job control specifications for subroutines, and standard packages. Problems of identification and security.

5. *Functions of file and communication systems (10%)*

Role of information acquisition, storage and transmission in an organization. Typical operations in file systems: file creation, maintenance, interrogation. Typical operations using communication systems. Issues of information availability, privacy, security.

6. *File systems (10%)*

Characteristics of auxiliary storage devices. Capacity, access, cost. Device types: tape, disk, mass storage. Data fields, records, files, hierarchies of files. Directories and indices, inverted files. Structure and access: sequential, direct indexed sequential, randomized, randomized with buckets. Storage allocation and control techniques. Estimating capacity and timing requirements.

7. *Review of data management systems and analysis (15%)*

Generalized data management systems. Directory maintenance, query languages, data description, job management. Characteristics of available systems.

8. *Review of communication systems (5%)*

Theoretical concepts: channels and channel capacity, noise, error detection and correction. Existing communication facilities; line types, exchanges; utilities and tariffs. Transmission modes. Line termination and terminal devices. Single line: point-to-point, multidrop. Networks: centralized, decentralized, distributed. Control and protocol; acknowledgment, wait-requests, contention, polling. Switched, store-and-forward. Data concentrators and distributors. Estimating line and terminal requirements.

9. *Examples (10%)*

The data base concept: integrated data approach, coordination, control, multiple use of data. The data administrator; the computer utility. System resiliency and integrity, privacy, and security considerations.

References: There is no single introductory text for a combined hardware/software course at this level. The references given for Curriculum 68 courses I3 and I4 are in general relevant. Text materials may also be drawn from the operating manuals of whatever large-scale computer system is available for use in the course.

1. *Hardware modules:* Bell and Newell (1971); Buchholz (1962); Gear (1969); Husson (1970); Martin (1967).

2. *Execution software, multiprogramming and multiprocessing:* Daley and Dennis (1968); Denning (1970); Dijkstra (1968b); Rosen (1967) Pt. 5; Stimler (1969); Watson (1970); Wilkes (1967).

3. *Operation software:* Rosin (1969).

4. *Data and program handling software:* Rosen (1967).

5. *Function of file and communications systems:* Gruenberger

(1969); Martin (1967); Meadow (1967); Minker and Sable (1967); Salton (1968); Senko (1969).

6. *File systems:* CODASYL (1969); CODASYL (1971); Dodd (1969); IBM (1969); IFIP (1969); Martin (1967); Meadow (1967); Salton (1968); Senko (1969).

7. *Review of data management systems and analysis:* CODASYL (1969); CODASYL (1971); Gruenberger (1969); IFIP (1969); McGee (1968); Martin (1967); Senko (1969).

8. *Review of communication systems:* Davenport (1971); Gentle (1965); Martin (1967); Martin (1969).

9. *Examples:* Martin (1967); Parkhill (1966).

Bibliography: The references are to the combined bibliography given below.

Combined Bibliography—Course Group UC

Bell, C.G., and Newell, A. (1971) *Computer Structures: Readings and Examples*. McGraw-Hill, New York. CR 12, 5(71)21, 279; also CR 12, 7(71)21, 618.

A systematic treatment of computer architecture, with examples drawn from existing computer families. A definitive work, containing notational and pedagogical innovations which should set the style for future works in this area.

Bertziss, A.T. (1971) *Data Structures: Theory and Practice*. Academic Press, New York CR 12, 11(71)22,086.

Combines an introduction to discrete structures with a treatment of modeling and storage structures. Example program segments use FORTRAN; last chapter describes several other programming languages for information structures. Contains an extensive bibliography.

Buchholz, W. (Ed.) (1962) *Planning a Computer System*. McGraw-Hill, New York, CR 4, 6(63)4786.

An earlier but still useful discussion of how a computer hardware system is designed. Based on the development of the IBM Stretch computer.

CODASYL Data Base Task Group (1969) *October 69 Report*. Report to the CODASYL Programming Language Committee. available through ACM. CR 11, 5(70)19,080.

Contains a proposal for a Data Description Language for describing a data base and a Data Manipulation Language which when associated with the facilities of a host language, allows manipulation of data bases described in the Data Description Language.

CODASYL Systems Committee (1971) *Feature Analysis of Generalized Data Base Management Systems*. Technical report, available from ACM, New York.

Gives the reader a good feel for the state of the art in some widely-used generalized data base management systems. The introduction to this report also appears in *Comm. ACM* 14, 5, 308-318.

Cuadra, C.A. (Ed.) (1966-1971) *Annual Review of Information Science and Technology*. Vols. 1-6, CR 8, 1(67)11,128 (Vol. 1); CR 10, 4(69)16,550 (Vol. 3); CR 11, 7(70)19,391 (Vol. 4).

An excellent survey and review publication, covering mainly information storage and retrieval systems.

Daley, R.C., and Dennis, J.B. (1968) Virtual memory, processes, and sharing in MULTICS. *Comm. ACM* 11, 5, 306-12. CR 9, 8(68)14,978.

Discusses the concepts of dynamic linking and loading, and the sharing of procedures and data in virtual memory.

Davenport, W.P. (1971) *Modern Data Communications*. Hayden, New York.

An introductory textbook covering many topics in data communications in an elementary way.

Denning, P.J. (1970) Virtual memory. *Computing Surveys* 2, 3, 153-89. CR 12, 4(71)21,031.

A thorough treatment of this fundamental concept for design of multiprogramming systems.

Dijkstra, E.W. (1968a) The structure of "THE"-multiprogramming system. *Comm. ACM* 11, 5, 341-46.

The techniques described here for program design, implementation, and verification are quite useful in illustrating program aesthetics.

- Dijkstra, E.W. (1968b) Co-operating sequential processes. In Genuys (1968).
A definitive article which sets forth the basic aspects of concurrently running processes in computer systems.
- Dodd, G.G. (1969) Elements of data management systems. *Computing Surveys* 1, 2, 117-33. *CR* 10, 11(69)17, 780.
Title is misleading. Actually a presentation of information structures in the context of data management systems.
- Flores, I. (1969a) *Computer Sorting*. Prentice-Hall, Englewood Cliffs, N.J. *CR* 10, 1(69)16,053.
Presents various sorting techniques, with assembly language level procedures.
- Galler, B., and Perlis, A. (1970) *A View of Programming Languages*. Addison-Wesley, Reading, Mass.
- Gauthier, R., and Ponto, S. (1970) *Designing Systems Programs*. Prentice-Hall, Englewood Cliffs, N.J. *CR* 12, 3(71)20,829.
See particularly Ch. 7 (Data Representation) and Ch. 8 (Search Structures).
- Gear, C.W. (1969) *Computer Organization and Programming*. McGraw-Hill, New York. *CR* 10, 9(69)17,372.
A good text in assembly-level programming, which treats both hardware and basic software in this context.
- Genuys, F. (Ed.) (1968) *Programming Languages*. Academic Press, New York.
- Gordon, G. (1969) *System Simulation*. Prentice-Hall, Englewood Cliffs, N.J. *CR* 11, 3(70)18,682.
- Griswold, R.E., Poage, J.F., and Polonsky, J.P. (1968) *The SNOBOL 4 Programming Language*. Prentice-Hall, Englewood Cliffs, N.J. *CR* 10, 11(69)17,858.
- Gruenberger, F. (Ed.) (1969) *Critical Factors in Data Management*. Prentice-Hall, Englewood Cliffs, N.J. *CR* 11, 2(70)18,384.
A collection of symposium papers.
- Hassitt, A. (1967) *Computer Programming and Computer Systems*. Academic Press, New York. *CR* 8, 4(67)12,355.
Ch. 8 on debugging philosophy and Ch.9 on the dynamic use of storage are most useful.
- Head, R.V. (1971) *A Guide to Packaged Systems*. Wiley, New York.
- Hopgood, F.R.A. (1969) *Compiling Techniques*. American Elsevier, New York. *CR* 10, 11(69)17,773.
Distinguishes between abstract data structures and internal storage structures. Ch. 2 (Data Structures), Ch. 3 (Data Structure Mappings), and Ch. 4 (Tables) are particularly relevant here.
- Husson, S. (1970) *Microprogramming: Principles and Practices*. Prentice-Hall, Englewood Cliffs, N.J.
Contemporary treatment of the implementation of computer control, using several existing processor designs as extended examples.
- IBM Corporation. (1969) *File Design Handbook*. Information Sciences Depart., IBM Research, San Jose, Calif.
A prototype file design handbook with special emphasis on providing equations, guidelines, and simulation data for use by the file designed in meeting user constraints on cost, storage capacity, response time, etc.
- IFIP. (1969) *File Organization*. Selected papers from File 68—an I.A.G. Conference. Swets and Zeitlinger N.V., Amsterdam, The Netherlands.
Papers ranging from the nature of management and information systems, and details of file structure design and programming support systems, through specific case studies.
- Iverson, K.E. (1962) *A Programming Language*. Wiley, New York.
Contains considerable material on data structures, graphs, trees, and sorting, as well as descriptions of these in APL.
- Johnson, L.R. (1970) *System Structure in Data, Programs and Computers*. Prentice-Hall, Englewood Cliffs, N.J. *CR* 12, 1(71)20,504. Systematic treatment of much of computer science, taking the tree as a basic structural element.
- Knuth, D.E. (1968) *The Art of Computer Programming, Vol. 1: Fundamental Algorithms*. Addison-Wesley, Reading, Mass. *CR* 9, 6(68)14,505.
An extensive compendium (Ch. 2. Information Structures) of information and techniques on data structures and storage management. Does not distinguish between modeling and implementation structures. Section 1.4 on subroutines, co-routines, and linking is must reading. Other related topics are thoroughly covered. Many good examples and exercises.
- Martin, J. (1967) *Design of Real-Time Computer Systems*. Prentice-Hall, Englewood Cliffs, N.J. *CR* 9, 2(68)13,607.
A thorough treatment of analysis and design methodology for implementing real-time computer systems. Contains much material on both file and communications subsystems. Ch. 2, 3, and 10 on basic techniques and building blocks for these programs are quite good. Ch. 9 on the dynamic use of memory is also relevant.
- Martin, J. (1969) *Telecommunications and the Computer*. Prentice-Hall, Englewood Cliffs, N.J. *CR* 11, 8(70)19,602-19,603.
See annotation in bibliography for Course Group D.
- Meadow, C.T. (1967) *The Analysis of Information Systems*. Wiley, New York. *CR* 9, 8(68)14,939.
Subtitled "A Programmer's Introduction to Information Retrieval." A thoughtful presentation from the standpoints of both library science and computer science.
- Mealy, G.H. (1967) Another look at data. *Proc. AFIPS 1967 FJCC*, Vol. 31, AFIPS Press, Montvale, N.J., 525-34.
Sketches a theory of data based on relations.
- Minker, J., and Sable, J. (1967) File organization and data management. In *Cuadra* (1967), Vol. 2, 123-60.
A report of then-recent developments in file organization and data management, organized in a tutorial and expository framework, with an extensive bibliography.
- Morris, R. (1968) Scatter storage techniques. *Comm. ACM* 11, 1, 38-44.
Surveys hashing schemes for symbol table algorithms. Presents analytic formulations of processing requirements.
- Parkhill, D. (1966) *The Challenge of the Computer Utility*. Addison-Wesley, Reading, Mass. *CR* 8 1(67)11,053.
Discusses the history, technology, economic, and legal aspects of computer utilities.
- Rosen, S. (Ed.) (1967) *Programming Systems and Languages*. McGraw-Hill, New York. *CR* 10, 1(69)15,975.
Pt. 2 of this collection contains reprint articles on the major programming languages, and Pt. 3 articles on compiling and assembling. Pt. 4 contains papers of historical interest on various string and list processing languages. Pt. 5 is devoted to operating systems.
- Rosin, R.F. (1969) Supervisory and monitor systems. *Computing Surveys* 1, 1, 37-54. *CR* 10, 8(69)17, 284.
A survey of operating system development, tracing the evolution from the earliest crude monitor systems to the present.
- Salton, G.J. (1968) *Automatic Information Organization and Retrieval*. McGraw-Hill, New York, *CR* 10, 6(69)16,841.
Concentrates on automatic computer-based information retrieval systems. Includes a selective bibliography in information storage and retrieval and related topics.
- Senko, M.E. (1969) File organization and management information systems. In *Cuadra* (1969), Vol. 4, 111-43.
A review of management information systems applications and structure, viewed from the standpoints of both the information scientist and the systems programmer. Includes an extensive bibliography.
- Stimler, S. (1969) *Real-Time Data-Processing Systems*. McGraw-Hill, New York. *CR* 10, 9(69)17,391.
A text on hardware configuration design, emphasizing file and communications subsystems. Develops and illustrates many evaluation and optimization techniques.
- Van Horn, E.C. (1968) Three criteria for designing computer systems to facilitate debugging. *Comm. ACM* 11, 5, 360-64. *CR* 9, 11(68)5,580.
- Walsh, D. (1969) *A Guide for Software Documentation*. Inter-ACT, McGraw-Hill, New York. *CR* 11, 7(70)19,392.
Contains a number of models for the design of documentation procedures.
- Wegner, P. (1968) *Programming Languages, Information Structures, and Machine Organization*. McGraw-Hill, New York. *CR* 10, 2(69)6,228.
There is a good section on coroutines, tasks, and asynchronous processing (4.10). Includes an extensive bibliography.
- Williams, R. (1971) A survey of data structures for computer graphics systems. *Computing Surveys* 3, 1, 1-21. *CR* 12, 7(71)21,621.
Includes an extensive bibliography.
- Wirth, N. (1971) Program development by stepwise refinement. *Comm. ACM* 14, 4, 321-27. *CR* 12, 8(71)21,630.
An interesting exposition of the program design process.

CURRICULUM '78

Recommendations for the Undergraduate Program in Computer Science

A Report of the ACM Curriculum Committee on Computer Science

Editors: Richard H. Austing, University of Maryland
Bruce H. Barnes, National Science Foundation
Della T. Bonnette, University of Southwestern Louisiana
Gerald L. Engel, Old Dominion University
Gordon Stokes, Brigham Young University

Contained in this report are the recommendations for the undergraduate degree program in Computer Science of the Curriculum Committee on Computer Science (C³S) of the Association for Computing Machinery (ACM).

The core curriculum common to all computer science undergraduate programs is presented in terms of elementary level topics and courses, and intermediate level courses. Elective courses, used to round out an undergraduate program, are then discussed, and the entire program including the computer science component and other material is presented. Issues related to undergraduate computer science education, such as service courses, supporting areas, continuing education, facilities, staff, and articulation are presented.

Key Words and Phrases: computer sciences courses, computer science curriculum, computer science education, computer science undergraduate degree programs, service courses, continuing education

CR Categories: 1.52

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

Contents

1. Introduction
 2. Core Curriculum
 - 2.1 Introduction
 - 2.2 Objectives
 - 2.3 Elementary Material
 - 2.4 Implementation Considerations
 - 2.5 Sample Elementary Level Courses
 - 2.6 Sample Intermediate Level Courses
 3. Computer Science Electives
 - 3.1 Introduction
 - 3.2 Elementary Level
 - 3.3 Advanced Level
 4. The Undergraduate Program
 - 4.1 Introduction
 - 4.2 Computer Science Requirements and Electives
 - 4.3 Mathematics Requirements
 - 4.4 Other Requirements and Electives
 5. Service Courses
 - 5.1 Introduction
 - 5.2 General Service Courses
 - 5.3 Supporting Areas
 - 5.4 Continuing Education
 6. Other Considerations
 - 6.1 Introduction
 - 6.2 Facilities
 - 6.3 Staff
 - 6.4 Articulation
- References
Appendix

1. Introduction

Curriculum development work in computer science has been a continuing effort of the Curriculum Committee on Computer Science (C³S) of the Association for Computing Machinery (ACM). The work leading to the material presented in this report was started under the chairmanship of C³S of Preston Hammer, and continued when John Hamblen was appointed chairman in 1976.

In the time since the publication of "Curriculum '68" [1] by C³S, many significant developments have occurred within computer science education, and many educational efforts have been undertaken by C³S, other groups within ACM, and other professional organizations. As part of the background work in preparation of this report, an extensive survey of the literature of computer science education since "Curriculum '68" was prepared and published [2]. The efforts of C³S since 1968 are summarized in this document.

The writing group, in its preparation of this set of recommendations, paid considerable attention to the developments as reported in the literature, and to informal comments received regarding "Curriculum '68." In addition to this, a variety of individuals, representing many different types of institutions, and many different interests within computer science, were brought into C³S meetings and working sessions to present their ideas. A working draft of the report was prepared and published in the June 1977 *SIGCSE Bulletin* in order that the material receive as wide a distribution as possible, and to provide an opportunity for input from interested individuals. Prior to the publication of the working paper, draft reports on specific areas were widely circulated and numerous panel and discussion sessions were held both to inform interested parties of the thinking of the Committee and to allow for comments and suggestions on the work done to that point.

The wide circulation of the various drafts and working papers resulted in numerous suggestions and constructive criticisms, many of which have been incorporated into this final document. In addition to this input, a relationship of mutual benefit has developed by interaction with the parallel, but independent, development of the Model Curricula Subcommittee of the IEEE Computer Society leading to the publication of their curriculum guidelines in *Computer Science and Engineering* [3].

The writing group is most grateful to all those individuals who contributed to the effort. The Appendix contains the names and affiliations of those people who contributed by serving on C³S, by supplying course outlines, by supplying comments on the draft report, and in other ways contributing to the final version presented here. The Committee, of course, assumes full responsibility for the substance of this material and the recommendations contained herein.

The report first presents the core curriculum common to all computer science undergraduate programs. This is presented in Section 2 in terms of elementary level

material and courses, and intermediate level courses. Section 3 presents computer science electives that may be used to round out an undergraduate program. In Section 4, the full course of study is presented which includes the computer science component, and other material necessary in a program at the bachelor degree level. The important areas of service courses, including general service courses, supporting areas, and continuing education are discussed in Section 5. The report concludes by addressing the areas of facilities, staff, and articulation in Section 6.

In studying this report, it should be recognized that it is a set of guidelines, prepared by a group of individuals working in a committee mode. As such, the recommendations will not satisfy everyone, nor is it intended that they be appropriate to all institutions. It is the hope of the Committee that this report will further stimulate computer science educators to think about their programs and, as appropriate, to share their thinking with others. If this is done, the primary objective of the preparation of these guidelines will have been met.

2. Core Curriculum

2.1 Introduction

Within the present work, C³S has considered the classification scheme of computer science as defined in "Curriculum '68" with a view to isolating those areas which should be common to all computer science undergraduate degree programs.

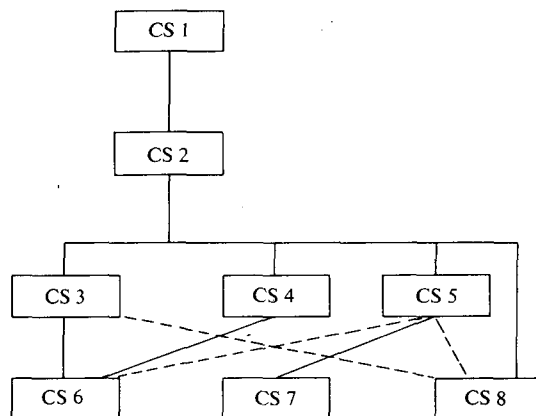
The core curriculum, described in this section, represents this refinement. The material is divided into a section on elementary material, including the specifications of topics at this level and the description of five sample courses, and the intermediate levels, including the description of three sample courses. This collection of eight courses represents one way to include the required core material in the computer science undergraduate major.

While the course material is detailed later on in the section, to gain perspective the eight courses (three semester hours each) are listed here:

- CS 1. Computer Programming I
- CS 2. Computer Programming II
- CS 3. Introduction to Computer Systems
- CS 4. Introduction to Computer Organization
- CS 5. Introduction to File Processing
- CS 6. Operating Systems and Computer Architecture I
- CS 7. Data Structures and Algorithm Analysis
- CS 8. Organization of Programming Languages

The structuring of the courses as to prerequisites is shown in Figure 1. The solid lines represent required prerequisites, while the dashed lines represent highly recommended prerequisites. This diagram includes courses representing only the computer science material considered to be essential to the program. The entire program, including relevant mathematics requirements, is illustrated in Figure 2 on page 160.

Fig. 1. Computer science core curriculum.



The discussion of the core course material in this section concentrates on the computer science components which are necessary for the undergraduate program. The relationship of this material to two-year programs (especially transfer programs) and the developing high school programs will be considered in Section 6.4.

The elementary core material represents subject matter necessary for all students in computer science in order to be able to achieve the objectives of the undergraduate major. The intermediate level core material follows naturally by providing the students who have been equipped with the basics of the field with the tools to be operational computer scientists.

2.2 Objectives

The core material is required as a prerequisite for advanced courses in the field and thus it is essential that the material be presented early in the program. In learning this material, the computer science student should be provided with the foundation for achieving at least the objectives of an undergraduate degree program that are listed below.

Computer science majors should:

1. be able to write programs in a reasonable amount of time that work correctly, are well documented, and are readable;
2. be able to determine whether or not they have written a reasonably efficient and well organized program;
3. know what general types of problems are amenable to computer solution, and the various tools necessary for solving such problems;
4. be able to assess the implications of work performed either as an individual or as a member of a team;
5. understand basic computer architectures;
6. be prepared to pursue in-depth training in one or more application areas or further education in computer science.

It should be recognized that these alone do not represent the total objectives of an undergraduate program, but only those directly related to the computer science component. Material addressing other requirements and electives is covered in Section 4.4.

2.3 Elementary Material

In order to facilitate the attainment of the objectives above, computer science majors must be given a thorough grounding in the study of the implementation of algorithms in programming languages which operate on data structures in the environment of hardware. Emphasis at the elementary level then should be placed on algorithms, programming, and data structures, but with a good understanding of the hardware capabilities involved in their implementation.

Specifically, the following topics are considered elementary. They should be common to all undergraduate programs in computer science.

Programming Topics

- P1. Algorithms: includes the concept and properties of algorithms; the role of algorithms in the problem solving process; constructs and languages to facilitate the expression of algorithms.
- P2. Programming Languages: includes basic syntax and semantics of a higher level (problem oriented) language; subprograms; I/O; recursion.
- P3. Programming Style: includes the preparation of readable, understandable, modifiable, and more easily verifiable programs through the application of concepts and techniques of structured programming; program documentation; some practical aspects of proving programs correct. (Note: Programming style should pervade the entire curriculum rather than be considered as a separate topic.)
- P4. Debugging and Verification: includes the use of debugging software, selection of test data; techniques for error detection; relation of good programming style to the use of error detection; and program verification.
- P5. Applications: includes an introduction to uses of selected topics in areas such as information retrieval, file management, lexical analysis, string processing and numeric computation; need for and examples of different types of programming languages; social, philosophical, and ethical considerations.

Software Organization

- S1. Computer Structure and Machine Language: includes organization of computers in terms of I/O, storage, control and processing units; register and storage structures, instruction format and execution; principal instruction types; machine arithmetic; program control; I/O operations; interrupts.
- S2. Data Representation: includes bits, bytes, words and other information structures; number representation; representation of elementary data structures; data transmission, error detection and correction; fixed versus variable word lengths.
- S3. Symbolic Coding and Assembly Systems: includes mnemonic operation codes; labels; symbolic addresses and address expressions; literals; extended machine operations and pseudo operations; error flags and messages; scanning of symbolic instruc-

- tions and symbol table construction; overall design and operation of assemblers, compilers, and interpreters.
- S4. Addressing Techniques: includes absolute, relative, base associative, indirect, and immediate addressing; indexing; memory mapping functions; storage allocation, paging and machine organization to facilitate modes of addressing.
 - S5. Macros: includes definition, call, expansion of macros; parameter handling; conditional assembly and assembly time computation.
 - S6. Program Segmentation and Linkage: includes subroutines, coroutines and functions; subprogram loading and linkage; common data linkage transfer vectors; parameter passing and binding; overlays; re-entrant subprograms; stacking techniques; linkage using page and segment tables.
 - S7. Linkers and Loaders: separate compilation of subroutines; incoming and outgoing symbols; relocation; resolving intersegment references by direct and indirect linking.
 - S8. Systems and Utility Programs: includes basic concepts of loaders, I/O systems, human interface with operating systems; program libraries.

Hardware Organization

- H1. Computer Systems Organization: includes characteristics of, and relationships between I/O devices, processors, control units, main and auxiliary storage devices; organization of modules into a system; multiple processor configurations and computer networks; relationship between computer organization and software.
- H2. Logic Design: includes basic digital circuits; AND, OR, and NOT elements; half-adder, adder, storage and delay elements; encoding-decoding logic; basic concepts of microprogramming; logical equivalence between hardware and software; elements of switching algebra; combinatorial and sequential networks.
- H3. Data Representation and Transfer: includes codes, number representation; flipflops, registers, gates.
- H4. Digital Arithmetic: includes serial versus parallel adders; subtraction and signed magnitude versus complemented arithmetic; multiply/divide algorithms; elementary speed-up techniques for arithmetic.
- H5. Digital Storage and Accessing: includes memory control; data and address buses; addressing and accessing methods; memory segmentation; data flow in multimemory and hierarchical systems.
- H6. Control and I/O: includes synchronous and asynchronous control; interrupts; modes of communication with processors.
- H7. Reliability: includes error detection and correction, diagnostics.

Data Structures and File Processing

- D1. Data Structures: includes arrays, strings, stacks, queues, linked lists; representation in memory; algorithms for manipulating data within these structures.

- D2. Sorting and searching: includes algorithms for in-core sorting and searching methods; comparative efficiency of methods; table lookup techniques; hash coding.
- D3. Trees: includes basic terminology and types; representation as binary trees; traversal schemes; representation in memory; breadth-first and depth-first search techniques; threading.
- D4. File Terminology: includes record, file, blocking, database; overall idea of database management systems.
- D5. Sequential Access: includes physical characteristics of appropriate storage media; sort/merge algorithms; file manipulation techniques for updating, deleting, and inserting records.
- D6. Random Access: includes physical characteristics of appropriate storage media; physical representation of data structures on storage devices; algorithms and techniques for implementing inverted lists, multi-lists, indexed sequential, hierarchical structures.
- D7. File I/O: includes file control systems (directory, allocation, file control table, file security); I/O specification statements for allocating space and cataloging files; file utility routines; data handling (format definition, block buffering, buffer pools, compaction).

2.4 Implementation Considerations

Throughout the presentation of the elementary level material, programming projects should be assigned; these projects should be designed to aid in the comprehension and use of language details, to exemplify the problem solving process, and/or to introduce more advanced areas of computer science.

Good programming style should be stressed in the teaching of all of this material. The discipline required to achieve style will promote the development of effective algorithms and should result in students writing correct, understandable programs. Thus emphasis in the programming exercises should be placed on efficient algorithms, structured programming techniques, and good documentation.

A specific course on structured programming, or on programming style, is not intended at the elementary level. The topics are of such importance that they should be considered a common thread throughout the entire curriculum and, as such, should be totally integrated into the curriculum. They provide a philosophy of discipline which pervades all of the course work.

Throughout the presentation of this elementary material, meaningful actual computer applications should be cited and reviewed. In the process of so doing, reference must be made to the social, philosophical, and ethical considerations involved in the applications. Like structured programming, these issues are of such importance to the development of the computer scientist that they must permeate the instruction at this level.

It would be desirable, though not necessary, for the

computer science major to be familiar with all of the elementary level topics before taking intermediate level courses. This, however, may not always be possible. Factors influencing how and when courses are offered which include the material are: the purpose and circumstances of a particular department within the context of its educational institution, the availability of computer resources, and whether an institution is on the quarter or semester system.

Most courses at this level should include laboratory sessions. These laboratories provide the student with the opportunity to gain practical experience by actually solving problems on the computer. Laboratory sessions should be implemented in such a way that the student can develop good programming techniques under close supervision. The instructor may or may not be the same as for the lecture portion of the course. The absence of a specific laboratory in a course description does not imply that programming should not be required.

2.5 Sample Elementary Level Courses

The following set of courses is provided merely as a sample to illustrate one of the ways in which core material at the elementary level might be presented. Other implementations are possible. No matter what implementation is attempted, however, all of the elementary material specified in Section 2.3 should be included so that students are equipped with adequate background for intermediate and advanced level material.

Each course described in the sample set is assumed to be offered on a semester basis. Suggested numbers of hours of credit are given in parentheses immediately after the course titles. For example, (2-2-3) indicates two hours of lectures and two hours of laboratory per week for a total of three semester hours of credit.

CS 1. Computer Programming I (2-2-3)

The objectives of this course are:

- (a) to introduce problem solving methods and algorithm development;
- (b) to teach a high level programming language that is widely used; and
- (c) to teach how to design, code, debug, and document programs using techniques of good programming style.

COURSE OUTLINE

The material on a high level programming language and on algorithm development can be taught best as an integrated whole. Thus the topics should not be covered sequentially. The emphasis of the course is on the techniques of algorithm development and programming with style. Neither esoteric features of a programming language nor other aspects of computers should be allowed to interfere with that purpose.

TOPICS

- A. *Computer Organization.* An overview identifying components and their functions, machine and assembly languages. (5%)

- B. *Programming Language and Programming.* Representation of integers, reals, characters, instructions. Data types, constants, variables. Arithmetic expression. Assignment statement. Logical expression. Sequencing, alternation, and iteration. Arrays. Subprograms and parameters. Simple I/O. Programming projects utilizing concepts and emphasizing good programming style. (45%)

- C. *Algorithm Development.* Techniques of problem solving. Flowcharting. Stepwise refinement. Simple numerical examples. Algorithms for searching (e.g. linear, binary), sorting (e.g. exchange, insertion), merging of ordered lists. Examples taken from such areas as business applications involving data manipulation, and simulations involving games. (45%)

- D. *Examinations.* (5%)

CS 2. Computer Programming II (2-2-3)

Prerequisite: CS 1

The objectives of this course are:

- (a) to continue the development of discipline in program design, in style and expression, in debugging and testing, especially for larger programs;
- (b) to introduce algorithmic analysis; and
- (c) to introduce basic aspects of string processing, recursion, internal search/sort methods and simple data structures.

COURSE OUTLINE

The topics in this outline should be introduced as needed in the context of one or more projects involving larger programs. The instructor may choose to begin with the statement of a sizeable project, then utilize structured programming techniques to develop a number of small projects each of which involves string processing, recursion, searching and sorting, or data structures. The emphasis on good programming style, expression, and documentation, begun in CS 1, should be continued. In order to do this effectively, it may be necessary to introduce a second language (especially if a language like Fortran is used in CS 1). In that case, details of the language should be included in the outline. Analysis of algorithms should be introduced, but at this level such analysis should be given by the instructor to the student.

Consideration should be given to the implementation of programming projects by organizing students into programming teams. This technique is essential in advanced level courses and should be attempted as early as possible in the curriculum. If large class size makes such an approach impractical, every effort should be made to have each student's programs read and critiqued by another student.

TOPICS

- A. *Review.* Principles of good programming style, expression, and documentation. Details of a second language if appropriate. (15%)
- B. *Structured Programming Concepts.* Control flow. Invariant relation of a loop. Stepwise refinement of

both statements and data structures, or top-down programming. (40%)

- C. *Debugging and Testing*. (10%)
- D. *String Processing*. Concatenation. Substrings. Matching. (5%)
- E. *Internal Searching and Sorting*. Methods such as binary, radix, Shell, quicksort, merge sort. Hash coding. (10%)
- F. *Data Structures*. Linear allocation (e.g. stacks, queues, deques) and linked allocation (e.g. simple linked lists). (10%)
- G. *Recursion*. (5%)
- H. *Examinations*. (5%)

CS 3. Introduction to Computer Systems (2-2-3)

Prerequisite: CS 2

The objectives of this course are:

- (a) to provide basic concepts of computer systems;
- (b) to introduce computer architecture; and
- (c) to teach an assembly language.

COURSE OUTLINE

The extent to which each topic is discussed and the ordering of topics depends on the facilities available and the nature and orientation of CS 4 described below. Enough assembly language details should be covered and projects assigned so that the student gains experience in programming a specific computer. However, concepts and techniques that apply to a broad range of computers should be emphasized. Programming methods that are developed in CS 1 and CS 2 should also be utilized in this course.

TOPICS

- A. *Computer Structure and Machine Language*. Memory, control, processing and I/O units. Registers, principal machine instruction types and their formats. Character representation. Program control. Fetch-execute cycle. Timing. I/O operations. (15%)
- B. *Assembly Language*. Mnemonic operations. Symbolic addresses. Assembler concepts and instruction format. Data-word definition. Literals. Location counter. Error flags and messages. Implementation of high level language constructs. (30%)
- C. *Addressing Techniques*. Indexing. Indirect Addressing. Absolute and relative addressing. (5%)
- D. *Macros*. Definition. Call. Parameters. Expansion. Nesting. Conditional assembly. (10%)
- E. *File I/O*. Basic physical characteristics of I/O and auxiliary storage devices. File control system. I/O specification statements and device handlers. Data handling, including buffering and blocking. (5%)
- F. *Program Segmentation and Linkage*. Subroutines. Coroutines. Recursive and re-entrant routines. (20%)
- G. *Assembler Construction*. One-pass and two-pass assemblers. Relocation. Relocatable loaders. (5%)
- H. *Interpretive Routines*. Simulators. Trace. (5%)
- I. *Examinations*. (5%)

CS 4. Introduction to Computer Organization (3-0-3) or (2-2-3)

Prerequisite: CS 2

The objectives of this course are:

- (a) to introduce the organization and structuring of the major hardware components of computers;
- (b) to understand the mechanics of information transfer and control within a digital computer system; and
- (c) to provide the fundamentals of logic design.

COURSE OUTLINE

The three main categories in the outline, namely computer architecture, arithmetic, and basic logic design, should be interwoven throughout the course rather than taught sequentially. The first two of these areas may be covered, at least in part, in CS 3 and the amount of material included in this course will depend on how the topics are divided between the two courses. The logic design part of the outline is specific and essential to this course. The functional, logic design level is emphasized rather than circuit details which are more appropriate in engineering curricula. The functional level provides the student with an understanding of the mechanics of information transfer and control within the computer system. Although much of the course material can and should be presented in a form that is independent of any particular technology, it is recommended that an actual, simple minicomputer or microcomputer system be studied. A supplemental laboratory is appropriate for that purpose.

TOPICS

- A. *Basic Logic Design*. Representation of both data and control information by digital (binary) signals. Logic properties of elemental devices for processing (gates) and storing (flipflops) information. Description by truth tables, Boolean functions and timing diagrams. Analysis and synthesis of combinatorial networks of commonly used gate types. Parallel and serial registers. Analysis and synthesis of simple synchronous control mechanisms; data and address buses; addressing and accessing methods; memory segmentation. Practical methods of timing pulse generation. (25%)
- B. *Coding*. Commonly used codes (e.g. BCD, ASCII). Parity generation and detection. Encoders, decoders, code converters. (5%)
- C. *Number Representation and Arithmetic*. Binary number representation, unsigned addition and subtraction. One's and two's complement, signed magnitude and excess radix number representations and their pros and cons for implementing elementary arithmetic for BCD and excess-3 representations. (10%)
- D. *Computer Architecture*. Functions of, and communication between, large-scale components of a computer system. Hardware implementation and sequencing of instruction fetch, address construction, and instruction execution. Data flow and control

block diagrams of a simple processor. Concept of microprogram and analogy with software. Properties of simple I/O devices and their controllers, synchronous control, interrupts. Modes of communications with processors. (35%)

- E. *Example.* Study of an actual, simple minicomputer or microcomputer system. (20%)
- F. *Examinations.* (5%)

CS 5. Introduction to File Processing (3-0-3)

Prerequisite: CS 2

The objectives of this course are:

- (a) to introduce concepts and techniques of structuring data on bulk storage devices;
- (b) to provide experience in the use of bulk storage devices; and
- (c) to provide the foundation for applications of data structures and file processing techniques.

COURSE OUTLINE

The emphasis given to topics in this outline will vary depending on the computer facilities available to students. Programming projects should be assigned to give students experience in file processing. Characteristics and utilization of a variety of storage devices should be covered even though some of the devices are not part of the computer system that is used. Algorithmic analysis and programming techniques developed in CS 2 should be utilized.

TOPICS

- A. *File Processing Environment.* Definitions of record, file, blocking, compaction, database. Overview of database management system. (5%)
- B. *Sequential Access.* Physical characteristics of sequential media (tape, cards, etc.). External sort/merge algorithms. File manipulation techniques for updating, deleting and inserting records in sequential files. (30%)
- C. *Data Structures.* Algorithms for manipulating linked lists. Binary, B-trees, B*-trees, and AVL trees. Algorithms for traversing and balancing trees. Basic concepts of networks (plex structures). (20%)
- D. *Random Access.* Physical characteristics of disk/drum and other bulk storage devices. Physical representation of data structure on storage devices. Algorithms and techniques for implementing inverted lists, multilist, indexed sequential, and hierarchical structures. (35%)
- E. *File I/O.* File control systems and utility routines, I/O specification statements for allocating space and cataloging files. (5%)
- F. *Examinations.* (5%)

2.6 Sample Intermediate Level Courses

Sample versions of three courses at the intermediate level are given to illustrate topics and material which should be required of all computer science majors. This material and the elementary level topics in Section 2.3

constitute the minimum requirements which should be common to all computer science undergraduate programs to achieve the basic objectives of those programs.

Courses which cover the intermediate level material contain a strong emphasis on fundamental concepts exemplified by various types of programming languages, architecture and operating systems, and data structures. Neither theoretical treatments nor case study approaches in and of themselves are adequate or appropriate at this level. Advanced level (elective) courses may be used for predominantly theoretical treatment of topics or for comprehensive case studies.

CS 6. Operating Systems and Computer Architecture I (2-2-3)

Prerequisite: CS 3 and CS 4

(CS 5 recommended)

The objectives of this course are:

- (a) to develop an understanding of the organization and architecture of computer systems at the register-transfer and programming levels of system description;
- (b) to introduce the major concept areas of operating systems principles;
- (c) to teach the inter-relationships between the operating system and the architecture of computer systems.

COURSE OUTLINE

This course should emphasize concepts rather than case studies. Subtleties do exist, however, in operating systems that do not readily follow from concepts alone. It is recommended that a laboratory requiring hands on experience be included with this course.

The laboratory for the course would ideally use a small computer where students could actually implement sections of operating systems and have them fail without serious consequences to other users. This system should have, at a minimum, a CPU, memory, disk or tape, and some terminal device such as a teletype or CRT. The second best choice for the laboratory experience would be a simulated system running on a larger machine.

The course material should be liberally sprinkled with examples of operating system segments implemented on particular computer system architectures. The interdependence of operating systems and architecture should be clearly delineated. Integrating these subjects at an early stage in the curriculum is particularly important because the effects of computer architecture on systems software has long been recognized. Also, modern systems combine the design of operating systems and the architecture.

TOPICS

- A. *Review.* Instruction sets. I/O and interrupt structure. Addressing schemes. Microprogramming. (10%)
- B. *Dynamic Procedure Activation.* Procedure activation and deactivation on a stack, including dynamic storage allocation, passing value and reference parameters, establishing new local environments, addressing mechanisms for accessing parameters (e.g. displays,

relative addressing in the stack). Implementing non-local references. Re-entrant programs. Implementation on register machines. (15%)

- C. *System Structure*. Design methodologies such as level, abstract data types, monitors, kernels, nuclei, networks of operating system modules. Proving correctness. (10%)
- D. *Evaluation*. Elementary queueing, network models of systems, bottlenecks, program behavior, and statistical analysis. (15%)
- E. *Memory Management*. Characteristics of the hierarchy of storage media, virtual memory, paging, segmentation. Policies and mechanisms for efficiency of mapping operations and storage utilization. Memory protection. Multiprogramming. Problems of auxiliary memory. (20%)
- F. *Process Management*. Asynchronous processes. Using interrupt hardware to trigger software procedure calls. Process stateword and automatic SWITCH instructions. Semaphores. Ready lists. Implementing a simple scheduler. Examples of process control problems such as deadlock, producer/consumers, readers/writers. (20%)
- G. *Recovery Procedures*. Techniques of automatic and manual recovery in the event of system failures. (5%)
- H. *Examinations*. (5%)

CS 7. Data Structures and Algorithm Analysis (3-0-3)

Prerequisite: CS 5

The objectives of this course are:

- (a) to apply analysis and design techniques to nonnumeric algorithms which act on data structures;
- (b) to utilize algorithmic analysis and design criteria in the selection of methods for data manipulation in the environment of a database management system.

COURSE OUTLINE

The material in this outline could be covered sequentially in a course. It is designed to build on the foundation established in the elementary material, particularly on that material which involves algorithm development (P1, P3) and data structures and file processing (D1, D7). The practical approach in the earlier material should be made more rigorous in this course through the use of techniques for the analysis and design of efficient algorithms. The results of this more formal study should then be incorporated into data management system design decisions. This involves differentiating between theoretical or experimental results for individual methods and the results which might actually be achieved in systems which integrate a variety of methods and data structures. Thus, database management systems provide the applications environment for topics discussed in the course.

Projects and assignments should involve implementation of theoretical results. This suggests an alternative way of covering the material in the course, namely to

treat concepts, algorithms, and analysis in class and deal with their impact on system design in assignments. Of course, some in-class discussions of this impact would occur, but at various times throughout the course rather than concentrated at the end.

TOPICS

- A. *Review*. Basic data structures such as stacks, queues, lists, trees. Algorithms for their implementation. (10%)
- B. *Graphs*. Definition, terminology, and property (e.g. connectivity). Algorithms for finding paths and spanning trees. (15%)
- C. *Algorithms Design and Analysis*. Basic techniques of design and analysis of efficient algorithms for internal and external sorting/merging/searching. Intuitive notions of complexity (e.g. NP-hard problems). (30%)
- D. *Memory Management*. Hashing. Algorithms for dynamic storage allocation (e.g. buddy system, boundary-tag), garbage collection and compaction. (15%)
- E. *System Design*. Integration of data structures, sort/merge/search methods (internal and external) and memory media into a simple database management system. Accessing methods. Effects on run time, costs, efficiency. (25%)
- F. *Examinations*. (5%)

CS 8. Organization of Programming Languages (3-0-3)

Prerequisite: CS 2 (CS 3 and CS 5 highly recommended)

The objectives of this course are:

- (a) to develop an understanding of the organization of programming languages, especially the run-time behavior of programs;
- (b) to introduce the formal study of programming language specification and analysis;
- (c) to continue the development of problem solution and programming skills introduced in the elementary level material.

COURSE OUTLINE

This is an applied course in programming language constructs emphasizing the run-time behavior of programs. It should provide appropriate background for advanced level courses involving formal and theoretical aspects of programming languages and/or the compilation process.

The material in this outline is not intended to be covered sequentially. Instead, programming languages could be specified and analyzed one at a time in terms of their features and limitations based on their run-time environments. Alternatively, desirable specification of programming languages could be discussed and then exemplified by citing their implementations in various languages. In either case, programming exercises in each language should be assigned to emphasize the implementations of language features.

TOPICS

- A. *Language Definition Structure*. Formal language concepts including syntax and basic characteristics of grammars, especially finite state, context-free, and ambiguous. Backus-Naur Form. A language such as Algol as an example. (15%)
- B. *Data Types and Structures*. Review of basic data types, including lists and trees. Constructs for specifying and manipulating data types. Language features affecting static and dynamic data storage management. (10%)
- C. *Control Structures and Data Flow*. Programming language constructs for specifying program control and data transfer, including DO . . . FOR, DO . . . WHILE, REPEAT . . . UNTIL, BREAK, subroutines, procedures, block structures, and interrupts. Decision tables, recursion. Relationship with good programming style should be emphasized. (15%)
- D. *Run-time Consideration*. The effects of the run-time environment and binding time on various features of programming languages. (25%)
- E. *Interpretative Languages*. Compilation vs. interpretation. String processing with language features such as those available in SNOBOL 4. Vector processing with language features such as those available in APL. (20%)
- F. *Lexical Analysis and Parsing*. An introduction to lexical analysis including scanning, finite state acceptors and symbol tables. An introduction to parsing and compilers including push-down acceptors, top-down and bottom-up parsing. (10%)
- G. *Examinations*. (5%)

3. Computer Science Electives

3.1 Introduction

In this section a variety of computer science electives will be considered which are appropriate at the elementary and advanced levels. Elective courses at the elementary level, while enhancing the program of a student, normally should not be used to meet the requirements of the major program. Elective courses at the advanced level should be selected to meet major requirements as well as to allow the student to explore particular areas of computer science in more detail.

3.2 Elementary Level

At the elementary level it would be highly desirable to provide a mechanism for offering courses in specific programming languages such as APL, Cobol, LISP, or PL/I which could be taken as electives by computer science majors or majors in other disciplines. The extent of the course, the number of credits offered and the prerequisites would depend on the language offered and the purpose for offering it. One convenient way to achieve this goal would be to include in the curriculum a Programming Language Laboratory for variable credit (i.e. one

to three semester hours). The prerequisite could be designated in general as "consent of instructor" or more specifically as CS 1 or CS 2 and the laboratory could be taken for repeated credit provided that different languages were taught. In addition to its function as an elective, the laboratory could be offered in conjunction with an intermediate or advanced course, thus enabling an instructor to require students to learn a specific language at the same time they take a course (e.g. LISP in the laboratory along with CS 7—Data Structures and Algorithm Analysis).

3.3 Advanced Level

Ten advanced level elective courses are specified. Computer Science departments should offer as many as possible of these courses on a regular basis, but few departments are expected to have sufficient resources to offer all, or even a large majority, of them. Possible additional courses which could be offered as special topics are listed in Section 3.4.

CS 9. Computers and Society (3-0-3)

Prerequisite: elementary core material

The objectives of this course are:

- (a) to present concepts of social value and valuations;
- (b) to introduce models which describe the impact of computers on society;
- (c) to provide a framework for professional activity that involves explicit consideration of and decisions concerning social impact;
- (d) to present tools and techniques which are applicable to problems posed by the social impact of computers.

Much debate surrounds the role of this course in the curriculum. While few will disagree that professional computer scientists should be instructed to evaluate social issues regarding that which they do, it has been argued that such a course is not a computer science course, but rather should be in the area of the social sciences. Another argument is presented which states that this material is so important that it should not merely be covered in a single course, but instead should be integrated throughout the curriculum. Although this latter argument has validity, it is difficult to insure sufficient coverage of topics when they are scattered throughout a number of courses. As a result it is recommended that this course be considered at least as a strongly recommended elective. If, in fact, the material to meet the above objectives is not covered in the other intermediate and advanced level courses in this program, then this course should be required.

A computer science major taking an advanced level computers and society course would be expected to be familiar with the elementary material described in the previous section. All of that material, however, is not necessarily prerequisite for such a course. The prerequisite should, in fact, be chosen in such a manner that non-majors would also be able to take the course. A mixture of majors in such a course would provide broadening

interchange and would benefit both the computer science students and the other majors. The course should be taught by the computer science faculty, but team-teaching with faculty from other disciplines should be encouraged. The course could be general and treat a number of computer impact topics, or specific, and treat in depth one of the topics (such as legal issues in computing). This recommendation is conditioned on the assumption that instructors who present material on societal impact, whether as an entire course or as part of other courses, will try to include both sides of or approaches to issues without instilling their own philosophical leanings on complex societal issues. For example, certain topics contain political overtones which should be discussed, but which, if not done carefully, can give the material a political science flavor it does not deserve.

A strict outline is not given. The number of topics and extent of coverage as well as the instructional techniques used can vary considerably and still meet the objectives of the course. A term project involving computer applications that are manifested in the local community is strongly recommended. Possible topics, but certainly not an exhaustive list, that could be included in such a course are as follows:

- A. History of computing and technology
- B. The place of the computer in modern society
- C. The computer and the individual
- D. Survey of computer applications
- E. Legal issues
- F. Computers in decision-making processes
- G. The computer scientist as a professional
- H. Futurists' views of computing
- I. Public perception of computers and computer scientists

CS 10. Operating Systems and Computer Architecture II (2-2-3)

Prerequisite: CS 6; Corequisite: a course in statistics

COURSE OUTLINE

This course continues the development of the material in CS 6. Emphasis should be on intrasystem communication.

TOPICS

- A. *Review.* I/O and interrupt structure. Addressing schemes. Memory management. (10%)
- B. *Concurrent Processes.* Concepts of processes in parallel. Problems associated with determinancy, freedom from deadlock, mutual exclusion, and synchronization. (15%)
- C. *Name Management.* Limitations of linear address space. Implementation of tree-structured space of objects for the support of modular programming. (15%)
- D. *Resource Allocation.* Queueing and network control policies. Concepts of system balancing and thrashing.

Job activation/deactivation. Process scheduling. Multiprogramming systems. (25%)

- E. *Protection.* Constraints for accessing objects. Mechanism to specify and enforce access rules. Implementation in existing systems. (15%)
- F. *Advanced Architecture and Operating Systems Implementations.* Pipelining and parallelism. User interface considerations. Introduction to telecommunications, networks (including minicomputers) and distributed systems. (15%)
- G. *Examinations.* (5%)

CS 11. Database Management Systems Design (3-0-3)

Prerequisites: CS 6 and CS 7

COURSE OUTLINE

This course should emphasize the concepts and structures necessary to design and implement a database management system. The student should become acquainted with current literature on the subject and should be given an opportunity to use a database management system if possible.

During the course the student should gain an understanding of various physical file organization and data organization techniques. The concept of data models should be covered and the network, relational, and hierarchical data models should be explored. Examples of specific database management systems should be examined and related to the data models discussed. The student should become familiar with normalized forms of data relations including canonical schema representations. Techniques of systems design and implementation should be discussed and practiced. Data integrity and file security techniques should be explored. The major experience of the course should be the design and implementation of a simple database management system that would include file security and some form of query into the system.

TOPICS

- A. *Introduction to Database Concepts.* Goals of DBMS including data independence, relationships, logical and physical organizations, schema and subschema. (5%)
- B. *Data Models.* Hierarchical, network, and relational models with a description of the logical and data structure representation of the database system. Examples of implementations of the various models. (15%)
- C. *Data Normalization.* First, second, and third normal forms of data relations. Canonical schema. Data independence. (5%)
- D. *Data Description Languages.* Forms, applications, examples, design strategies. (10%)
- E. *Query Facilities.* Relational algebra, relational calculus, data structures for establishing relations. Query functions. Design and translation strategies. (15%)

- F. *File Organization*. Storage hierarchies, data structures, multiple key systems, indexed files, hashing. Physical characteristics. (25%)
- G. *Index Organization*. Relation to files. Inverted file systems. Design strategies. (5%)
- H. *File Security*. Authentication, authorization, transformation, encryptions. Hardware and software techniques. Design strategies. (10%)
- I. *Data Integrity and Reliability*. Redundancy, recovery, locking, and monitoring. (5%)
- J. *Examinations*. (5%)

CS 12. Artificial Intelligence (3-0-3)

Prerequisite: CS 7

COURSE OUTLINE

This course introduces students to basic concepts and techniques of artificial intelligence, or intelligent systems, and gives insights into active research areas and applications. Emphasis is placed on representation as a central and necessary concept for work in intelligent systems. Strategies for choosing representations as well as notational systems and structures should be discussed. Students should understand, for example, that the selection of a programming language is really a basic representational choice and that an important component of that choice is whether the programming language is really the basic representational mode or whether it is a translator/interpreter of an intermediate representational mode such as the predicate calculus or other notational system (e.g. modal or fuzzy logics).

Other issues of importance in this course are natural language, vision systems, search strategies, and control. The extent and type of coverage will vary. The use of natural language and vision systems in applications of intelligent systems research to other disciplines should be emphasized. Search strategies should be seen as being implicit in representation and control. General issues related to control should be discussed and illustrated by examples of existing systems. A variety of applications could be mentioned at the beginning of the course as motivation for studying intelligent systems. These applications could then be elaborated on at appropriate times throughout the course or at the end.

Students could profit from a background in LISP because of its widespread use in artificial intelligence work. A Programming Language Laboratory as described in Section 3.2 could be used to provide this background either concurrently or with CS 7. If neither alternative is possible, then an introduction to LISP could be included in the course during the discussion of representation, but there would not be enough time for an in-depth treatment of the language.

TOPICS

- A. *Representation*. Constraints and capabilities of notational systems such as logics and programming languages. Notational structures such as trees, networks, statistical representations, and frames. Strategies for

- choosing representations (e.g. exploiting natural constraints in data, representation of similar patterns as in analogies). Introduction to LISP. (40%)
- B. *Search Strategies*. Tree and graph searches (e.g. depth and breadth first, minimax, alpha-beta). Heuristics. (15%)
- C. *Control*. General characteristics of production and procedurally oriented systems. Parallel vs. serial processing. Existing systems to illustrate issues (e.g. HEARSAY II, DENDRAL, MYCIN). (20%)
- D. *Communication and Perception*. Introduction to concepts related to current research in natural language and in vision systems. Use of tactility in intelligent systems. (10%)
- E. *Applications*. Sampling of current work in such areas as psychology, medicine, science, architecture, and such machines as industrial robots. (10%)
- F. *Examinations*. (5%)

CS 13. Algorithms (3-0-3)

Prerequisites: CS 7 and CS 8

COURSE OUTLINE

This course should develop students' abilities as writers and critics of programs by exposing students to problems and their algorithmic solution. As programming is both art and science, student programmers can benefit considerably from analysis of case studies in a wide variety of areas. All options for presenting algorithms in a very high level language should be considered, without regard for whether a processor exists for that language. Translation of each algorithm to a more machine-readable form can be given separately, if necessary. Careful choice of the level of abstraction appropriate to a given problem should be made as a means of adjusting students' load in the course.

Domain independent techniques should emerge during the course as algorithm-rich topics are presented from various areas. One convenient classification of topics into areas to ensure breadth of coverage is: combinatorics, numerical analysis, systems programming, and artificial intelligence. Algorithms from a majority of these areas should be analyzed, although not necessarily in the order indicated in the outline. The percentage ranges are intended to give instructors flexibility in choosing areas and topics.

TOPICS

- A. *Combinatorics*. Algorithms for unordered and ordered sets, graphs, matrices (within the semi-ring paradigm), bit vectors. (10-25%)
- B. *Numerical Analysis*. Algorithms for integer arithmetic (fast multiplication, prime testing, sieves, factoring, greatest common denominator, linear Diophantine equations), real arithmetic (Taylor series, how various calculators work), polynomial arithmetic, random numbers, matrix operations (inversion, determinants). (10-25%)

- C. *Systems Programming*. Algorithms in text processors (pattern matching) language processors (parsing, storage management), operating systems (scheduling, synchronization), database management (sorting, searching). (10-25%)
- D. *Artificial Intelligence*. Algorithms in natural language processing (concordances, context-free parsers), robotics (vision, manipulator operation), theorem proving and problem solving (decision methods, search heuristics). (10-25%)
- E. *Domain Independent Techniques*. Divide-and-conquer. Solution of recurrence equations. Dynamic programming. (15%)
- F. *Examinations*. (5%)

CS 14. Software Design and Development (3-0-3) or (2-2-3)

Prerequisites: CS 7 and CS 8

COURSE OUTLINE

This course presents a formal approach to state-of-the-art techniques in software design and development and provides a means for students to apply the techniques. An integral part of the course is the involvement of students working in teams in the organization, management, and development of a large software project. The team project aspect can be facilitated either by scheduling separate laboratories or by using some of the lecture periods to discuss practical aspects of the team projects.

TOPICS

- A. *Design Techniques*. Formal models of structured programming. Demonstrations of code reading and correctness. Stepwise refinement and reorganization. Segmentation. Top-down design and development. Information hiding. Iterative enhancement. Structured design. Strength and coupling measures. (50%)
- B. *Organization and Management*. Milestones and estimating. Chief programmer teams. Program libraries. Walk-throughs. Documentation. (15%)
- C. *Team Project*. Organization, management, and development of a large scale software project by students working in teams. (30%)
- D. *Examinations*. (5%)

CS 15. Theory of Programming Languages (3-0-3)

Prerequisite: CS 8

COURSE OUTLINE

This is a course in the formal treatment of programming language translation and compiler design concepts. Course material builds on the background established in CS 8, specifically on the introduction to lexical analysis, parsing, and compilers. Emphasis should be on the theoretical aspects of parsing context-free languages, translation specifications, and machine-independent code improvement. Programming projects to demonstrate various concepts are desirable, but extensive projects to write compilers, or major components of compilers,

should be deferred to a special topics course on compiler writing.

TOPICS

- A. *Review*. Grammars, languages, and their syntax and semantics. Concepts of parsing and ambiguity. BNF description of Algol. (15%)
- B. *Scanners*. Finite state grammars and recognizers. Lexical scanners. Implementation of symbol tables. (20%)
- C. *Parsers*. Theory and examples of context-free languages and push-down automata (PDA). Context-free parsing techniques such as recursive descent, LL(k), precedence, LR(k), SLR(k). (40%)
- D. *Translation*. Techniques of machine-independent code generation and improvement. Inherited and synthesized attributes. Syntax directed translation schema. (20%)
- E. *Examinations*. (5%)

CS 16. Automata, Computability, and Formal Languages (3-0-3)

Prerequisites: CS 8 and MA 4 (see Sect. 4.1)

COURSE OUTLINE

This course offers a diverse sampling of the areas of theoretical computer science and their hierarchical interconnections. Basic results relating to formal models of computation should be introduced. Stress should be given to developing students' skills in understanding rigorous definitions in computing environments and in determining their logical consequences. In this regard strong emphasis should be placed on problem assignments and their evaluations.

Material need not be presented in the order specified, but it is important to give nearly equal emphasis among the major areas. Topics within each area can be covered in greater depth in appropriate special topics courses.

TOPICS

- A. *Finite State Concepts*. Acceptors (including non-determinism). Regular expressions. Closure properties. Sequential machines and finite state transducers. State minimization. (30%)
- B. *Formal Grammars*. Chomsky hierarchy grammars, pushdown acceptors and linear bounded automata. Closure properties and algorithms on grammars. (35%)
- C. *Computability and Turing Machines*. Turing machine as acceptor and transducer. Universal machine. Computable and noncomputable functions. Halting problem. (30%)
- D. *Examinations*. (5%)

CS 17. Numerical Mathematics: Analysis (3-0-3)

Prerequisites: CS 1 and MA 5

COURSE OUTLINE

This course with CS 18 forms a one-year introduction to numerical analysis. The courses are intended to

be independent of each other. Students should be expected not only to learn the basic algorithms of numerical computation, but also to understand the theoretical foundations of the algorithms and various problems related to the practical implementations of the algorithms. Thus each topic implies a discussion of the algorithm, the related theory, and the benefits, disadvantages, and pitfalls associated with the method. Programming assignments should be given to illustrate solutions of realistic problems rather than just the coding of various algorithms. Topics such as convergence and error analysis for specific algorithms should be treated in a theoretical manner. Floating point arithmetic and use of mathematical subroutine packages are included in both courses because they should be discussed throughout the courses as they relate to specific problems. All other topics in each course should be covered sequentially. The depth to which topics are treated may vary, but most, if not all, topics should be discussed.

TOPICS

- A. *Floating Point Arithmetic*. Basic concepts of floating point number systems. Implications of finite precision. Illustrations of errors due to roundoff. (15%)
- B. *Use of Mathematical Subroutine Packages*. (5%)
- C. *Interpolation*. Finite difference calculus. Polynomial interpolation. Inverse interpolation. Spline interpolation. (15%)
- D. *Approximation*. Uniform approximation. Discrete least-squares. Polynomial approximation. Fourier approximation. Chebyshev economization. (10%)
- E. *Numerical Integration and Differentiation*. Interpolatory numerical integration. Euler-McLaurin sum formula. Gaussian quadrature. Adaptive integration. Fast Fourier transform. Richardson extrapolation and numerical differentiation. (15%)
- F. *Solution of Nonlinear Equations*. Bisection. Fixed point iteration. Newton's method. Secant method. Muller's method. Aitken's process. Rates of convergence. Efficient evaluation of polynomials. Bairstow's method. (15%)
- G. *Solution of Ordinary Differential Equations*. Taylor series methods. Euler's method, with local and global error analysis. Runge-Kutta methods. Predictor-corrector methods. Automatic error monitoring—change of step size and order. Stability. (20%)
- H. *Examinations*. (5%)

CS 18. Numerical Mathematics: Linear Algebra (3-0-3)

Prerequisites: CS 1 and MA 5

COURSE OUTLINE

The same remarks apply to this course as to CS 17.

TOPICS

- A. *Floating Point Arithmetic*. Basic concepts of floating point number systems. Implications of finite precision. Illustrations of errors due to roundoff. (15%)

- B. *Use of Mathematical Subroutine Packages*. (5%)
- C. *Direct Methods for Linear Systems of Equations*. Gaussian elimination. Operational counts. Implementation, including pivoting and scaling. Direct factorization methods. (20%)
- D. *Error Analysis and Norms*. Vector norms and matrix norms. Condition numbers and error estimates. Iterative improvement. (15%)
- E. *Iterative Methods*. Jacobi's method. Gauss-Seidel method. Acceleration of iterative methods. Overrelaxation. (15%)
- F. *Computation of Eigenvalues and Eigenvectors*. Basic theorems. Error estimates. The power method. Jacobi's method. Householder's method. (15%)
- G. *Related Topics*. Numerical solution of boundary value problems for ordinary differential equations. Solution of nonlinear systems of algebraic equations. Least-squares solution of overdetermined systems. (10%)
- H. *Examinations*. (5%)

3.4 Special Topics

The special topics courses should be offered whenever departmental resources are sufficient to do so. Thus content and prerequisites may vary each time they are offered because the available material is changing rapidly and different faculty members may have widely differing opinions of what should be included in a course. Most importantly, the material should be current and topical. In time, some of the material should be integrated into courses previously specified or may replace entire courses in the curriculum. Monitoring of this phase of the program should be a continuing activity of individual departments and C³S.

Examples of special topics courses include:

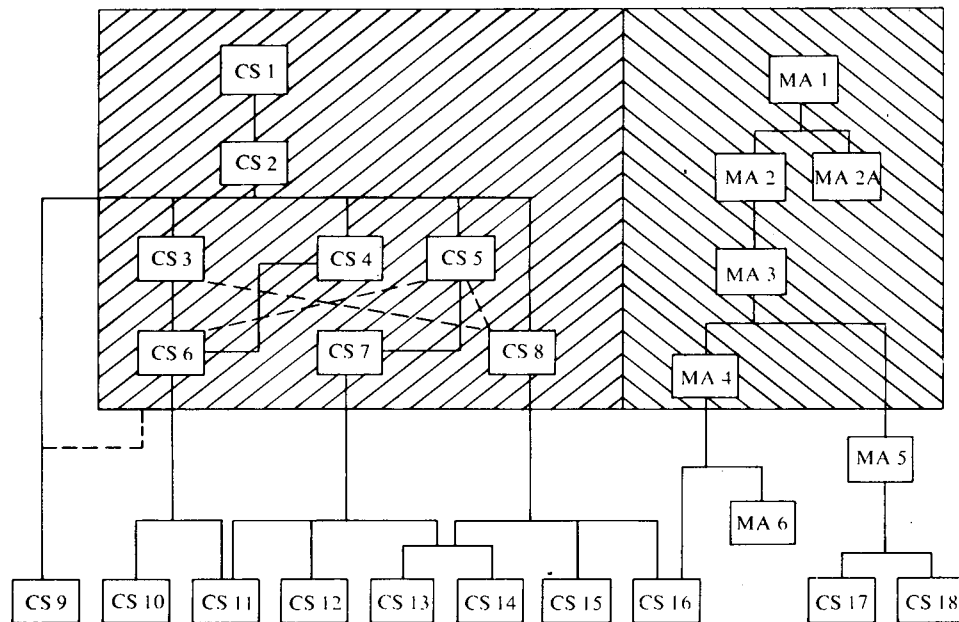
- A. Microcomputer Laboratory
- B. Minicomputer Laboratory
- C. Performance Evaluation
- D. Telecommunications/Networks/Distributed Systems
- E. Systems Simulation
- F. Advanced Systems Programming
- G. Graphics
- H. Compiler Writing Laboratory
- I. Structured Programming
- J. Topics in Automata Theory
- K. Topics in Computability
- L. Topics in Formal Language Theory
- M. Simulation and Modeling

4. The Undergraduate Program

4.1 Introduction

Outlines of eighteen computer science courses are included in previous sections. Eight of the courses indicate one of the ways in which the core material might be presented. Ten courses along with thirteen topics courses

Fig. 2. Recommended computer science and mathematics courses.



illustrate the kind of elective material to be offered at an advanced level.

The eighteen computer science courses are as follows:

- CS 1. Computer Programming I
- CS 2. Computer Programming II
- CS 3. Introduction to Computer Systems
- CS 4. Introduction to Computer Organization
- CS 5. Introduction to File Processing
- CS 6. Operating Systems and Computer Architecture I
- CS 7. Data Structures and Algorithm Analysis
- CS 8. Organization of Programming Languages
- CS 9. Computers and Society
- CS 10. Operating Systems and Computer Architecture II
- CS 11. Database Management Systems Design
- CS 12. Artificial Intelligence
- CS 13. Algorithms
- CS 14. Software Design and Development
- CS 15. Theory of Programming Languages
- CS 16. Automata, Computability, and Formal Languages
- CS 17. Numerical Mathematics: Analysis
- CS 18. Numerical Mathematics: Linear Algebra

The structure of these courses is given in Figure 2. The following set of mathematics courses is included in the structure for completeness and because of its relevance to an undergraduate program in computer science:

- MA 1. Introductory Calculus
- MA 2. Mathematical Analysis I
- MA 2A. Probability
- MA 3. Linear Algebra
- MA 4. Discrete Structures
- MA 5. Mathematical Analysis II
- MA 6. Probability and Statistics

Their role and the extent to which they conform to the needs of a computer science major are discussed in Section 4.3.

Solid and dashed lines represent, respectively, absolute and recommended prerequisites. The shaded area depicts the core curriculum in computer science and required mathematics courses.

4.2 Computer Science Requirements and Electives

The computer science major will consist of the eight courses of the core material plus four additional courses selected from the recommended computer science advanced electives with no more than two in any one specific subfield of the disciplines. Within the requirements for the four elective courses, the special topics courses specified in Section 3.4 should also be considered as possible electives for the major.

It should be noted that as students proceed through the computer science portion of the program, they begin at a very practical level and as they progress the work becomes more conceptual and theoretical. At the junior level the program is strongly conceptual while in the senior year the program may be fully theoretical, or involve a significant amount of theory supplemented with laboratory activities.

4.3 Mathematics Requirements

An understanding of and the capability to use a number of mathematical concepts and techniques are vitally important for a computer scientist. Analytical and algebraic techniques, logic, finite mathematics, aspects of linear algebra, combinatorics, graph theory, optimization methods, probability, and statistics are, in various ways, intimately associated with the development of computer science concepts and techniques. For example, probability and statistics develop the required tools for measure-

ment and evaluation of programs and systems, two important aspects of computer science. Analysis, as commonly contained in calculus courses, gives the mathematical bases for important concepts such as sets, relations, functions, limits, and convergence. Discrete structures provides the bases for semigroups, groups, trees, graphs, and combinatorics, all of which have applications in algorithms analysis and testing, as well as in data structure design. Thus mathematics requirements are integral to a computer science curriculum even though specific courses are not cited as prerequisites for most computer science courses. Unfortunately, the kind and amount of material needed from these areas for computer science usually can only be obtained, if at all, from the regular courses offered by departments of mathematics for their own majors.

Ideally, computer science and mathematics departments should cooperate in developing courses concentrating on discrete mathematics which are appropriate to the needs of computer scientists. Such courses, however, if offered by mathematics departments, would substantially increase their service course load and would constitute a heavy additional commitment of their resources. On the other hand, these course offerings could constitute an applied mathematics component which, in turn, might provide attractive alternatives for some mathematics departments. Suitable computer oriented mathematics course offerings constitute an important topic which should be explored more thoroughly both on local (i.e. individual institutions) and national levels. Specific course recommendations, however, are outside the domain of this report.

Until such time as suitable courses become readily available, it will be necessary to rely on the most commonly offered mathematics courses for the mathematical background needed by computer science majors. One set of such courses was recommended in 1965 by the Committee on Undergraduate Programs in Mathematics (CUPM) of the Mathematical Association of America. Courses MA 1, 2, 2A, 3, 5, and 6 in the structure included in Section 4.1 are intended to be CUPM recommended courses. Details on course contents can be found in the CUPM report [5].

MA 4 represents a more advanced course in discrete structures than that given in "Curriculum '68". The course will build on concepts developed by the study of calculus and linear algebra and will emphasize applications of discrete mathematics to computer science. In particular, if techniques in probability are not included in an earlier course, some emphasis should be given to them in this course. A number of examples of suitable outlines for this course have appeared in the literature, primarily in the *SIGCSE Bulletin* [6, 7, 8, 9, 10].

If courses of the type cited above are the only kind of mathematics courses available, then MA 1, MA 2, MA 2A, MA 3, and MA 4 should be required of all computer science majors. In addition, MA 5 or MA 6 may be required depending on which advanced level

computer science electives are selected. If more appropriate courses are provided as a result of interaction between computer science and mathematics departments, then the specification of required mathematics courses and the prerequisite structure should be reconsidered.

4.4 Other Requirements and Electives

As specified in this report, the minimum requirements are 36 semester hours in computer science and 15 semester hours in mathematics. This is certainly less than half of the required hours of a typical undergraduate degree program.

Additional requirements and electives will vary with the requirements of the individual institutions and hence only the most general of recommendations can be given.

It is certainly recognized that writing and communication skills must be emphasized throughout the program. This must be accomplished by requiring appropriate courses in the humanities, and also by emphasis on these skills in the courses within the computer science program itself. Surveys of employers stress the need for these skills as a requirement for employment.

Science and engineering departments represent fruitful areas for support of a computer science program. For those institutions with access to an engineering program, courses such as switching circuits and digital logic should be utilized. Within the science departments, a number of options are available to meet general university requirements. In addition to courses in fields such as physics, it should be noted that the increasing emphasis on computing in the biological and environmental sciences offers additional options for students.

A large portion of the job market involves work in business oriented computer fields. As a result, in those cases where there is a business school or related department, it would be most appropriate to take courses in which one could learn the technology and techniques appropriate to this field. For those students choosing this path, business courses develop the background necessary to function in the business environment.

The general university requirements in the social sciences, with careful advising, will generally be adequate, although it should be recognized that increasing use of computers in these fields may make it appropriate for some students to devise a minor in such an area if that is within their interests.

In consideration of this entire area of general requirements and electives, it must be recognized that a person who is going into the computer job market at the bachelor's level will, in all likelihood, initially be a systems, scientific, engineering, or business programmer. As a result, the student is well advised to work out a program with an advisor that will provide a meaningful and thorough background in the area of the student's interest. The general liberal arts requirements of the institution will give the necessary breadth to the program. A well developed concentration in an area other than com-

puter science will put the student in a position to develop and grow in that area as well as in computer science.

5. Service Courses

5.1 Introduction

There is a great need and demand for computer science material by students who do not intend to major in computer science. Faculty of computer science departments must be willing to offer different courses for those students than for majors when that is appropriate. Service courses should be offered by computer science faculty rather than by faculty in other departments. This, of course, implies that the courses must be made appealing by providing appropriate computer science content in a manner that is attuned to the needs, levels, and backgrounds of the students taking such courses.

There is some possibility that certain courses can be team-taught by faculty from computer science and from one or more other disciplines, but it must be recognized that this approach is difficult. Heads of departments must make difficult decisions regarding how much of the department's teaching resources is to be used for majors and how much is to be used for students in other disciplines. In making these decisions, it is essential that the department and institution properly acknowledge and reward faculty who are working in this area, if the courses are to maintain a high level of excellence.

A variety of service courses must be considered to satisfy the diverse needs of groups of students. Among the categories of undergraduate level courses are the following: (a) liberal arts or general university requirements; (b) supporting work for majors in other disciplines; and (c) continuing education.

5.2 General Service Courses

Students taking a course to satisfy a requirement such as a general university requirement may come from any discipline other than computer science. Some of the science, engineering, or mathematics oriented students may profit most by taking the same first course recommended for computer science students (CS 1). This has an immediate advantage for students who become interested enough in computing to want additional computer science courses. They will have the prerequisite for the second (and subsequent) courses for the computer science major. Those students who stop after one or two of these courses at least have excellent basic programming techniques to apply to computer oriented work in their discipline.

Other students will require more specialized study than that listed in CS 1. For many of these students the courses listed in the section on elementary computer science electives may be more appropriate.

It must still be recognized that a different course (or courses) must be provided for majors in the fields mentioned above as well as for majors in business oriented

fields, social sciences, education, and humanities. Service courses for these students normally should include a combination of computer appreciation, programming, applications, and societal impact. Different mixes of these broad areas should be considered for different groups, and the amount of each is best determined by each institution. Topics within each area should be as pertinent to the group served as possible, especially in the language chosen to illustrate programming. To meet this goal, feedback from students is important and communication between computer science and other departments, including periodic review of the courses, is essential. The course should have no prerequisites and it should be made clear to the students that the course is not intended for those who want additional work in computer science. If local conditions warrant, the material could be presented in two semesters rather than one.

Though as indicated, full specification of such courses is impossible, an example can be given to illustrate the kind of course under consideration:

CSS 1. Computer Applications and Impact (3-0-3)

COURSE OUTLINE

A survey of computer applications in areas such as file management, gaming, CAI, process control, simulation, and modeling. Impact of computers on individuals and society. Problem solving using computers with emphasis on analysis, formulation of algorithms, and programming. Projects chosen from various application areas of student interest.

TOPICS (percentages dependent on local situations)

- A. *Computer Systems*: Batch and interactive, real time, information management, networks. Description of each system, how it differs from the others, and kinds of applications for which each system is best suited.
- B. *Databases*: Establishment and use. Data definition and structures.
- C. *Errors*: Types, effects, handling.
- D. *Social Implications*: Human-machine interface. Privacy. Moral and legal issues.
- E. *Future Social Impact*: Checkless society. CAI. National data banks.
- F. *Languages*: As appropriate, introduction to a business oriented language, a symbol manipulation language, and/or a procedure oriented language. Brief exposition of characteristics which make these languages appropriate for particular classes of problems.
- G. *Concepts and Techniques Used in Solving Problems*: Selected from appropriate application areas such as CAI, data management, gaming, information retrieval, and simulation.
- H. *Projects and Examinations*.

5.3 Supporting Areas

A number of students will choose computer science as a supporting (or minor) area. Various possibilities

for sets of courses should be available. One of the ways to achieve this by using the same courses as taken by a computer science major is to require courses CS 1 and CS 2; at least two of the courses CS 3, CS 4, CS 5; and at least two of the courses CS 6, CS 7, CS 8. Additional courses could then be taken as student interest and program requirements would allow. Computer science faculty should communicate with faculty from other departments to determine the needs of the other departments and to indicate how certain courses or course combinations might satisfy the needs.

In those cases where existing courses are not appropriate as supporting work for other majors, new courses should be created, probably to be offered as upper division level courses. Two alternatives for establishing sets of courses for use as supporting work are as follows: (a) CS 1 and CS 2, one course combining material from CS 5 and CS 7, and one course combining material from CS 3, CS 4, and CS 6; and (b) CS 1 and CS 2, one course combining material from CS 3 and CS 5, and one course combining material from CS 4, CS 6, and CS 7. Alternative (a) attempts to combine similar topics from different levels while alternative (b) attempts to combine different topics from similar levels. It should be recognized that students who complete either of the latter two alternatives may not be well enough prepared to take a more advanced computer science course for which any of the courses CS 6, CS 7, or CS 8 are prerequisite.

5.4 Continuing Education

Continuing education is an area which has grown so rapidly and includes such a large variety of interests that it is virtually impossible to specify course possibilities. Nevertheless, computer science departments must address the needs appropriate to their local situations. Some of the possibilities which should be considered are: (a) adult education courses, probably versions of the courses suggested to meet general university requirements; (b) professional development seminars, usually consisting of one day to several weeks devoted to a specific topical area (e.g. minicomputers, database management systems); and (c) courses offered in the evenings or on weekends (on or off campus), possibly regular course offerings or modifications of them primarily for employed persons who need to acquire or enhance their computer science background. The latter possibility would include full-scale baccalaureate or master's degree programs.

6. Other Considerations

6.1 Introduction

Implementation of the computer science curriculum recommendations given in this report implies more than the development of a coherent program of courses. Articulation with other educational institutions and with employers of graduates of such programs must be given

serious attention, and a commitment must be made to provide and maintain these resources. In most cases, such commitments go well beyond the boundaries of computer science departments.

Specific requirements involving such areas as staff, equipment, and articulation will vary among institutions depending on such things as size, location, capability, and mission of the school and program. As a result, specific recommendation in these areas cannot be given. However, in this section, general guidelines for implementation in these areas are discussed.

6.2 Facilities

In order to implement the full set of recommendations contained in this report, a wide range of computing facilities will be required. Equipment such as data entry devices, microcomputers, minicomputers, and medium or large-scale computer systems all play separate and important roles in the development of the computer scientist.

Data entry devices such as card punches, teletype-writers, and display terminals should be provided for program preparation and communication between student and computer. Such equipment should be conveniently located and in a large enough area for both easy and convenient student access and use. This equipment may be provided and maintained by the central computing facility at the institution for general student and faculty use, or, if enrollments in the computer science program and demands for service warrant, the equipment may be located and maintained by the department with some restriction on the use by other departments. To implement successfully an adequate program that insures easy and ready access to such facilities, close cooperation and planning is necessary that will involve the computer science department, the computer center, and, perhaps, other departments which use these computer facilities.

Microcomputers are quite desirable in teaching details of computer architecture previously only attainable by extensive programming of "hypothetical computers," simulators, or textbook discussions. They have provided a relatively inexpensive and highly versatile resource which can be used in a variety of ways including combining several such units into reasonably sophisticated and powerful computer systems. Their use is becoming so widespread that in addition to using microcomputers in a systems course, under some circumstances, consideration may be given to offering a laboratory course in which each student, or a group of students in the course, would purchase a suitable kit and construct a computer.

The availability of one or more minicomputers in a department allows the students to obtain "hands-on" experience as well as the opportunity to utilize interactive systems and programming languages which may not be available, or practical, on a medium or large-scale computer system. This kind of equipment also allows the

student to work on software development projects, and other projects that might not be possible due to restrictions on the use of the central facility. It is desirable that the department maintain and schedule such minicomputer facilities in such a way that student usage and software development can proceed in an orderly fashion through laboratory course work and individual projects.

A medium or large-scale computer, normally operated and maintained as a central facility at the institution for use by all departments, should provide appropriate hardware and software support for the major program. Auxiliary memory is required in order to store files so that access methods specified in the core courses can be implemented and tested. Suitable input/output devices and system facilities are needed so that rapid turnaround of student jobs is possible, interactive computing is available, and programming languages used in the curriculum are supported.

Regardless of what specific items of computer equipment are available to support a curriculum in computer science, effective teaching and research in the field require laboratory facilities. Computer science is in part an empirical science which involves implementing procedures as well as studying theoretically based processes. Because systems, algorithms, languages, and data structures are created, studied, and measured via combinations of hardware and software, it is essential that appropriate laboratory facilities be made available that are comparable to those necessary in the physical and biological sciences and engineering disciplines. This implies that appropriate laboratory facilities are available for student and faculty use, and may imply that additional laboratory space is required by certain faculty and students for special purposes. The initial budgetary support for establishing these laboratories may be substantial, and continuing regular budgetary support is essential for successful implementation of a program.

While we have thus far stressed the hardware facilities necessary for the recommended curriculum, equal attention must also be given to software. In order for the student to master the material in the core and elective courses, sufficient higher level languages must be available. Additionally, special purpose systems such as statistical systems, database management systems, information storage and retrieval systems, and simulation systems should be available for student use. It must be recognized in planning that many of these systems require a significant initial and continuing investment on the part of the institution. Where possible, fast turnaround or interactive systems should be considered in order to provide as much access as possible for the student.

In addition to the computer related facilities required for the recommended curriculum, there is also a requirement for those resources of a university that are normally associated with any discipline. Adequate library facilities, including significant holdings of periodicals are ab-

solutely necessary, and the implementor of this report is referred to the basic library list [4] for a basis of establishing a library collection to support the instructional program.

While traditional library support is essential to the computer science program, it must be recognized that the field requires some additional resources that may not be necessary in other disciplines. Specifically, the student of computer science must have available, in some form, language, programming, and systems manuals as well as documentation for programs and other materials directly related to the development and use of systems. This material must be easily and conveniently available to the student at all times.

6.3 Staff

Insofar as it is possible, the vast majority of faculty members in departments offering the curriculum that has been recommended in this report should have their primary academic training in computer science. At the same time, it remains the case that demand exceeds supply for these individuals and it is often necessary, and in some cases desirable, to acquire faculty with degrees in other disciplines, but who have experience in computing through teaching or employment in government, business, or industry.

The size of the department will depend on available resources, required teaching loads, commitments to offering service courses, and commitments to continuing education programs. Approximately six full-time equivalent faculty members are necessary to offer a minimal program that would include the core courses as well as a selection of elective and service courses. Most of these faculty members should be capable of offering all of the core courses in addition to elective courses in their areas of specialization. Additional continuing instructional support may be available from the computer center, and from other departments such as mathematics which may offer numerical analysis or other applied mathematics courses that could be cross-listed by both departments. In addition, adjunct faculty from local government, business, or industry are valuable additions in many cases. Such individuals are often able to bring a different perspective to the program; however, care must be taken to insure that the program does not become overly dependent on individuals who may be unable to perform continuing service.

Because of the rapid growth of this field, consideration must be given to providing ongoing opportunities for faculty development, such as a sabbatical leave program, opportunities to attend professional development seminars, and interchange programs with industry.

A department which operates its own laboratory facilities should consider obtaining a full-time staff member to maintain such systems, be responsible for necessary documentation and languages, and coordinate other activities connected with the laboratory. Such a staff

member would provide continuity in the development of the laboratory resource.

The field is still developing rapidly, and as was indicated earlier, is at least in part empirical in nature. As a result faculty will be required to devote a great deal of time to course development, software development, development of laboratory resources, and development of service offerings. To provide for continuing excellence in these areas, it must be recognized that they are essential contributions to the program and profession, and as such should be considered within the context of the reward structure of the institution.

6.4 Articulation

It is imperative that departments offering computer science programs keep in close contact with secondary schools, community and junior colleges, graduate schools, and prospective employers of their graduates. This requires a continuing, time consuming effort. Primary responsibility for this effort could be placed with one faculty member, whose teaching load should then be reduced. Experience has shown that person-to-person contact on a continuing basis is necessary for successful articulation.

Usually, a central office in a four-year institution has direct contact with secondary schools. With computing becoming more prevalent at that level, however, it is highly useful and appropriate for a departmental representative to maintain contact with those local secondary schools which offer, or desire to offer, courses in computing.

Articulation agreements exist in many areas between four-year institutions and community and junior colleges. These agreements need to be updated frequently as programs or courses change, and personal contact between departments is necessary to keep abreast of these changes. Transfer programs in community and junior colleges are often geared to programs at four-year institutions. As a result, proposed changes in the four-year program which influence transfer programs should be promulgated as soon as possible so that the community and junior colleges can incorporate such changes, thereby reducing the lag between programs to the benefit of transfer students.

Some of the graduates of the recommended program will continue academic work in computer science in graduate school, but most will seek employment upon graduation. Departments must be aware of the graduate school requirements so that their programs prepare students adequately for advanced work in the field, but they must also maintain communication with employers in order to know what job requirements exist so that the faculty can advise students more effectively. Feedback from recent graduates of the program is quite useful in this regard and should be encouraged as much as possible. In order to most effectively implement this aspect of the program, faculty members should have

available to them graduate school brochures, Civil Service Commission documents, and whatever else can come from personal contacts with employees in government and industry, as well as from the professional societies.

References

1. Curriculum Committee on Computer Science (C³S). Curriculum '68, recommendations for academic programs in computer science. *Comm. ACM* 11, 3 (March 1968), 151-197.
2. Austing, R.H., Barnes, B.H., and Engel, G.L. A survey of the literature in computer science education since Curriculum '68. *Comm. ACM* 20, 1 (Jan. 1977), 13-21.
3. Education Committee (Model Curriculum Subcommittee) of the IEEE Computer Society. A curriculum in computer science and engineering. Committee Report, IEEE Pub. EH0119-8, January 1977.
4. Joint Committee of the ACM and the IEEE Computer Society. A library list on undergraduate computer science—computer engineering and information systems. Committee Report, IEEE Pub. EH0131-3, 1978.
5. Committee on the Undergraduate Program in Mathematics. A general curriculum in mathematics for colleges. Rep. to Math. Assoc. of America, CUPM, Berkeley, Calif., 1965.
6. Special Interest Group on Computer Science Education. SIGCSE Bulletin, (ACM) 5, 1 (Feb. 1973).
7. Special Interest Group on Computer Science Education. SIGCSE Bulletin, (ACM) 6, 1 (Feb. 1974).
8. Special Interest Group on Computer Science Education. SIGCSE Bulletin, (ACM) 7, 1 (Feb. 1975).
9. Special Interest Group on Computer Science Education. SIGCSE Bulletin, (ACM) 8, 1 (Feb. 1976).
10. Special Interest Group on Computer Science Education. SIGCSE Bulletin, (ACM) 8, 3 (Aug. 1976).

Appendix

Contributors to the C³S Report

Robert M. Aiken, University of Tennessee
Michael A. Arbib, University of Massachusetts
Julius A. Archibald, SUNY at Plattsburgh
William Atchison, University of Maryland
Richard Austing, University of Maryland
Bruce Barnes, National Science Foundation
Victor R. Basili, University of Maryland
Barry Bateman, Southern Illinois University
Della T. Bonnette, University of Southwestern Louisiana
W.P. Buckley, Aluminum Company of America
Frank Cable, Pennsylvania State University
Gary Carlson, Brigham Young University
B.F. Caviness, Rensselaer Polytechnic Institute
Donald Chand, Georgia State University
Sam Conte, Purdue University
William Cotterman, Georgia State University
Daniel Couger, University of Colorado
John F. Dalphin, Indiana University—Purdue University at Fort Wayne
Gene Davenport, John Wiley and Sons
Charles Davidson, University of Wisconsin
Peter Denning, Purdue University
Ed Desautels, University of Wisconsin
Benjamin Diamant, IBM
Karen A. Duncan, MITRE Corporation
Gerald Engel, Old Dominion University
Michael Faiman, University of Illinois
Patrick Fischer, Pennsylvania State University
Arthur Fleck, University of Iowa
John Gannon, University of Maryland
Norman Gibbs, College of William and Mary
Malcolm Gotterer, Florida International University
David Gries, Cornell University

(Appendix continued on next page)

(Appendix continued from preceding page)

H.C. Gyllstrom, Univac
Douglas H. Haden, New Mexico State University
John W. Hamblen, University of Missouri-Rolla
Preston Hammer, Grand Valley State Colleges
Richard Hamming, Naval Postgraduate School
Thomas R. Harbron, Anderson College
Stephen Hedetniemi, University of Oregon
Alex Hoffman, Texas Christian University
Charles Hughes, University of Tennessee
Lawrence Jehn, University of Dayton
Karl Karlstrom, Prentice-Hall
Thomas Keenan, National Science Foundation
Sister M.K. Keller, Clarke College
Douglas S. Kerr, The Ohio State University
Rob Kling, University of California, Irvine
Joyce C. Little, Community College of Baltimore
Donald Loveland, Duke University
Robert Mathis, Old Dominion University
Daniel McCracken, President, ACM
Robert McNaughton, Rensselaer Polytechnic Institute
M.A. Melkanoff, University of California, Los Angeles
John Metzner, University of Missouri-Rolla
Jack Minker, University of Maryland
Howard Morgan, University of Pennsylvania
Abbe Mowshowitz, University of British Columbia
Michael Mulder, Bonneville Power Administration

Anne E. Nieberding, Michigan State University
James Ortega, North Carolina State University
F.G. Pagan, Memorial University of Newfoundland
John L. Pfaltz, University of Virginia
James Powell, North Carolina State University
Vaughn Pratt, Massachusetts Institute of Technology
Anthony Ralston, SUNY at Buffalo
Jon Rickman, Northwest Missouri State College
David Rine, Western Illinois University
Jean Sammet, IBM
John F. Schrage, Indiana University—Purdue University at
Fort Wayne
Earl Schweppe, University of Kansas
Sally Y. Sedelow, University of Kansas
Gary B. Shelly, Anaheim Publishing
James Snyder, University of Illinois
Theodor Sterling, Simon Fraser University
Gordon Stokes, Brigham Young University
Alan Tucker, SUNY at Stony Brook
Ronald C. Turner, American Sign and Indicator Corporation
Brian W. Unger, The University of Calgary
James Vandergraft, University of Maryland
Peter Wegner, Brown University
Patrick Winston, Massachusetts Institute of Technology
Peter Worland, Gustavus Adolphus College
Marshall Yovits, The Ohio State University
Marvin Zerkowitz, University of Maryland

Recommendations for Master's Level Programs in Computer Science

A Report of the ACM Curriculum Committee on Computer Science

Editors: Kenneth I. Magel, University of Missouri-Rolla
Richard H. Austing, University of Maryland
Alfs Berztiss, University of Pittsburgh
Gerald L. Engel, Christopher Newport College
John W. Hamblen, University of Missouri-Rolla
A.A.J. Hoffmann, Texas Christian University
Robert Mathis, Old Dominion University

The ACM Committee on Curriculum in Computer Science has spent two years investigating master's degree programs in Computer Science. This report contains the conclusions of that effort. Recommendations are made concerning the form, entrance requirements, possible courses, staffing levels, intent, library resources, and computing resources required for an academic, professional, or specialized master's degree. These recommendations specify minimum requirements which should be met by any master's programs. The Committee believes that the details of a particular master's program should be determined and continually updated by the faculty involved. A single or a small number of model programs are not as appropriate at the graduate level as at the bachelor's level.

Key Words and Phrases: computer science courses, computer science curriculum, computer science education, computer science graduate programs, master's programs.

CR Categories: 1.52

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

Contents

1. Introduction
 2. The Need for Masters Programs
 3. Goals of Program
 - 3.1 Basic Intent
 - 3.2 Communication Skills
 - 3.3 Current Literature Level
 - 3.4 Professionalism
 4. Entrance Requirements
 - 4.1 Admission Requirements
 - 4.2 Prerequisites
 5. Program Organization
 - 5.1 Course Work
 - 5.2 Culminating/Unifying Activity
 - 5.3 Seminar
 - 5.4 Thesis or Project
 - 5.5 Comprehensive Examination
 6. Resource Requirements
 - 6.1 Faculty
 - 6.2 Computing Equipment
 - 6.3 Library
 7. Specializations
 8. Conclusions
- Appendices**
- A. Contributors
 - B. Course Descriptions
- References**

1.0 Introduction

The Committee on Curriculum in Computer Science (C³S)* of the Association for Computing Machinery has within its charter the obligation to address computer science education at the baccalaureate level and above. The Committee intends that this document establish a basis for master's degree programs of substance while at the same time permitting sufficient flexibility to allow for adaptation to the objectives and resources of individual colleges and universities. A second objective of the report is to provide guidance to those institutions which have begun or are about to begin a master's program without specifying a rigid blueprint for the establishment of such programs. Finally, and perhaps most importantly, the Committee hopes this report will foster meaningful interchange among computer science educators regarding instructional programs at the master's level.

Graduate programs in computer science preceded the introduction of undergraduate programs, the earliest programs appearing in the early 1960s. "Curriculum '68" [5] concentrated on the definition and specification of undergraduate programs but did consider master's programs also. Specifically the following recommendation was given:

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 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 [5, p. 163].

The section on the master's curriculum concludes with the statement:

This proposed program embodies sufficient flexibility to fulfill the requirements of either an "academic" degree obtained in preparation for 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 master's degree programs [5, p. 164].

The Committee believes that the discipline has matured enough that we can now see this distinction between academic and professional programs beginning to appear. We reject, however, the concept of an utterly terminal program. In our view all programs should provide the possibility of additional study in the field. This report tries to establish the common aspects of master's programs in computer science and indicates possible differences and distinctions.

Some attention was given to master's level programs by C³S following the publication of "Curriculum '68."

The results of this work were presented by Melkanoff [8] in 1973. Further work in this area was deferred, however, while work progressed on the new C³S recommendations at the undergraduate level. The new undergraduate recommendations were published as "Curriculum '78" in the March 1979 *Communications of the ACM* [2].

In an independent effort, the ACM Curriculum Committee on Computer Education for Management (C³EM) (now the Subcommittee on Curriculum in Information Systems) developed guidelines for a master's program in Information Systems [1, 9]. They clearly define a related professional degree program. The scope and extent of existing graduate programs in computer science have been recently surveyed [4, 7].

2.0 The Need for Master's Programs

The classical objective of academic master's programs is the preparation for study at the doctoral level, and this remains an important aspect of such programs. Different goals exist for professional programs, but we believe that all programs should prepare the student for study beyond the master's level.

Among the objectives for students in master's programs is entry into the computer field at a relatively high level of responsibility and expertise. Computer Science is such a new and rapidly expanding field that individuals entering with a master's degree in this field will almost immediately move to positions with great responsibility. This, in turn, implies the requirement for an advanced level of prior training in both technical and related areas (e.g., communication skills).

Many people already in the field desire additional training in computer science. These individuals may have undergraduate degrees in computer science and desire to advance; or they may have had considerable experience in computing, but little formal education in the field. While this latter group should be declining in number as more undergraduate computer science majors enter the job market, the demand does exist and will continue to do so in the foreseeable future. In addition, there will be a continuing need for individuals with a bachelor's degree in computer science to update their training.

In all of these cases, the master's degree provides both motivation for the student and a standard for reward by the employer.

The two-year colleges are offering a large number of courses in data processing and related topics. For most faculty positions in such institutions, a master's degree is a minimum requirement and a master's in computer science is an appropriate preparation.

Increasingly, precollege instruction in computer science is being offered. Consequently, there is a need for a master's program to prepare individuals to teach computer science at the precollege level. Further exploration of such a master's program should be done jointly by

* The Curriculum Committee on Computer Science (C³S) became a subcommittee of the Curriculum Committee on Computer Education in 1978.

this Committee and the ACM Subcommittee on Elementary and Secondary Education.

Graduate enrollments in computer science, information systems, and other related programs have grown steadily since their inception in the early 1960s. Even though growth rates are substantial, estimates of demand for personnel with graduate degrees in such programs far exceed the supply. During the 80s, the need for master's graduates is estimated to be approximately 34,000 annually. During this same period, annual production will only increase from about 3,000 to 4,000 [6].

3.0 Goals

3.1 Basic Intent

The basic intention of a master's program in computer science is to develop the student's critical professional thinking and intuition. The curriculum must be structured to provide a balanced mixture of learning experiences to make the graduate capable of sound professional decisions. As a result the graduate should be able to assume responsible positions in business, government, and education at the research, development, and planning levels. The program should also provide an excellent foundation for further formal training.

The primary emphasis of the program should be on the concepts, theory, and practice of computer science. Students should have a broad understanding of the field. Techniques and methodologies of computer science should be discussed and used. Intensive education in specific areas of computer science and/or training in an application area is desirable. An academically oriented program will encourage students to develop and use abstract models. A professionally oriented program will encourage students to apply abstract models and computer science concepts in practical situations.

Academically oriented programs will tend to attract full-time students, and these students are generally oriented toward further education and research. Students in professional programs are generally oriented toward careers in industry or government, and such programs are frequently designed to accommodate part-time students.

3.2 Communication Skills

Computer scientists require special communication skills. They must be able to communicate with the rest of their organizations in understandable terms, both orally and in writing. They must be able to communicate with their co-workers, users of their computer systems, and other professionals who require computer expertise. They must be able to produce documentation for a complex computing system which is clear, concise, unambiguous, and accurate. They must be able to produce well organized reports which clearly delineate objectives, method of solution, results, and conclusions for a complex task.

3.3 Current Literature Level

Graduates should be cognizant of the pertinent literature in their field of choice and be able to read, interpret, and use this material. They should find it a normal procedure to review current journals to keep abreast of new trends and ideas. They should be able to recognize and use techniques relevant to their present endeavors.

3.4 Professionalism

Since graduates could assume responsible positions in some organizations they should be able to function effectively as members of teams. They should possess qualities of leadership along with technical skills so as to effectively lead a group to the successful completion of a task.

Master's students should take an active part in the activities of any local professional computer science organization which may exist. They should be aware of the societal impact of computing as incorporated in the ACM Code of Ethics [12].

4.0 Entrance Requirements

4.1 Admission Requirements

The Graduate Record Examinations (GRE) Advanced Computer Science Test has been available since October 1976. Its purpose is to help graduate committees assess the qualifications of applicants with a bachelor's degree in Computer Science for advanced study in computer science. The Advanced Test in Computer Science is one of a number of measures that might be used to evaluate a candidate for admission to the M.S. degree program. The verbal part of the GRE may help measure the communication skill level of applicants and the quantitative part is a good general indicator of numeric manipulation capabilities.

A "B" average for the undergraduate degree is a common requirement for admission to graduate study. Some schools provide a "special" status for those who do not meet entrance requirements with subsequent re-evaluation for admission to full status.

4.2 Prerequisites

The student entering a master's program ideally should have a B.S. in Computer Science or at least the material included in CS 1 through CS 8 of "Curriculum '78" [2] or SE-1 through SE-4 and CO-1 through CO-4 of the IEEE Computer Society Model Curriculum [11], and mathematics through calculus, linear algebra, and one course in statistics. Course titles for CS 1 through CS 8 are given in Appendix B. Discrete structures, maturity in both abstract reasoning and the use of models, and one or more years of practical experience in computer science are desirable. Of course, the applicant must satisfy the general entrance requirements of the institution's graduate school or department.

Some schools may admit students who do not meet the entrance requirements listed above. These students will have to remove deficiencies early in their graduate studies.

Removal of academic deficiencies might be through any or all of the following approaches:

- a. Require students to take specific existing undergraduate courses for no credit toward the master's degree;
- b. Establish special "immigration" courses that rapidly cover the material in the areas of deficiency; or
- c. Provide the students with self-study outlines in conjunction with appropriate proficiency examinations.

Any courses taken to remove deficiencies must be in addition to the program required for the master's degree.

5.0 Program Organization

5.1 Course Work

Formal course work is provided to give the students a mixture of practical and theoretical work. Such courses will typically begin at a level in which the courses may be taken by advanced undergraduate students or graduate students.

The specific graduate courses which are offered reflect the expertise and judgment of the faculty involved. Graduate programs reflect their specific environments far more than do undergraduate programs. It is possible to envision several independent axes, e.g., software/hardware, theory/practice, and numeric/nonnumeric computation. Each department should determine where on each axis its program should be, consistent with available resources and expertise. These emphases should be re-evaluated at least every three years.

Nevertheless, the Committee believes all master's programs should have some aspects in common. Accordingly, a list of possible courses is given below. Departments planning master's programs should start with this list. In preparing these course descriptions, the Committee drew on material from well-established master's degree programs at

Georgia Institute of Technology
University of Illinois
University of Maryland
University of Missouri-Rolla
Northwestern University
University of North Carolina at Chapel Hill
Ohio State University
Purdue University
Rutgers: The State University of New Jersey
Stanford University
The University of Texas at Austin [13].

Computer Science is a rapidly changing field. The courses listed here reflect the present state of the field

and will require periodic updating. Descriptions of these courses are given in Appendix B. They provide a starting point for developing or updating a master's degree program.

Typical courses which should be offered, under the topical areas within which they fall, might be as follows. Courses CS 9 through CS 18 are described in [2]. Courses CS 19 through CS 38 are described in Appendix B.

A. *Programming Languages*

- CS 14 Software Design and Development
- CS 15 Theory of Programming Languages
- CS 19 Compiler Construction
- CS 20 Formal Methods in Programming Languages
- CS 21 Architecture of Assemblers
- CS 25 High Level Language Computer Architecture

B. *Operating Systems and Computer Architecture*

- CS 10 Operating Systems and Computer Architecture II
- CS 22 Performance Evaluation
- CS 23 Analytical Models for Operating Systems
- CS 24 Computer Communication Networks and Distributed Processing
- CS 26 Large Computer Architecture
- CS 27 Real-Time Systems
- CS 28 Microcomputer Systems and Local Networks

C. *Theoretical Computer Science*

- CS 13 Algorithms
- CS 16 Automata, Computability, and Formal Languages
- CS 29 Applied Combinatorics and Graph Theory
- CS 30 Theory of Computation

D. *Data and File Structures*

- CS 11 Database Management Systems Design
- CS 31 Information Systems Design
- CS 32 Information Storage and Access
- CS 33 Distributed Database Systems

E. *Other Topics*

- CS 9 Computers and Society
- CS 12 Artificial Intelligence
- CS 34 Pattern Recognition
- CS 35 Computer Graphics
- CS 36 Modeling and Simulation
- CS 17 Numerical Mathematics: Analysis
- CS 18 Numerical Mathematics: Linear Algebra
- CS 37 Legal and Economic Issues in Computing
- CS 38 Introduction to Symbolic and Algebraic Manipulation

These courses are representative of those being offered today in established master's programs. Some over-

lap considerably with others, e.g., CS 19 and CS 21 or CS 22 and CS 23. These pairs are included to provide alternative examples. The Committee does not propose that both members of a pair be offered. Further, the Committee expects the appropriate courses to change frequently as the field matures. Additional courses may be offered to reflect the interests of the faculty.

In considering the courses that should be taken in the master's program it should be recognized that one of the purposes of such a program is to supply the opportunity for additional course work over that possible in an undergraduate program. Some of the courses, appropriately, are available for the graduate student and advanced undergraduates, although the reasons for selection may be different.

The master's program should provide both breadth in several areas, and depth in a few. In addition, it should allow a degree of flexibility to address individual needs. The typical program will consist of 30 to 36 semester hours.

The program should include at least two courses from A, two courses from B, and one course from each of C, D, and E. The student who has not been exposed to numerical analysis as an undergraduate should take CS 17. Students with strong undergraduate backgrounds in computer science may have already satisfied some of these requirements and may thus proceed to more advanced courses. Their degrees probably will be more specialized than those of students with weaker backgrounds.

The entire program should contain at least four computer science courses which are for graduate students only.

5.2 Culminating/Unifying Activity

Beyond the course work, each student should be required to participate in some summarizing activity.

A thesis, project, seminar, or comprehensive examination exemplifies a kind of culminating activity for the program. They provide a format for a student to combine concepts and knowledge from a number of different courses. They also provide a method of judging a student's performance outside the narrow confines of a single course. They may also be useful in insuring a uniform standard in a program that may cover many years and use many different instructors.

These culminating activities can be very time-consuming for both students and faculty. Faculty loads must allow the necessary time for the preparation, supervision, and evaluation of these activities.

5.3 Seminar

A seminar in which the students make presentations can be useful for providing experience and improving the communication skills of students. The seminar provides an opportunity for the student to explore the literature and make formal presentations. The seminar is also useful in developing and encouraging the habit of

reading and discussing the current literature in computer science.

5.4 Thesis or Project

A thesis or project usually taking more than one semester should be done by each student. This is suggested to extend the student's experience in analysis and design and the evaluation and application of new research findings or technological advances. Relating projects to work environments can strengthen a professional program.

The thesis or project provides the primary means by which the student gains practical experience in applying computing techniques and methodologies. It also provides a basis for developing written and oral communication skills and documentation experience. Finally, it provides the opportunity for exploring recent concepts in the literature and demonstrating an understanding of those concepts.

A project is much more difficult to evaluate than course work or a thesis. However, because of the importance of the project within the program its careful evaluation is vitally important. Successful completion means:

- a. The product produced performs as prescribed.
- b. The project has been properly documented both in terms of nontechnical descriptions and in terms of technical diagrams and formal documentation.
- c. A formal public oral presentation has been given. This is seen as a mechanism for encouraging both a high level of presentation and a high technical standard for the project.

Through a seminar and a project the student can gain practical experience in the evaluation, selection, and decision making process.

5.5 Comprehensive Examination

An alternative to the thesis may be a comprehensive examination. This examination serves a purpose similar to the thesis or project discussed previously. It summarizes the entire program. The examination should consist of:

- a. review and analysis of articles from current literature; and/or
- b. questions that integrate material from more than one course.

The period of time over which degree requirements are satisfied will be considerably longer for part-time students than for full-time students. The former, therefore, should be supplied with reading guides prior to the comprehensive examination. Indeed, in order to encourage a reading habit in all students some examination questions should be related to required readings rather than to course work.

6.0 Resource Requirements

6.1 Faculty

Most faculty are qualified to teach in more than one area of specialization. Although in the past most computer science faculty received degrees in other disciplines, it is recommended that master's programs not be implemented without experienced computer science faculty or faculty formally trained in computer science. A minimum of five computer science faculty members is required to provide adequate breadth for a stand-alone master's program. If the department offers a bachelor's program as well, then at least eight faculty members would be required for both programs. Limited use of qualified adjunct faculty is appropriate in some special circumstances, but at least three quarters of the courses must be offered by regular full-time faculty.

6.2 Computing Equipment

Every computer science master's program must have access to adequate computer systems. The amount of computing power which must be available depends on how many students will be in the program at one time and their specializations.

An area of specialization such as computer graphics or design automation requires very special and possibly dedicated computing facilities, both hardware and software. Programs specializing in information systems place heavy demands on a large computer system with appropriate software.

For the study of computer systems and languages a variety of languages and operating systems must be available. A dedicated system under departmental control is optimal for hands-on experience. Programs and experiments dealing with the security of systems usually require a dedicated system.

Proper arrangements must be made for maintenance of the computing facility and laboratory equipment. Plans and provisions also need to be made for growth and periodic modernization of equipment resources.

6.3 Library

The list of books and magazines for undergraduate programs prepared by a joint committee of ACM and IEEE is the only available list of reference material [10]. It is a good starting point, but suffers from being for undergraduates rather than graduate students and being current only to 1977. Additional materials, particularly selected applied and theoretical journals, are required for a master's program. Besides faculty and computing equipment, substantial library resources are essential for an adequate master's program in computer science. Sizable current expenditure funds are needed to maintain collections, but at most universities such funds will be insufficient by themselves. An additional special allocation of tens of thousands of dollars will be needed to establish a basic holding in the first place.

7.0 Specializations

A specialization program in the context of "master's level programs in computer science" is defined as a professional program which, in general, would be administered by a Computer Science department, but which differs from traditional and/or academic programs in several important aspects. Here the emphasis is on the "specialist." A number of schools have already developed such programs. Almost always the title of the program is the key to the area of specialization and alerts the potential student to the nontraditional (in the computer science sense) nature of the offering. Examples of such programs include health computing (also called medical information science), library information science, and software design and development. In each case the professional practitioner produced by these specialist programs is expected to draw upon a broadly based knowledge of the technical foundations of computer science and be able to apply these concepts in the context of a particular application area, e.g., medicine or software development. These specialists are expected to be the professional level link between computer science and another specific technical area. Justification for suggesting that these programs be administered by computer science rests with the degree to which computer science dominates the course load imposed on the student.

There are intrinsic benefits from Computer Science departments having specialist programs. Nevertheless, it is not feasible for a particular department to have a specialist program unless it has a nucleus of faculty with appropriate similar interests and expertise. Emphasis and content will vary widely.

On the other hand, the Committee wants to discourage a somewhat frivolous proliferation of programs with specialist names. A specialist program should build on a more general Computer Science master's program rather than be a relatively inexpensive shortcut to a master's level program. Therefore, the following guidelines are presented:

- a. There must be a clear and continuing need for individuals with a particular training both locally and nationally. This need must be expected to last for several years.
- b. There must be a distinct body of knowledge which these individuals need and which is not provided by a generalist degree of the type presented earlier in this report.
- c. There must be at least three full-time faculty members available with expertise in this body of knowledge.
- d. Any needed resources (e.g., special hardware, databases) must be available in sufficient quantity locally. Provision must be made for periodic updating and improvement of these resources to keep pace with the state of the art.

6 A Conclusions

This report is the product of compromise. More than 200 Computer Science educators were consulted in its preparation. The Committee started out to produce a model curriculum for Computer Science master's degree programs similar to the model curriculum for bachelor's degree programs described in [2]. We quickly determined that even a small group of computer scientists could not agree on a model curriculum. We tried to develop separate model curricula for academic, professional, and specialization programs, but could not reach a consensus on any of those. Next we tried to develop a list of core concepts which every master's graduate should know. Lists of anywhere from five to thirty concepts were generated and rejected. What one person felt should be in the core another felt was relatively unimportant.

Computer Science is a volatile field. The Committee tried to determine in which directions the field was moving. We wanted to produce a forward looking report. Again, we could not reach a consensus. Each expert disagreed with the others.

This report makes some recommendations for what a master's degree in Computer Science should be and what it should not be. The report does not provide a blueprint for a master's program because the Committee believes the field is too new to have just one or even a small number of blueprints. The Computer Science faculty at an institution must be the ultimate determiners of what should and what should not be in the program. This report provides some recommendations for minimums which should be in every program. Beyond that we must defer to the mature, reasoned judgments of the local faculty.

* * * *

Appendix A

Contributors

The following people have made substantial contributions to this report:

- Robert M. Aiken, University of Tennessee
- * Richard H. Austing, University of Maryland
- * Bruce Barnes, National Science Foundation
- * Alfs T. Bertziss, University of Pittsburgh
- * Della T. Bonnette, University of Southwestern Louisiana
- Stephen E. Cline, Prentice-Hall, Inc.
- * John F. Dalphin, Indiana-Purdue University at Fort Wayne
- * Gerald L. Engel, Christopher Newport College
- Richard E. Fairley, Colorado State University
- ** John W. Hamblen, University of Missouri-Rolla
- * Alex A. J. Hoffman, Texas Christian University
- Lawrence A. Jehn, University of Dayton
- * William J. Kubitz, University of Illinois
- Joyce Currie Little, Community College of Baltimore
- * Kenneth I. Magel, University of Missouri-Rolla
- * Robert F. Mathis, Old Dominion University
- * John R. Metzner, University of Missouri-Rolla
- David Moursund, University of Oregon
- * James D. Powell, Burroughs-Wellcome Co.
- David Rine, Western Illinois University
- Kenneth Williams, Western Michigan University
- Anthony S. Wojcik, Illinois Institute of Technology
- Marshall C. Yovits, Indiana-Purdue University at Indianapolis

* Committee members

** Committee chairman

Appendix B

Course Descriptions

The descriptions are very brief to allow faculty to adjust these courses to their own environments. The Committee recognizes the need to develop an objective list of acceptable textbooks. For some of these courses, no textbook yet exists. Articles from the recent literature must be used. The Committee anticipates the availability of a textbook list in the SIGCSE Bulletin within the next two years.

Courses CS 1 through CS 18 are described in [2]. Courses CS 1, through CS 8 are prerequisite to a master's program.

- CS 1 Computer Programming I
- CS 2 Computer Programming II
- CS 3 Introduction to Computer Systems
- CS 4 Introduction to Computer Organization
- CS 5 Introduction to File Processing
- CS 6 Operating Systems and Computer Architecture I
- CS 7 Data Structures and Algorithm Analysis
- CS 8 Organization of Programming Languages
- CS 9 Computers and Society
- CS 10 Operating Systems and Computer Architecture II
- CS 11 Database Management Systems Design
- CS 12 Artificial Intelligence
- CS 13 Algorithms
- CS 14 Software Design and Development
- CS 15 Theory of Programming Languages
- CS 16 Automata, Computability, and Formal Languages
- CS 17 Numerical Mathematics: Analysis
- CS 18 Numerical Mathematics: Linear Algebra

The three numbers in parentheses following the course names below are: classroom hours per week, laboratory hours, and total course credit.

CS 19 Compiler Construction (3-0-3)
Prerequisite: CS 8

An introduction to the major methods used in compiler implementation. The parsing methods of LL(k) and LR(k) are covered as well as finite state methods for lexical analysis, symbol table construction, internal forms for a program, run time storage management for block structured languages, and an introduction to code optimization.

CS 20 Formal Methods in Programming Languages (3-0-3)
Prerequisite: CS 8

Data and control abstractions are considered. Advanced control constructs including backtracking and nondeterminism are covered. The effects of formal methods for program description are explained. The major methods for proving programs correct are described.

CS 21 Architecture of Assemblers (3-0-3)
Prerequisite: CS 6

Anatomy of an assembler: source program analysis, relocatable code generation, and related topics. Organization and machine language of two or three architecturally different machines; survey and comparison of these machines in various programming environments.

CS 22 Performance Evaluation (3-0-3)
Prerequisite: CS 6

A survey of techniques of modeling concurrent processes and the resources they share. Includes levels and types of system simulation, performance prediction, benchmarking and synthetic loading, hardware and software monitors.

CS 23 Analytical Models for Operating Systems (3-0-3)
Prerequisite: CS 6

An examination of the major models that have been used to study operating systems and the computer systems which they manage. Petri nets, dataflow diagrams, and other models of parallel behavior will be studied. An introduction to the fundamentals of queueing theory is included.

CS 24 Computer Communication Networks and Distributed Processing (3-0-3)
Prerequisite: CS 6

A study of networks of interacting computers. The problems, rationales, and possible solutions for both distributed processing and distributed databases will be examined. Major national and international protocols including SNA, X.21, and X.25 will be presented.

CS 25 High Level Language Computer Architecture (3-0-3)
Prerequisite: CS 6

An introduction of architectures of computer systems which have been developed to make processing of programs in high level languages easier. Example systems will include SYMBOL and the Burroughs B1700.

CS 26 Large Computer Architecture (3-0-3)
Prerequisite: CS 6

A study of large computer systems which have been developed to make special types of processing more efficient or reliable. Examples include pipelined machines and array processing. Tightly coupled multiprocessors will be covered.

CS 27 Real-Time Systems (3-0-3)
Prerequisite: CS 6

An introduction to the problems, concepts, and techniques involved in computer systems which must interface with external devices. These include process control systems, computer systems embedded in aircraft or automobiles, and operating systems which interface with operating system software for these systems.

CS 28 Microcomputer Systems and Local Networks (2-2-3)
Prerequisite: CS 6

A consideration of the uses and organization of microcomputers. Typical eight or sixteen bit microprocessors will be described. Microcomputer software will be discussed and contrasted with that available for larger computers. Each student will gain hands-on experience with a microcomputer.

CS 29 Applied Combinatorics and Graph Theory (3-0-3)
Prerequisites: CS 7, 13

A study of combinatorial and graphical techniques for complexity analysis including generating functions, recurrence relations, Polya's theory of counting, planar directed and undirected graphs, and NP complete problems. Applications of the techniques to analysis of algorithms in graph theory and sorting and searching.

CS 30 Theory of Computation (3-0-3)
Prerequisites: CS 7, 16

A survey of formal models for computation. Includes Turing Machines, partial recursive functions, recursive and recursively enumerable sets, the recursive theorem, abstract complexity theory, program schemes, and concrete complexity.

CS 31 Information System Design (3-0-3)
Prerequisites: CS 6, 11

A practical guide to Information System Programming and Design. Theories relating to module design, module coupling, and module strength are discussed. Techniques for reducing a system's complexity are emphasized. The topics are oriented toward the experienced programmer or systems analyst.

CS 32 Information Storage and Access (3-0-3)
Prerequisites: CS 6, 11

Advanced data structures, file structures, databases, and processing systems for access and maintenance. For explicitly structured data, interactions among these structures, accessing patterns, and design of processing/access systems. Data administration, processing system life cycle, system security.

CS 33 Distributed Database Systems (3-0-3)
Prerequisites: CS 11, 24

A consideration of the problems and opportunities inherent in distributed databases on a network computer system. Includes file allocation, directory systems, deadlock detection and prevention, synchronization, query optimization, and fault tolerance.

CS 34 Pattern Recognition (3-0-3)
Prerequisites: CS 6, 7

An introduction to the problems, potential, and methods of pattern recognition through a comparative presentation of different methodologies and practical examples. Covers feature extraction methods, similarity measures, statistical classification, minimax procedures, maximum likelihood decisions, and the structure of data to ease recognition. Applications are presented in image and character recognition, chemical analysis, speech recognition, and automated medical diagnosis.

CS 35 Computer Graphics (3-0-3)

Prerequisites: CS 6, 7

An overview of the hardware, software, and techniques used in computer graphics. The three types of graphics hardware: refresh, storage, and raster scan are covered as well as two-dimensional transformations, clipping, windowing, display files, and input devices. If a raster scan device is available, solid area display, painting and shading are also covered. If time allows, three-dimensional graphics can be included.

CS 36 Modeling and Simulation (3-0-3)

Prerequisites: CS 6, 7

A study of the construction of models which simulate real systems. The methodology of solution should include probability and distribution theory, statistical estimation and inference, the use of random variates, and validation procedures. A simulation language should be used for the solution of typical problems.

CS 37 Legal and Economic Issues in Computing (3-0-3)

Prerequisites: CS 9, 12

A presentation of the interactions between users of computers and the law and a consideration of the economic impacts of computers. Includes discussion of whether or not software is patentable, as well as discussion of computer crime, privacy, electronic fund transfer, and automation.

CS 38 Introduction to Symbolic and Algebraic Manipulation (3-0-3)

Prerequisite: CS 7

A survey of techniques for using the computer to do algebraic manipulation. Includes techniques for symbolic differentiation and integration, extended precision arithmetic, polynomial manipulation, and an introduction to one or more symbolic manipulation systems. Automatic theorem provers are considered.

References

1. Ashenurst, R. L. (Ed.) Curriculum recommendations for graduate professional programs in information systems, a report of the ACM Curriculum Committee on Computer Education for Management. *Comm. ACM* 15, 5 (May 1972), 363-398.
2. Austing, R.H., Barnes, B.H., Bonnette, D.T., Engel, L., and Stokes, G. (Eds.) Curriculum '78: Recommendations for the undergraduate program in computer science, a report of the ACM Curriculum Committee on Computer Science. *Comm. ACM* 22, 3 (March, 1979), 147-166.
3. Austing, R.H., Barnes, B.H., and Engel, G.L. A survey of the literature in computer science education since curriculum '68. *Comm. ACM* 20, 1 (Jan. 1977), 13-21.
4. Bertiss, A.T. The M.S. program in computer science. SIGCSE Bulletin (ACM) 11, 1 (Feb. 1979), 61-69.
5. Curriculum Committee on Computer Science (C³S). Curriculum '68: Recommendations for academic programs in computer science, a report of the ACM Curriculum Committee on Computer Science. *Comm. ACM* 11, 3 (March 1968), 151-197.
6. Hamblen, J.W. *Computer Manpower—Supply and Demand—by States*. Information Systems Consultants, R.R. 1, Box 256A, St. James, Mo., 1973, 1975, and 1979.
7. Hamblen, J.W., and Baird, T.B. *Fourth Inventory of Computers in U.S. Higher Education, 1976-77*. EDUCOM, Princeton, N.J., 1979.
8. Melkanoff, M.A. An M.S. program in computer science. SIGCSE Bulletin (ACM) 5, 1 (Feb. 1973), 77-82.
9. Teichroew, D. (Ed). Education related to the use of computers in organizations, position paper by the ACM Curriculum Committee on Computer Education for Management. *Comm. ACM* 14, 9 (Sept. 1971), 575-588.
10. Joint Committee of ACM and IEEE-CS. *A Library List on Undergraduate Computer Science, Computer Engineering, and Information Systems*. ACM, New York, 1978.
11. IEEE Computer Society. A curriculum in computer science and engineering. EH 0119-8, Los Alamitos, Calif., Nov. 1976.
12. Association for Computing Machinery. *Professional Code of Ethics*. ACM, New York.
13. Association for Computing Machinery. *Administrative Directory*. ACM, New York, 1980.

Educational Programs in Information Systems

A Report of the ACM Curriculum Committee on Information Systems

Editor: Jay F. Nunamaker, Jr., University of Arizona

This report describes the status of educational programs in Information Systems at the B.S., M.S., and Ph.D. levels. A survey was conducted during the period June 1977-June 1979 of schools of Business Administration, Departments of Computer Science, Engineering Colleges, and academic units offering programs in Information Systems. A one-page description of each program was then generated according to a standard format. This standardized description was used as a guide to summarize information about each program.

The report outlines career opportunities in Information Systems and lists brief descriptions of positions available to graduates of Information Systems programs. The need for an Information Systems program and problem areas with respect to teaching information systems are discussed. The results of the survey include a listing of the most common names for the Information Systems program and an evaluation of the number of programs that met the guidelines established by the Curriculum Committee on Computer Education for Management in 1972 and 1973. A list of institutions by degree level that met the proposed guidelines is presented.

Key Words and Phrases: education, management systems, systems analysis, management information systems, information systems development, information analysis, systems design, curriculum, data processing.

CR Categories: 1.52, 3.51.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.
© 1981 ACM 0001-0782/81/0300-0124 \$00.75.

Preface

The purpose of this report is to summarize the status of Information Systems (IS) programs in the United States, a project undertaken by the ACM Curriculum Committee on Information Systems (C²IS).^{*} The need for degree programs in information systems summarized in this report was documented in an earlier committee position paper [1]. Comprehensive curriculum recommendations for undergraduate and graduate programs in information systems were presented in later reports [2, 3].

The tasks of the present project included: (a) the collection of college and university catalogs, brochures, general descriptions of information systems programs; and (b) the classification of information systems with respect to minimal criteria.

Many people assisted the Committee in its work in preparing this report; a draft version was circulated to members of the academic and professional community. The Committee would like to thank Ron Harding for his work in collecting, analyzing, and summarizing the responses from the many universities. Without his devotion to the project, this report would not have been possible. The Committee would also like to thank Benn Konsynski for his assistance during the past two years. The Committee, of course, assumes full responsibility for the substance of the report and the conclusions and recommendations contained in it.

The Curriculum Committee on IS membership dur-

^{*} Formerly known as the Curriculum Committee on Computer Education for Management (C³EM). The Curriculum Committee on Information Systems is a subcommittee of the Curriculum Committee on Computer Education.

ing the preparation of this report was:

Jay F. Nunamaker, Jr., (Chairman), University of Arizona.

William Cotterman, Georgia State University

J. Daniel Couger, University of Colorado

Gordon B. Davis, University of Minnesota

Benjamin Diamant, IBM Corporation

Andrew B. Whinston, Purdue University

Marshall Yovits, Indiana-Purdue University at Indianapolis

The validity of the list of programs contained in this report is accurate as of October 1979.

1. Introduction

Since there is a wide spectrum of interests among students, faculty, and administrators, IS programs vary, but in general they fall into one of two groups:

(1) Those designed for the person who wants to learn computers as a part of preparation for a general management career; and

(2) Those designed for the person who wants a lifetime career in information systems.

The universities and colleges listed in this report play many different roles in the educational process. The large teaching-oriented institutions do not pursue the same goals and offer the same kinds of courses as do strongly research-oriented universities. A wide variety of existing courses and programs reflect the orientation of the institution of which each is a part. A brief description of each program satisfying the minimum criteria established by the Committee is presented in the full report [4].

The term "information system" has come to refer to a computer-based system for providing information to members of an enterprise. The term "management information system" indicates a major emphasis in information systems. The IS discipline provides the analytical framework and the methodology to analyze, design, implement, and manage complex information/decision systems. An IS is defined as "a set of personnel, computer hardware, software packages, computer programs, data files, communication systems, decision models, organizational procedures and practices, so structured and assembled as to ensure data quality, transmission, processing, and storage in accordance with a given performance criterion to assist decision-making." An IS integrates systems analysis, statistics, management, management science, accounting, economics, finance, marketing, production, and computer and communications technology to accomplish these tasks.

In this respect, IS is the embodiment of the so-called "systems approach," i.e., viewing organizations as a complex whole composed of interacting subparts. The single most distinguishing feature of an information system, however, is its emphasis upon the flow of information

within the organization. Thus, it is "information" that is the common link binding the organizational subparts. As organizations grow in size and complexity, the need for better and more timely information and for improved decision-making techniques becomes critical. Recent advances in computer and communications technology are making it practical to integrate the planning and control of operations across functional areas and geographic distances. Concurrent with these developments is the recognition that information is a resource which should be subject to managerial planning and control in the same way as other resources such as land, labor, and capital.

The information system usually includes the concept of a comprehensive database accessible by computer and available for analysis, processing, and decision-making purposes by the entire organization. To be integrated, an information system must provide support to the three areas of management: planning, control, and operations. Viewed from a systems perspective, the information system must provide transactional processing, decision-making, and planning capabilities designed to support the management process by providing information, analysis, and programmed decision-making both routinely and on request. The system typically incorporates quantitative analytical tools (e.g., work order scheduling, economic order quantity determination) and integrates planning (e.g., modeling, simulation) with current operations to provide a dynamic mechanism for the planning and control of the organization.

2. Career Opportunities in Information Systems

The purpose of IS curricula is to prepare systems analysts, systems designers, application programmers, database administrators, information retrieval specialists, and communication systems specialists. The emphasis is on the functional areas of management, including accounting, production, marketing, finance, and the applications of computers in those functional areas.

With more and more companies processing information to aid management decision-making and with the dynamic increase of the use of computers, graduates with a background in IS are in great demand. Career opportunities in information systems exist at numerous levels wherever computers are applied to industry and government. The demand has been so great that the postgraduates with limited course work in IS have been actively recruited. Excellent opportunities continue to be available in the field because of the rapid expansion of data processing systems in business and government.

Although some attempts have recently been made to standardize and describe job classifications in the computer field, there is still much room for improvement. The IS curricula presented in this report were designed to prepare individuals for the following career paths:

Information Systems Manager

The information systems manager must be able to assume responsibility for:

- formulating and defining requirements for the organization.
- interacting with other managers in order to provide solutions to management, scientific, and business problems.
- evaluating proposed information systems and managing a team of analysts/programmers and other specialists in implementing a system.

Systems Analyst/Designer

The systems analyst/designer is involved with the analysis, design, and implementation of the information system and is responsible for:

- gathering facts about the existing system and analyzing them to determine the effectiveness of the proposed system.
- assisting the user in determining information needs, preparing software and hardware specifications.
- designing new systems, recommending system changes, and being involved in testing and implementing these systems.

Applications Programmer

The applications programmer is primarily concerned with the actual construction of the system and is responsible for:

- designing detailed logic and diagrams for programs.
- coding the programs.
- verifying the accuracy and completeness of programs.
- documenting the programs and operating instructions according to organizational standards.

Database Administrator

The database administrator's responsibilities include:

- developing and maintaining a dictionary of standard data definitions.
- assisting in the development and design of the database, maintaining secrecy, completeness, and timeliness of the database.
- designing and operating the database security systems against unauthorized access and use of the files.
- providing liaison between analysts and users of the database.
- advising analysts/designers, programmers, and users of the most efficient ways to use the database.

Communications Analyst

The communications analyst is responsible for:

- working with the systems analyst/designer on the design and implementation of distributed information systems.
- taking responsibility for the data processing/data communications interface.
- being knowledgeable in data communications and computer hardware/software systems.

Systems Librarian

The systems librarian is responsible for:

- the control and maintenance of the files and programs being developed and maintained in the systems.

The position of Systems Librarian varies from organization to organization, with some organizations viewing the position as clerical and others seeing it as providing entry level experience for junior programmers.

Some organizations separate the tasks of analysis and design into two jobs; others combine the jobs of analysis and programming. In either case, the tasks performed are properly identified above. Other jobs in a Management Information Systems department, or an Information Systems department, are typically handled by graduates of computer science degree programs: computer hardware specialists, systems programmers, and scientific/mathematical application programmers.

3. Need for an Information Systems Degree Program

Computer-related occupations range from those requiring heavy technical skills in computer hardware and software and almost no organizational knowledge to those which demand extensive organizational skills and only modest computer hardware/software knowledge. As illustrated in Figure 1, in the area of programming, systems programming requires a heavy technical knowledge while applications programming requires a mix of both technical and organizational skills. In the area of systems analysis and design, the physical system design task of configuring hardware and software requires extensive understanding of hardware and software information analysis, a moderate level of technical expertise, and a thorough knowledge both of organizational functions and processes and of human behavior in systems and organizations.

It is generally agreed in the computer industry that there is a shortage of trained personnel across all occupational categories. Many studies have assessed the shortage of trained personnel needed for effective use of computer technology. While the results are not always quantitatively consistent, all support the position that the shortage is acute. The experience of the Curriculum Committee on IS suggests that the shortage of trained personnel is not uniform across the technical/organizational spectrum. The observation of this Committee is that the demand for personnel having a combination of technical and organizational skills is relatively much greater than the demand for solely technical skills. (Figure 1 pictures the Committee experience.)

A hidden but significant impact of the imbalance of supply and demand as observed by the Committee is the drawing of people trained toward the technical end of the spectrum into positions toward the organizational

Fig. 1. Comparison of Activities, Degree Programs, and Supply and Demand with Respect to Technical and Organizational Knowledge Dimensions

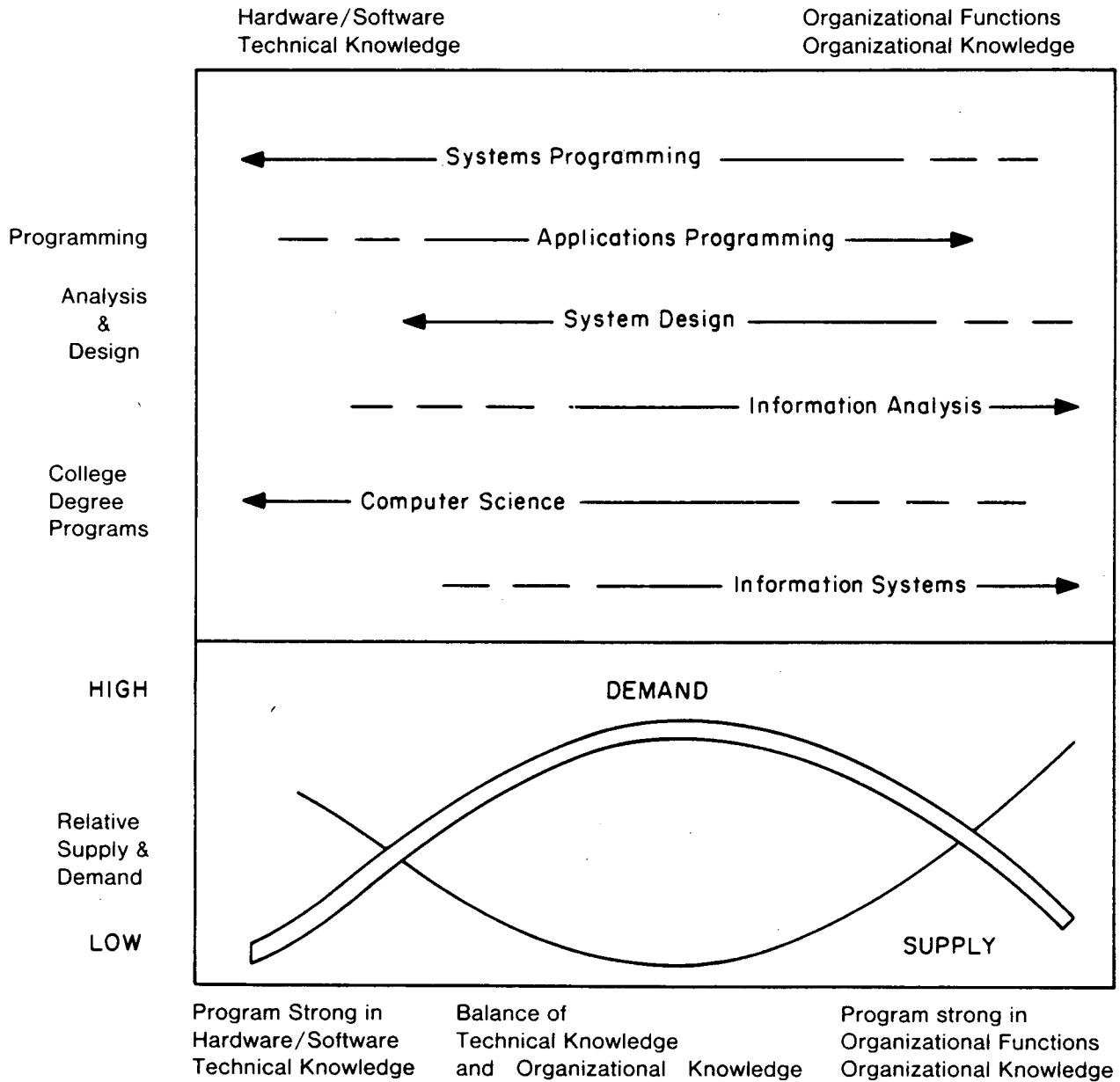
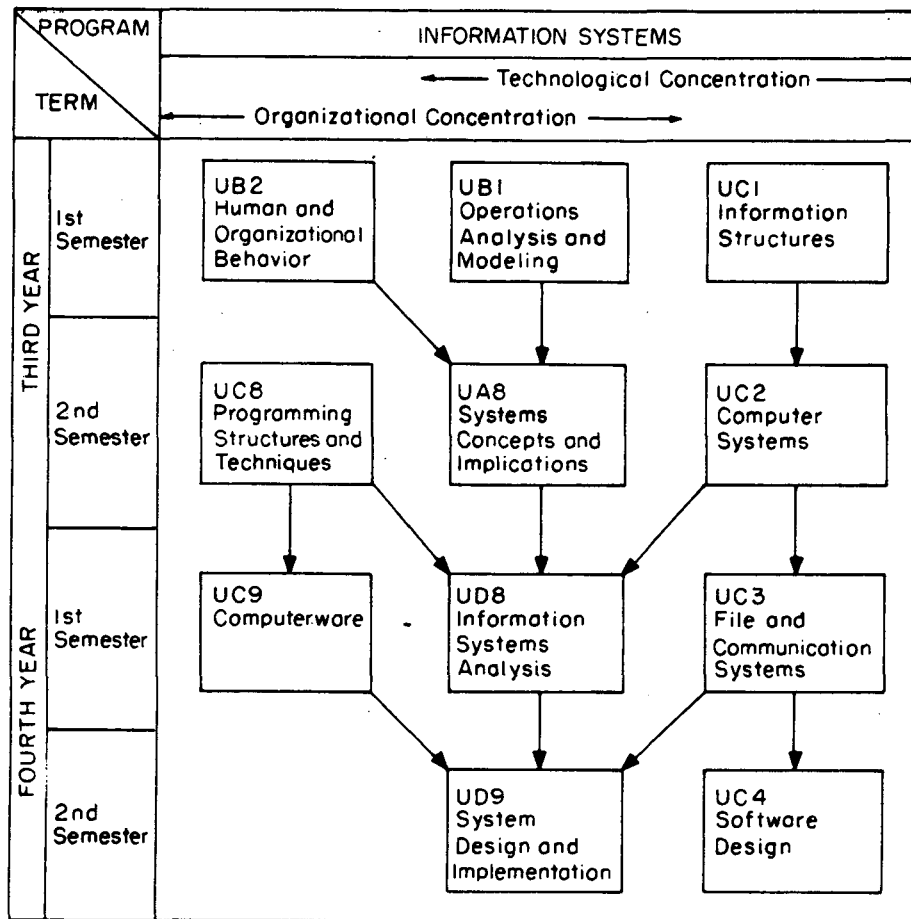


Fig. 2. Core Courses and Sequence for Undergraduate IS Program



end. In other words, positions needing heavy organizational skills are being filled with persons having heavy technical but very low organizational training. This mismatch creates problems in the analysis and design of information systems. It also makes it appear that the shortage of trained personnel is uniform across the entire discipline when, in fact, it is not.

Just as programs stressing strong technical-weak organizational skills provide inadequate background for analysis and design, programs that produce students with strong organizational-weak technical skills prepare them poorly for handling the complexities of systems analysis, design, and implementation.

The need, then, is for a degree program which provides both technical and organizational knowledge. Operationally, this means that the IS curriculum must include subject matter from both the traditional disciplines of computer science and those of administration and management.

The use of computers in support of organizational processes such as data processing, decision support, and

information storage and retrieval requires systems so designed and implemented that they:

- identify information requirements (based on an understanding of organizational functions, organizational processes, and decision-making).
- fit technical characteristics into the behavioral framework of the organization.
- match technical design with human characteristics.

Computer science degree programs typically emphasize hardware and software technical knowledge and exclude the organizational dimensions. Computer science curricula, therefore, serve to meet the needs of those occupations needing a technical emphasis (Figure 1). There is need for another program (IS) to meet the other range of positions. The IS curriculum has some subject matter also contained in computer science but has necessary organizational and behavioral coverage.

Not only is the demand not uniform across the technical/organizational dimension, but academic programs supplying trained personnel are radically out of

balance with demand. For example John Hamblen, in the 1979 study of computer manpower supply and demand, found a ratio of almost five computer science degree programs for every information systems/data processing degree program [5].

ACM Curriculum Recommendations for IS Programs

The programs recommended by C³EM in 1972 [3] and 1973 [2] are summarized below, for comparison with the programs listed in the following section. The prerequisites for both the undergraduate and graduate programs are:

- finite mathematics, including the fundamentals of formal logic, sets and relations, and linear algebra.
- elementary statistics, including the fundamentals of probability, expected value, and construction of sample estimates.
- elementary computer programming, including prob-

lem analysis and algorithm synthesis, and competence in a higher language.

- elementary economics, including microeconomics and theory of the firm, and price theory.
- elementary psychology, including fundamentals of personality formation, attitudes, and motivation.

The courses at both the undergraduate and graduate level were divided into four groups. The four groups are:

- A. Analysis of Organizational Systems
- B. Background for Systems Development
- C. Computer and Information Technology
- D. Development of Information Systems

The 11 courses and course sequence required for the undergraduate program are described in Figure 2 (reproduced from [2]). The 13 courses and course sequence

Fig. 3. Core Courses and Sequence for Graduate IS Program

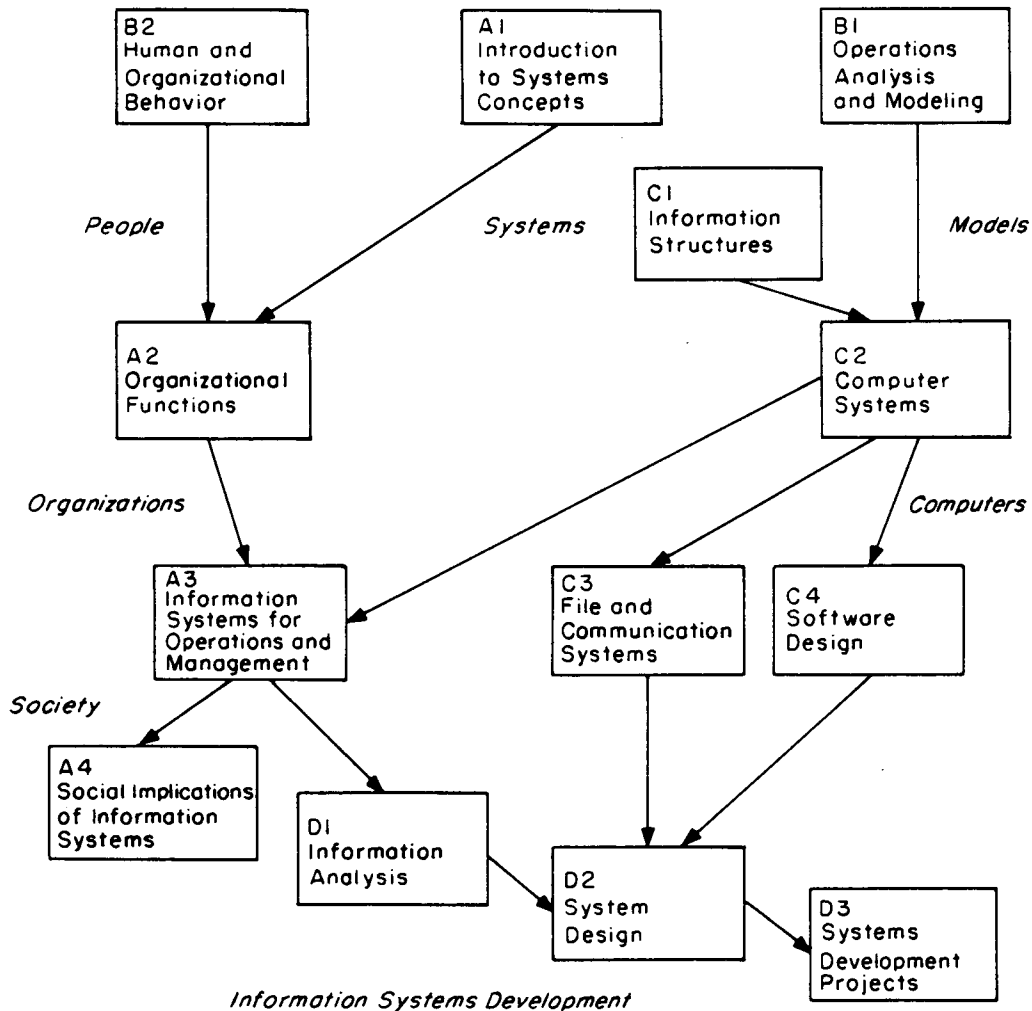


Table 1. Bachelor's Degree Programs Meeting the Criteria

School	Department	Degree
Alabama-Birmingham, Univ. of	Dept. of Computer & Information Sciences	B.S.
Arizona State Univ.	Dept. of Quantitative Systems	B.S.B.A.
Arizona, Univ. of	Dept. of Management Information Systems	B.S.B.A.
Appalachian State Univ.	Dept. of Bus. Ed. & Office Admin.	B.S.B.A.
Bowling Green State Univ.	Dept. of Acctg. & M.I.S.	B.S.B.A.
Boise State Univ.	Dept. of Acctg. & Data Processing	B.B.A.
California State Univ. at Dominguez Hills	Dept. of Business Administration	B.S.B.A.
California State Univ. at Fullerton	Dept. of Quantitative Methods	B.A.B.A.
California State Univ. at Los Angeles	Dept. of Acctg. & Business Info. Systems	B.S.B.A.
California State Univ. at Sacramento	Dept. of Acctg. & Management Info. Systems	B.S.B.A.
California State Polytechnic Univ.	Dept. of Information Systems	B.S.
Colorado State Univ.	Dept. of Management Science & Info. Systems	B.S.B.A.
Colorado, Univ. of	Dept. of Management Science	B.S.B.A.
Drexel Univ.	Dept. of Accounting	B.S.B.A.
Eastern Michigan Univ.	Dept. of Operations Research & Info. Systems	B.B.A.
Eastern New Mexico Univ.	Dept. of Acctg. & Computer Info. Science	B.S.B.A.
Eastern Washington Univ.	Dept. of Acctg. & Decision Science	B.A.B.A.
Florida Atlantic Univ.	Dept. of Administration & Systems	B.A.S.
Georgia State Univ.	Dept. of Information Systems	B.B.A.
Hawaii, Univ. of		B.B.A.
Houston, Univ. of	Dept. of Management	B.B.A.
Idaho State Univ.	Dept. of Information & Computer Science	B.B.A.
Indiana Univ.	Dept. of Admin. Systems & Business Educ.	B.S.B.
Indiana Univ. of PA	Business Management Department	B.S.
Long Island Univ.	School of Business Administration	B.S.
Mankato State Univ.	Dept. of Computer Science	B.S.
Maryland, Univ. of	Dept. of Information Systems Management	B.S.
Massachusetts, Univ. of	Dept. of Accounting	B.S.B.A.
Miami, Univ. of	School of Business Administration	B.B.A.
Michigan, Univ. of	School of Business Administration	B.B.A.
Minnesota, Univ. of	Dept. of Management Sciences	B.S.B.A.
Murray State Univ.	Dept. of Management	B.S.B.
Nebraska-Lincoln, Univ. of	Dept. of Management	B.S.B.A.
Nevada-Reno, Univ. of	Dept. of Acctg. & Information Systems	B.S.B.A.
New Mexico State Univ.	Dept. of Accounting & Finance	B.B.A.
New York Univ.	Dept. of Computer Applic. & Info. Systems	B.S.
Northern Arizona Univ.	College of Business Administration	B.S.B.A.
North Florida, Univ. of	Dept. of Accounting	B.B.A.
Ohio State Univ.	Dept. of Computer & Information Science	B.S.B.A.
Oklahoma State Univ.	Dept. of Administrative Sciences	B.S.B.A.
Purdue Univ.	Dept. of Computer Science; Dept. of Comp. Tech.	B.S.C.S.; B.S.
Rochester Institute of Technology	School of Computer Science & Technology	B.T.
Saint Cloud State Univ.	Dept. of Quant. Methods & Info. Systems	B.S.
San Diego State Univ.	Dept. of Information Systems	B.S.B.A.
San Francisco State Univ.	Dept. of Data Systems & Quant. Methods	B.S.
South Carolina, Univ. of	Dept. of Management Science	B.S.B.A.
Southern Mississippi, Univ. of	Dept. of Accounting	B.S.B.A.
Temple Univ.	Dept. of Computer & Information Sciences	B.B.A.
Wisconsin-Whitewater, Univ. of	Dept. of Management	B.B.A.
Virginia Commonwealth Univ.	Dept. of Information Systems	B.S.B.
West Virginia Institute of Technology	Division of Business Administration	B.S.
Xavier Univ.	Dept. of Management & Information Systems	B.S.B.A.

required for the graduate program are described in Figure 3 (reproduced from [3]).

4. A Survey of Existing Degree Programs in Information Systems

The C²IS surveyed information systems programs to determine how well the curriculum guidelines described earlier have been implemented.

Step one of this project collected college and university catalogs, brochures, and general descriptions of in-

formation systems programs. Letters requesting undergraduate and graduate catalogs, brochures, and other material identifying or describing current program requirements and course offerings in the information systems area were sent to the following organizations:

205 business schools meeting AACSB (American Assembly of Collegiate Schools of Business) accreditation standards;

149 computer science department heads; and

159 collegiate chapters of the ACM.

There was a 53 percent response rate from the AACSB

Table 2. Master's Degree Programs Meeting the Criteria

School	Department	Degree
Alabama-Birmingham, Univ. of	Dept. of Computer & Information Sciences	M.S.C.I.S.
Arizona, Univ. of	Dept. of Management Information Systems	M.S.M.I.S.
Bentley College	Computer Systems Department	M.S.
California State Univ. at Sacramento	Dept. of Acctg. & Mgmt. Info. Science	M.S.M.I.S.
California-Los Angeles, Univ. of	Dept. of Management	M.S.M.
Chicago, Univ. of	Graduate School of Business	M.B.A.
Colorado State Univ.	Dept. of Management Science & Info. Systems	M.S.
Colorado, Univ. of	Dept. of Management Science	M.S.
Duquesne Univ.		M.S.B.I.S.
Eastern Michigan Univ.	Dept. of Operations Research & Info. Systems	M.S.I.S.
Florida Atlantic Univ.	Dept. of Administration & Systems	M.A.S.
Georgia Institute of Technology	School of Information & Computer Science	M.S.I.C.S.
Georgia State Univ.	Dept. of Information Systems	M.B.I.S.
Harvard Univ.	Division of Applied Science	M.E.
Houston, Univ. of	Dept. of Management	M.B.A.
Indiana University	Dept. of Operations & Systems Management	M.B.A.
Massachusetts Institute of Technology	Sloan School of Management	M.S.
Michigan, Univ. of	Graduate School of Business Admin.	M.B.A.
Minnesota, Univ. of	Dept. of Management Science	M.B.A.
Nebraska-Lincoln, Univ. of	Dept. of Management	M.A.B.A.
New Mexico, Univ. of	School of Business & Admin. Sciences	M.B.A.
New York Univ.	Dept. of Computer Applic. & Info. Systems	M.B.A.
Northwestern Univ.	Dept. of Acctg. & Information Systems	M.M.
Ohio State Univ.	Dept. of Computer & Information Science	M.S.C.I.S.
Pennsylvania State Univ.	Dept. of Acctg. & Management Info. Systems	M.S.B.A.
Purdue Univ.	Graduate School of Industrial Admin.	M.S.M.
Saint John's Univ.	Dept. of Quantitative Analysis	M.B.A.
San Diego State Univ.	Dept. of Information Systems	M.B.A.
San Jose State Univ.		M.S.C.I.S.
Temple Univ.	Dept. of Computer & Information Sciences	M.B.A.
Texas A&M Univ.	Dept. of Business Analysis & Research	M.S.
Texas Technology Univ.	Dept. of Information Systems & Quant. Sci.	M.S.B.A.
Wisconsin-Madison, Univ. of	Dept. of Accounting & Quant. Analysis	M.S.B.A.
Wisconsin-Milwaukee, Univ. of	Dept. of Business Administration	M.S.M.

Table 3. Doctoral Degree Programs

School	Department	Degree
Arizona, Univ. of	Dept. of Management Information Systems	Ph.D.
California, Univ. of at Los Angeles	Dept. of Management	Ph.D.
California, Univ. of at Irvine	Dept. of Computer Science	Ph.D.
Carnegie-Mellon Univ.	Graduate School of Industrial Administration	Ph.D.
Case Western Reserve Univ.	Graduate School of Management	Ph.D.
Chicago, Univ. of	Graduate School of Business	Ph.D.
Georgia, Univ. of	Graduate School of Business	Ph.D.
Georgia State Univ.	Dept. of Information Systems	Ph.D.
Harvard Univ.	Control Department	D.B.A.
Houston, Univ. of	Dept. of Management	Ph.D.
Indiana, Univ. of	School of Business	Ph.D.
Iowa, Univ. of	School of Business	Ph.D.
Massachusetts Institute of Technology	Sloan School of Management	Ph.D.
Michigan, Univ. of	Graduate School of Business Administration	DBA
Minnesota, Univ. of	Management Science Dept.	Ph.D.
Nebraska-Lincoln, Univ. of	School of Business	Ph.D.
New York Univ.	Dept. of Computer Applications and Information Systems	Ph.D.
North Texas State Univ.	School of Business	Ph.D.
Ohio State Univ.	Dept. of Computer and Information Science	Ph.D.
Pennsylvania, Univ. of	Dept. of Decision Science	Ph.D.
Pittsburgh, Univ. of	School of Business	Ph.D.
Purdue Univ.	Krannert School of Industrial Administration	Ph.D.
Rochester, Univ. of	Graduate School of Management	Ph.D.
Southern California, Univ. of	School of Business	Ph.D.
Southwestern Louisiana, Univ. of	Dept. of Computer Science	Ph.D.
Stanford Univ.	Graduate School of Business	Ph.D.
Temple Univ.	Dept. of Computer and Information Sciences	Ph.D.
Texas Technology Univ.	Dept. of Information Systems and Quantitative Science	D.B.A.

schools; a 49 percent response rate from the computer science departments; and an 11 percent response rate from the ACM chapters.

From a review of the materials, it was determined that 91 schools offered some form of IS course study. The 91 colleges and universities offered 70 programs at the bachelor's level and 54 programs at the master's level. Many institutions offered both an undergraduate and a graduate program.

Names of Programs

From the original list of 124 reported IS programs, there were 37 different names associated with the field. The two most common, by far, were "Management Information Systems" and "Information Systems."

Most Common Names	
No.	Name
27	Management Information Systems
18	Information Systems
5	Business Information Systems
4	Business Data Processing
4	Computer Information Science
4	Management Systems
3	Computer Information Systems
3	Information Processing
3	Information Systems Analysis and Design
3	Information Science
9	Names Used Twice
18	Names Used Once

5. Criteria for Inclusion of Degree Programs

It was agreed by committee members that the programs should prepare students in the knowledge areas listed below.

- Computer Systems
 - Hardware Components
 - Software Components
- Communication Systems
 - Hardware Components
 - Software Components
- Programming Systems
- Data Systems
 - Data Structure/File Structures
 - Database Management
- Systems Analysis and Design
- Modeling of Systems and/or Performance Evaluation
- Organizational and Administrative Functions
 - Accounting
 - Finance
 - Marketing
 - Production
 - Organizational Behavior
 - Organizations and Management

Very few programs met these desired standards. One of the problems with the original curriculum recommendations was that they call for the implementation of many courses; but most universities do not have the resources to add 10 to 15 new courses. The Committee, therefore, settled for minimum criteria consisting of five (semester) courses in technical areas (computer science, information systems) and four courses in organization and administrative functions.

Minimum Criteria for Selection

Five semester courses in the following areas:

- Programming
- Data/File Structures
- Systems Analysis/Design

Four semester courses in the following areas:

- Organization and Administrative Functions
- Organization and Management
- Accounting
- Finance
- Marketing
- Organizational Behavior

Evaluation Results

The lists of 70 bachelor's and 54 master's programs were evaluated as meeting or not meeting minimum requirements for an "ACM Information Systems Program."

	No. of Programs	Satisfied Criteria	Did not Meet the Criteria
Bachelor's	70	53	17
Master's	54	34	20
Doctoral	28	not evaluated	

The doctoral programs were not evaluated with respect to minimum criteria since most programs can be customized to the interests of the individual student.

Location of Programs Meeting Criteria

The results indicated that only 53 undergraduate and 34 graduate programs satisfied the minimum criteria for classification as an information systems program based on the ACM curriculum. The colleges of Business or Management were found to be the home for a majority of Information Systems Programs.

	Total	Business or Management College	CS Dept., Engineering College
Bachelor's	53	42	11
Master's	34	25	9

Problem Areas in IS Curriculum

Another aspect of the project was to identify problem areas in the curriculum. The single most critical problem area appeared to be contents of the systems analysis and design courses. Many new techniques such as structured design and top-down approaches are available, but systems analysis and design instruction appears still to be on an *ad hoc* basis. Course descriptions and content revealed very little commonality in what is taught in

systems analysis and design courses. Content ranged from the very general overview (a sort of "gee whiz" course) to a "nitty-gritty, nuts and bolts" type of course. The latter was often essentially a programming course, largely ignoring the problems of analysis and design.

The original curriculum recommendations provided a general model for all universities and colleges. The next project of this Committee will be to examine the original model to assess its current applicability and to determine whether additional models are required.

6. Lists of Institutions Meeting the Criteria

This section includes lists of bachelor's (Table 1) and master's (Table 2) programs, meeting the criteria established by the Committee, which appear on the preceding pages. All of the Ph.D. (Table 3) programs that were submitted were included. The Ph.D. programs are very specialized and, therefore, it was not appropriate to evaluate them with respect to minimal criteria.

The list of institutions contained in Tables 1, 2, 3, and Reference [4] can be considered valid as of October 1979. Sometimes it was difficult to determine whether or not an institution met the guidelines. The complete descriptions of individual programs [4] can be obtained from ACM Headquarters.

In many institutions where a program exists in one college or department, there is a good chance that a similar program also exists in a complementary department. In some instances programs at the same institution that appeared to be very similar were not included; the IS program listing was included for the department that offered the primary courses.

7. Summary

The decision to specify whether or not a school met the criteria was often based on the course titles. In many instances, the Committee did not have course outlines so schools may have been classified as not meeting the criteria when in fact they do. The Committee recognizes that very often the course content is not reflected by the course title for one reason or another.

In summary, the objective of the report is to illustrate that many IS programs exist and the programs vary tremendously from university to university.

We know that the list of programs in this report is incomplete. We invite those schools not listed or listed incorrectly to write us and to send information about their programs, which we will include in the next revision of the report. Information on the programs should follow the format used in Reference [4].

Having attempted to identify programs that satisfy a minimum common core in IS, the authors of this report hope it will provide information and guidelines to institutions planning to implement programs in IS.

References

1. Teichroew, D. (Ed.). Education related to the use of computers in organizations (Position Paper—ACM Curriculum Committee on Computer Education for Management). *Comm. ACM* 14, 9 (Sept. 1971), 573–588.
2. Couger, J. (Ed.). Curriculum recommendations for Undergraduate programs in information systems. *Comm. ACM* 16, 12 (Dec. 1973), 727–749.
3. Ashenurst, R.L. (Ed.). Curriculum recommendations for graduate professional programs in information systems. *Comm. ACM* 15, 5 (May 1972), 363–398.
4. Nunamaker, J.F., Cotterman, W., Couger, J.D., Davis, G., Diamant, B., Whinston, A., and Yovits, M. Status and description of programs in information—ACM Special Report. To appear.
5. Hamblen, J.W. *Computer Manpower Supply and Demand—By States, 1979*. Information Systems Consultants, St. James, Mo., 1979, pp. 2–3.

Recommendations and Guidelines for an Associate Level Degree Program In Computer Programming

A Report of the ACM Committee on Curriculum
for Community and Junior College Education

Editors: Joyce Currie Little, Community College of Baltimore
Marjorie Leeson, Delta College
Harice Seeds, Los Angeles City College
John Maniotes, Purdue University Calumet Campus
Richard H. Austing, University of Maryland
Gerald L. Engel, Christopher Newport College

This report presents recommendations and guidelines for a two-year associate degree career program in computer programming. It represents the result of five years of effort by the Association for Computing Machinery's Committee on Curriculum for Community and Junior College Education and is the consensus of a large number of computer educators and industry representatives.

The program is designed to prepare computer personnel for entry-level jobs in applications computer programming. Graduates should be qualified to work on project teams with a systems analyst on projects requiring medium-to-large-scale computer equipment. Specialized options within the program allow a variety of career paths in the computer field.

The report gives the purpose and objectives of the curriculum, a detailed topical outline of recommended subject matter content, and guidelines for local curriculum design. Recommendations are given regarding resources needed for the implementation of the program.

These guidelines are applicable to programs in vocational-technical institutes, community and junior colleges, and colleges and universities interested in the education of computer personnel for computer programming. They may be of use in evaluating existing programs or in establishing new ones.

Key Words and Phrases: associate level programs, career programs, community and junior colleges, computer education, computer programming, computer science education, curriculum, data processing education, education, information systems education, two-year programs, vocational-technical institutes.

CR Categories: 1.52, 2.42, 3.5.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage. The ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.
© 1981 ACM.

The current address of Joyce Currie Little is Towson State University, Baltimore, MD 21204.

Contents

Preface

1. Introduction

- 1.1 Purpose
- 1.2 Historical Perspective

2. Computer Programming Jobs: Prospects, Skills, and Environments

- 2.1 Duties of the Computer Programmer
- 2.2 Job Descriptions
- 2.3 Job Environment
- 2.4 Career Paths and Advancement
- 2.5 Employment Supply and Demand
- 2.6 Qualities for Success

3. Objectives of the Curriculum

- 3.1 Purpose of the Curriculum
- 3.2 Performance Objectives

4. Content of the Curriculum

- 4.1 Required Computer Content
 - 4.1.1 Principles and Techniques of Programming
 - 4.1.2 Programmer Environment
- 4.2 Optional Computer Content
- 4.3 Related Content
- 4.4 Design Considerations
- 4.5 Major Programming Language
- 4.6 Access to Equipment
- 4.7 Structured Programming

5. Resources for Implementation

- 5.1 Organizational Matters
- 5.2 Faculty and Staff
- 5.3 Equipment and Access
- 5.4 Instructional Materials
- 5.5 Adaptation for Continued Relevancy

6. Articulation

- 6.1 Industry
- 6.2 Other Educational Institutions
- 6.3 The Computer Profession

7. Summary

References

Appendices

A. Reference Materials

- 1. Computer Programming as a Career
- 2. Instructional Bibliographies
- 3. Periodicals and Newsletters

B. Curriculum Implementation Material

C. Student Activities and Memberships

D. Source of Job Descriptions

E. Contributors and Workshop Participants

F. ACM Committee on Curriculum for Community and Junior College Education, 1981

Preface

This publication of the Association for Computing Machinery (ACM) represents the first effort of a scientific and technical society to provide curriculum recommendations and guidelines for the education of computer programmers at the associate degree level. Prepared by the ACM Committee on Curriculum for Community and Junior College Education, under the auspices of the Education Board of the ACM, this work was coordinated by the Chair, Joyce Currie Little, Community College of Baltimore, Maryland, with the assistance of Richard H. Austing, University of Maryland, College Park, and Gerald L. Engel, Christopher Newport College, Newport News, Virginia.

The initial meeting of the Committee was held in 1975 with support from the ACM Special Interest Group on Computer Science Education (SIGCSE). Thirteen community and junior college educators established curriculum work as the highest priority with emphasis on the improvement of existing programs to educate computer programmers. Several workshops followed and a draft report resulted which appeared in the *SIGCSE Bulletin* in June 1977. Numerous presentations were given at conferences and revisions of the draft were sent to several hundred interested persons. Reactions were discussed by the Committee for incorporation into the report.

The Committee would like to thank all those who made this report possible. We would especially like to thank Richard H. Austing and Gerald L. Engel for encouragement and support given to the Committee and to this project. Professor Austing was the chairman of SIGCSE and the vice-chairman of the Education Board during most of this time, and Professor Engel was chairman of the Curriculum Committee on Computer Education. Without these two persons, the Committee might not have existed. We also thank William F. Atchison, University of Maryland, College Park, who was chairman of the Education Board during most of this work, and David Kniefel, New Jersey Educational Computer Network, who was chairman during its publication. The Committee would like to thank Maureen Shumate for her manuscript review.

The Committee is grateful to all those persons who helped by contributing their comments, reactions, and advice. Many people participated in workshops, reviewed drafts, published announcements, and gave presentations. All those who contributed toward the report are named in Appendix E. Special thanks go to the Virginia Institute for Marine Science, Gloucester Point, Virginia, for hosting several sessions and to the National Technical Institute for the Deaf at Rochester Institute of Technology for assistance with manuscript production.

This work is intended for use by the computer industry, vocational-technical schools and institutes, community and junior colleges, and colleges and universities interested in the education of personnel for computer programming. ACM and the Committee would be pleased to hear your reactions.

1. Introduction

1.1 Purpose

These recommendations are given for a two-year associate degree career program in computer programming. The program is designed to prepare computer personnel for entry-level jobs in applications computer programming. Graduates should be qualified to work on applications project teams with a systems analyst on projects which require medium-to-large-scale computer equipment. The recommendations given should ensure sufficient foundation so that graduates with experience and continued learning may advance in any of a wide variety of career paths appropriate to their own particular interests and abilities.

The report gives the purpose and objectives of the curriculum and provides a detailed topical outline of required core subject matter content. Recommendations are also given for content in other related subjects and in an applied area for specialized applications work. Suggestions for laboratory facilities and staff resources are made. Guidelines for implementation are given to assist an institution in its own curriculum design to meet the needs of local industry. Lists of resource materials are given in the Appendices.

We hope that this report will encourage the reevaluation of existing programs, serve as a guideline in the creation of new programs, stimulate interaction among educational levels and disciplines, and promote improvement of the education of computer programmers.

1.2 Historical Perspective

This is the first set of curriculum recommendations to be produced by a scientific and technical computer society for an associate degree career program in the computer field. It is gratifying to the Committee that ACM has supported this work. It is consistent with ACM's early leadership in establishing curriculum reports at the baccalaureate and graduate levels in both computer science and information systems [6, 8]. These two earlier ACM Curriculum reports published in 1968 and 1973, respectively, credited community and junior colleges with providing education for the applications programmer.

Other curricula guidelines for two-year programs in the data processing area have been prepared at the state and national levels. Under the auspices of the U.S. Department of Health, Education, and Welfare, Office of Education, a business data processing curriculum covering a broad range of jobs was prepared and released in 1963 [28]. In 1964 a companion publication gave suggested techniques for courses of study in vocational and technical programs for electronic data processing in engineering, science, and business [29]. In 1970 a curriculum in scientific data processing technology was released [30], and in 1973 the earlier 1963 report was updated [27]. Work on information processing in scientific work was also done in the early 1960's by the National Science Teachers Association, and

a regular newsletter was published [26]. One scientific type curriculum was published by a computer manufacturing company [4]. The national association of two-year colleges, now known as the American Association for Community and Junior Colleges, sponsored a curriculum report published in 1970 [2]. Several state reports have addressed curriculum in the business data processing area, often covering the entire gamut of career opportunities available to two-year graduates [18, 24]. Recent increased emphasis on career education should encourage more awareness of the differing characteristics required for the different types of jobs in this field.

Computer-oriented courses and curricula in community and junior colleges began in the 1950's with a few experimental courses and programs in business and mathematics departments. Local business and industry often provided computer facilities, materials, and instructors for the early programs. Cooperation between schools and industries has existed from the beginning and has been a continuing characteristic of computer-oriented program development in community and junior colleges.

During the early 1960's, the National Defense Education Act provided matching funds for the rental or purchase of computers and other data processing equipment. The colleges used the federal funds provided to acquire small computer systems (e.g., the IBM 1401 and 1620) and unit-record equipment. Data processing curricula in these institutions reflected the type of equipment available.

The 1963 curriculum guidelines in business data processing featured five areas of study: (1) computer data processing basics and equipment, (2) assembler and compiler languages, (3) organization of business, (4) business applications, and (5) supporting sciences and electives. The report's effect on curriculum was extensive and its influence is still apparent in many community and junior college programs. Although these areas are still important in the training of an applications programmer, the content in each area has changed substantially over the years.

In the late 1960's and early 1970's many colleges gradually converted to third-generation computer systems which had auxiliary storage capabilities. The increased capability of the equipment and the increased awareness of its potential for college-wide use led to greater usage of the facility for administrative purposes. A large number of colleges acquired a computer services manager, or director, to serve the entire college community. Often the computer formerly used in a hands-on laboratory environment became physically inaccessible to students. Instead, a closed-shop, scheduled, batch-processing environment was provided. Many colleges acquired terminals to offer students immediate access by timesharing. Others installed remote job entry batch-processing stations serviced by the centralized computer center. A few colleges acquired systems large enough to service their own as well as local high schools' computing needs. Many colleges obtaining a computer for the first time have acquired a minicomputer, often shared with administration. Depart-

ments often acquire their own individual microcomputers.

The equipment changes and new technology necessitated adaptation in courses and curricula to incorporate different programming techniques, new data manipulation methods, and a variety of laboratory access methods. New courses were developed and the course content of existing courses modified. Changes came about slowly and were not extensive enough. As a result some curricula still emphasize outdated or inappropriate programming techniques and languages, and sometimes use obsolete equipment.

Since the early 1960's, successive surveys of degree programs in the computer field have been carried out on a national scale by John Hamblen [17]. In the 10-year period between 1966 and 1976, the number of programs at the associate level doubled to reach almost 400, and the number of graduates more than quadrupled to reach slightly more than 6000. Most of the associate level programs in the survey were named data processing and were in a department with the same name. Other programs were in the departments of business or business administration. Associate level programs named computer science or computer programming were few, but increasing in number. Some of the programs named computer science were primarily for transfer students and did not have the training of an applications programmer as a major objective.

The relationship between career and transfer programs is very close and is very complicated. For many years there were few four-year programs available to which students could transfer. By 1976 the number of baccalaureate level programs had grown to almost equal the number at the associate level [17]. As more four-year programs become available, there will be more need for emphasis on transfer programs.

Developments in computers in the 1980's are occurring at an even faster rate than changes occurred a decade ago. Advances in computer technology are having a profound effect on business and industry. Many programmers now have microcomputers available along with timesharing terminals and remote job entry stations. Changes in the way the programmer works will continue [5, 23, 34]. The increased maturity of the field has led to a more standardized vocabulary, better defined job tasks, and a generally accepted body of knowledge and skills required for competence [1, 3, 33]. The disciplined approach to the job, brought about by recent trends in structured programming, will perhaps help bring order to what has been a rather chaotic environment. Studies on productivity and programming practices [40], research on motivation [19], initiation of certification and self-assessment tests, and the development of software engineering [41] all promote a more professional role for the programmer.

2. Computer Programming Jobs: Prospects, Skills, and Environment

There are many kinds of computer programming jobs which require different levels of competency. This section describes the duties and requirements for the application computer programmer. Other more specialized titles are

often used, such as business programmer or scientific programmer. Duties and requirements for computer programmers are not necessarily standardized.

Some jobs in computer programming are more advanced and require more education and experience. These include the position of programmer/analyst, which combines the work of the programmer with the work of the systems analyst. Other programmers, called systems programmers, do not perform applications work for an end user but are instead responsible for the systems software, such as the operating system, the programming languages, and the utility packages. Opportunity will most likely exist for programmers to be promoted into these other related areas.

2.1 Duties of the Computer Programmer

A computer programmer assists in the development of automated systems and is responsible for the instructions to the computer. Job tasks include analysis of program specifications provided by an analyst, design of the program logic, coding of the program in the chosen programming language, preparing test data and verifying its accuracy, and providing documentation according to company standards.

A computer programmer who works in the applications area must also understand the problem being solved. For example, programmers working in the business field must have knowledge of the language of business; programmers doing work in an engineering field need to be knowledgeable in the language of engineering. The preparation of a productive applications programmer requires knowledge and awareness of the applications area.

Entry-level computer programmers are usually given an initial in-house training session. This allows the trainee to become familiar with the specific brand of equipment in use, the operating system and utilities, and the differences in the programming languages. It also gives the company a chance to establish its ground rules for computer usage and to ensure that new employees are aware of the data processing standards set by the company. After this session, the trainee is usually assigned as a junior member of a programming team.

Computer programmers at the entry level are often initially assigned to teams for maintenance programming [31]. These programmers analyze existing production programs to isolate problems and to make requested changes. They must use existing documentation, modify the program, expand the existing test data, and retest. In many cases, programmers are assigned to convert these existing programs to more up-to-date programming languages or to convert them to a new system. Programmers must know both the old and new programming languages and the old and new system.

2.2 Job Descriptions

Job descriptions for computer programming personnel are published periodically. Some sources are listed in Appendix D. Several levels of the job are usually specified, ranging from junior to senior. Task analyses of computer programming have led to differentiations among the ap-

plications specialties. Most differentiation has been made between the business programmer and the scientific programmer [1]. However, not all programmers fit these two categories. There are many other specialties, many of which do not have a formal job description. For example, a computer programmer might specialize in the development of computer-assisted-instruction software, in computer-aided design, or in library information retrieval systems.

2.3 Job Environment

The environment in which a computer programmer works varies with the applied area, the type of company, the geographic area, and the size of the facility. Applications programmers assigned to teams for work in business are usually a part of a computer center staff, reporting to a team leader, a senior programmer, or a systems analyst. In other cases, a programmer might be a part of a team in the user area. A variety of different team structures could be used [23].

Offices for work are usually similar to most other offices. Some programmers work at their desk with a terminal or a microcomputer. Others submit programs for keypunching and do compilation and testing through a programming coordinator. Still others may use the machines in a hands-on environment for testing. Sometimes programmers take a portable terminal home at night to do extra work. The programmer's work area and the tools needed for use have been referred to as a work bench [7].

Computer programmers must be able to work under pressure. Often they are required to work overtime hours to complete projects by deadline dates. The shortage of personnel has driven salaries to attractive levels [39]. However, turnover of programming personnel is high, presently averaging 25 percent per year [15, 25].

2.4 Career Paths and Advancement

There is ample opportunity for advancement in the computer field. Usually gradual development and maturity proceeds along one of three parallel lines of movement: increasing knowledge of business programming for the applications areas within the company; an increasing interest in the wider perspective of the entire application system; or increasing technical skills. The most typical promotion path for the computer programmer is to the more senior positions, or to programmer/analyst, then to systems analyst. The systems analyst often moves into computer center management. Persons with a more technical orientation tend to go toward systems software programming. Advanced work within the systems programming group can be in areas such as telecommunications or database management systems. Most of these more advanced positions require additional education and experience.

2.5 Employment Supply and Demand

There are not nearly enough personnel trained to work in computer programming. In a survey by National Personnel Associates, out of more than 100 occupations, com-

puter programmers ranked first in demand [11]. The U.S. Department of Labor confirms the serious shortage of personnel, predicting that jobs for computer programmers should increase 102 percent by 1990 [9, 36]. One annual mid-year survey for 1980 shows that, in all data processing, applications programmers rank first in demand, having increased by 33.3 percent in one year [13]. Second in rank were systems software programmers, with 26.7 percent increase. Programmer/analysts were ranked seventh, with 14.1 percent increase.

The personnel shortage is compounded by several factors. Colleges have not been able to graduate nearly enough computer programmers [16]. Although numbers of students increased sharply, associate degree programs graduated only an average of 15 to 18 per program each year over the last 10 years. Some of the graduates were in operations [21]. Teachers have not increased in numbers proportional to the number of students, but have actually decreased [21]. Sufficient new positions for faculty have not been created, due partially to the anticipation of the predicted decline in college age population. Even where positions exist, teachers are leaving them to go into higher paying jobs in industry.

A comparison of national computer personnel production in higher education to industry demand has been given by Hamblen [16]. Need is based on existing jobs as well as on estimates of the number of positions available due to replacement and growth. Production estimates came from data collected with funding from a National Science Foundation grant and from data obtained by the U.S. Office of Education. The 1981 report stated that the two-year colleges are producing slightly more personnel than needed (assumed to be mostly in the operator category), and that four-year colleges are producing far too few. He suggested that the weaker two-year programs should be strengthened and that approximately half of the two-year graduates should plan to transfer to four-year programs. It appears that many two-year colleges have already begun to do that.

In the period from 1971 to 1976, the number of associate degrees granted in computer programming technologies increased 29.6 percent while general data processing technologies decreased 15.1 percent [12]. Industry has accepted the philosophy that a college education should be required for programmers [24], and the two-year colleges have accepted that challenge.

In the midst of these predictions are others which forebode drastic change in the opposite direction. Some say the need for applications programmers will sharply decline [14, 38]. The present applications programmer situation could become somewhat alleviated as more software tools are developed. Custom programming in today's programming languages has become highly labor intensive, and work in the future may be done in other ways. As these changes occur, the demand for programmers will also change. So will their function. Some people feel that the programmer position as we now know it may eventually become obsolete.

Whether or not these predictions come true in the long

term, the Committee concludes that the need for applications programmer will exist for some time to come. However, changes in their roles and responsibilities are likely. In particular, there will be an increased interest in productivity, more emphasis on disciplined and structured methods, more in-house training programs, more use of software packages, more use of programming product tools, and more use of contract programmers.

Many associate level graduates continue their college education in related fields of business administration or mathematics and often on a part-time continuing education basis [22]. Some employers, especially those in government service, show a preference for employees with the general cultural level usually associated with the baccalaureate degree, regardless of major field. One recent follow-up study of graduates from a two-year program and a nearby four-year program revealed that both groups had high placement and satisfaction with their education [20]. As more baccalaureate programs in computing are created, it is likely that industry will begin to utilize graduates with majors in computing. More of the associate level graduates will attempt to enter these transfer programs as they progress in their career path in the computer profession.

2.6 Qualities for Success

Community and junior colleges offer open admission education to all those who wish to learn. Some specific curricula, however, such as those with limited space for students, limit enrollments in order to provide adequate education to those who are admitted. In these situations, selection of students is made on this basis of various factors such as academic preparedness for the program, maturity, aptitude tests, or economic need. Admission to computer-related programs has typically been open. Satisfactory completion of a first course serves as a prerequisite to further work. High drop-out rates have resulted due to a lack of awareness about the purpose of the program, insufficient readiness for college, inability in logical thinking, and shortage of patience and persistence in handling details.

It is advisable to counsel entering students so that they are more aware of characteristics needed to be a successful programmer. A brochure on careers is available at a reasonable cost for distribution to prospective students (see Appendix A1). Aptitude tests are available and may be used for counseling. Several authors characterize those who succeed in and enjoy the work of a programmer as usually having good reasoning and logical problem-solving ability and being observant and alert to detail. Programmers also usually enjoy continuing challenges, are able to work under pressure, and are persistent in pursuing a problem to its completion. Efforts to spread career education information should be encouraged so that students may become more aware of job characteristics and requirements in the computer field.

3. Objectives of the Curriculum

3.1 Purpose of the Curriculum

Community and junior colleges often serve as the starting point for the education of many persons who might not otherwise attend a four-year college. Some people use the community college to gain knowledge to change from one career type to another or as a place to update skills. Courses are available full-time or part-time, day or evening, on-campus or off-campus. In most cases, classes are small, facilities excellent and available, and faculty accessible on a personal basis. Students are of all types. By offering courses and curricula in the computer field for all these different types of people, community colleges serve as a viable source of qualified personnel for entry-level computer programming jobs.

The purpose of the curriculum described is to offer students the opportunity to enter the computer programming field as entry-level applications computer programmers. Graduates should be qualified to work on project teams with a systems analyst on applications which require medium-to-large-scale computer equipment. Graduates also should be prepared to work on applications which support the general, administrative, and organizational information processing functions of industry, business, commerce, and government service. Several options within the curriculum allow the selection of specialized areas for more in-depth study, leading to a wide range of potential career paths within the computer field. The curriculum is designed as a two-year associate degree career program for the preparation of graduates for jobs, but it also provides a sufficient foundation as a basis for continued learning in the field.

Computer programming work is found in many different applied areas. Almost every local community will have a need for a business-oriented program. Some communities may need other types as well, such as for scientific work, or in programming for service-oriented institutions, such as hospitals, schools, and churches. Persons already educated in an applied area may wish to do computer programming in that field. Students entering college for the first time should establish an applied area.

The Committee has attempted to work within the mission of the community and junior colleges: to serve community needs. Therefore, instead of prescribing a fixed set of courses, the Committee has specified subject matter topics. Instead of prescribing that all computer programmers be specialists in business work, the Committee has specified that any applied area for which there is a local or a personal need could be combined with core courses. The Committee hopes this approach will provide a flexible framework within which an institution can design and develop its own curriculum to meet the needs of the local community, the variety of aspirations of the students, and the rapidly changing technology with which we work.

3.2 Performance Objectives

The main objective of the curriculum is to produce a graduate who will be able to write application programs. To accomplish the main objective, the graduate must be able to

- (1) analyze problem specifications prepared by an analyst with respect to the data format, the method of processing, and the required user results;
- (2) plan detailed program logic; necessary program steps should be defined using a program logic plan or schemata such as flowcharts, decision tables, structured programming charts, or pseudocode;
- (3) use a problem-oriented procedural language to convert a detailed program logic plan into an efficient and well-structured applications program;
- (4) modify an existing program or program module to accomplish requested changes in requirements;
- (5) verify and thoroughly test the accuracy and completeness of computer programs by preparing sample data and by using debugging techniques and software aids;
- (6) prepare appropriate documentation of computer programs for utilization by the user, production personnel, analysts, or maintenance programmers;
- (7) use preprogrammed subroutines and utilities, the computer system library, the job control language or system commands, and associated reference manuals and documentation;
- (8) interpret and use the program specifications and documentation provided by system analysts;
- (9) illustrate the basic functions of computer hardware at the assembler level by writing assembler language coded programs.

The corollary objective to the main objective, weighing equally with it, is to produce a graduate with the necessary communication skills and related talents who can work in an organization responsibly, effectively, and productively. Toward that end, the graduate must be able to

- (1) communicate effectively with other programmers and with clerical personnel, analysts, managers, and users using acceptable oral and written English;
- (2) utilize mathematical and statistical skills to better facilitate problem description, analysis, solution, and digital computation;
- (3) utilize the standard methods of an organization for information flow through its accounting and information retrieval system;
- (4) adapt to the pace of constantly changing equipment, high-level programming languages, and programming products; use reference manuals and updates;
- (5) work effectively as a member of a team in an open, objective manner;
- (6) be knowledgeable about where, how, and why computers are used to increase awareness of ethical, social, and economic implications of computing.

It is a temptation to do only specific training for industry in specific programming languages or tools. Two-year colleges must not, at the associate degree level, yield to the temptation to train a specialized technician who may be obsolete with the next configuration of computers and unprepared to continue to learn. The associate degree program must provide a foundation of knowledge and skills sufficient to serve as a base for continued learning. The graduate should be able to do a competent job and have potential for growth. The ability to transfer knowledge about a computer system and programming language to a different mode or form will be necessary as technological changes continue over the next decade.

4. Content of the Curriculum

A topical approach was chosen for the definition of subject matter needed for the core requirements of the curriculum. These topics are purposely not structured into one fixed set of courses. Each two-year college offering this program will wish to combine the material into courses specifically designed for its own needs. The structure chosen by the college for implementation will depend on a variety of factors, including type of industry need, type of student clientele and their previous background, degree of expertise of the faculty, degree of cooperation between industry and the college, type of calendar and scheduling used by the college, degree of innovation desired by the faculty/staff implementers and feasible for the institution, and type of upper level programs nearby. Colleges with existing programs may wish to implement changes gradually or very rapidly, depending on resources available. Colleges starting new programs may wish to try totally new nontraditional course structures and scheduling.

The subject matter has been categorized below into required or core computer content, optional computer content, and related content.

4.1 Required Computer Content

The required computer-related content has been identified and classified into two major areas: *principles and techniques of programming*, and *programmer environment*. The first stresses in-depth programming knowledge, techniques, and skills supported by training in problem-solving and logical analysis which facilitate optimum organization of the data and program for computer solution. The second includes fundamental concepts of hardware and software, and organizational, social, psychological, and economic matters which make up the environment of the programmer within the analysis, design, implementation, and operation phases of an applications project.

These two areas should not be construed to be a course structure. Introductory and advanced material on a topic

Table I. Topics in Principles and Techniques of Programming

<ol style="list-style-type: none"> 1. Data Representation, Structure, Storage, and Processing <ol style="list-style-type: none"> a. Characters, fields, records, files b. Data representation, coding, and conversion c. Input—process—output cycle d. Summarizing, selecting, classifying, sequencing methods e. Sorting, searching, merging methods f. Records and block lengths: fixed and variable, blocked and unblocked g. Tables, arrays, stacks, queues, lists, trees h. Direct and inverted files; multi-record files i. Sequential, indexed-sequential, virtual, and direct access files j. Multivolume files and multifile volumes k. File activity, volatility, volume l. File updating m. File maintenance and security n. Reorganizing of files: conversion, modification, restructuring o. Calculation concerns: truncation, rounding, accuracy, modes, precision 2. Programming Languages and Logic <ol style="list-style-type: none"> a. Problem analysis b. Program logic plans: flowcharts, decision tables, structure and hierarchy charts, Nassi-Shneiderman charts, algorithm statements, data-flow diagrams, pseudocode c. Data input and output: types and format d. Use of structured programming constructs and extensions: linear sequence, IF-THEN-ELSE, DO-WHILE, DO-UNTIL, and CASE e. Routines: housekeeping, initialization, end-of-file, end-of-job f. Multiple-level control breaks g. Use of parameter data, sentinel fields, program indicators h. Use of validation and control features: check digits, hash totals, cross totals, batch totals 	<ol style="list-style-type: none"> i. Use of compound logical functions based on AND, OR, NOT j. Test data creation k. Debugging methods; traces, snapshots, dumps l. Program correctness: logical equivalence, accuracy m. Program efficiency, timing, and style n. Program documentation: internal and external o. Basic and advanced syntax and semantics of, and in-depth practical projects in, a problem-oriented procedural language: data formats, delimiters, statements, precedence rules, sequence of control, subroutines and linkage p. Basic syntax and semantics of an assembler level language: instruction format, operand structure, register and storage structure, storage allocation, mnemonic operation codes, labels, symbolic addresses, declaratives, macro instructions, subroutine linkages, address and operation modification, interrupts q. Basic syntax and semantics of one additional specialized language r. Use of language reference manuals s. Anticipation of differences due to implementation <ol style="list-style-type: none"> 3. Interface with Hardware and Software <ol style="list-style-type: none"> a. Basic computer architecture b. Overview of operating systems c. Job stream control language and usage d. Use of utility programs, text editors, macros, functions e. Use programming products which facilitate programming f. Program modules: storage and usage g. Input—output routines: buffering, paging, overlap, timing h. Compilation—linkage—execution considerations i. Resources accountability j. Use of assembler routines by non assembler languages and compatibility of data formats k. Operational implications of program style upon operator l. Operational implications of program style upon user
---	---

is listed only once, although it may be introduced in one course, used in programming exercises, and then covered in more depth in an advanced course. This topical approach is not an ordered teaching outline, but is a list of topics recommended for coverage in a core curriculum.

4.1.1 Principles and Techniques of Programming

The content recommended for a core curriculum, given in Table I, should provide the student with programming techniques, knowledge, and skills for use of the computer as a tool in the performance of practical computer applications. The prospective programmer should be able to analyze programs to determine quality, correctness, appropriateness, and completeness of documentation. The major computer language should be selected on the basis of need in local industry served by the college, and it should be taught in depth. This language should be supplemented by more limited experience with at least one other computer language to allow students to make comparisons between the languages and to give graduates more versatility. The core also includes a foundation in assembler concepts in order to provide a better understanding of hardware/software interaction and compiler-generated code.

Concepts and skills in programming in a procedural

language, some foundation in assembler, and exposure to one other language must be combined with the ability to interface the hardware and software by means of an operating system. The ability to access the system library by the job control language or system commands is necessary for use of utility programs and for control of hardware components.

4.1.2 Programmer Environment

The content recommended for knowledge of a programmer's environment is given in Table II. It includes those topics which prepare the programmer to function in the environment of hardware, software, and people. It includes fundamental concepts of computers and peripheral devices, computer languages, and the organizational, procedural, social, psychological, and economic concepts necessary for the day-to-day performance of the programmer. Emphasis should be placed on the role and responsibility of the programmer in the successful analysis, design, implementation, and maintenance of a computer-based information system. This content provides the programmer with an overview of the complete applications project, from source data to distribution of results, and of the role of all personnel involved in the project. It should also include aspects of societal impact, ethics, professionalism, and information

Table II. Programmer Environment Topics

1. Computer Equipment and Function	b. Mode and format
a. Internal functions of central processor unit	c. Data validation and control: mode checking, limit checking, consistency between elements, check digits
b. Storage devices: primary and secondary	d. File validation and control, internal labels, external labels, file backup, hash totals, batch totals
c. Peripheral devices: data collection, reporting	e. File structures
d. Computer configurations	f. File security and techniques
e. Comparisons of hardware components	g. Concern for privacy
2. Programming Languages in Organizations	8. Report Requirements and Forms Control
a. Compilers, interpreters, assemblers	a. Survey of business forms
b. Generators, file management languages	b. Input forms: types, evaluation, development, control
c. Languages relative to equipment	c. Output forms layout
d. Languages appropriate to the application	d. Output forms: distribution and control
e. Mode of processing: demand versus batch	e. Programming considerations dependent upon input and output forms
3. Computers in Organizations	9. Quality Programming with Structured Approach
a. Evaluation of computers and their usage	a. Fundamentals of structured analysis and design of programs
b. Computer center in the overall organization structure	b. Top-Down programming, debugging, testing, implementation
c. Computer centers: structure, staff, job descriptions	c. Analysis of a program's structure for improvement: efficiency, style
d. Relationship of programmer to the computer center organization	d. Review of control structures
e. Statistical data and trends	e. Determination of logical equivalence of two differently written programs
f. Costs: hardware, software, other	f. Comparison of efficiency of two differently written programs: execution times, storage requirements, programming time, maintenance time
4. Overview of an Existing Applications System	g. Conversion of unstructured design to structured design
a. Necessary reports	h. Effect of program languages in structured programming
b. Source data	10. Programming Projects Concepts
c. Systems flowcharts	a. Evaluation of programs for good structure
d. Process flowcharts	b. Team approach: design and code walkthrough
e. Necessary programs	c. Program project coordinator (librarian): function and responsibility
f. Equipment/configuration needed	d. Standards: use of, examples, advantages
g. Proper documentation for usage of the system	e. Overview of project management methods: personnel assignment activity sheets, Gantt charts, PERT charts
5. Overview of the Systems Cycle and the Programmer's Role in the Project	11. The Programming Profession
a. Analysis phase	a. Career paths
b. Development phase	b. Professional ethics
c. Implementation phase	c. Licensing and certification considerations
d. Production phase	d. Professional societies
e. Evaluation phase	e. Need for continuing education
f. Program maintenance	
6. Documentation	
a. Using systems documentation	
b. Using systems flowcharts	
c. Using flow process charts	
d. Evaluation of program documentation	
7. Data Elements and Files	
a. Collection, forms, and coding	

concerning potential career options. The student must be made aware of the role and responsibility of the programmer in the organization and in society as a whole.

The content should be taught from the point of view of the programmer. Practical projects should provide the student with an opportunity to analyze prewritten programs, to examine well-written documentation of programs, and to work with well-designed input and output forms. Students should be given poorly designed unstructured programs to convert to structured form. The student should be prepared to work with the tools of, and within the constraints of, the analyst's profession. Use of the traditional documentation forms should therefore be extensive.

Fundamentals of hardware should stress computer equipment and its function, individually and in a configuration. Knowledge of various peripheral equipment—

input, output, and storage—should encompass a broad range of systems, from programmable calculators to large-scale configurations. Students should understand the equipment appropriate for a variety of typical applications. Software concepts should expose the student to the variety of software levels and languages and to the purposes of each. Concepts of metalanguages should offer some insight into the structure of languages. This should lead to an awareness of specialized concepts such as data definition languages, report generators, and database management and inquiry systems.

4.2 Optional Computer Content

Some colleges may wish to offer content in addition to the required computer content. Special interests of the student often lead to the selection of additional topics, which can lead to special types of employment or to transfer to

four-year programs. Some students may wish to elect a course in numerical methods, while others may wish to learn more about operating systems, data communications, or systems analysis. Optional additional topics recommended for coverage are given in Table III.

Each two-year program should regularly perform an assessment of the needs of local industry. Based on the documentation of such a study, the institution may wish to include additional topics in the core requirements. Colleges serving communities with relatively advanced computer work in high school may choose to design their program to include more advanced courses.

These topics are not intended to reflect courses even though some may warrant a full course. Several of these topics may be combined to form one course. Colleges should have the flexibility to offer topics or programs for the satisfaction of specialized community needs by providing short courses, modules, or community service seminars.

4.3 Related Content

In addition to being skilled technically as a computer programmer, a graduate must be able to use communications and other skills to work in an organization responsibly, effectively, and productively. Subject matter content must therefore include materials to assure proper communication, fundamentals of mathematics and statistics, and knowledge of organizational matters and information flow within a company

English Communications

The importance of speaking, listening, and writing proficiently is increasingly recognized in both academic and professional spheres. Employers have declared that these abilities are as important as specific technical skills [35]. Graduates must work closely with other computer and data processing personnel and must be able to communicate effectively. Written and oral instructions must be both received and given; technical reports and program documentation must be developed, written, and presented. Courses such as English grammar, English writing, technical writing, and speech should be a part of the curriculum.

It is not enough, however, to specify that courses be taken. Computer-related courses should build upon, use, and thereby strengthen these skills. Opportunities to give presentations in class should be found. Verbal instructions must be given to strengthen listening skills. Writing and interpreting narrative instructions should be part of student practice. Oral presentation must be required. Program documentation should be examined and evaluated for correct English as well as for correct form. Students should be expected to use these skills in all classes just as they will be expected to do on the job.

Business Knowledge

The computer programmer working in the computer applications area should be acquainted with certain fundamentals of the business and economic systems, at least as practiced in the local area being served by the college.

Knowledge of the functions of a commercial organization and a comfortable use of its vocabulary are important. Skills needed in this area are often met through introductory business or economics courses. Further knowledge of information flow in an organization can be gained from a course in accounting.

Class examples must include applications from the functions of business. Certain typical applications may be chosen for use in laboratory programming exercises. The presentation of each laboratory project should be thorough, so the student will understand both the application and the processing needed.

The programmer is most involved in the implementation phase of the systems cycle and must understand the application being programmed. Information analysis must be performed by the analyst/designer to determine the best way to handle the application. Although the programmer needs an overall perspective of the entire system, more intensive emphasis must be given to the programmer's role in the implementation of the project.

Mathematical Knowledge

The computer programmer has many occasions to use communication skills. Mathematics also serves as a communication language, for one's self and others. Logical thinking and reasoning skills can be improved and organization skills strengthened by using the structure of mathematics. It is therefore recommended that a minimal level of mathematical skills and vocabulary be attained by all those in the program. Students should complete at least one course in finite mathematics, including set theory and logic, probability, permutations and combinations, series and sequences, functions, linear equations, and matrix notation and manipulation. A course in elementary non-calculus-based statistics should provide the necessary logical and vocabulary foundations of statistics. It should include the use of computerized statistical software packages. Students entering the program with a good mathematics background may choose more advanced courses such as advanced algebra, differential and integral calculus, and linear algebra.

Faculty in computer courses have the responsibility of seeing that students apply mathematical knowledge to their computer work. Mathematical vocabulary and usage should be brought into computer programming classes where they apply. The use of Venn diagrams, summation notation, logical tautologies, sums of sequences, and use of terms such as "frequency distribution," etc., should be part of the curriculum. Courses in mathematics, separate from other majors and especially designed to include applications from the computer field, would be helpful for computer programming majors.

Perhaps more than any other area, the mathematics background and expertise of the graduate determines the number of future options available and the variety of directions open as career paths. For example, students expecting to transfer into programs in computer science, engineering, the sciences, or mathematics must necessarily

Table III. Optional Additional Topics

-
- | | |
|--|--|
| <ol style="list-style-type: none"> 1. Additional Specialized Programming Skills <ol style="list-style-type: none"> a. COBOL, BASIC, FORTRAN, PL/I, Pascal, APL, or other languages not already included in the chosen option b. Assembler languages (advanced) c. Microprocessor specialized statements d. Report program generators (e.g. RPG) e. DBMS query languages f. Other specialized languages 2. Operating Systems and Job Control <ol style="list-style-type: none"> a. Monitors and executives b. Multiprogramming, multiprocessing, and multiaccessing considerations c. Virtual storage and paging d. New advances in operating systems concept e. New advances in data storage 3. Data Communications and Teleprocessing <ol style="list-style-type: none"> a. Overview of data transmission theories b. Coding systems and compatibility (ASCII, EBCDIC, etc.) c. Overview of error detection and correction d. Transmission methods e. Transmission equipment (data sets, acoustic couplers, etc.) f. Terminals g. Computer access and processing 4. Database and Its Management <ol style="list-style-type: none"> a. Concepts b. File structures c. Data dictionaries d. Database management systems concepts e. Sources of software f. Costs and benefits of software g. Administration h. Standards for organization use | <ol style="list-style-type: none"> 5. Computer Operations and Operations Management <ol style="list-style-type: none"> a. Operations of peripheral devices (tapes, disks, bursters, etc.) b. Console operation c. Use of job control language d. Job scheduling e. Control checks f. Dispatching and disbursing g. Routine production operations h. Record keeping and statistical data i. Billing 6. Computer Hardware and Services <ol style="list-style-type: none"> a. Microprocessors and microcomputers b. Programmable calculators c. Standalone systems d. Terminals and leased computer services e. Remote job entry stations f. Timesharing systems g. Service bureaus 7. Applications Software Packages <ol style="list-style-type: none"> a. Sources of packages b. Costs of implementation c. Evaluation of product d. Legal implications e. Contracts 8. Systems Analysis for Information Systems <ol style="list-style-type: none"> a. Systems planning b. Feasibility studies c. Development of a system d. Evaluation of a system 9. Systems Design for Information Systems <ol style="list-style-type: none"> a. Specification of logical systems b. Suggested implementation into a physical system c. Evaluation of design based on user needs |
|--|--|
-

be equipped with mathematical skills generally attained in the traditional course sequence leading up to and through calculus. Students choosing an emphasis in a scientific applied area would also choose the traditional calculus sequence of courses. Those expecting to transfer into business-related fields, however, will usually find that the courses recommended in this report are sufficient for continued study in quantitative methods. Students who plan to transfer should select mathematics courses suitable for their transfer program and institution.

Applied Area of Specialty

Each student anticipating a career in computer programming should choose one applied area of specialty, rather than a series of elementary courses in several fields. Twelve semester credits or more should be taken in an applied area to give the graduate an area of expertise upon which to base computer programming skills. The selection of an applied area may be determined by the institution or may be chosen by the student. In most cases, the selection of the applied area should depend on criteria such as the ultimate goal of the student and job placement priorities. Mature returning students and students who already have a baccalaureate in another field often have an applied area established and need only to complete computer-related courses to obtain employment in the field.

The predominant applications area for computer programmers lies in the supportive functions of business and administration. For that reason, students in the curriculum choosing this area should learn as much as they can about specialized applications—payroll, inventory, financial management, banking, and similar courses. Courses chosen from advanced accounting, personnel management, banking, and marketing could be elected for an applied area specialty.

Another applied area might be mathematics and statistics to support the statistical data gathering, sampling, and the reporting functions of governmental agencies, as well as to support the programming needs of certain research and development agencies. Courses in this case include advanced algebra, differential and integral calculus, linear algebra, and calculus-based statistics.

Many other applied areas could be chosen which lead to productive and interesting careers. The increasing use of computers in such diverse fields as biology, law, chemistry, psychology, the allied health fields, the humanities, and law enforcement offers a large spectrum of opportunities for a challenging profession.

General Education

Most states and institutions require 24 semester credits or the equivalent for general education in each associate level degree program. The requirements are usually chosen

by the student from categorical lists of accepted courses. Categories often included are English and the humanities, mathematics and science, and social and behavioral sciences. General education courses in psychology and speech are recommended. The others chosen should contribute to the general cultural level of the student or reflect special areas of interest. Generally, courses in business and computer subjects are not considered general education subjects.

4.4 Design Considerations

Each institution must design its curriculum and the courses within it to fit its own mission and philosophy, the characteristics of its student clientele, the characteristics of its commercial environment, and the other educational opportunities available to students in this field in the community.

Options within the curriculum are possible by means of choices: in the applied area; within the general education category; in the major procedural programming language; and in mathematics. Some institutions may, after study of the community being served, wish to require that this applied area be in a business field. Others may wish to emphasize the scientific programmer option or the health services programmer preparation. Some institutions may instead prefer to advise students concerning job availability in the area and leave the choice of applied area to the student.

The associate degree curriculum should be institutionally designed within the philosophy of the guidelines presented. The program should be at least 60 semester credits, with at least 24 semester credits in computer-related subjects. It should consist of

- (1) the required computer topics,
- (2) local-option computer topics and electives,
- (3) communications, mathematics, and business requirements,
- (4) student-option electives for an emphasis in an applied field,
- (5) institutional and state general education requirements.

The Committee has encouraged the publication of papers authored by interested community and junior college educators to provide several sample curricula which meet the requirements of this report. Several of these are listed in Appendix B. By examining several different ways to accomplish the educational aims of this report, curriculum planners may be better able to analyze their own situation and to establish a program especially suited to their needs.

4.5 Major Programming Language

Selection of the major programming language will depend on the applied area and on local need. Recent national surveys indicate that the predominant problem-oriented, procedural language for business work is COBOL. Some colleges may find, however, that another

language, such as BASIC, FORTRAN, Pascal, or PL/I, is preferred for work in their local community. The major programming language chosen should be justified by documented results of recent surveys of the local area being served.

With the tremendous growth of small computers, an increased need exists for programming with report program generators such as RPG. Likewise, an increased growth in the number of large job shops using some form of database management system (DBMS) has led to an increased use of query languages by users as well as by applications programmers. The question about the appropriateness of a report program generator or a query language to serve as the major language in these recommendations has been a much debated and controversial one. The Committee examined the issue and concluded that although capability in the use of only nonprocedural programming tools might prepare the graduate for specialized technical employment, the graduate's potential career advancement might be limited. It is generally agreed that knowledge of only a nonprocedural language would be less transportable to procedural types of computer programming and that graduates would therefore be less capable of transferring the knowledge of that single programming medium to other languages. The Committee recommends that these specialized application-oriented package systems, such as the report generators and database query languages, be the second language in the program, a local-option requirement, or, in cases where the need is sufficient, the basis for a totally separate program.

The initial exposure to programming need not be in the major programming language. There is much support for the benefit gained when a simple introductory language is used in a timesharing environment or on microcomputers. Many secondary schools now teach BASIC in this manner. This approach has been found to be highly motivational and is relatively successful in exposing students to general programming concepts. With the advent of microcomputers, immediate response demand processing is now feasible and is highly recommended as an initial introduction to programming.

The quality of work of a computer programmer is greatly enhanced by a basic understanding of computer functions at the level of execution of instructions. For this reason, it is recommended that instruction in an assembler language be part of the core. Assembler language could be taught as a separate hands-on programming course in a laboratory setting on whatever machine is available, as a course in elementary computer architecture with assembler programming, or as part of the instruction in the major programming language through the use of compiler-generated code.

4.6 Access to Equipment

Strong arguments have been given for hands-on experience [10]. With availability of computer system access ranging from desk-top computer to network, it is now possible and desirable for students to have experience with

different types of computer access. These should include (1) programming for a batch-oriented computer service facility, (2) programming for a timesharing system where each program is entered into the system once, accessed for a debugging session, and then replaced in updated form into the system, (3) programming in an interactive mode, where line-by-line diagnostics are provided and execution-time input of data is in interactive mode, (4) hands-on access to online devices such as card readers and printers to see the relationship of program to job stream set up and to the use and control of forms, and (5) hands-on access to a central processing unit, which provides increased understanding of hardware/software interaction in program loading and execution. Experience with these different types of access need not be threaded throughout the entire curriculum; rather, each course may be designed to include one or more types of access. When not all of these types of access are available on site, special arrangements could be made with industry to provide limited experiences in other access methods.

Instruction in the major programming language requires extensive laboratory practice. The major language programming courses should include a scheduled laboratory period of at least two hours each week. This should be scheduled in one two- or three-hour block of time and should preferably be taught by the class instructor. Scheduled laboratory sessions should be provided with rapid, if not immediate, turnaround during the laboratory class. This is necessary to ensure that during the laboratory class each student be able to obtain experience with the compilation and execution cycle with classmates and instructor present to discuss and explain situations as they arise. This could be provided by hands-on access to a remote job entry station, with students placing their program decks into the computer input devices, initiating the run, and receiving their results within a very short time on the output printer. It could be attained on terminals by use of a text editor, by use of a computer system with an interactive procedural language, or by direct access to the console of a computer. Additional access should be available during certain other hours of the day so that students may return to work on their programming projects. Should that be impossible, computer service should be scheduled at least three times each day, or continuous service provided with no more than two hours turnaround.

The controlled use of turnaround during the scheduled laboratory allows interaction between the instructor and the student, between the student and the computer, and among students. Such rapid turnaround has in some cases been abused when students eager to get a problem working tend to "change one thing and try it again," without really examining the logic of the program. When laboratories are used properly, groups of 15 to 20 students can have effective interaction with the computer, their instructor, and their classmates.

The interface between the program and the computer system requires that the programmer be able to work with and control the operating system. It is recommended that

both instruction in and use of an operating system be required. The student should have experience using program modules and utility program packages and should be able to catalog and recall programs from an on-line library. The student should be made aware of the job accounting systems used for recording time usage, storage usage, and supplies usage for the computer facility.

4.7 Structured Programming

Concepts of structured and disciplined programming should be threaded throughout all programming instruction. Students should be made aware of the basic control structures as they are first encountered and used. They should be taught to read and recognize well-structured programs. They should have experience with the use of structured walkthroughs and should be able to read and evaluate their own programs or programs written by others. Benefit can be gained from requiring written and oral explanation for the rationale behind the decisions made in coding a particular routine. Students should have practice in working alone and also in being a part of a programming team. Class-monitored and instructor-monitored project management of the teams give the student an awareness of scheduling and timing considerations which are needed on the job.

This curriculum content should give the student a good foundation with which to enter the job market. Materials from industry, such as standards manuals, project management techniques, and program logic plan guidelines, would contribute toward making the student aware of the job requirement in that local area. Guest speakers from industry and visits to local industry settings are recommended.

5. Resources for Implementation

Attention must be given to the many problems of implementing and maintaining a computer-related curriculum. Colleges should not attempt to implement a program unless they are fully committed to the necessary investment in time and money to acquire and maintain a competent faculty and up-to-date computer resources. The acquisition of qualified staff and access to appropriate computer equipment and services is critical. The organization and placement of the academic department in relation to other college departments and to the computer facility affects the ease in maintaining the viability of the curriculum. The rapidly changing nature of the topical content, programming languages, and equipment leads to a need for easy adaptability for change. Ways must be found to keep both faculty resources and computer resources up to date.

5.1 Organizational Matters

The need for talented graduates in computer-related occupations continues to increase as computers are more widely used. Courses may be initiated in a number of existing departments, with possible diffusion of effort and undesirable competition among departments. It is recom-

mended that, where possible, one academic department or group be chosen as the host for computer course development to house the faculty and the curricular planning efforts. Where that is not possible, strong emphasis should be given to liaison between departments to allow shared courses, faculty, and equipment.

The relationship between the academic department and its computing facility must be a close one. Although no one organizational structure is being recommended, the structure must supply equipment requirements and access specified. Some colleges could choose to have a chair-director, responsible for both the academic program and the computer services center. Another common structure has an administrative computer services manager at the same organizational level as the academic department chair. Separate facilities, one for the academic department and one for the computer services center, might be considered in some cases. Each community college should analyze its college-wide needs and weigh the several alternatives. No matter which alternative is employed, close communication and mutual respect for interests and needs must be maintained to ensure the cooperation necessary for the viability of the curriculum.

5.2 Faculty and Staff

A substantial portion of the faculty should be maintained in full-time positions to provide coverage of the topics required, especially since part-time day faculty are extremely difficult to find. Qualifications for faculty in the community colleges are generally based on the master's degree as a minimal academic requirement. Graduates from data processing, computer science, or information systems programs are preferred as instructors. Experience in industry as a programmer or analyst is recommended. Graduates from related areas such as business, accounting, economics, engineering, or mathematics are often qualified after several years of work experience in the computer field. People with a degree and recent experience in business data processing are also qualified.

Graduate programs in data processing, computer science, and information systems are producing a very limited number of potential faculty. Community colleges have not been able to compete with what industry and the universities have to offer these graduates: an opportunity to work with up-to-date equipment on interesting new applications for a higher salary. Many faculty now teaching in the programs have degrees in noncomputer-related fields, with training and experience from industry. Many others have come from the ranks of faculty in other related departments, as enrollments in their field decreased, or as their interests in the computer field were aroused. Hence community colleges must currently work for professional development of existing faculty, while also attempting to recruit new teachers from industry and the graduate schools.

Laboratory attendants are often needed, either full time or part time, to work in open laboratory settings to allow students continued access to equipment. With the help of these personnel, valuable equipment can be attended and

maintained, supplies can be readily available, and specialized assistance to students can be given. Desirable qualifications for laboratory attendants are an associate degree in the field, an interest in helping students, familiarity with equipment and programming languages, and responsibility toward established duties.

At least one additional person must be responsible for the systems programming effort necessary to maintain the computer system, its operating system, library, programming languages, and software. As updates to programming language packages are received from the manufacturer, they must be inserted into the existing system. This person may be faculty or staff, depending on the organizational structure of the academic department and the computer facility used in the institution. In cases where this person is a teaching faculty member, released time must be provided.

5.3 Equipment and Access

It is vital that a computer resources laboratory be available for class work to be done in this curriculum. There are several ways to supply this laboratory resource, including the use of one or more on-site computers, a computer shared with administration, leased services, or data communications access to a remote computer. Computer usage may also be obtained from local industry or from a neighboring college. Computer resources are available through manufacturers, leasing companies, and computer service centers. A great deal of effort may be needed to determine the most effective and efficient way to provide the computer resources needed for the curriculum and college.

The computer resources laboratory should provide computer service and access and ample data entry key devices for program and data preparation. Space must also be provided for data and program preparation and for checkout. The preparation area should be located near the data entry area and should have work tables. Space must also be provided for computer reference manuals. This area could also serve as a central location for current computer-related periodicals and newspapers.

It is impossible to specify one computer configuration that would be suitable for all two-year programs. A mini-computer may be suitable for one and a medium-scale computer better for another program. A microcomputer laboratory could be used in a network configuration. Department needs are often tied through the equipment administration to those of other departments. When other departments have an extensive use of computer-assisted instruction, sharing resources can present a critical problem. It is important that the entire college community be aware that, for the curriculum described in this report, the facility is a classroom laboratory. Stated needs and access must not be overrun by well-meant efforts to centralize and economize.

It is difficult to specify needs solely on the basis of total enrollment in the department. A course requiring hands-on access and immediate turnaround may meet within the computer room itself, may use a microcomputer labor-

atory, or may use a remote job entry station. A course requiring batch-processing will need data entry machines. The quantity depends on the number of students in the course and the programming language being taught. The number of timesharing terminals or microcomputers needed will depend on the number of students in courses requiring their use. The number of hours of computer time needed for runs outside those scheduled for class laboratories will depend on the number of projects assigned in each course and the average number of runs required for each student. Data should be collected for an analysis of computer facility needs.

5.4 Instructional Materials

Many good textbooks are now available. A survey of current books is published annually by the *Computing Newsletter*. A library list of current books has been published jointly by ACM and IEEE (see Appendix A2). Exhibits of current books are held each year at annual conferences of the various computer associations, as well as at the National Computer Conference, sponsored by AFIPS, the Computer Science Conference, sponsored by ACM, and the National Educational Computing Conference. Many sources of material and ideas are available through publications of the professional computer societies and proceedings of their conferences. Content articles in publications of the Data Processing Management Association, the Association for Educational Data Systems, and newsletters of the ACM Special Interest Groups for Business Data Processing, Computer Science Education, and Computer Personnel Research are especially helpful (see Appendix A3).

Laboratory materials are less readily available and are often unsuited to particular equipment and/or courses. Development of laboratory exercises often must be done by the faculty. Development involves a major time commitment on the part of the faculty; this commitment should be recognized by the institution and assigned as a part of the faculty load.

The use of technology as an aid to instruction has been much debated and discussed. Computer-assisted instruction and computer-managed instruction have not been generally available for the teaching of computer-related subjects at the two-year college level. More programmed material is becoming available for use on microcomputers. Some equipment manufacturers provide tutorial packages for programming language instruction with lease or purchase of equipment; these packages should be used where they suit the needs of the curriculum and the equipment being used. Information about progress in this area is available through the ACM Special Group for Computer Uses in Education (SIGCUE) (see Appendix A3).

Many videotape and audio cassette tapes have been produced. Videotape and cassette player/recorders may be used in the classroom and laboratory. Students may replay lessons for individual learning and may watch themselves giving oral presentations. Faculty may use these media to bring in recorded material for in-class activities or may

record assignments or solutions to assignments. Videodisk materials will be available soon.

Commercially available film and film strips are becoming more prevalent. Many good films are available, without charge, from equipment manufacturers and university suppliers. A bibliography of films is available (see Appendix A2).

5.5 Adaptation for Continued Relevance

The computing field is one of rapid change. Many of the changes occurring are transparent to the general computer user, but they are a part of the computer professional's livelihood. The half-life of a computer professional (the time it takes the knowledge to decay to a state in which it can provide only half of the facts or techniques to do the job) was estimated in the 1950's to be about ten years, and in the 1960's to be about five years. In 1970 it was estimated to be three years. It is perhaps now even less. Planning is therefore vital to ensure that continual adaptation is as easy to accomplish as possible.

Faculty

The most critical factor contributing to the success of this program is a well qualified and dedicated faculty. It is vital that faculty members participate in workshops with other computer professionals, attend educational conferences, and continue to get experience in industry, either through consulting work, by supervision of field projects of students, through summer employment in industry, or by sabbatical leaves. Faculty should read computer journals and literature in order to keep up with new hardware and software developments. It is essential for faculty to understand the differences in implementations of programming languages on more than one configuration. Their own experience on different equipment allows them to instruct students in such a way that the students are aware of the differences that exist. Faculty should be encouraged to participate in self-assessment procedures and take certification examinations.

Equipment

Community and junior colleges cannot always have the exact version of the hardware configuration that the student is likely to encounter on the job. It is possible, however, to offer the student an acquaintance with each of the types of access to a computer likely to be encountered and to give an exposure to the prevalent configurations of equipment found in the area. College equipment need not be the only computer access nor the only configuration shown to the student.

Field trips to other locations and usage of other computers make it possible for students to be aware of changes in computers and configurations. Colleges with a batch-processing computer and no timesharing capability could consider using the services of a local university, either by telephone or by taking students to the university for their laboratory sessions. Colleges with their own timesharing, but no access to a large batch machine, could arrange for a local company to provide daily service for certain courses.

Microcomputers could be used to allow interactive programming experience. Internships in local industry could provide practical experience with operating computers in a hands-on access manner when not available on-site.

Curriculum

Each college must choose a course structure for its topical content. Progress in educational research may well contribute toward ways to improve course structuring and methods of presentation and instruction. Advisory committees can provide valuable information about community customs and needs. As students complete the program and become employed, they can give invaluable advice concerning what changes in the program may be needed. Continual evaluation of the suitability of the graduates for employment, and examination of their professional success and career development within the industry, must be done.

6. Articulation

6.1 Industry

An important measure of the success of this program is that obtained from job placement records of the graduates. Their ability to get, and to retain, pertinent employment in commerce, industry, and government is one criteria for evaluating the program.

Throughout the history of two-year programs, placement of graduates in computing occupations has been good. In many communities, the community or junior college has been the only source of new computer personnel. Many companies, however, have consistently hired non-computer-educated baccalaureate and master level applicants and provided them with in-house training. In some cases, experience on the specific computer configuration to be used has often been the most important measure for evaluation of applicants. Graduates have in many cases been offered positions a level below their preparation, often at the operator level. Community and junior colleges must recognize that although a strong need for computer programmers exists, and is increasing, the quality of the programs must be maintained and matched to the needs of industry.

Good relations with local industry must be developed. Industry and the institution can both benefit. These relationships can lead to the continuing relevance of the curriculum and to an increasing opportunity for industry—college interaction such as cooperative work—study, internships, or part-time employment for students. These are extremely valuable, but require administrative coordination. Assignments of this nature should be a part of the faculty teaching load. On-the-job experiences are valuable in giving the student early exposure to the job environment and offer an opportunity to make more informed career decisions. Often motivation is increased and a better job placement can be secured upon graduation.

Surveys of the needs of industry in the community being served should be made and viewed in light of the offerings of other neighboring educational programs in an effort to

match the educational program to the expected need two years hence. One study done in an effort to determine whether educational offerings were matched to corresponding industry requirements revealed that they were not [32]. Increased interactions with industry are vital both to industry and to the educational institution. It is important for educators to have increased awareness of industry needs and practices and for industry to be aware of the educational offerings in their area. This mutual awareness can lead to more consistent and uniform hiring practices in specific job categories, which should be beneficial to industry and education, employer and employee.

Formal ties between the institution and local industry representatives should be established through curriculum advisory committees. Advisors should meet on a regular basis with college faculty and staff. This regular contact between faculty and advisory committee can be used to ensure continuing appropriateness for local needs. Reports giving advice on how to initiate and use an advisory board are available (see Appendix B).

Department members should maintain direct communication with employers. Too often formal contact with prospective employers is maintained by other offices within the college. Curriculum managers and teachers benefit tangibly by involving themselves in sessions with company representatives. The exchange of information regarding current and future systems, methods, and languages contributes toward interaction and increases the opportunity for placement of graduates. Opportunity for this interaction in a relaxed atmosphere is often possible through local chapters of professional societies such as ACM, ACPA, AEDS, ASM, or DPMA.

Timely feedback from graduates of the program presently employed in industry is also important in keeping the curriculum in line with local needs. Surveys concerned with the preparedness of recent graduates for their employment situation should be done every two years. Follow-up studies on all alumni are useful for tracing the career paths of graduates. Workshops can be offered for alumni to update skills and provide feedback on the program.

Contact should be made with local, state, and federal governments so that faculty become acquainted with the requirements for government programmers. Some of these jobs require entrance tests, experience, and four-year degrees. The present United States government position of computer programmer, for example, ranges from GS-5 through GS-15 and is not open to community college graduates without experience. Faculty and students are advised to obtain the most recent job requirements and descriptions as they are subject to change [37].

6.2 Other Educational Institutions

Secondary Schools.

The number of secondary schools which offer computing is increasing rapidly. A suggested curriculum for grades 9–12 has been published (see Appendix A1). It is important that high school students in computer-oriented pro-

grams or courses be made aware of transfer possibilities to two-year colleges. College faculty should maintain close contact with the respective departments at the high school level in order to improve articulation. Secondary school students who complete courses equivalent to the material recommended in this report should be able to obtain advanced placement in the program, early admissions, or have a means of obtaining credit by examination. Courses and programs should be reviewed periodically, in order to keep the high school student informed about the equivalence or nonequivalence of the course content.

Four-Year Programs

The curriculum outlined in this report is designed to be a two-year career program to prepare students for jobs. However, departments which offer this program must articulate with representatives in appropriate departments of nearby baccalaureate degree granting institutions in order to facilitate transfer.

Articulation is complicated because of the varied types of course structures existing in the different educational levels and the variety of departments teaching computer-related courses. A great deal of time can be consumed in this effort. However, the benefits to students and faculty are well worth the effort. Articulation guidelines should be prepared and made available to students entering computer-oriented curricula.

Students who make a decision to transfer should select courses suitable for transfer. Graduates of the program who work for a time and then continue toward a baccalaureate degree may find that additional lower division courses are needed as prerequisites to certain upper division level courses. Students who transfer into computer science programs will need to take the sequence of courses in calculus to prepare for later work. Students who transfer into data processing, information processing, or information systems management may find that most courses taken as part of the recommended program will transfer for credit, but not necessarily as equivalent courses. Some courses may transfer only as lower division electives.

One or more two-year college department members should be assigned to serve in continuing liaison to the transfer institution. They must keep in close contact with the four-year college representative from the appropriate department to facilitate the identification of course equivalences. College-wide liaison resulting in formal transfer agreements has been documented (see Appendix B). Articulation of this kind is an ongoing process and requires an ongoing effort.

Other Two-Year Programs

Although students do not often transfer between community and junior colleges, circumstances sometimes lead students to take one or more courses at a neighboring college. Colleges should cooperate in making this student movement as easy as possible. Regular meetings of faculty from area colleges could facilitate this cooperation and are encouraged.

Accreditation Agencies

Guidelines for use by accreditation agencies could help ensure that technically oriented career programs produce quality graduates. Their recommendations could influence course content and could encourage responsiveness to community needs. The ACM Accreditation Committee has initiated an effort to establish such recommendations for the two-year level. Contact ACM for the status of this work.

6.3 The Computer Profession

Certification examinations in computer programming are now available for the senior level programmer in three specialties: business applications, scientific applications, and systems programming. They are sponsored by eight professional societies through the Institute for Certification of Computer Professionals (ICCP). The senior level of this certification test is beyond the scope of an associate degree graduate without experience. An entry-level examination suitable for the graduates of a program such as that given in this report is in progress. Contact ICCP for more information (see Appendix A1).

Activities to assist students to become familiar with the profession are encouraged. Professional societies can provide sponsorship for student clubs and guest speakers for special activities. Attendance at regional computer exhibits offers an opportunity for professional interactions for students and faculty. Many colleges sponsor computer fairs, often in conjunction with high schools. Local programming competitions can be held each year for students, with winners encouraged to enter regional and national competitions regularly sponsored by ACM (see Appendix C).

7. Summary

This report is the consensus of a large number of educators and industry practitioners. It gives guidelines and recommendations for the design of a curriculum to educate entry-level applications computer programmers. It provides a flexible framework within which a two-year associate level degree career program can be constructed to prepare students for local jobs and at the same time give them a foundation for continued learning.

Rapid changes which affect the work of computer programmers will no doubt continue to occur. It is important that curriculum developers plan for ways to allow for almost continued adaptation. It is also critical that career advisement for this field include information about its dynamic and volatile nature. Those entering this field must expect continual change, continual adaptation, and continual challenges. Those who have talent and enjoy this type of work can expect to find much opportunity for an exciting career.

Many members of the Committee and participants of the workshops have indicated that they have benefited from their involvement with this effort. The Committee, listed in Appendix F, welcomes queries about, and continuing feedback from, these recommendations.

References

1. Berger, R. *Computer Programmer Job Analysis Reference Text*. AFIPS Press, Arlington, Va., 1974.
2. Brightman, R. W. (Ed.) *The Computer and the Junior College Curriculum*. American Association of Community and Junior Colleges, One DuPont Circle, N.W., Washington, D.C., 1970.
3. The Center for Vocational and Technical Education. *Business Data Processing Occupational Performance Survey*. The Ohio State University, Columbus, Ohio, March 1973.
4. Charp, S. *Scientific Data Processing Courses in Vocational and Secondary Schools*. IBM Data Processing Report E20-0092-0.
5. Cottrell, S., IV, and Fertig, R. T. Applications software trends: evolution or revolution. *Government Data Systems* 7, 1 (Jan./Feb. 1978), 12-13, 26.
6. Couger, J. D. (Ed.) *Curriculum recommendations for undergraduate programs in information systems: a report of the ACM Curriculum Committee on Computer Education for Management*. *Communications of the ACM* 16, 12 (Dec. 1973), 727-749.
7. Couger, J. D. The programmer's work bench. *Computing Newsletter* (Feb. 1980), 2-3.
8. Curriculum Committee on Computer Science. Curriculum 68: recommendations for academic programs in computer science; a report of the ACM Curriculum Committee on Computer Science. *Communications of the ACM* 11, 3 (March 1968), 151-197.
9. DeCamp, D. Employment: the 1981 outlook. *Computerworld* XV, 1, (Dec. 29/Jan. 5, 1980), 44-48.
10. Desautels, E. J. On computing facilities for computer science. *Computer* (Nov. 1974), 39-48.
11. Dooley, A. Programmers number 1 in employer demand. *Computerworld* (Oct. 6, 1980), 1.
12. Education Division, U.S. Department of Health, Education, and Welfare. *Associate Degrees and other Formal Awards Below the Baccalaureate: Analysis of Six-Year Trends*. U.S. Government Printing Office, Washington, D.C., 1978.
13. Fox-Morris Recruitment Network. *The Fox-Morris Report III*, 3 (1981), 1.
14. Gordon, R. Applications programmers: an endangered species? *Computer Careers News* 2, 5 (April 6, 1981), 1.
15. Half, R. Look (within) before you leap to new job. *Data Management* 18, 5 (May, 1980), 21-23.
16. Hamblen, J. W. *Computer Manpower—Supply and Demand by States*. Information Systems Consultants, R.R. 1, Box 256A, St. James, Mo., 1973, 1975 and 1981.
17. Hamblen, J. W., and Baird, T. B. *Fourth Inventory of Computers in U.S. Higher Education, 1976-77*. EDUCOM, Princeton, N.J., 1979.
18. Korn, W. and Laub, J. *A Suggested Curriculum for the Two-Year Associate Degree Business Data Processing Program*. Wisconsin Board of Vocational Technical and Adult Education, 4802 Sheboygan Avenue, Madison, Wis., 1973.
19. LeDuc, A. L., Jr. Motivation of programmers. *Data Base*, (A Quarterly Publication of SIGBDP), (Summer 1980), 4-12.
20. Lee, A. L., III. *A Followup Study of Computer Science Graduates of East Texas State University and El Centro Community College*. University Microfilms, International, 1977, 77-27, 555.
21. Little, J. C. Computer science-related degree programs at the associate level. In J. Hamblen and C. P. Landis, *The Fourth Inventory of Computers in Higher Education: An Interpretive Report*. EDUCOM and Westview Press, Boulder, Colo., 1980.
22. Little, J. C. Where are our students going? *Proceedings of the June 1973 SIGUCC Symposium*. Association for Computing Machinery, New York, N.Y., 1973.
23. Mantei, M. The effect of programming team structures on programming tasks. *Communications of the ACM* 24, 3 (March 1981), 106-113.
24. Maryland State Department of Education. *Maryland Business Data Processing Advisory Committee Report, 1974*. Maryland State Department of Education, Division of Vocational-Technical Education, P.O. Box 8711, BWI Airport, Baltimore, Md., 1974.
25. McLaughlin, R. A. That old bugaboo, turnover. *Datamation* 25, 11 (Oct. 1979), 96-101.
26. National Science Teacher Association. *PIP Newsletter, An Informational Publication of the Project on Information Processing I*, 1 (Feb. 1963) 1.
27. Office of Education. *Data Processing Technology—A Suggested Two Year Post High School Curriculum*. U.S. Department of Health, Education, and Welfare, Office of Education, Stock Number 1780-01240, U.S. Government Printing Office, Washington, D.C., 1973.
28. Office of Education. *Electronic Data Processing—1, A Suggested Two Year Post High School Curriculum for Computer Programmers and Business Applications Analysts*. U.S. Department of Health, Education, and Welfare, Office of Education, FS5. 280:80024, U.S. Government Printing Office, Washington, D.C., 1963.
29. Office of Education. *Electronic Data Processing in Engineering, Science and Business*. U.S. Department of Health, Education and Welfare, Office of Education, FS5. 280:80030, U.S. Government Printing Office, Washington, D.C., 1964.
30. Office of Education. *Scientific Data Processing Technology—A Suggested Two Year Post High School Curriculum*. U.S. Department of Health, Education, and Welfare, Office of Education, FS5. 280:80068, U.S. Government Printing Office, Washington, D.C., 1970.
31. Philips, J. W. Entry-level position of computer programmer: a survey. *SIGSCE Bulletin* 12, 1 (Fall, 1980), 198-202.
32. Pollack, M. *An Abstract of Principal Employers' Personnel Requirements and Higher Education Course Offerings in Data Processing*. Bronx Community College, New York, N.Y., 1973.
33. Rohrbach, J. J., Jr. (Ed.) *A Catalog of Performance Objectives, Criterion-Referenced Measures, and Performance Guides for Programming*. Vocational-Technical Educational Consortium of States, Georgia Department of Education, Atlanta, Ga., 1976.
34. Shelly, G. The future: a step forward or a step backward. *Software* 1, 1 (Jan. 1981), 34-38.
35. Taylor C. A. Knowing business and how to communicate best learned skills, study reveals. *Data Management* 19, 6 (June 1981), 34-37.
36. United States Bureau of Labor Statistics. *Occupational Outlook Handbook*, 1979-1980.
37. United States Civil Service Commission. Standards for computer specialist series, GS-334. Standards Division, Washington, D.C., most recent edition.
38. Winkler, C. Martin: applications development without programmers is here. *Computer Career News* 2, 7 (May 4, 1981), 1-4.
39. Withington, F. G. 1980: separating fact from fantasy. *Datamation* 26, 7 (July 1980), 76-82.
40. Yohe, J. M. An overview of programming practices. *Computing Surveys* 6, 4 (Dec. 1974), 221-245.
41. Zelkowitz, M. V. Perspectives on software engineering. *Computing Surveys* 10, 2 (June 1978), 197-216.

Appendices

Appendix A. Reference Materials

1. Computer Programming as a Career

1. American Federation of Information Processing Societies, Inc. A Look into Computer Careers. 1815 North Lynn Street, Arlington, VA 22209, 1980.
2. Association for Computing Machinery. Self-Assessment Procedures. 1133 Avenue of the Americas, New York, NY 10036.
3. Babb, P. A computer career: how to get there from here. *AEDS Monitor* 19, 4-6 (Nov. 1980), 7-23.
4. *Computers and Careers, A Suggested Curriculum for Grade 9-12*. Superintendent of Documents, U.S. Government Printing Office, Washington, DC, Stock Number 1780-1241.
5. Data Processing Management Association. Your Career in Data Processing. 505 Busse Highway, Park Ridge, IL 60068.
6. Griffin, J.P. The job outlook in brief. *Occupational Outlook Quarterly*, U.S. Government Printing Office, Washington, DC 20402 (most recent issue).
7. *Hansen's Weber Salary Survey on Data Processing Positions*, 1080 Green Bay Road, Lake Bluff, IL 60044, latest edition.
8. Institute for the Certification of Computer Professionals (ICCP). The Psychological Corporation. *Report of the 1980 Certificate in Computer Programming Examinations*. 757 Third Avenue, NY 10017, Feb. 1981.
9. Maniotes, J. and Quasney, J.S. *Computer Careers*. Hayden Publishing Company, 1974.
10. McDaniel H. *Careers in Computers and Data Processing*. Petrocelli Books, 1978.
11. Northwest Regional Educational Laboratory. *Elements of Computer Careers*. Prentice-Hall, Inc., 1977.
12. *Occupational Outlook Handbook*, U.S. Department of Labor, Bureau of Statistics, 1980-1981.
13. Stibbens, S. The movement is up, up, and away. *Infosystems* (Dec. 1980), 75-76.
14. Taylor, R.P., and Fisher, J. The relative importance of sources of information for keeping programmers up-to-date. *Proceedings of the Sixteenth Annual Computer Personnel Research Conference, ACM SIGCPR*, Aug. 16-17, 1979, 34-43.
15. Willoughby, T.C. Programmer exemption. *Computer Personnel* (A Quarterly Publication of the SIGCPR) 8, 2 (March 1980), 10-15.
16. Zalud, B. Ascent of EDP career cliff: dangers, set-backs and rewards. *Data Management* (Sept. 1977), 11-13.
4. *Computer Decisions*, Hayden Publishing Co., 50 Essex St., Rochelle Park, NJ 07662.
5. *Computer Personnel*, A Quarterly Publication of the SIGCPR, ACM, 1133 Avenue of the Americas, New York, NY 10036.
6. *Computerworld*, 797 Washington Street, Newton, MA 02160.
7. *Computing Reviews*, ACM, 1133 Avenue of the Americas, New York, NY 10036.
8. *Computing Newsletter*, J. Daniel Couger (Ed.), Box 7375, Colorado Springs, CO 80933.
9. *Computing Surveys*, ACM, 1133 Avenue of the Americas, New York, NY 10036.
10. *Data Base*. A Quarterly Newsletter of SIGBDP, Special Interest Group for Business Data Processing, ACM, 1133 Avenue of the Americas, New York, NY 10036.
11. *Data Management*. Data Processing Management Association, 505 Busse Highway, Park Ridge, IL 60068.
12. *Datamation*, P.O. Box 200, Greenwich, CT 06830.
13. *DP/ED*, North American Publishing Co., 401 North Broad Street, Philadelphia, PA 19108.
14. *EDP Analyzer*, 925 Anza Avenue, Vista, CA 92083.
15. *Government Data Systems*, United Business Publications, Inc., 750 Third Avenue, New York, NY 10017.
16. *Infosystems*, Hitchcock Publishing Company, P.O. Box 3007, Wheaton, IL 60187.
17. *Interface, The Computer Education Quarterly*, Mitchell Publishing Company, 116 Royal Oak, Santa Cruz, CA 95066.
18. *Mini-Micro Systems*, Cahners Publishing Co., 211 Columbus Ave., Boston, MA 02116.
19. *MIS Quarterly*, 269 19th Avenue South, University of Minnesota, Minneapolis, MN 55455.
20. *SIGCAS Bulletin*, A Quarterly Publication of the Special Interest Group for Computers and Society, ACM, 1133 Avenue of the Americas, New York, NY 10036.
21. *SIGCSE Bulletin*, A Quarterly Publication of the Special Interest Group on Computer Science Education, ACM, 1133 Avenue of the Americas, New York, NY 10036.
22. *SIGCUE Bulletin*, A Publication of the Special Interest Group on Computer Uses in Education, ACM, 1133 Avenue of the Americas, New York, NY 10036.
23. *The Computing Teacher*, International Council for Computers in Education, Eugene, OR 97403.

2. Instructional Bibliographies

1. Annual bibliography of computer-oriented books. *Computing Newsletter*. Box 7345, Colorado Springs, CO 80933.
2. Cougar, J.D. Ed. Interesting films to use in dp instruction. *Computing Newsletter XIV*, 9 (May 1981), 6.
3. Lidtke, D. *Computers and Computer Applications: A Film Bibliography*. OCCE Special Report—A Publication of the Curriculum Group of the Oregon Council for Computer Education, 4015 S.W. Canyon Road, Portland, OR 97221, Feb. 1977.
4. Joint Committee of the ACM and IEEE. *A Library List on Undergraduate Computer Science, Computer Engineering, and Information Systems*. IEEE Publication EH0131-3, 1978.

3. Newsletters and Periodicals

1. *AEDS Journal*, Association for Education Data Systems, 1201 Sixteenth Street, N.W., Washington, DC 20036.
2. *Communications of the ACM*, 1133 Avenue of the Americas, New York, NY 10036.
3. *Computer*, IEEE Computer Society, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720.

Appendix B. Curriculum Implementation Material

1. Beil, D. H. The data processing curriculum of the National Technical Institute for the Deaf: a suggested implementation for an AAS degree program in computer programming. *Proceedings of the ACM Annual Conference*, 1978, 812-821.
2. Dillman, R.W., Anderson, W.H., Choper, D.L., Lloyd, J.M., Simms, K.B., and Williams, J.F. Two-year curricula in computer studies—implementing the guidelines. *SIGCSE Bulletin* 10, 3 (Aug. 1978), 140-150.
3. Gorgone, J. T., and Schrage, J. F. A design for the education of computer programmers at the associate level within a baccalaureate level institution. *Proceedings of the ACM Annual Conference*, 1978, 807-811.
4. Lee, I. H., and Plog, C. E. A design for a community/junior college curriculum with options for two neighboring institutions. *Proceedings of the ACM Annual Conference*, 1978, 798-806.
5. Illinois Office of Education. *A Guide for Planning, Organizing, and Utilizing Advisory Councils*. Division of Vocational Technical Education, Illinois Office of Education, 100 North First Street, Springfield, IL.

6. Little, J. C. An overview of ACM guidelines and recommendations for a community and junior college career program in computer programming. *Proceedings of the ACM Annual Conference, 1978, 793-797.*
7. Little, J. C. The status of computer education in the community and junior colleges—needs and alternatives. *Proceedings of the National Computer Conference, AFIPS Press, Arlington, VA, 1978.*
8. University of Illinois. A Transfer Handbook for Junior College Students, Academic Advisors, and Counselors. University of Illinois at Urbana-Champaign, Urbana, IL.
9. Maniotes, J. The state of undergraduates computer and data processing programs at public universities in Indiana. *SIGCSE Bulletin 6, 1 (Feb. 1974), 53-58.*

Appendix C. Student Activities and Memberships

1. Association for Computing Machinery, 1133 Avenue of the Americas, New York, NY 10036.
2. Association for Systems Management, 24587 Bagley Road, Cleveland, OH 44138.
3. Data Processing Management Association, 505 Busse Highway, Park Ridge, IL 60068.

Appendix D. Sources of Job Descriptions

1. *Datamation* magazine, Technical Publishing, 666 Fifth Ave., New York, NY 10103.
2. *Infosystems* magazine, Hitchcock Publishing Co., Hitchcock Building, Wheaton, IL 60187.
3. Philip H. Weber Services, A. S. Hansen, Inc., 1080 Green Bay Road, Lake Bluff, IL.
4. *Installation management: Organizing the DP Activity.* IBM Corporation, Dept. 824, Technical Publication/Systems Department, 1133 Westchester Avenue, White Plains, NY 10604.
5. *Occupations in Electronic Computing Systems.* U.S. Department of Labor, Superintendent of Documents, Washington, DC 20402.

Appendix E. Contributors and Workshop Participants

Workshops Participants, August 11-13, 1975, Gloucester Point, Virginia

Richard H. Austing, University of Maryland
 Bruce Barnes, National Science Foundation
 Frank Cable, Pennsylvania State University
 Richard Ciero, Thomas Nelson Community College, Virginia
 Richard Dempsey, Pennsylvania State University, Worthington Scranton Campus
 John Dineen, Middlesex County College, New Jersey
 Gerald L. Engel, Virginia Institute for Marine Science, Virginia
 B. Albert Friedman, Sinclair Community College, Ohio
 John Gorgone, Purdue University, Indiana University at Fort Wayne, Indiana
 Harold Highland, SUNY/Farmingdale, New York
 Ron Lenhart, Yavapai College, Arizona
 Joyce Currie Little, Community College of Baltimore, Maryland
 John Maniotes, Purdue University, Calumet Campus, Indiana
 Morris Pollack, Bronx Community College, New York
 Richard Reynolds, Orange Coast College, California
 Harice Seeds, Los Angeles City College, California

Workshop Participants, May 27-28, 1976, Gloucester Point, Virginia

Richard H. Austing, University of Maryland
 Frank Cable, Pennsylvania State University
 Richard Ciero, Thomas Nelson Community College, Virginia

Benjamin Diamant, IBM Corporation, Woodbridge, Virginia
 Gerald L. Engel, Virginia Institute for Marine Science, Virginia
 Richard Gehrt, Employers Insurance of Wausau, Wisconsin
 Malcolm Gotterer, Florida International University
 Majorie Leeson, Delta College, Michigan
 Ronald Lenhart, Yavapai College, Arizona
 Doris K. Lidtke, Towson State College, Maryland
 Joyce Currie Little, Community College of Baltimore, Maryland
 John Maniotes, Purdue University, Calumet Campus, Indiana
 Robert R. Pearson, North Carolina Educational Computer Services
 Harice Seeds, Los Angeles City College, California
 George Sotas, U.S. Government Accounting Office
 Robert Thorn, Computer Related Services, Norfolk, Virginia
 John W. Westley, Illinois Central College, East Peoria, Illinois
 James D. Wintress, Aetna Life and Casualty, Hartford, Connecticut

Workshop Participants, February 1977, Atlanta, Georgia

Richard H. Austing, University of Maryland, College Park, Maryland
 Donald Beil, Rochester Institute of Technology, Rochester, New York
 Priscilla Cairra, Whittier Regional Voc-Tech High School, Haverhill, Massachusetts
 Tom Cashman, Long Beach Community College, Long Beach, California
 Richard Ciero, Thomas Nelson Community College, Hampton, Virginia
 Sylvia Charp, Philadelphia School District, Philadelphia, Pennsylvania
 Gerald L. Engel, Virginia Institute for Marine Sciences
 Gary Gleason, Pensacola Junior College, Pensacola, Florida
 Dan Kameron, Robert J. Brady Co., Bowie, Maryland
 Iva Helen Lee, McLennan Community College, Waco, Texas
 Majorie Leeson, Delta College, University Center, Michigan
 John Lloyd, Montgomery College, Rockville, Maryland
 Joyce Currie Little, Community College of Baltimore, Maryland
 John Maniotes, Purdue University—Calumet Campus, Hammond, Indiana
 Robert Marcus, Ethnotech, Inc., Lincoln, Nebraska
 Michael Meehan, Winthrop Publishers, Inc., Cambridge, Massachusetts
 Harice Seeds, Los Angeles City College, Los Angeles, California
 Gary D. Shelly, Anaheim Publishing Co., Fullerton, California
 Richard Slocum, University of Maine at Augusta, Augusta, Maine
 Erwin Vernon, Sinclair Community College, Dayton, Ohio
 Gerald Weinberg, Ethnotech, Inc., Lincoln, Nebraska

Workshop Participants, August 1977, Gloucester Point, Virginia

Richard H. Austing, University of Maryland, College Park, Maryland
 Richard Ciero, Thomas Nelson Community College, Hampton, Virginia
 John Dineen, Middlesex County College, Edison, New Jersey
 Gerald L. Engel, Virginia Institute for Marine Sciences, Virginia
 Donna Hutcheson, East Texas State University, Commerce, Texas
 Iva Helen Lee, McLennan Community College, Waco, Texas
 Marjorie Leeson, Delta College, University Center, Michigan
 Joyce Currie Little, Community College of Baltimore, Maryland
 John Lloyd, Montgomery College, Rockville, Maryland
 Claudia E. Plog (Bette), El Centro College of the Dallas County Community College District, Dallas, Texas
 Joseph Ramach, State of Maryland, Baltimore, Maryland
 Harice Seeds, Los Angeles City College, Los Angeles, California
 John P. Stonebeck, Northhampton County College, Bethlehem, Pennsylvania
 John Sweeney, National Technical Institute for the Deaf, Rochester Institute of Technology, Rochester, New York
 Erwin Vernon, Sinclair Community College, Dayton, Ohio
 John W. Westley, Illinois Central College, East Peoria, Illinois
 Jacqueline A. Wood, University of Alabama, Birmingham, Alabama
 Raymond W. Woodcock, Boulder Valley Vocational-Technical Center, Boulder, Colorado

Additional Contributors or Reviewers

Walter Anderson, U.S. Government Accounting Office,
Washington, D.C.
Julius A. Archibald, Jr., SUNY at Plattsburgh, New York
Barry L. Bateman, Southern Illinois University at Carbondale
Robert Bise, Orange Coast College, California
Hoyle Blalock, Jr., Central Piedmont Community College
Joseph J. Cebula, Community College of Philadelphia, Pennsylvania
Jane Chapman, Paducah Community College, Kentucky
Frank Connolly, Montgomery Community College, Maryland
William W. Cotterman, Georgia State University, Atlanta, Georgia
J. Daniel Couger, University of Colorado, Colorado
Joan Culverhouse, McLennan Community College, Texas
John Dalphin, Indiana University, Purdue University, Fort Wayne,
Indiana
Richard W. Dillman, Western Maryland College, Westminster,
Maryland
Karen Eck, McDonnell-Douglas Automation Company, California
Robert Fox, College of duPage, Illinois
Michael Faiman, University of Illinois, Urbana, Illinois
Dan Fullerton, Albuquerque Technical-Vocational Institute,
Albuquerque, New Mexico
David Gilmore, U.S. Office of Personnel Management,
Washington, D.C.
Gary Gleason, Pensacola Junior College, Florida
Jim Gross, University of Wisconsin Center
William Gruener, Addison-Wesley Publishing Co., Reading,
Massachusetts
John Hamblen, National Bureau of Standards
Carl Hammer, UNIVAC, a Division of Sperry Rand
Preston Hammer, Grand Valley State College, Michigan
Patricia Hendershot, U.S. Department of Agriculture,
Washington, D.C.
Laureen M. Hendry, IBM Corporation, Atlanta, Georgia
Larry Jehn, University of Dayton, Ohio
Karl Karlstrom, Prentice-Hall, Inc., Englewood Cliffs, New Jersey
Ed Keith, Citrus College, California
Alton Kindred, Manatee Junior College, Florida
Ralph Lee, University of Missouri at Rolla
Charles Leidlich, Miami-Dade Community College, Florida
Theodore S. Lewis, Trident Technical College, Charleston,
South Carolina
Donald W. Loveland, Duke University, North Carolina
Bernard Luskin, Coast College District, California
Sister Patricia Marshall, Xavier University of Louisiana
William McCartin, Noxell Corporation, Baltimore, Maryland
Daniel D. McCracken, McCracken Associates
Larry Newcomer, York Campus of Pennsylvania State University

Tom Oatney, Clark Technical College, Springfield, Ohio
Charles G. Peterson, Northwest State, Maryvale, Missouri
Montgomery Phister, Jr., Systems Consultant, Santa Monica,
California
Lawrence D. Pickens, Del Mar College, Texas
Roger M. Polay, Washtenaw Community College, Ann Arbor, Michigan
Doug Ruby, McDonnell-Douglas Automation Company, California
Martin Sandman, National Cash Register, Inc.
Gary Shelly, Anaheim Publishing Company, California
Bobby Smith, City of Portsmouth, Virginia
B. D. Sorell, Kilgore College, Texas
Leslie S. Spencer, Indiana University of Pennsylvania
Nancy Stern, Hofstra University, Hempstead, New York
Walter S. Szalajka, Lewis University, Lockport, Illinois
Ralph A. Szweda, Monroe Community College, Rochester, New York
James Tuedio, Pasadena City College, Pasadena, California
Gerald M. Weinberg, Ethnotech, Inc., Lincoln Nebraska
Eric Weiss, Sun Company, Radnor, Pennsylvania
Frank M. White, Catonsville Community College, Maryland
Eric Whiteside, Miami-Dade Community College, Florida

Appendix F.

ACM Committee on Curriculum for Community and Junior College Education, 1981

Chair: Joyce Currie Little, Community College of Baltimore,
Maryland*
Members: John (Jack) Dineen, Middlesex County College, Edison,
New Jersey
Dennis Fletcher, Informatics, Inc., Canoga Park, California
Iva Helen Lee, McLennon Community College,
Waco, Texas
Marjorie Leeson, Delta College, University Center,
Michigan
John F. Schrage, So. Ill. Univ. at Edwardsville, Illinois
Harice Seeds, Los Angeles City College, California
John Sweeney, National Technical Institute for the Deaf,
Rochester, New York
Erwin Vernon, Sinclair Community College, Dayton, Ohio
Ray Woodcock, Boulder Valley Voc-Tech Center, Boulder,
Colorado

*Joyce Currie Little's current address is Towson State University,
Dept. of Mathematics and Computer Science, Stephens Hall,
Room 111, Baltimore, MD 21204.

