INFORMATION TO USERS

This material was produced from a microfilm copy of the original document. While the most advanced technological means to photograph and reproduce this document have been used, the quality is heavily dependent upon the quality of the original submitted.

The following explanation of techniques is provided to help you understand markings or patterns which may appear on this reproduction.

1. The sign or "target" for pages apparently lacking from the document photographed is "Missing Page(s)". If it was possible to obtain the missing page(s) or section, they are spliced into the film along with adjacent pages. This may have necessitated cutting thru an image and duplicating adjacent pages to insure you complete continuity.

2. When an image on the film is obliterated with a large round black mark, it is an indication that the photographer suspected that the copy may have moved during exposure and thus cause a blurred image. You will find a good image of the page in the adjacent frame.

3. When a map, drawing or chart, etc., was part of the material being photographed the photographer followed a definite method in "sectioning" the material. It is customary to begin photoing at the upper left hand corner of a large sheet and to continue photoing from left to right in equal sections with a small overlap. If necessary, sectioning is continued again — beginning below the first row and continuing on until complete.

4. The majority of users indicate that the textual content is of greatest value, however, a somewhat higher quality reproduction could be made from "photographs" if essential to the understanding of the dissertation. Silver prints of "photographs" may be ordered at additional charge by writing the Order Department, giving the catalog number, title, author and specific pages you wish reproduced.

5. PLEASE NOTE: Some pages may have indistinct print. Filmed as received.

University Microfilms International
300 North Zeeb Road
Ann Arbor, Michigan 48106 USA
St. John's Road, Tyler's Green
High Wycombe, Bucks, England HP10 8HR

Competency analysis:
Business computer programming

by

Ralph LeRoy Keul

A Dissertation Submitted to the
Graduate Faculty in Partial Fulfillment of
The Requirements for the Degree of
DOCTOR OF PHILOSOPHY

Department: Industrial Education
Major: Industrial Education (Industrial
Vocational-Technical Education)

Approved:

Signature was redacted for privacy.
In Charge of Major Work

Signature was redacted for privacy.
For the Major Department

Signature was redacted for privacy.
For the Graduate College

Iowa State University
Ames, Iowa

1978

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

INTRODUCTION

From the darkest dawn of history, until this very day, man has been a perpetrator, a victim, a creature of change. The brute lessons of survival insist that stagnation invites extinction, and that each person must learn whatever is necessary to survive in his own time. Now as our society plunges haphazardly forward we wonder if our skills and knowledge today will be adequate tomorrow. With our rapid innovations in technology the skills and competencies necessary for job entry are constantly changing, especially in the field of computer programming. In a relatively short span of years computer technology has exploded from simple cryptic adding machines to phenomenally intricate and omnipresent controllers of many aspects of our lives. The impact of such unsettled change was described by Taba (61), "The future itself is a direction in which we no longer look with confidence but with vauge forebodings and a sense of unpreparedness" (p. 36). In this age of galloping technology where nothing is static we must persistently strive to meet this challenge of change. As society has witnessed the fading need for unskilled labor and the ever increasing need for technical specialists, education has expanded and grown to meet those changing needs.

The Smith-Hughes Act of 1917, the National Defense Education Act of 1958, the 1963 Vocational Education Act, and Iowa Senate File 550 all focused upon the demands of our

changing society and affected many levels of our education system, including establishment of the Iowa area community colleges. Chapter 280A of the Code of Iowa (1968) stipulated:

> It is hereby declared to be the policy of the state of Iowa and the purpose of this chapter to provide for the establishment of not more than seventeen areas which shall include all of the area of the state and which may operate either area vocational schools or area community colleges offering to the greatest extent possible, educational opportunities and services .... (p. 4)

The Des Moines Area Community College structured vocational training programs in response to the skilled needs survey of 1967 (Langerman), and now offers many career programs to develop the skills and competencies necessary for success in a chosen occupation.

The community college curricula have changed over the years but in a somewhat unpredictable manner. Sometimes a change in curriculum content is initiated by an unhappy employer, sometimes by a manufacturer of new equipment, but always with a persistent lack of systematic reliable facts to base decisions upon. A typical vocational program at the community college level that has gone through this pattern of erratic change is the Computer Programming program at Des Moines Area Community College and thus served as the subject of this evaluation study.

Problem of the Study

The problem of this study was twofold:

1. To identify the competencies essential to entry level

business computer programmers.

2. To analyze the existing computer programming curriculum to determine the extent of essential competencies included in the curriculum at Des Moines Area Community College.

Purpose of the Study

The study was undertaken with the following purposes:

1. To compile a list of competencies identified as needed by employed business computer programmers.
2. To provide a systematic method of analyzing the competencies required of entry level computer programmers.
3. To obtain reliable information to identify discrepancies of industry need and existing curriculum content so as to provide data relative to curriculum revision and improvement.

Need for the Study

This study attempted to improve understanding, cooperation and exchange of ideas between educators and industry relative to computer programming. The competencies expected by industry are constantly changing, and it is essential that curriculum content be evaluated periodically to reflect the current priorities of industry. The analysis of current computer education by Heiker and Galli (29) stressed that the education community, not industry, will have to solve the

curriculum problem by research into industry skill requirements. The need was summed up by Shelly (56) in a study of why industry does not hire computer programming school graduates. He concluded:

> Schools are often not responsive to the critical needs of industry, while students are taught a great deal of subject matter that is not useful to them in the data processing profession, and funds are not directed to the updating of curriculum. (p. 3)

There are trends appearing in data processing literature as evidenced above and the introduction of new computer products that suggest the community college may not be keeping pace with changing technology. The computer hardware has developed into larger and larger computers on the one hand and smaller, even micro computers on the other. Further, the techniques of programming are becoming increasingly dependent upon more structured approaches and team planning. Computer programming was once considered a highly individual art, but current thought is shifting towards disciplined teamwork and increased predictability in programming. As Weinberg (68) described:

> We have too long suffered under the illusion that programming is an individual activity, when the most casual observation on real programming projects will reveal the intensely cooperative nature of the work. (p. 98)

An even more powerful stimulus for change has been our social trends toward privacy with demands that access to any computer data be protected and controlled from abuse (31). The present curriculum does not address these trends.

Additional support of the need for this study was found in the literature of several authorities on curriculum. Taba (61), in her discussion of curriculum warned us:

> Often the curriculum is loaded with insignificant detail because there is no way to determine what is important and what is not. Without a reference to basic ideas any one detail is as important as any other. ... It is easy to indulge in broad coverage of facts under the illusion of providing for depth when actually this practice invites neglect of insight, prevents thoughtful reactions, and stultifies inquiry. (p. 270)

The curriculum is constantly accommodating additions of new material; but obsolete material thoroughly integrated into the curriculum requires analytical and conscious effort to ultimately delete such material (28,15).

In addition Cashman (20), a recognized expert in the field of computer education, criticized data processing educators:

> As one analyzes the fact that public education trains personnel successfully in many areas and disciplines, why has education not been successful in meeting the needs of business data processing industry as evidenced by the extensive in-house training in all phases of data processing? In the past fifteen years there has been virtually no leadership from data processing industry through its related professional associations relative to curriculum development in career education in such areas as business programming. (p. 3)

Educators at the community college level have an obligation to take the initiative in updating and redeveloping curriculum to provide students more relevant and usable skills.

There has been a conspicuous thread through history insisting on the updating of education. Nearly forty years

ago Carver (19) related the need as:

> The problem of curriculum construction and revision in the field of education is one we must face squarely and endeavor to solve. What our philosophy regarding curriculum changes shall be, what we teach and how we teach it are paramount issues. (p. 2)

The community college must recognize that the curriculum can grow stale and that the content should be periodically evaluated to verify that student competencies meet and fulfill employer expectations.

The content and structure of the programming curriculum at Des Moines Area Community College is supported by similarities found in numerous data processing articles, programmer job descriptions, and college catalogs. An example of this content agreement is found in the objectives suggested by Brightman (13):

1. Understanding computer concepts and capabilities.
2. Problem solution utilizing flowcharts and fundamental processes.
3. Ability to code solutions in one or more programming languages.
4. Basic concepts of communications and interactive technique.
5. Understanding software concepts.
6. Data file organizations.
7. Basic understanding of a business system. (p.36)

Although curriculum agreement is a very positive indication of appropriate content, educators should not be lulled into assuming that the content is current. There should be some method of periodic assessment of industry needs and analysis systems established. However, the curriculum guidelines discussed by Little (40) offered the caution:

> No one structure could be designed that would suit all community and junior colleges. Each institution must design its curriculum and the courses within it to fit within its own mission and philosophy, the characteristics of its student clientele, the characteristics of its commercial environment and other educational opportunities available to students. (p. 30)

Although other competency research studies may be beneficial and used as models, each community college must vigorously endeavor to search out and delineate its own unique curriculum requirements.

Evidence of the need for this study is clear and specific. If the vocational programs at the community college level are to fulfill their purpose they must be guided by evaluation and curriculum redevelopment. This study was an evaluation of curriculum using competency analysis.

Questions of the Study

The questions to be researched by this study were:

1. What are the competencies considered to be needed by entry level business computer programmers?
2. Does the existing curriculum at Des Moines Area Community College lack emphasis of competencies identified as needed ?
3. Does the existing curriculum at Des Moines Area Community College contain excess emphasis of competencies identified as needed?
4. Can the evaluation procedure developed and applied

facilitate systematic curriculum development?

Assumptions of the Study

This study was based upon several assumptions which made the study more efficient and pragmatic. The assumptions included:

1. The entry level competencies needed by industry are best perceived by employed business computer programmers because they must actually perform the required tasks.
2. The entry level competencies provided by the Des Moines Area Community College curriculum are best perceived by analysis of course outlines and consultation with the instructional staff as to the constituent competencies.
3. The business computer programmers surveyed provided accurate, unbiased, and objective information concerning essential competencies.
4. The competency items of analysis were thorough and representative of the knowledge, skills, and activities needed by entry level business computer programmers as presented and revised in the pilot survey.

Limitations of the Study

The scope of this research was constrained by the following limitations:

1. This study was restricted to observations of a sample of business computers employed in Iowa.
2. This study investigated only competencies needed within the employment region and presented by the Des Moines Area Community College Computer Programming curriculum.

Definition of Terms

To help the reader of this study, several terms were defined:

1. Business computer programmer: One who prepares the logic necessary to process commercial functions such as accounting or inventory but excluding the fields of scientific, analog, mathematical, and software programming.
2. Business computer programming curriculum: A seven quarter program of approximately 119 credits leading to the associate of applied science degree.
3. Competencies: Those skilled tasks and levels of understanding required of job entry level business computer programmers.
4. Programming languages: A group of computer programs which translate human oriented syntax into machine executable commands. Commonly identifed by acronyms such as COBOL, FORTRAN, and RPG.

Organization

This study is organized in six chapters: Introduction, Review of Literature, Method and Procedure, Findings, Discussion, and Summary, Conclusion, and Recommendations. The initial preparation of problem definition, development, and organization of the research proposal was undertaken during the spring and summer quarters of 1977. The review of literature, with development of the pilot instrument, and administration of the pilot survey was conducted during the fall quarter of 1977. Development of the revised survey instrument and administration of the statewide survey occurred during the winter quarter of 1978. Analysis of the data, interpretation of the findings, and discussion of the results were completed during the spring quarter of 1978.

REVIEW OF LITERATURE

The review of literature was considered carefully and undertaken with three primary goals. The first, was to identify as much information as possible to define the job of a computer programmer from a macro viewpoint. A thorough research of the scope of programmer expectations was viewed to be necessary to give meaning and direction to the competency analysis. Second, the literature was reviewed to find relevant information concerning task analysis methodology. The third goal, was to study significant research into the specific area of computer programmer competency analysis.

Historical Influences

The enterprise and technology of electronic computers and data processing is fairly new by all measures of history. IBM installed their first commercial computer in 1953, just 25 years ago (45). Yet in that short span whole generations of machines and techniques have come and gone; at times so quickly people did not have a chance to learn to use them effectively. The computer has always been just a complex adding machine, a device to process information and at the same time accomplish useful results. The oldest known machine of this type was the abacus, consisting of rows of beads strung on wires, and dating back to the sixth century

B.C. (53).

In 1614 John Napier, the inventor of logarithms, developed a set of numbering rods to aid in multiplication (called Napiers bones). Edmund Gunter, in 1620, devised a line of numbers scaled in proportion to the logarithm of one through ten, and this was used by William Oughtred to form the slide rule in 1633.

The mechanical adding machine was invented by Blaise Pascal in 1642, followed by the calculating machine which Gottfried Leibnitz invented in 1673. In 1804, Joseph Jacquard developed the automatic loom controlled by punched cards. Then in 1823 an analytical engine that would use punched card input was designed by Charles Babbage; however technology of that time could not produce all the necessary parts so it was never finished. Lady Lovelace designed many programs for the analytical engine and so became the world's first programmer (53).

The U.S. census of 1890 was a milestone in history. The previous census had taken seven and a half years to complete. Herman Hollerith developed a punched card sorting machine that enabled completion of the census in one-third the time. In 1930, Dr. Vannevar Bush built a mechanical analog computer used for army ballistics calculations, and in 1944 Howard Aiken completed the Mark I relay computer. It was forty years ago at Iowa State University that John Atanasoff conceived the first electronic computer. It was

completed in 1942 but ironically neither IBM nor Remmington Rand felt it had commercial value. Based on the ideas of Dr. Atanasoff, Dr. John Mauchly and Dr. J. Presper Eckert completed the ENIAC in 1945. It was designed to add 5000 numbers per second and weighed 30 tons (37).

Another radical milestone in history occured in 1946 when Dr. John von Neuman conceived the idea that instructions, as well as data, could be stored in memory; and that numbers could be represented more efficiently in binary notation. The first programming was performed in machine language, and the first compiler was developed in 1952 by Dr. Grace Hopper. Magnetic core storage was invented in 1951 by Dr. Jay Forrester. FORTRAN language was developed in 1956, and COBOL in 1960 (42).

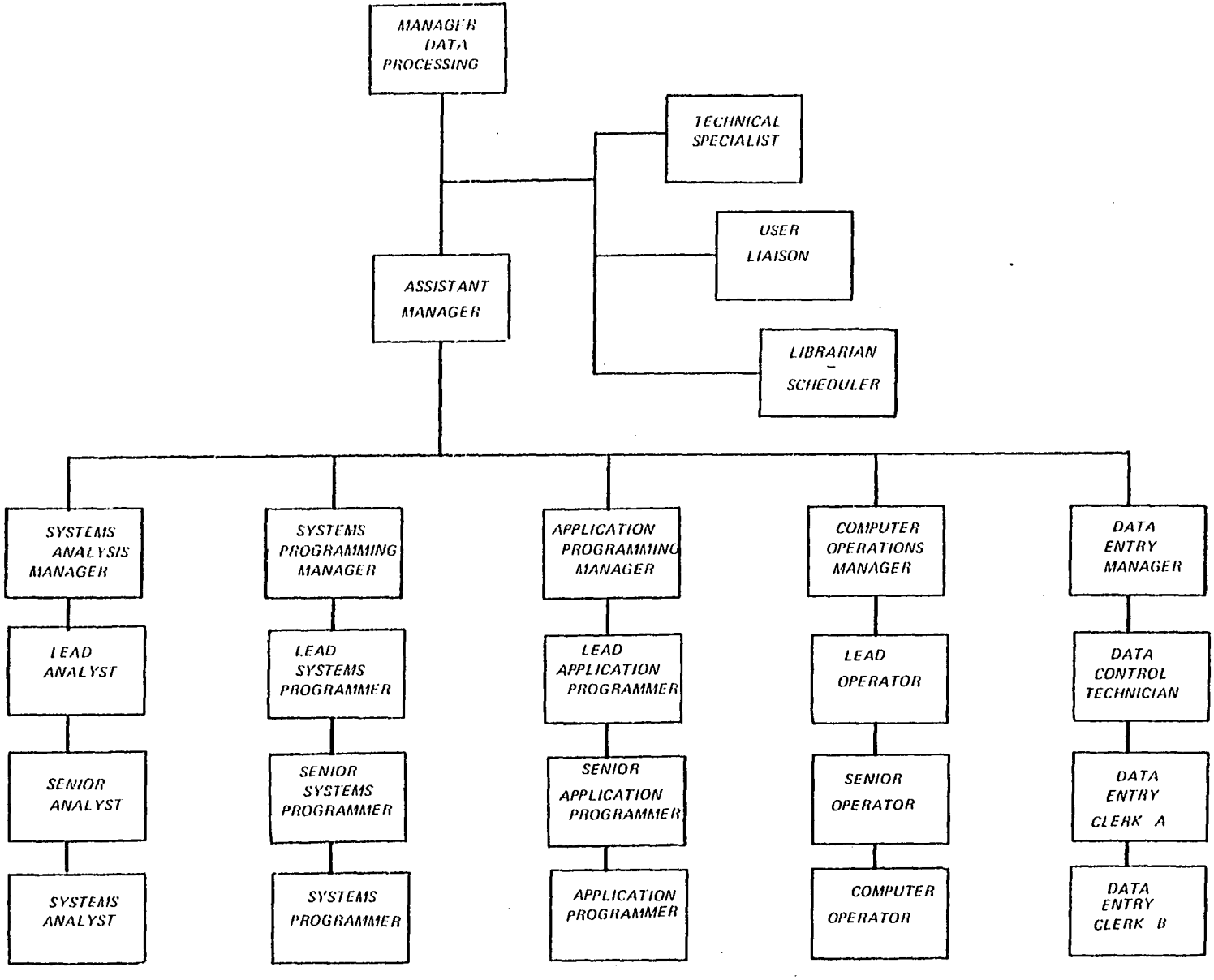
Thus began the widespread use of electronic digital computers and the inseparable profession of the computer programmer. The purpose of this brief history serves to point out two important perspectives to consider in task analysis. First, that the occupation of computer programmer is part of and a result of rapid, persistent, and dramatic technological changes. Second, that the occupation lacks the sustained tradition, stability, and leadership found in many professions. For this reason, rules or standards are at best scanty, conformity or convention are ill-defined, and standards have not reached widespread acceptance (2, 22).

Job Descriptors

A business computer programmer is a person who converts ideas into machine actions. The ideas are always of a problem solving nature, while the machine performs such actions as arithmetic manipulation. The programmer must synthesize many bits and pieces of information together into some meaningful form, then convert that input into machine syntax (program), then test and correct in a seemingly endless loop until the program works, then gather up all that has been done so that someone else can understand it. As Weinberg (69) said in his book, The Psychology of Computer Programming: "Documentation is the castor oil of programming; managers think it is good for programmers, and programmers hate it. In fact, the managers know it must be good because programmers hate it so much" (p. 262).

There are numerous levels of programmers encompassing the trainee, the programmer, the programmer analyst, the lead programmer, the systems programmer, and the programming manager. There are also specializations such as test programmer, application programmer, software programmer, and data base programmer (48). The computer programmer as a member of the data processing department serves a staff or support function to line departments. The exception to this is where the company product is the production of programs. Figure 1 shows the formal organization of a typical data

Figure 1. Data processing organization chart


MANAGER DATA PROCESSING
TECHNICAL SPECIALIST
USER LIAISON
ASSISTANT MANAGER
LIBRARIAN - SCHEDULER
SYSTEMS ANALYSIS MANAGER
SYSTEMS PROGRAMMING MANAGER
APPLICATION PROGRAMMING MANAGER
COMPUTER OPERATIONS MANAGER
DATA ENTRY MANAGER
LEAD ANALYST
LEAD SYSTEMS PROGRAMMER
LEAD APPLICATION PROGRAMMER
LEAD OPERATOR
DATA CONTROL TECHNICIAN
SENIOR ANALYST
SENIOR SYSTEMS PROGRAMMER
SENIOR APPLICATION PROGRAMMER
SENIOR OPERATOR
DATA ENTRY CLERK A
SYSTEMS ANALYST
SYSTEMS PROGRAMMER
APPLICATION PROGRAMMER
COMPUTER OPERATOR
DATA ENTRY CLERK B

processing department (25, 48).

A brief job description for computer programmer might read: Analyzes problems as outlined by the systems analyst, designs logic flowcharts, codes programs in computer language, tests the program to find and repair errors, then documents the program, and may consult with program users (17, 74).

A programmer becomes immersed in the singular world of analysis and evaluation while following spider-web paths of logic and syntax. Much of the work is clerical: writing, sorting, organizing, filing, and sitting long hours behind a desk. A programmer must become accustomed to constant training regardless of years of experience; first because the technology is constantly changing, but also because the programmer floats from being an expert in inventory this year to being an expert in payroll next year (17). The functional duties of a programmer will vary between any two installations. In a large installation there is more scope for specialization while a smaller installation requires a person to be able to function in multi-purpose roles.

Working Environment

A programmer usually works in an office, with two to five other people, and if the group is larger, each person will generally have a cubicle. The programmers in a small installation may have access to the computer or even actually

operate it, but larger installations will severely restrict or forbid access to the computer. Job opportunities often coincide with metropolitan population areas and are relatively scarce in rural areas. Some programmers belong to a union, depending upon the type of company and geographic location, but the majority do not. The recognized professional organizations are the Association of Systems Management and the Data Processing Management Association, both are national in scope. The typical computer programmer works an eight a.m. to five p.m. day, although there is a hazard of late night telephone calls from the computer center when a program fails to work properly. It is also necessary for a programmer to work long hours of overtime so as to meet a deadline, often under intense stress and pressure from management (29, 67).

It does not require unusual physical strength or agility to be a programmer. Studies indicate that male and female programmers perform equally well, and there are many successful blind or handicapped programmers (47, 55).

In his book The Mythical Man-Month, Brooks (14) described the joys and woes of programming:

> Why is programming fun? What delights may its practitioner expect as his reward? First is the sheer joy of making things. As a child delights in his mud pie, so the adult enjoys building things, especially things of his own design. Second is the pleasure of making things that are useful to other people. Deep within, we want others to use our work and to find it helpful. Third is the facination of fashioning complex puzzle-like objects of interlocking moving parts and watching them work in subtle cycles, playing out the consequences

> of principles built in from the beginning. The programmed computer has all the facination of the pinball machine or the jukebox mechanism, carried to the ultimate. Fourth is the joy of always learning, which springs from the non-repeating nature of the task. In one way or another the problem is ever new, and its solver learns something new: sometimes practical, sometimes theoretical, and sometimes both. Finally, there is the delight of working in such a tractable medium. The programmer, like the poet, works only slightly removed from pure thought-stuff. He builds his castles in the air, from air, creating by exertion of the imagination. Few media of creation are so flexible, so easy to polish and rework, so readily capable of realizing grand conceptual structures. Yet the program construct, unlike the poet's words, is real in the sense that it moves and works, producing visible outputs separate from the construct itself. (p. 7)

It is important to recognize the stark reality of the real world, as many people associate the word "computer" with ideas of flashing lights and unquestionable accuracy. When in truth, nearly all business programs running presently contain an error, a bug, a flaw. The wrongness may be hidden and only surface under certain conditions at certain times, but none the less they are wrong (24, 36). Bunyan (17) explains: "We have reached the point now where our systems are too complicated for any individual to grasp their totality" (p. 87).

The working environment is usually one of stress, which is a result of the work being primarily problem solving. The occupation also has a good deal of negative aspects. Each day the programmer may have to deal with customers who are disgruntled; if they did not have a nagging problem, they would not need to talk to the programmer. There is usually

a demanding deadline, and as Brooks (14) commented:

> First, one must perform perfectly. The computer resembles the magic of legend in this respect too. If one character, one pause, of the incantation is not strictly in proper form, the magic does not work. Human beings are not accustomed to being perfect, and few areas of human activity demand it. Adjusting to the requirement of perfection is the most difficult part of learning to program. Next, other people set one's objectives, provide one's resources, and furnish one's information. One rarely controls the circumstances of his work or even its goals. (p. 9)

Programming managers have long recognized wide variations in productivity between good programmers and poor ones. Sackman, Erikson, and Grant (52) measured performances of a group of programmers and found the ratios between best and worst performance averaged about ten to one (indicating the $20,000 programmer may be ten times as productive as the $15,000 per year one, or even vice versa).

In his current study of programmer productivity, Johnson (32) noted that the average lifespan of a computer program is about seven years and during that lifespan, including initial development, about ninety percent of the cost of each program is for maintenance, changes, and repairs. Brooks (14) stated: "The fundamental problem with program maintenance is that fixing a defect has a fifty percent chance of producing another. So the whole process is two steps forward and one step back" (p. 122).

Lehman and Belady (39) studied productivity and found:

"All repairs tend to destroy the program structure, to increase the entropy and disorder of the system. Less and less effort is spent on fixing original design flaws; more and more time is spent on fixing flaws introduced by earlier fixes" (p. 61).

The half-life of a computer professional is defined (14) to be 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. The half-life of a computer professional 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.

Among the hazards of programming is the problem of exactness. The language tools available to a programmer are extremely demanding in their syntax rules (54). A misplaced comma, one period, or a misspelled word can be disasterous. At times the syntactical errors can be especially frustrating because they appear to be correct. That is, they may not be flagged by the language as errors, and therefore operate incorrectly yet undected. Figure 2 is an example of syntactical precision where instruction 1 is wrong and instruction 2 is correct. Both sentences appear to say the same thing; yet instruction 1 would never do

Instruction 1

If month is not less than 1 and greater than 12
go to process
else go to error.

Instruction 2

If month is not less than 1 and not greater than 12
go to process
else go to error.

Figure 2. Example of programming language precision

what it appears to say, and the language would not give the slightest indication anything is wrong.

Another unpleasant reality the programmer must face is the unbelievable array of computer languages in use. In her book Programming Languages Jean Sammet (53) described no less than 117 significant programming languages in use. Each language may have a different dialect for each computer manufacturer and each language may have a different syntax for each host operating system. In short, every programmer must expend as much or more effort in solving the syntax of the language, as in solving the target problem.

Aptitudes

The profession of computer programming is nearly universally agreed to be a frustrating mixture of art and science. An experiment by Shneiderman (57) reported that

programmers display an ability to divide a complex problem into parts. This ability is known to psychologists as "chunking" and involves grouping or organizing information into "chunks" which are as easy to handle as individual units.

Sackman and Gold (51) conducted a factor analytic research in which they studied 53 variables related to computer programming. Some of the main factors identified were: problem solving speed, exploitation of the computer system, exploration of alternatives, conceptualization ability, and problem solving strategy.

In his article on behavioral factors, Testa (62) said that creative people are field independent (able to separate figure from background). They can be identified using the Myers-Briggs Type Indicator (MBTI) of perception traits; are intuition-thinking or intuition-feeling, and tend to focus upon concepts and ideas rather than the environment.

Bell (7) conducted research to measure the aptitudes of programmer trainees so as to predict their future success. The investigation was a concurrent validation study of an established predictor test, namely, the Computer Programmer Aptitude Battery. The study systematically measured the frequency and type of errors committed by the subjects. The data on errors committed were then compared to the test variables. The results indicated no relationship between

the aptitude tests and programmer errors.

As Luftig (43) described desirable aptitudes: "In selecting programmer employees look for people with an aptitude for logical thinking and the exacting kind of analysis which is part of the job. The work also calls for patience, persistence, and ability to do the work with extreme accuracy. Ingenuity and imagination are particularly important" (p. 10).

It is also interesting to note that Weinberg (69) observed one necessary trait as being: "Someone without the ability to tolerate stressful situations for a period of a week or more is not good programmer material. It is unlikely that a programmer will go through a month without having to face the psychological shock of having his whole work pulled out from under him, or at least changed sufficiently so that his previous efforts become garbage" (p. 149)

Career Paths

In the past, short as the period may seem because the computer industry is less than a human generation old, the career path has traditionally been from computer operator to computer programmer to systems analyst (1, 23). The mature practioners could not possibly have had original university training in programming, so it was quite common

ten years ago to place a programmer in production after one week of training. In the beginning of computers, many programmers came out of the world's best science and engineering laboratories and applied a great deal of skill and ingenuity to the problems of those times. As the limited supply of really good programmers ran out, programming became more of a mix ranging from lower level clerical coders to a few prima donna superprogrammers (5).

As the computer industry matured and the functional areas became better established, the jobs became clearly differentiated from each other. Career progression at this time begins with the trainee entering the job from five main sources: on-the-job training, private vocational school, community college, university, and military training. Any given installation may have a mix of programmers from any or all of these sources all working side by side (1). The phenomenal growth of data processing has produced numerous pressures for practical accomplishments. The result has been that many businesses recruit programmers from wherever they can, stretching standards to an extreme. The further result from this growth pressure is that the profession of computer programmer has an unbelievably wide range of persons; some with very limited skills, some with brilliant skills. Very often management is unable to detect the good programmers from the bad, producing an endless cycle of

personnel problems (5).

The career path for a contemporary business programmer can extend vertically from a trainee to a programmer I, to programmer II, to programmer III as a person's level of competence increases. Beyond that point, begin levels of management from senior programmer to lead programmer to programming manager. There are also several horizontal branches, with considerable opportunity to jump from one branch to another. First, there is the computer operations branch where increasingly sophisticated hardware demand more and more employee knowledge of programming. Then, there is the branch of application programming where the business and industrial production occurs. Next, there is the systems software branch where specialists lend support with compilers and operating systems. Also there is the branch of systems analysis where company workflow and data processing are integrated. Finally, there is the career path from one company to another, exploiting the demand for programmers and transferability of computer skills (6, 33).

Competency Research

The review of literature revealed a limited number of research projects directly concerned with computer programmer competency analysis. Although few in number, the past research produced a wealth of ideas and information of great

value in formulating this research effort. The following discussion of the review of literature is not inclusive of all available but rather, a selection of research with particular significance to this topic.

In 1970 Emily Gloster (27) conducted a curriculum study of data processing in North Carolina. The curriculum was developed based upon a survey of four-year degree programs in the state. The survey instrument identified general data processing tasks, one on each page, with space for six items of criteria in performing the task:

1. Tools, equipment, materials.
2. Performance knowledge.
3. Safety hazards.
4. Decisions.
5. Cues.
6. Errors.

The study was conducted in sufficient detail to develop a general curriculum several years ago, but current critique of this research would pose several problems. First, that the data processing field has evolved into several specialized occupations that require unique sets of criteria for analysis. For instance, today it would be necessary to separate the data entry tasks from computer operations (10). Next, the research did not identify the needs of industry upon which to construct task statements (34).

An analysis of the content of computer technology with implications for technical education was conducted by Jordan (34) at the University of Missouri in 1969. The purpose of the study was to measure the extent of computer technology used by industry, and to identify the body of content related to computer technology. He found that the most frequent use of computers was business applications, and that the batch processing technique was used most often. The survey used a rating checklist sent to 34 computer specialists to identify 22 related blocks of competencies at 213 industries. This technique of organizing competencies into related blocks or groups was adapted into the present research instrument. Jordan concluded there were eleven items of content that received "essential" ratings and that those items should be emphasized in training programs. He also suggested that different types of computer technicians require significantly different competencies. He further suggested that computer curriculum content should be based upon the needs of industry in the geographic area.

An excellent model including criteria to be used when writing competency statements for a survey was found in a research project by Borcher and Joyner (10). Their business data processing occupational survey, completed in 1973, identified five job classifications: manager of data processing, supervisor of operations, systems analyst,

programmer, and computer operator. The purpose of the survey was to validate job descriptions for the job titles, to determine frequency of tasks performed, and to determine tasks common to all jobs within the data processing cluster. The research focused upon identifying 14 duty categories of 445 task statements. A sample of 38 data processing installations was randomly selected, with 406 survey forms returned. The analysis performed upon the data established a rank for each task depending upon whether it was performed by one job title or sets of job titles. One interesting observation found in this project was that a higher percentage of return was obtained by instructing the respondents to give the survey to their supervisor rather than mail it to the researcher.

A thorough discussion of the data processing technology curriculum was produced by Ashworth and Gebolys (4). Their guidebook presented a broad analysis of employment opportunities, faculty, laboratory, library, advisory committee, and curriculum considerations. It is an excellent source of data for someone who is establishing a new data processing program. Some specific items of curriculum content in the present research evolved from suggestions of Ashworth and Gebolys, particularly the items of personal qualities. The relationship between competency and curriculum is specified as being:

1. The training should prepare the graduate to be a productive employee in an entry level job.
2. The formal program, together with a reasonable amount of experience should enable the graduate to advance to positions of increasing responsibility.
3. The foundations provided by the training must be sufficiently comprehensive to enable the graduate to pursue further study.

In addition, a valuable suggestion was offered that student interest and motivation are enhanced by practical aspects of instruction. If the first term consists entirely of general subjects, students often lose interest. For a student who is enrolled to study programming it is important to begin actual training in this area immediately. Curriculum organization should be such that students may change vocational specialization without loss of credit. Also of interest was the suggestion of a special projects course for senior students. The projects should be planned and organized individually to provide maximum experience in a supervised realistic work environment.

A study conducted by Wier (70) compared the business data processing curriculum of 42 junior colleges from 27 states. The results of the study indicated a similarity among the types of courses and credit hours, and an identifiable pattern in curriculum content. The data were

analyzed by calculating standard deviations to determine the extent of variation from the mean. A small standard deviation, in terms of percent of the mean, indicated a homogenous pattern of subject matter. The courses found to be alike at the different schools were COBOL programming, FORTRAN programming, job control language, and accounting.

Berger (8) in his study of computer programmer job analysis devised five task categories:

1. Planning and analysis.
2. Program Development.
3. Program implementation.
4. Program testing.
5. Program support.

Each of the categories was then subdivided into design, methodology, and documentation task classifications. The survey was completed by 53 computer programming managers, all male, with an average of over ten years experience. The sample consisted of scientific, software, engineering, and business programmers. The sampling method determined there was no accurate census of the programmer population, and the best procedure was to sample individuals within important strata of the general population of programmers.

A task inventory questionnaire was used by Ammerman and Pratzner (3) in their 1974 programmer occupational survey report. The purpose of the study was to compare

worker and supervisor perceptions of programmer tasks. The results showed strong indications that supervisor expectations for the average programmer were much higher than actual performance of individual programmers. There were also many tasks being performed by programmers that the supervisors rated as being not needed. As the survey asked 12 questions about each of 474 tasks, it required an extreme amount of time on the part of the respondent to fill out the survey. The sample in the study consisted of 60 programmers and 40 supervisors who volunteered a minimum of four hours each to complete the questionnaire. If less than ten percent of the respondents performed a particular task, it was considered irrelevant and was subsequently dropped. A total of 161 tasks were dropped by applying the above criterion. The content of the questionnaires contained, in addition to programmer tasks, many task statements concerning department management, computer operations, systems analysis management, software programming, and scientific programming. Although the questionnaire contained many duplicate and overlaping tasks, it provided a basis for several items adaptable to the present research effort.

A report on the recommended minimum vocational/technical program objectives, published by the U.S. Office of Education (65), provided a model for task analysis. The method of identifying computer programming skills consisted

of a performance objective statement, the conditions of performance examination, and the criteria of successful performance. The skills in programming language were identified as COBOL, FORTRAN, ASSEMBLER, PL/1, and RPG. The items found in the report were stated at level of detail to be used in a course lesson plan rather than a competency survey. It should be noted however, that the general categories of skills proved useful in validating the present research instrument.

Toyne (63) conducted an evaluation of data processing training in Georgia in 1974. The purpose of the research was to provide behavioral objectives for data processing curriculum development. The study concluded:

1. Programmers need more training in the area of conceptual skills.
2. Computer installations use programming languages more to handle business activities rather than solve scientific problems.
3. The same skills are required even though core capicity and computer types differ.
4. Unit record equipment is being phased out.
5. There is significant difference between the skills needed inside and outside the greater metropolitan Atlanta area.

The 36 tasks identified by the study included building loops

within programs, reading sequential files, writing sub-programs, and testing programs.

Considerable curriculum structure information was gathered from the several literature sources of Berryman (9), Couger (21), Kearney (35), Peck (50), and Storer (60). The studies offered similar curriculum recommendations and were valuable in validating the item categories of computer hardware and language specifications in the present research instrument. The studies suggested that the technical content should be determined in relation to the types and size of business community being served, while general development content should address interpersonal communication skills.

A study by Bryant (16) presented five duties which were sub-divided into a number of tasks. The duties were:

1. Preparing data entry information.
2. Operating computers.
3. Supervising programming.
4. Maintaining a systems library.
5. Business applications.

For each task a two-page table was given showing tools used, equipment, materials, objects acted upon, decision knowledge, safety hazards, cues, and errors encountered in the job. This study provided useful ideas on the technique of decomposing general duties into discrete tasks because it also listed a broad range of criteria needed to recognize

relevant elements.

In an article on teaching computer science, Snyder (58) proposed a curriculum that helped validate the survey method used in the present research. He suggested including skills in the area of information structures, systems programming, systems analysis, and computer applications which are included in the present research instrument.

A 1976 research study by Lyon and Christiansen (44) of data processing curricula used a technique which restricted the number of choices to a four point scale. The rating scale was composed of must know, should know, desirable to know, and nice to know. The study was intended to determine if a correlation existed among the technicians, supervisors, and instructors perceptions of data processing tasks. The survey contained 25 task statements each for computer programmer, computer operator, and keypunch operator. The study revealed dichotomies in rankings that imply industry and education are sometimes out of phase on task ratings. Also, the study concluded that the technician may be the only person who can realistically rate or rank tasks which he actually performs.

The curriculum guidline of the Association for Computing Machinery by Little (40) offers many recommendations for a community college program. The report was based on the results of workshop discussions between educators,

industry, and professional society representatives held in 1975 and 1976. The report contained broad guidelines for establishing a computer programming curriculum including faculty selection, equipment needed, industrial relations, general education, and subject matter. The report offered caution against curriculum development problems such as the temptation to do only specific training for certain industries. One recommendation was to avoid the temptation to train a specialized technician who may be outdated with the next generation of equipment. The ability to transfer knowledge about a computer system and language to a different system will be necessary as technological changes continue over the next decade. Among the specific subject matter items offered in the report were:

1. Interpret specifications.
2. Analyze problems.
3. Plan detailed program logic.
4. Use a problem oriented procedure language.
5. Modify existing programs.
6. Verify and thoroughly test programs.
7. Prepare adequate documentation.
8. Use associated reference manuals.

The report also criticized the usual "open-door" policy of admission to data processing departments which result in high drop-out rates. The student's apparent lack of being

ready, lack of ability in logical thinking, and lack of persistence in handling details should be the target of pre-admission counseling.

The correlation between classroom performance and programming performance was found to be not significant in a 1977 study by Love (41). However, there was significant correlation of information processing abilities and programming performance. The study automatically included for analysis each program change submitted by 61 students. The 1500 programs collected were analyzed to identify the type of change and type of instruction; then interpreted as levels of programming performance to compare to classroom grades. The information processing abilities were determined by measuring memory, speed and accuracy, and information organization; then compared to individual programming performance. The study also conducted a laboratory experiment to measure student ability to understand programs. The experiment determined that a simplified program structure helped experienced programmers follow instruction flow, but had no effect on students.

A research study by Johnson (33) in 1977 focused upon job cluster specifications for computer operations personnel. The study determined the occupational definitions for job titles and performance standards by interviewing managers. The importance of worker traits and career path opportun-

ities were determined by surveying workers on the job site. Equipment used daily by workers and current data processing methods employed by industry were also surveyed. This study presented a thorough perspective of the computer operations job cluster in relation to the present data processing community.

In addition to the competency related research reviewed there was a study conducted in 1974 of computers used by industry. Burnett (18) concluded the distribution of computers on a percentage basis has remained remarkably stable since 1968. In manufacturing industries, computers have penetrated 1 in 25 plants to account for 38 percent of all installations. Public service industries make up 30 percent of the 59,000 installations in the United States. The number of computers found in education is nearly equal to all computers in all levels of government, and the number of computer specialists has grown to more than 300,000. The overall penetration of computers for users is considerably slower than the growth predicted by experts in 1969. The slower growth is probably due to mini-computers, software firm's capability, and service bureaus. The content of this study points to several stabilizing influences which would indicate that any results of research into computer programming competencies should remain relatively stable and reliable for the near future.

Task Analysis

The field of educational research contains many relevant textbooks, dissertations, and journal articles about research methodology. Although time, space, and focus prevent a complete review, a number of items were presented due to their particular significance in formulating this research model.

Mager and Beach (46), in their textbook on developing vocational education instruction, offered this practical suggestion on course development:

> The first step is describing in general terms that which someone does when performing the job. The second step is to describe job performance in finer detail, listing each of the tasks of which the job is composed and describing the steps in each of these tasks (task analysis). (p. 4)

A sample task listing sheet was presented, with columns for the task statement, frequency of performance, importance, and learning difficulty. This technique of using three criteria measurements with each task statement was adapted for use in the present research instrument.

In their textbook on educational research, Borg and Gall (11) presented a thorough discussion of the methods and tools used in survey research. The problem of selecting the sample is identified and suggestions to assure that adequate proportions of subgroups are represented by strat-

ified sampling. When the sampling unit is restricted by time and money available, the technique of cluster sampling may be used (73). Survey research often yields a type of normative description where the mean score of the entire sample on each attitude item is determined. In addition, a more interesting and complete analysis may be found by breaking the total sample down into subgroups by the differentiated description technique. The methods of constructing questionnaire items, attitude scale measurement, and pretesting considerations for the instrument are presented. Borg and Gall offered an excellent discussion of research bias and an extensive checklist that researchers may use to assure that thorough methodology is followed.

Another viable reference text on research methodology by Borgen and Davis (12) contained a handbook for completing surveys. A checklist of survey procedures was presented with suggestions for sampling, instrument construction, instrument evaluation, and data interpretion. An especially useful idea on data interpretation was also suggested: "When using a scaled instrument such as an importance or agreement scale, it is best to assign a range of values to each category. The reason for the range is that the data are really not discrete but rather represent a continuum; therefore the range of values is needed" (p. 81). In selecting the sample, the size should depend upon the extent

to which the selected individuals are representative of the population, the types of groups involved, and the method of data analysis. Also, the mail survey may introduce ambiguities because of respondent misinterpretation of items, incomplete returns, poorly answered items, and low rate of return. For surveys of career education programs, it is necessary to consider the amount of general education needed. It must be recognized that career education program focus is readiness for employment, and relevant general education should be included accordingly. Borgen and Davis further suggest a competency identification and analysis worksheet that contains columns for the competency item, frequency, importance, conditions, and criteria of application on the job.

Herschbach (30), in an article on deriving instructional content advised:

> Analysis of job tasks focuses primarily on task and task elements involved in work activity. The objective is to dissect job activity into its different skill and knowledge components in order to identify training content. ... The purpose of the task inventory step is to collect background information and develop a differentiated list of significant tasks performed by incumbent workers. (p. 63)

After the task inventory step is completed, the next step in content development should be to describe the actions, conditions, standards, and contingencies of job performance.

The question of which type of scale to use in the survey instrument was presented in the textbook by Wiersma (71):

> A Thurstone-type scale is more laborious to construct than a Likert-type scale. Both scales are susceptible to invalid self-report. However, the Thurstone-type is not affected by response-sets. The Likert-type may yield more information in that it is required to respond to all items. In studies that used comparable scales of both types, the correlations between the scores were found to be quite high, some as high as the .90's. On this basis, the scales may be considered interchangeable. (p. 210)

The problem of instrument validity is a primary concern of the attitude measurement survey. Any self-report device has the possibility of being faked; that is, the respondent may report attitudes which are quite different from his true feelings. It is difficult to find an external criterion with which to compare the reported attitude.

Gay (26) in the textbook Educational Research, suggested that reliability of the instrument for the survey be determined by split-half analysis:

> The coefficient of internal consistency is a type of reliability which is based on the internal consistency test. (1) administer the test to a group; (2) divide the test group into two comparable halves; (3) compute each subject's score on the two halves; (4) correlate the two sets of scores. If the coefficient is high, the test has good split-half reliability. (p. 94)

Reliability can also be expressed in terms of the standard error of measurement; a small standard error of measurement indicates high reliability and a large standard error of measurement indicates low reliability.

Review of Literature Summary

The review of literature was conducted to trace the historical development of business computer programming, to select appropriate task analysis methodology models, and to identify significant research in the area of computer programmer competency analysis.

The many sources of review aided greatly in the success of the present research effort; providing recognition of potential problems and establishment of proper procedures. An overall perspective of what the computer programmer job descriptors consist of, and the historical events that led to their formulation was attempted. The job of computer programmer was found to be relatively recent in origin, probably still in the formative stages, and anticipating future changes.

A description of the working environment was presented. Among the factors examined were: The satisfactions offered by the work, the stressful demands upon the incumbent, and some realities of imperfection that may be encountered in the job. In addition, the literature addressed the aptitudes required in computer programming as being logical thinking and stress tolerance. The career paths were described as providing opportunity for both horizontal progress and vertical expansion.

Several research reports in competency analysis and survey methodology were cited. Many of the reports touched on the periphery of the problem statement, yet as a whole contributed to the validity of this research. Some of the research reviewed was cited as being directly adapted to the questionnaire instrument design, the specific items of competency content, and the survey methodology development.

Although no one source contributed sufficient information to base the present research upon, as an integrated aggregate, the review of literature provided invaluable direction and congruence in approaching the stated problem.

METHOD AND PROCEDURE

The purpose of this study was to identify the competencies needed by business computer programmers in Iowa, and to identify discrepancies of industry need and existing curriculum content of the Computer Programming Department at Des Moines Area Community College. The procedures outlined in the following sections of this chapter were followed to fulfill the purpose.

Instrument

Several competency survey instrument formats were examined during the review of literature (49, 59, 64, 66). None of the existing survey instruments provided adequate application of the needed information. Therefore, a questionnaire was developed to satisfy the purpose of this study.

Based upon numerous task statement models found in the review of literature, extensive discussion with professional business programmers, and consultation with selected professors at Iowa State University a pilot questionnaire was designed. An inclusive list of business computer programmer competencies was compiled. The list of competencies was analyzed and organized into ten categories to eliminate overlaping and duplicate statements. The criteria used in constructing the competency statements appears in Appendix A.

The pilot instrument was reviewed by staff members of the Des Moines Area Community College Computer Programming Department and selected Iowa State University Computer Science faculty. The pilot survey was then administered to 23 employed business computer programmers to confirm the accuracy and clarity of the competency statements.

Each competency item was evaluated by the respondents on three criteria measures.

1. The degree of importance associated with the competency.
2. The level of difficulty attributed to the competency.
3. The frequency a particular competency must be performed.

In addition, the pilot instrument contained one competency statement primarily to indicate how carefully the respondent read the statements; it was not an actual computer programmer competency. All pilot respondents identified the fictitious statements as "not used", indicating a degree of internal consistency.

The criteria used to eliminate a competency statement from the pilot instrument was if 75 percent of the respondents identified the item as "not used". The pilot respondents also identified four additional competencies to be

included in the revised instrument. The category of "operate computer and peripheral equipment" was also deleted from the revised instrument.

The pilot instrument and pilot survey results were reviewed by the graduate committee members. Revision was made according to the committee member's comments and suggestions. Some of the important suggestions were:

1. Expand the Likert scale to five response levels.
2. Add a demographic question concerning the type of computer programmed.

The final instrument contained 62 competency statements within nine categories. The categories were:

1. Analyze data processing problems to devise and specify logical solutions.
2. Design computer programs and systems to implement problem solutions.
3. Write computer language programs to implement system specifications.
4. Test computer programs to prove system correctness.
5. Document system and program elements.
6. Understand computer hardware concepts and capabilities.
7. Understand computer software concepts and capabilities.

8. Understand business organization and standard procedures.
9. Develop professional and interpersonal qualities.

Sampling Plan

There was no known statewide population list of employed business computer programmers from which to draw the sample. Therefore, a list of employers was constructed from statewide membership lists in professional organizations, college placement office lists, and telephone books of metropolitan communities (73). The employer list was stratified by type of business to ensure usable proportions in the sample. A stratified random sample of 30 computer programmer employers was selected. Each employer was mailed a packet of six questionnaires with a personal memorandum asking that they be distributed to computer programmers under their supervision. In addition, a random sample of 50 employed business computer programmers, obtained from professional membership lists, were mailed the questionnaire with self-addressed stamped return envelopes.

Data Collection and Analysis

A total of 230 survey questionnaires were mailed. A relatively low percent of response was anticipated due to

the unavailability of accurate population lists.

Two weeks after the questionnaires were mailed, the number of returned responses had reached 96. A follow-up letter, with another questionnaire, was sent to those who had not returned the survey. As a result, a total of 139 complete responses were obtained, making the rate of return 60.1 percent.

Personal meetings were then held with Dr. Wolins and Dr. Warren of the Iowa State University Statistical Department to discuss data analysis. It was suggested that the most meaningful analysis of data would be obtained by:

1. Calculating the mean ratings of importance, difficulty, and frequency for each competency.
2. Constructing tables of competencies by rank of the mean values for importance, difficulty, and frequency.
3. Constructing scatter diagram plots to interpret the data relationships.

With the exception of two respondents who did not complete the questionnaire, all of the returned questionnaires were coded and keypunched into data cards, then processed by computer analysis.

The findings chapter presented an analysis of the

results of the competency survey. Tables which identified the competencies in rank order of importance, difficulty, and frequency were constructed and analyzed. The patterns of relationships were also investigated by analysis of scatter diagrams.

The discussion chapter presented a comparative analysis of the recommended curriculum content (derived from the survey results) with the actual curriculum content (displayed in Appendix F).

FINDINGS

The purpose of this study was to answer two preliminary questions:

1. What are the competencies perceived to be needed by entry level computer programmers?
2. Is the computer programming curriculum at Des Moines Area Community College congruent with identified needs of industry?

The first question is examined in the findings chapter, while the second question is considered in the discussion chapter.

Data were collected by a random sample survey of employed business computer programmers. The analyses of data were organized into seven sections. The sections were:

1. General Information on respondents.
2. Perceptions of Competency Importance.
3. Perceptions of Competency Difficulty.
4. Perceptions of Competency Frequency.
5. Relationship of Importance to Difficulty.
6. Relationship of Importance to Frequency.
7. Relationship of Difficulty to Frequency.

A summary of data analyses is presented at the end of the analysis sections.

General Information on Respondents

This section describes the survey findings of respondent characteristics concerning type of business, prior training, sex, age, number of employees, computer programmed and geographic location.

The final return of 139 questionnaires included 59 from businesses in the customer services category including banking, insurance, and publishing. There were 34 questionnaires returned from the consumer goods category which included manufacturing, retailing, and distribution industries. In addition, there were 46 returns from the public service categories of government, education, and utilities. Table 1 is a summary of the usable questionnaire returns by respondent type of business.

Of the 139 survey questionnaires returned, 43 were female programmers to account for 30.9 percent of the respondents. There were 96 questionnaires returned by male programmers, or 69.1 percent of the respondents. Table 2 is a summary of the survey returns by sex of the respondent.

The analysis of survey returns provided data concerning prior training experienced by the respondents. There were 6 questionnaires returned by programmers with a high school education. Of the respondents, 65 attended a vocational school, and 59 obtained training at a university. There

Table 1. Analysis of survey sample by respondent's type of business

Type of business	Number	Percent
Customer services	59	42.4
Consumer goods	34	24.5
Public service	46	33.1
Total	139	100.0

Table 2. Analysis of survey sample by respondent's sex

Sex	Number	Percent
Male	96	69.1
Female	43	30.9
Total	139	100.0

Table 3. Analysis of survey sample by respondent's prior training

Prior training	Number	Percent
High school	6	4.3
Vocational school	65	46.8
University	59	42.4
On the job	9	6.5
Total	139	100.0

were 9 respondents who experienced on the job training, and none of the respondents indicated they were trained while in military service. Table 3 is a summary of returns by prior training of the respondents.

The survey questionnaires were grouped by age of the respondent. Of the 139 returns, 26 were programmers between the age of 18 to 24. There were 79 programmers between the age of 25 to 31, and 31 programmers were between 32 to 38 years old. Only 3 of the respondents were 39 years or older. Table 4 is a summary of returns by age of the respondent.

The survey questionnaires were also grouped by the number of employees in the data processing department. The survey returns that represented small data processing installations numbered 54. There were 51 returns that represented medium size departments with 21 to 49 employees in data processing. Large departments with more than 50 employees accounted for 34 survey returns. Table 5 is a summary of the returns by number of employees in the data processing department.

The types of computers primarily programmed were grouped into small scale computers, medium scale computers, and large scale computers. Of the 139 survey returns, 43 were from installations with small scale computers, 26 had medium size computers, and 70 installations had large scale

Table 4. Analysis of survey sample by respondent's age group

Age group	Number	Percent
18 to 24	26	18.7
25 to 31	79	56.8
32 to 38	31	22.3
39 to 50	3	2.2
Total	139	100.0

Table 5. Analysis of survey by size of department

Size of department	Number	Percent
Small department	54	38.8
Medium department	51	36.6
Large department	34	24.6
Total	139	100.0

Table 6. Analysis of survey by respondent's computer

Computer	Number	Percent
Small scale	43	30.9
Medium scale	26	18.7
Large scale	70	50.4
Total	139	100.0

Table 7. Analysis of survey sample by respondent's location

Location	Number	Percent
Greater Des Moines area	88	63.3
Remainder of Iowa	51	36.7
Total	139	100.0

computers. Table 6 is a summary of the survey returns by type of computer programmed.

Table 7 is a summary of questionnaire returns by company location. There were 88 from metropolitan Des Moines area comprising 63.3 percent of the survey. The rest of Iowa was represented by 36.7 percent, or 51 returns.

Due to unavailable current population lists of employed business computer programmers, the results of this survey should be used with caution. The findings section which described the respondent characteristics may not be predictive of the population. The respondent analysis was included only to demonstrate that the survey sample was representative of the population surveyed.

Table 8 is a summary of the mean values for importance, difficulty, and frequency for each competency statement included in the survey questionnaire. The table is in sequence as the items appeared on the questionnaire.

Table 8. Mean values of importance, difficulty, and frequency for each competency

Item	Description	Importance	Difficulty	Frequency
1	Analyze data processing problems	4.00	3.24	4.08
1-a	Gather information	3.61	3.02	3.61
1-b	Analyze existing documentation	3.71	3.16	3.73
1-c	Determine impact of alternatives	3.65	3.48	3.40
1-d	Evaluate results of modification	4.06	3.10	3.85
1-e	Analyze problem cause	3.63	3.08	3.65
1-f	Prepare time and cost estimates	2.30	2.63	2.24
1-g	Define system error conditions	3.20	3.00	2.87
2	Design computer programs	3.81	3.34	3.26
2-a	Develop system flowcharts	3.24	2.61	2.71
2-b	Write system narratives	2.83	2.48	2.28
2-c	Develop program flowcharts	3.20	2.69	2.91
2-d	Write program specifications	3.24	2.73	2.97
2-e	Design data file structures	3.71	2.89	3.12
2-f	Design output forms	3.79	2.63	3.30
2-g	Use decision tables	2.02	2.10	1.95

Table 8. (Continued)

Item	Description	Importance	Difficulty	Frequency
2-h	Establish program requirements	3.02	2.69	3.02
2-i	Develop library routines	2.59	2.42	2.71
3	Write computer programs	4.02	2.91	3.77
3-a	Interpret program flowcharts	3.69	2.81	3.55
3-b	Code COBOL programs	4.00	2.71	3.97
3-c	Code ASSEMBLER programs	2.75	2.65	2.53
3-d	Code interactive programs	2.38	2.61	2.14
3-e	Develop routine libraries	3.12	2.32	3.02
3-f	Code sequential file routines	3.79	2.79	3.83
3-g	Code indexed file routines	3.30	2.85	3.26
3-h	Code table-look-up routines	3.65	2.91	3.51
3-i	Code internal sort routines	2.77	2.89	2.61
3-j	Study language manuals	3.53	3.04	3.26
4	Test computer programs	4.59	3.22	4.30
4-a	Develop test files	4.22	3.36	3.75
4-b	Create test transactions	4.28	3.30	3.79

Table 8. (Continued)

Item	Description	Importance	Difficulty	Frequency
4-c	Test all possible errors	4.28	3.34	3.73
4-d	Provide audit data	2.93	2.55	2.46
5	Document system and programs	4.06	2.81	3.44
5-a	Prepare operator instructions	4.24	2.55	3.73
5-b	Prepare program compilations	3.63	2.30	3.97
5-c	Prepare job stream statements	3.93	2.69	4.20
5-d	Write user procedures	3.18	2.71	2.55
5-e	Maintain data formats	3.63	2.57	3.34
5-f	Develop easy to follow logic	4.26	3.22	4.34
5-g	Train system user	3.14	2.83	2.51
6	Understand computer hardware	3.28	3.44	3.42
6-a	CPU physical storage	3.18	3.04	3.28
6-b	Tape storage capabilities	2.95	2.53	3.46
6-c	Disk storage capabilities	3.32	2.81	3.79
6-d	Teleprocessing device capabilities	2.69	2.61	2.65
6-e	Data entry capabilities	2.81	2.61	3.08

Table 8. (Continued)

Item	Description	Importance	Difficulty	Frequency
7	Understand computer software	3.28	3.24	3.38
7-a	Supervisor and interrupt concepts	2.44	2.75	2.57
7-b	Software utility packages	3.24	2.93	3.61
7-c	System library cataloging	3.24	2.57	3.53
8	Understand business procedures	3.42	3.02	3.87
8-a	Data processing user interaction	3.73	2.97	3.93
8-b	Business accounting practices	2.69	2.71	2.75
8-c	Business policy manuals	3.20	2.79	3.38
8-d	Compensation and promotion paths	3.28	2.61	2.69
9	Develop professional qualities	3.81	3.14	4.04
9-a	Establish job goals	3.95	3.26	3.61
9-b	Interact with user and management	4.32	3.28	4.65
9-c	Develop problem solving technique	4.30	3.40	4.48
9-d	Develop personal initiative	4.34	3.26	4.59

Perceptions of Competency Importance

The data collected included a quantified perception of how important a respondent indicated each competency statement was for entry level programmers. The units of measurement used were:

1 = Not used

2 = Least important

3 = Important

4 = Very important

5 = Essential

A summary of competency statements in descending rank order by mean value of importance appears in Table 9. There were 12 competencies identified by employed programmers as "essential" to entry level job performance. The criteria of a competency being interpreted as essential was determined to be those items with a mean value in excess of 4.00. The category of testing programs was measured as most important, followed by the category of personal development.

In addition, table 9 indicates none of the competencies were identified as "not needed" by entry level programmers, while 13 were identified as "least important." The competency presenting the lowest rating of importance was 2-g, "use decision tables."

Table 9. Rank order of competency statements by mean value of importance

Rank	Item	Description	Importance
1	4	Test computer programs	4.59
2	9-d	Develop personal initiative	4.34
3	9-b	Interact with user and management	4.32
4	9-c	Develop problem solving technique	4.30
5	4-b	Create test transactions	4.28
6	4-c	Test all possible errors	4.28
7	5-f	Develop easy to follow logic	4.26
8	5-a	Prepare operator instructions	4.24
9	4-a	Develop test files	4.22
10	1-d	Evaluate results of modification	4.06
11	5	Document system and programs	4.06
12	3	Write computer programs	4.02
13	1	Analyze data processing problems	4.00
14	3-b	Code COBOL programs	4.00
15	9-a	Establish goals	3.95
16	5-c	Prepare job stream statements	3.93
17	9	Develop professional qualities	3.81
18	2	Design computer programs	3.81
19	2-f	Design output forms	3.79
20	3-f	Code sequential file routines	3.79
21	8-a	Data processing user interaction	3.73

Table 9. (Continued)

Rank	Item	Description	Importance
22	1-b	Analyze existing documentation	3.71
23	2-e	Design data file structures	3.71
24	3-a	Interpret program flowcharts	3.69
25	1-c	Determine impact of alternatives	3.65
26	3-h	Code table-look-up routines	3.65
27	1-e	Analyze problem cause	3.63
28	5-b	Prepare program compilations	3.63
29	5-e	Maintain data formats	3.63
30	1-a	Gather information	3.61
31	3-j	Study language manuals	3.53
32	8	Understand business procedures	3.42
33	6-c	Disk storage capabilities	3.32
34	3-g	Code indexed file routines	3.30
35	6	Understand computer hardware	3.28
36	7	Understand computer software	3.28
37	8-d	Compensation and promotion paths	3.28
38	2-a	Develop system flowcharts	3.24
39	2-d	Write program specifications	3.24
40	7-b	Software utilitiy packages	3.24
41	7-c	System library cataloging	3.24
42	1-g	Define system error conditions	3.20

Table 9. (Continued)

Rank	Item	Description	Importance
43	2-c	Develop program flowcharts	3.20
44	8-c	Business policy manuals	3.20
45	5-d	Write user procedures	3.18
46	6-a	CPU physical storage	3.18
47	5-g	Train system user	3.18
48	3-e	Develop routine libraries	3.12
49	2-h	Establish program requirements	3.02
50	6-b	Tape storage capabilities	2.95
51	4-d	Provide audit data	2.93
52	2-b	Write system narratives	2.83
53	6-e	Data entry capabilities	2.81
54	3-i	Code internal sort routines	2.77
55	3-c	Code ASSEMBLER programs	2.75
56	6-d	Teleprocessing device capabilities	2.69
57	8-b	Business accounting practices	2.69
58	2-i	Develop library routines	2.59
59	7-a	Supervisor and interrupt concepts	2.44
60	3-d	Code interactive programs	2.38
61	1-f	Prepare time and cost estimates	2.30
62	2-g	Use decision tables	2.02

Perceptions of Competency Difficulty

The survey questionnaire collected data measuring each respondent's opinion of how difficult each competency was for entry level programmers to perform. The units of measurement were:

1 = Not used

2 = Easy to perform

3 = Somewhat hard to perform

4 = Difficult to perform

5 = Very difficult perform

A summary of competency statements in descending rank order by mean value of difficulty appears in Table 10. The criteria of mean value in excess of 3.00 was interpreted as being "difficult." There were 22 competency statements identified as being "difficult." The categories of competencies which were identified as being more difficult to perform than others were:

1-c Determine the impact of alternative solutions

6 Understand computer hardware

9-c Develop problem solving techniques

4-a Develop test files

It is significant to note that none of the competency statements were identified as "very difficult", also none of the competencies were identified by the mean values as being "not used."

Table 10. Rank order of competency statements by mean value of difficulty

Rank	Item	Description	Difficulty
1	1-c	Determine impact of alternatives	3.48
2	6	Understand computer hardware	3.44
3	9-c	Develop problem solving technique	3.40
4	4-a	Develop test files	3.36
5	2	Design computer programs	3.34
6	4-c	Test all possible errors	3.34
7	4-b	Create test transactions	3.34
8	9-b	Interact with user and management	3.28
9	9-a	Establish job goals	3.26
10	9-d	Develop personal initiative	3.26
11	1	Analyze data processing problems	3.24
12	7	Understand computer software	3.24
13	4	Test computer programs	3.22
14	5-f	Develop easy to follow logic	3.22
15	1-b	Analyze existing documentation	3.16
16	9	Develop professional qualities	3.14
17	1-d	Evaluate results of modification	3.10
18	1-e	Analyze problem cause	3.08
19	3-j	Study language manuals	3.04
20	6-a	CPU physical storage	3.04
21	1-a	Gather information	3.02

Table 10. (Continued)

Rank	Item	Description	Difficulty
22	8	Understand business procedures	3.02
23	1-g	Define system error conditions	3.00
24	8-a	Data processing user interaction	2.97
25	7-b	Software utility packages	2.93
26	3	Write computer programs	2.91
27	3-h	Code table-look-up routines	2.91
28	2-e	Design data file structures	2.89
29	3-i	Code internal sort routines	2.89
30	3-g	Code indexed file routines	2.85
31	5-g	Train system user	2.83
32	3-a	Interpret program flowcharts	2.81
33	5	Document system and programs	2.81
34	6-c	Disk storage capabilities	2.81
35	3-f	Code sequential file routines	2.79
36	8-c	Business policy manuals	2.79
37	7-a	Supervisor and interrupt concepts	2.75
38	2-d	Write program specifications	2.73
39	3-b	Code COBOL programs	2.71
40	5-d	Write user procedures	2.71
41	8-b	Business accounting practices	2.71
42	2-c	Develop program flowcharts	2.69

Table 10. (Continued)

Rank	Item	Description	Difficulty
43	2-h	Establish program requirements	2.69
44	5-c	Prepare job stream statements	2.69
45	3-c	Code ASSEMBLER programs	2.65
46	1-f	Prepare time and cost estimates	2.63
47	2-f	Design output forms	2.63
48	2-a	Develop system flowcharts	2.61
49	3-d	Code interactive programs	2.61
50	6-d	Teleprocessing device capabilities	2.61
51	6-e	Data entry capabilities	2.61
52	8-d	Compensation and promotion paths	2.61
53	5-e	Maintain data formats	2.57
54	7-c	System library cataloging	2.57
55	4-d	Provide audit data	2.55
56	5-a	Prepare operator instructions	2.55
57	6-b	Tape storage capabilities	2.53
58	2-b	Write system narratives	2.48
59	3-e	Develop routine libraries	2.42
60	2-i	Develop library routines	2.32
61	5-b	Prepare program compilations	2.30
62	2-g	Use decision tables	2.10

Perceptions of Competency Frequency

A quantified opinion of how frequently each competency must be performed by entry level programmers was included in the questionnaire. Each competency statement was measured by the following increments:

1 = Not used

2 = Seldom used

3 = Used monthly

4 = Used weekly

5 = Used daily

A summary of the questionnaire results in rank order by descending mean value of each competency frequency is presented in Table 11. The criteria of mean values in excess of 4.00 was interpreted as a competency "used daily." The survey findings included eight competencies as being used daily, while 36 were identified as being used more often than weekly. The competencies identified as being used most frequently were:

9-b Interact with user

9-c Develop problem solving techniques

9-d Develop personal initiative

4 Test computer programs

5-f Develop easy to follow logic

5-c Prepare job stream statements

Table 11. Rank order of competency statements by mean value of frequency

Rank	Item	Description	Frequency
1	9-b	Interact with user and management	4.65
2	9-d	Develop personal initiative	4.59
3	9-c	Develop problem solving technique	4.48
4	5-f	Develop easy to follow logic	4.34
5	4	Test computer programs	4.30
6	5-c	Prepare job stream statements	4.20
7	1	Analyze data processing problems	4.08
8	9	Develop professional qualities	4.04
9	3-b	Code COBOL programs	3.97
10	5-b	Prepare program compilations	3.97
11	8-a	Data processing user interaction	3.93
12	8	Understand business procedures	3.87
13	1-d	Evaluate results of modification	3.85
14	3-f	Code sequential file routines	3.83
15	4-b	Create test transactions	3.79
16	6-c	Disk storage capabilities	3.79
17	3	Write computer programs	3.77
18	4-a	Develop test files	3.75
19	1-b	Analyze existing documentation	3.73
20	4-c	Test all possible errors	3.73
21	5-a	Prepare operator instructions	3.73

Table 11. (Continued)

Rank	Item	Description	Frequency
22	1-e	Analyze problem cause	3.65
23	1-a	Gather information	3.61
24	7-b	Software utility packages	3.61
25	9-a	Establish job goals	3.61
26	3-a	Interpret program flowcharts	3.55
27	7-c	System library cataloging	3.53
28	3-h	Code table-look-up routines	3.51
29	6-b	Tape storage capabilities	3.46
30	5	Document system and programs	3.44
31	6	Understand computer hardware	3.42
32	1-c	Determine impact of alternatives	3.40
33	7	Understand computer software	3.38
34	8-c	Business policy manuals	3.38
35	5-e	Maintain data formats	3.34
36	2-f	Design output forms	3.30
37	6-a	CPU physical storage	3.28
38	2	Design computer programs	3.26
39	3-g	Code indexed file routines	3.26
40	3-j	Study language manuals	3.26
41	2-e	Design data file structures	3.12
42	6-e	Data entry capabilities	3.08

Table 11. (Continued)

Rank	Item	Description	Frequency
43	2-h	Establish program requirements	3.02
44	3-e	Develop routine libraries	3.02
45	2-d	Write program specifications	2.97
46	2-c	Develop program flowcharts	2.91
47	1-g	Define system error conditions	2.87
48	8-b	Business accounting practices	2.75
49	2-a	Design computer programs	2.71
50	2-i	Develop library routines	2.71
51	8-d	Compensation and promotion paths	2.69
52	6-d	Teleprocessing device capabilities	2.65
53	3-i	Code internal sort routines	2.61
54	7-a	Supervisor and interrupt concepts	2.57
55	5-d	Write user procedures	2.55
56	3-c	Code ASSEMBLER programs	2.53
57	5-g	Train system user	2.51
58	4-d	Provide audit data	2.46
59	2-b	Write system narratives	2.28
60	1-f	Prepare time and cost estimates	2.24
61	3-d	Code interactive programs	2.14
62	2-g	Use decision tables	1.95

1 Analyze data processing problems

9 Develop professional qualities

The competency identified as being used less frequently than any other listed on the questionnaire was 2-g, use decision tables.

Several methods of data analysis were considered in this study. One method of data analysis was suggested by Dr. Wolins of the Iowa State University Statistical Department, and was explained in the textbook by Willemsen (72), Understanding Statistical Reasoning. When the variables under study are measured with many values, and representing their relationship is complex, a scatter diagram may be used.

However, analysis of difference among various subgroups of the survey sample was not attempted. One purpose of the study was to identify programmer competencies needed by the computer programming industry as a whole. Although there may be variations in perceived values of competencies among various subgroups of the population, the curriculum must be directed towards satisfying the common needs of all groups served. It was not the purpose of this study to identify the special competency interests of subgroups, but rather to identify competencies which would enable students to secure employment in any of the industries employing computer programmers.

Relationship of Importance and Difficulty

An important consideration in curriculum design is to identify those competencies needed by industry that are both important to job success and also difficult to perform. The identification of such competency needs allows proper emphasis of material in the educational delivery system. The data were therefore analyzed and the mean values of importance and difficulty for each competency were plotted on the scatter diagram presented in Figure 3. Due to insufficient space only those items set sufficiently apart from the main pattern are identified on the graph. It is significant to note that the graph appears to present a noticeable relationship between those competencies which are more important and also those which are more difficult to perform. None of the competencies were presented as being difficult but not important. The one item presented as being neither important nor difficult was 2-g, use decision tables. The outermost items that may be interpreted as being important but not difficult were:

5-a Prepare operator instructions

5-b Prepare program compilations

3-e Develop library routines

The graph also presents two competencies as being important and somewhat difficult:

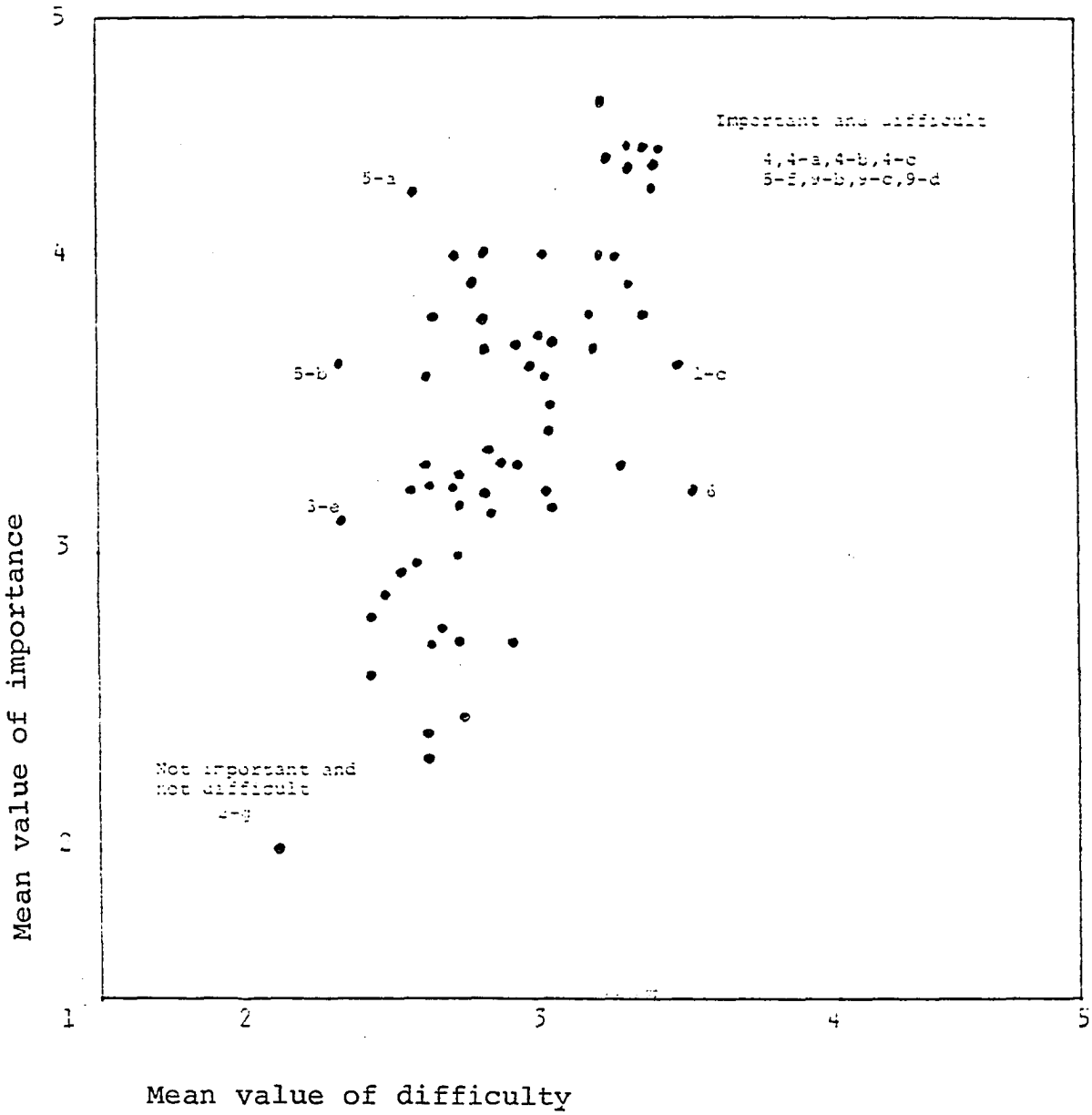


Figure 3. Relationship of importance and difficulty

1-c Determine the impact of alternative solutions

6 Understand computer hardware

There were eight competencies identified by the graph as being both essential and also difficult to perform, they were:

4 Test computer programs

4-a Develop test files

4-b Create test transactions

4-c Test all possible errors

5-f Develop easy to follow logic

9-b Interact with user

9-c Develop problem solving techniques

9-d Develop personal initiative

It should be noted that Figure 3 also presented graphic evidence that the overall list of competency statements were relatively grouped together, rather than spread out, which might indicate a consistency to be considered in curriculum planning.

Relationship of Importance and Frequency

Another substantial consideration in curriculum design is to identify those competencies that are both important to job success and also performed frequently. The survey data were analyzed and the mean values of competency importance and frequency were plotted on the scatter diagram presented in Figure 4. The figure presents a distinct linear re-

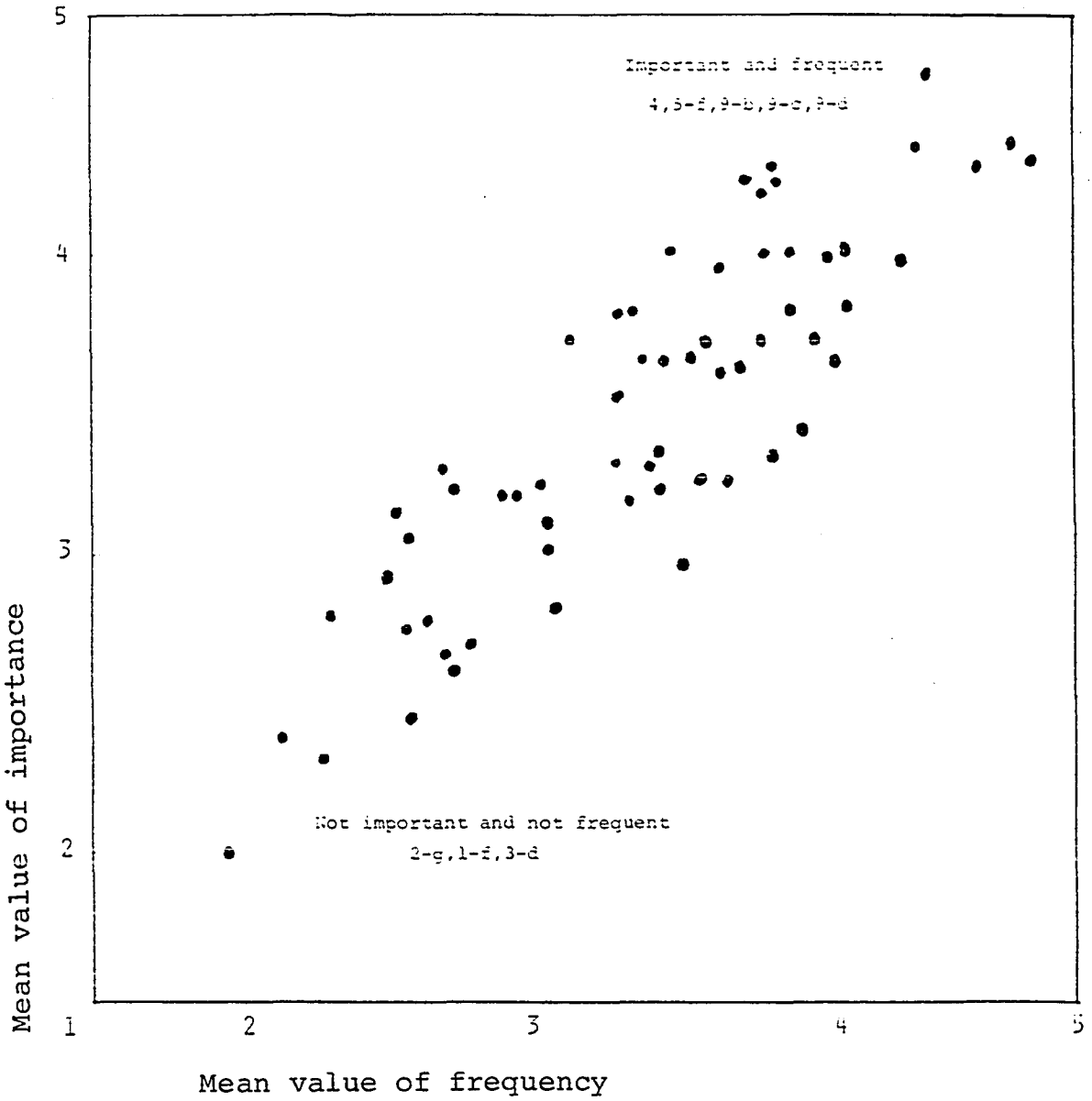


Figure 4. Relationship of importance and frequency

lationship between the importance of a competency and the frequency which that competency must be performed. The graphic data of where competencies do not appear on the diagram may be as significant to curriculum planning as where the competencies do appear. For instance, the fact that Figure 4 does not present any competencies in the lower right hand corner indicated there were no competencies used frequently which were not important. Also there were no competencies identifed as important but not used frequently.

The outermost competencies identified in figure 4 as being relatively more important and also used daily were:

4 Test computer programs

5-f Develop easy to follow logic

9-b Interact with user

9-c Develop problem solving techniques

9-d Develop personal initiative

The diagram also presents three competencies as being neither important nor used frequently, they were:

1-f Prepare time and cost estimates

3-d Code interactive programs

2-g Use decision tables

Relationship of Difficulty and Frequency

Another aid in curriculum design is information concerning how difficult each competency is to perform in

relation to how frequently it is done. The mean values of difficulty and frequency were plotted for each competency in Figure 5 as a graphic summary of the survey responses. the scatter diagram shows the competencies that should be considered as used daily and also somewhat difficult to perform were:

4 Test computer programs

5-f Develop easy to follow logic

9-b Interact with user

9-c Develop problem solving techniques

9-d Develop personal initiative

The diagram also identified the outermost competency 5-b, prepare program compilations, as being used frequently but not difficult to perform. The competency 3-e, develop routine libraries, is used less frequently but also is not difficult to perform. Special note should be made that the competency 2-g, use decision tables, appears on the graph as being neither very difficult to perform not used very often.

Summary of Findings

The survey yielded 139 usable questionnaire returns which were tabulated by a computer program to provide information meaningful to the purpose of the study. The data were analyzed from several perspectives which were then

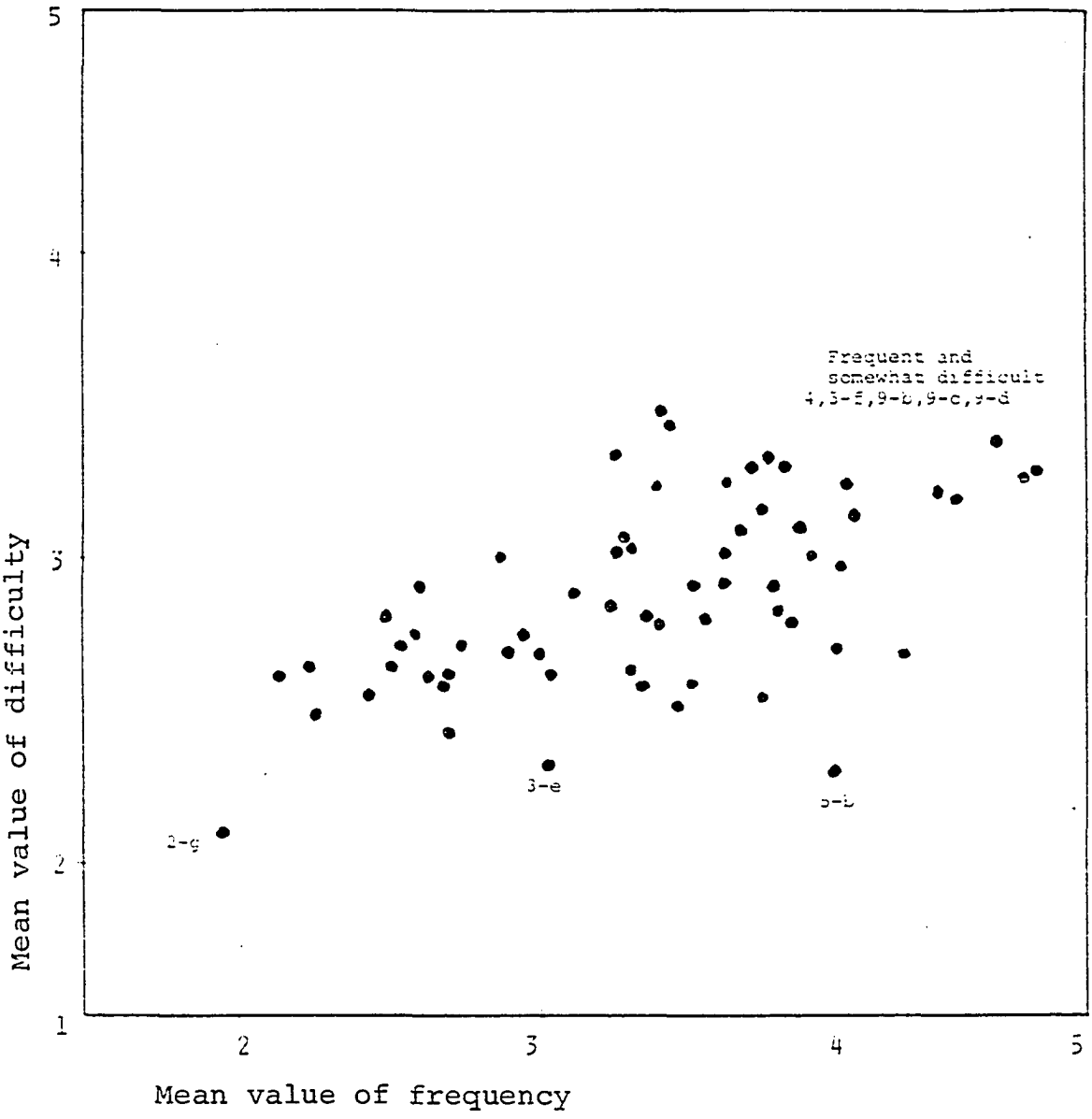


Figure 5. Relationship of difficulty and frequency

further discussed in this chapter of the study.

First, the data analysis provided general information concerning the survey respondents. The survey sample contained reasonable proportions as to type of business, sex, prior training, age, size of department, size of computer, and geographic location of employment.

Secondly, the mean values of importance, difficulty, and frequency were calculated for each competency statement, then presented in Table 8. In addition, the competencies were sorted into three separate rank orders. Table 9 displays the competencies in rank order by importance, Table 10 presents rank order by difficulty, and Table 11 displays rank order by how frequently the competency is used.

Finally, the patterns of relationship were also investigated. A scatter diagram showing the relationship of importance and difficulty was presented in Figure 3. A scatter diagram displaying the relationship of importance and frequency was presented in Figure 4. Then, a scatter diagram showing the relationship of difficulty and frequency was presented in Figure 5. Each of the scatter diagrams show a relatively close grouping of the competencies as a whole. Each scatter diagram also displayed a few outlying competencies which deviate from the overall graph pattern. These outlying competencies, due to their unique attributes, deserve critical attention to give them

proper emphasis in curriculum content planning. It should be noted also, that the scatter diagrams present a few competencies consistently as being both important, difficult, and frequent. Closer examination of Tables 9, 10, and 11 revealed wider variations in the rank order of each competency than might be assumed from the scatter diagrams. For instance, questionnaire item 3, write computer programs, appeared in rank position 12 of importance, then in rank position 26 of difficulty, and then in rank position 17 of frequency.

The significant findings included several items and relationships. First, the data analysis identified those competencies which were perceived by industry to be essential to entry level job performance. Such information is critical to correct curriculum design. Second, the data analysis identified those competencies perceived by industry as being relatively more difficult to perform than others. It should be noted that the industry programmers' perspectives were much lower for difficulty than they were for either importance or frequency. Such opinion may contrast sharply with students or educators. Third, the data analysis identified those competencies perceived by employed programmers as being used most frequently.

The most important competency was identified as category 4, test computer program to prove system correctness.

The most difficult competency was identified as 1-c, determine the impact of alternative solutions upon the total system.

The competency identified as being used most frequently was 9-b, interact with user, co-worker, management, and non-computer personnel in a productive manner.

The data also identified competency 2-g, use decision tables as being least important, least difficult, and used less frequently than other competencies.

DISCUSSION

One of the purposes of the study was to identify discrepancies between industry need and existing curriculum content of the Computer Programming Program at Des Moines Area Community College. The findings chapter presented the results of a random survey of employed computer programmers and of an analysis of the survey data regarding competency needs. This chapter presents an analysis of curriculum content and comparison with identified needs.

The discussion chapter was organized to answer two questions:

1. Does the existing curriculum at Des Moines Area Community College lack emphasis of needed competencies?
2. Does the existing curriculum at Des Moines Area Community College contain excess emphasis on needed competencies?

Proceeding with the philosophy that curriculum content should be based upon identified need, the following discussion focused upon the nine competency categories presented in the survey questionnaire. Each competency was analyzed by comparing the identified survey findings with the curriculum content emphasis at the Des Moines Area Community College Computer Programming Program.

In order to obtain an objective measure of the curriculum content it was necessary to devise a method of appraising the curriculum emphasis on the specified competencies. Dr. Milton Brown of the Iowa State University Professional Studies Department was consulted on the problem of curriculum content appraisal. A plan for the assessment of curriculum content emphasis was formulated. The curriculum content appraisal consisted of seven steps:

1. Review the summary of course names and credit hours offered at the Des Moines Area Community College Computer Programming Program. The summary is presented in Appendix D.
2. Read the Des Moines Area Community College Computer Programming course descriptions. The course descriptions are presented in Appendix E.
3. Define the criteria to measure curriculum content emphasis. For the purpose of this study, committee members Dr. Wolansky and Dr. Brown suggested that the instructional-unit method was the proper approach to measure the curriculum content emphasis on the competencies identified. An instructional-unit was defined to be equal to one class hour that the student is exposed to a specific competency during a lecture session or is required to apply a competency skill during a laboratory assignment. It was suggested that the criteria for assessing

instructional-unit not be considered for those competencies which occupy a minor or trivial portion of the lesson. Therefore, the instructional-unit values were applied only to those competencies which comprised a significant part of a lesson. The range of instructional-unit values which may be assigned to a given competency was dependent upon the length of the course and how often a course was conducted each week. For instance, if the duration of a course was ten weeks and if the course was conducted three times a week, the maximum instructional-units that could be appraised to a given competency would be 30 units. However, a course might not emphasize a given competency every day of every week. Therefore, the range of instructional-units was less than the maximum instructional hours possible.

4. Review the Des Moines Area Community College Computer Programming course outlines and conduct discussions with the programming instructional staff to analyze the course outlines.
5. Conduct discussions with the Des Moines Area Community College Computer Programming staff to appraise which competencies were emphasized in each course

and to assess the instructional-units to be assigned each competency in each course.

6. Construct a summary list consisting of the name of the course followed by the competencies which were emphasized in each course and the instructional-units assessed each competency. The summary of the instructional-unit assessment is presented in Appendix F.
7. Construct tables that summarize the total instructional-units assessed each competency. The total instructional-unit table revealed the actual curriculum emphasis and was used as a basis for comparison discussions.

The problem of defining the criteria to apply in deciding whether the curriculum content was excessive, adequate, or deficient was then considered. Due to the complexity of the relationships between the curriculum content emphasis and the importance, difficulty, and frequency of competencies, committee members were consulted for advise. For the purpose of this study, it was suggested that the analysis most meaningful to student learning should focus upon the level of difficulty attributed to each competency. Therefore, the discussion which follows was based upon analysis of the assessed instructional-units for each competency compared to the recommended instructional-units which

were derived from the competency level of difficulty.

Further discussions were conducted with the Des Moines Area Community College Computer Programming instructional staff to recommend a range of instructional-units to apply to levels of competency difficulty. The recommendation was to consider the level of difficulty for each competency by surveyed quarter rank. Table 10 was referenced to apply the criteria. For those competencies ranking in the upper 25 percent, a range of 35 to 50 instructional-units was recommended. Those competencies that rank in the second quarter were recommended a range of 20 to 34 instructional-units. The third quarter of competencies was recommended a range of 10 to 19 instructional-units. The lowest ranking 25 percent of the competencies were recommended a range of 1 to 9 instructional-units. Table 12 presents a summary of the recommended instructional-units for each level of difficulty surveyed.

Table 12. Summary of recommended instructional-unit ranges

Quarter	Difficulty Rank	Recommended Units
1	1 to 15	35 to 50
2	16 to 30	20 to 34
3	31 to 45	10 to 19
4	46 to 62	1 to 9

In addition, Table 13 was constructed to facilitate the analysis of the curriculum content emphasis of each competency compared to the recommended emphasis. If the assessed instructional-units for a competency were more than the recommended instructional-units, it was interpreted that the curriculum emphasis exceeded the identified need of industry. If the assessed instructional-units for a competency were fewer than the instructional-units recommended, it was interpreted that the curriculum was not sufficient to meet the need of entry level computer programmers.

It should be noted that the industry programmer competency needs survey was based upon the opinion of the respondents, and that the curriculum appraisal was based upon the judgement of the instructional staff. The following discussion should therefore be interpreted within the limitations and assumptions specified for this research.

Competency Category One

This competency category involved the analysis of data processing problems. The survey findings identified this competency category as ranking 13 in importance, 11 in difficulty, and 7 in frequency.

Examination of Table 13 revealed that four competencies lacked sufficient emphasis on curriculum content based upon the analysis of difficulty. The competencies were:

Table 13. Summary of assessed and recommended instructional-units

Item	Description	Difficulty Rank	Quartile	Assessed Units	Recommended Units
1	Analyze data processing problems	11	1	41	35 - 50
1-a	Gather information	21	2	15	20 - 34
1-b	Analyze existing documentation	15	1	14	35 - 50
1-c	Determine impact of alternatives	1	1	26	35 - 50
1-d	Evaluate results of modification	17	2	28	20 - 34
1-e	Analyze problem cause	18	2	14	20 - 34
1-f	Prepare time and cost estimates	46	4	1	1 - 9
1-g	Define system error conditions	23	3	19	10 - 19
2	Design computer programs	5	1	56	35 - 50
2-a	Develop system flowcharts	48	4	23	1 - 9
2-b	Write system narratives	58	4	12	1 - 9
2-c	Develop program flowcharts	42	3	34	10 - 19
2-d	Write program specifications	38	3	20	10 - 19
2-e	Design data file structures	28	2	27	20 - 34
2-f	Design output forms	47	4	33	1 - 9

Table 13. (Continued)

Item	Description	Difficulty Rank	Quartile	Assessed Units	Recommended Units
2-g	Use decision tables	62	4	1	1 - 9
2-h	Establish program requirements	43	3	5	10 - 19
2-i	Develop library routines	60	4	11	1 - 9
3	Write computer programs	26	2	108	20 - 34
3-a	Interpret program flowcharts	32	3	18	10 - 19
3-b	Code COBOL programs	39	3	25	10 - 19
3-c	Code assembler programs	45	3	51	10 - 19
3-d	Code interactive programs	49	4	14	1 - 9
3-e	Develop routine libraries	59	4	8	1 - 9
3-f	Code sequential file routines	35	3	11	10 - 19
3-g	Code indexed file routines	30	2	8	20 - 34
3-h	Code table-look-up routines	27	2	7	20 - 34
3-i	Code internal sort routines	29	2	2	20 - 34
3-j	Study language manuals	19	2	61	20 - 34
4	Test computer programs	13	1	35	35 - 50
4-a	Develop test files	4	1	18	35 - 50

Table 13. (Continued)

Item	Description	Difficulty Rank	Quartile	Assessed Units	Recommended Units
4-b	Create test transactions	7	1	12	35 - 50
4-c	Test all possible errors	6	1	10	35 - 50
4-d	Provide audit data	55	4	10	1 - 9
5	Document system and programs	33	3	25	10 - 19
5-a	Prepare operator instructions	56	4	4	1 - 9
5-b	Prepare program compilations	61	4	5	1 - 9
5-c	Prepare job stream statements	44	3	30	10 - 19
5-d	Write user procedures	40	3	17	10 - 19
5-e	Maintain data formats	53	4	1	1 - 9
5-f	Develop easy to follow logic	14	1	15	35 - 50
5-g	Train system user	31	2	12	20 - 34
6	Understand computer hardware	2	1	14	35 - 50
6-a	CPU physical storage	20	2	6	20 - 34
6-b	Tape storage capabilities	57	4	5	1 - 9
6-c	Disk storage capabilities	34	3	18	10 - 19
6-d	Teleprocessing device capabilities	50	4	9	1 - 9

Table 13. (Continued)

Item	Description	Difficulty Rank	Quartile	Assessed Units	Recommended Units
6-e	Data entry capabilities	51	4	3	1 - 9
7	Understand computer software	12	1	20	35 - 50
7-a	Supervisor and interrupt concepts	37	3	10	10 - 19
7-b	Software utility packages	25	2	11	20 - 34
7-c	System library cataloging	54	4	23	1 - 9
8	Understand business organization	22	2	22	20 - 34
8-a	Data processing user interaction	24	2	21	20 - 34
8-b	Business accounting practices	41	3	50	10 - 19
8-c	Business policies and procedures	36	3	13	10 - 19
8-d	Compensation and promotion	52	4	2	1 - 9
9	Develop professional qualities	16	2	1	20 - 34
9-a	Establish job goals	9	1	9	35 - 50
9-b	Interact with user and management	8	1	14	35 - 50
9-c	Develop problem solving techniques	3	1	12	35 - 50
9-d	Develop personal initiative	10	1	3	35 - 50

1-a Gather information

1-b Analyze existing documentation

1-c Determine impact of alternatives

1-e Analyze problem cause

None of the competencies in this category exceeded the recommended emphasis. The remaining four competencies were interpreted to have received curriculum content emphasis consistent with the survey rankings. The competencies were:

1 Analyze data processing problems

1-d Evaluate results of modification

1-f Prepare time and cost estimates

1-g Define system error conditions

Competency Category Two

This competency category involved the design of computer programs to implement systems. The category was identified by the survey as ranking 18 in importance, 5 in difficulty, and 38 in frequency.

Review of Table 13 disclosed that competency 2-h, establish program requirements, was deficient in curriculum emphasis. The table displayed seven competencies which were assessed instructional-units in excess of the recommended range. They were:

2 Design computer programs

2-a Develop system flowcharts

2-b Write system narratives

2-c Develop program flowcharts

2-d Write program specifications

2-f Design output forms

2-i Develop routine libraries

The remaining items in category two were found to receive adequate curriculum emphasis. Those competencies included:

2-e Design data file structures

2-g Use decision tables

Competency Category Three

This competency category concerned the actual writing of computer programs to implement system specifications. The survey findings identified the competency category as ranking 12 in importance, 26 in difficulty, and 17 in frequency.

Table 13 exhibited three competencies in this group which were assessed insufficient curriculum content to fulfill the surveyed needs. The competencies were:

3-g Code indexed file routines

3-h Code table look-up-routines

3-i Code internal sort routines

In addition, there were five competencies revealed as emphasized in excess of the recommended curriculum content:

3 Write computer programs

3-b Code COBOL programs

3-c Code assembler programs

3-d Code interactive programs

3-j Study language manuals

The three competencies in this category found to have received proper emphasis on curriculum content were:

3-a Interpret program flowcharts

3-e Develop routine libraries

3-f Code sequential file routines

Competency Category Four

This competency category involved the testing of computer programs to prove system correctness. The category was identified by the survey findings as ranking first in importance, 13 in difficulty, and 5 in frequency.

Investigation of Table 13 indicated that there were three competencies which were not emphasized adequately in the curriculum. They were:

4-a Develop test files

4-b Create test transactions

4-c Test all possible errors

Competency 4-d, provide audit data, was identified as having received curriculum emphasis in excess of that justified by the analysis of difficulty. Also, the emphasis upon competency 4, test computer programs, was interpreted to

have been congruent with the survey ranking.

Competency Category Five

This competency category related to the function of documenting the elements of a system and program. The category was identified by the survey findings as ranking 11 in importance, 33 in difficulty, and 30 in frequency.

Examination of Table 13 revealed that two competencies appeared to have lacked sufficient emphasis on curriculum content. The competencies were:

5-f Develop easy to follow logic

5-g Train system user

Table 13 also showed two competencies that received excessive assessment of curriculum content:

5 Document system and programs

5-c Prepare job stream statements

Four competencies were found to have received sufficient emphasis to meet the level of difficulty identified in the survey findings. The competencies were:

5-a Prepare operator instructions

5-b Prepare program compilations

5-d Write user procedures

5-e Maintain data formats

Competency Category Six

This competency category concerns understanding computer hardware concepts and capabilities. The survey findings identified the category as ranking 35 in importance, 2 in difficulty, and 31 in frequency.

From the review of Table 13, there appeared to be two competencies of this group which had not received sufficient emphasis on curriculum content. Those competencies were:

6 Understand computer hardware

6-a CPU physical storage

None of the competencies were identified as having received excessive curriculum emphasis. The remaining four competencies were interpreted as having received curriculum emphasis congruent with the identified need. They were:

6-b Tape storage capabilities

6-c Disk storage capabilities

6-d Teleprocessing device capabilities

6-e Data entry capabilities

Competency Category Seven

This category of competencies concerned understanding computer software concepts and capabilities. The survey findings identified this category as ranking 36 in importance, 12 in difficulty, and 33 in frequency.

The comparison of data in Table 13 disclosed that two

of the competencies in this group were deficient in curriculum emphasis. The competencies were:

7 Understand computer software

7-b Software utility packages

The table also showed that competency 7-c, system library cataloging, was emphasized to an extent beyond that justified by the survey level of difficulty. The competency identified as having received curriculum emphasis congruent with the surveyed need was 7-a, supervisor and interrupt concepts.

Competency Category Eight

This category of competencies involved an understanding of business organization and standard procedures. The survey findings identified the competency category as ranking 32 in importance, 22 in difficulty, and 12 in frequency.

There were no competencies from this category which were interpreted as lacking curriculum emphasis. However, Table 13 did reveal that item 8-b, business accounting practices, had excessive instructional-units assigned to it compared to the level of difficulty identified.

The remaining four competencies were found to have received curriculum emphasis congruent with the survey findings. Those competencies were:

8 Understand business organization

8-a Data processing user interaction

8-c Business policies and procedures

8-d Compensation and promotion

Competency Category Nine

This competency category concerns the development of personal and professional qualities. The survey identified the category as ranking 17 in importance, 16 in difficulty, and 8 in frequency.

Examination of Table 13 revealed that all five of the competencies within this category were deficient in curriculum emphasis based upon comparison with the survey ranking of level of difficulty. The competencies were:

9 Develop professional qualities

9-a Establish job goals

9-b Interact with user and management

9-c Develop problem solving techniques

9-d Develop personal initiative

Summary of the Discussion

The discussion chapter presented an analysis of the survey findings compared to the emphasis in curriculum content. The criteria to measure actual curriculum content was defined and applied as presented in Appendix F. Also, the criteria defined to establish recommended curriculum content was presented in Table 12.

The comparison of assessed instructional-units with recommended instructional-units was accomplished by constructing and analyzing data presented in Table 13. The comparison procedure identified 22 competencies out of a total of 62 which appeared to be lacking curriculum content emphasis sufficient to satisfy the level of difficulty attributed to the competencies by the survey findings. The competencies were:

1-a Gather information

1-b Analyze existing documentation

1-c Determine impact of alternatives

1-e Analyze problem cause

2-h Establish program requirements

3-g Code indexed file routines

3-h Code table-look-up routines

3-i Code internal sort routines

4-a Develop test files

4-b Create test transactions

4-c Test all possible errors

5-f Develop easy to follow logic

5-g Train system user

6 Understand computer hardware

6-a CPU physical storage

7 Understand computer software

7-b Software utility packages

9 Develop professional qualities

9-a Establish job goals

9-b Interact with user and management

9-c Develop problem solving techniques

9-d Develop personal initiative

In addition, the comparison process revealed 17 competencies which appeared to have recieved execssive curriculum emphasis based upon the survey levels of difficulty. The competencies included:

2 Design computer programs

2-a Develop system flowcharts

2-b Write system narratives

2-c Develop program flowcharts

2-d Write program specifications

2-f Design output forms

2-i Develop routine libraries

3 Write computer programs

3-b Code COBOL programs

3-c Code assembler programs

3-d Code interactive programs

3-j Study language manuals

4-d Provide audit data

5 Document system and programs

5-c Prepare job stream statements

7-c System library cataloging

8-b Business accounting practices

The discussion also disclosed that there were 23 competencies which appeared to have received curriculum emphasis to an extent congruent with the survey findings. The competencies were:

1 Analyze data processing problems

1-d Evaluate results of modification

1-f Prepare time and cost estimates

1-g Define system error conditions

2-e Design data file structures

2-g Use decision tables

3-a Interpret program flowcharts

3-e Develop routine libraries

3-f Code sequential file routines

4 Test computer programs

5-a Prepare operator instructions

5-b Prepare program compilations

5-d Write user procedures

5-e Maintain data formats

6-b Tape storage capabilities

6-c Disk storage capabilities

6-d Teleprocessing capabilities

6-e Data entry capabilities

7-a Supervisor and interrupt concepts

8 Understand business organization

8-a Data processing user interaction

8-c Business policies and procedures

8-d Compensation and promotion

It should be noted that the discussion chapter analysis was based upon the level of difficulty obtained from the responses to the survey questionnaire. Utilization of the data within the discussion chapter may be contingent upon influences of the importance levels and frequency levels which were also surveyed.

SUMMARY, CONCLUSION, AND RECOMMENDATION

Summary

The educational requirements of computer programmers has undergone rapid and important changes in technical skills needed at the job entry level. Awareness of potential curriculum obsolescence prompted the design of this study. Research was conducted to determine the extent to which the computer programming curriculum at Des Moines Area Community College met the identified requirements of industry. More specifically, the purpose of the study was to answer four questions:

1. What are the competencies considered to be essential to entry level business computer programmers?
2. Does the existing curriculum at Des Moines Area Community College lack emphasis of competencies identified as needed?
3. Does the existing curriculum at Des Moines Area Community College contain excess emphasis of competencies identified as needed?
4. Can the evaluation procedure developed and applied facilitate systematic curriculum improvement?

The tatic used in the research effort was to first construct a study proposal that specified in detail the nature of the problem, why the problem deserved attention, and the

research steps to be followed. Next, a review of literature was conducted to determine the historical influences, job descriptors, environment, and appitudes characteristic of computer programmers. A further review of literature was then conducted to investigate relevant past research in the use of task statement analysis and competency surveys.

The review of literature provided applicable information both in the area of specific competency statements, and also in the area of suitable research procedure.

The first question of the study was to determine the competencies needed by entry level business programmers. The research was essentially a comparison study of curriculum content and business competency needs. The results obtained from this question provided the factual foundation for the comparison portion of the study. The review of literature provided direction in obtaining three criteria for curriculum content consideration. The criteria were:

1. How important the competency was to successful job entry level performance.
2. How difficult the competency was to perform.
3. How often the competency must be performed.

These criteria provided necessary decision making input to the structuring of curriculum content emphasis.

The findings chapter of this study presented analysis of the results of the competency survey. The discussion chap-

ter then presented a comparison analysis of the assessed curriculum content and the recommended curriculum content based upon the level of difficulty surveyed.

The data obtained from the competency survey were analyzed to determine general information obtained from the respondents. Thirteen tables were presented to provide meaningful examination of the data. The results of the survey determined that there were 69.1 percent male respondents and 30.9 percent female. These data were consistent with the review of literature information which identified computer programming as a profession where opportunity exists for responsible members of either sex.

Many types of businesses were represented in the survey. There were 59 respondents from the customer services (banking, insurance, publishing), 34 from consumer goods (manufacturing, retailing, distribution), and 46 from the public service functions of government, education, and utilities.

The prior training of respondents was indicated by 65 from vocational schools, 59 from universities, 9 from on-the-job training, and 6 with a high school education.

The survey respondents were comprised of the largest age group of 25 to 31 years old (56.8 percent), followed by 32 to 38 years old (22.3 percent), 18 to 24 years old (18.7 percent), and 39 to 50 years old (2.2 percent).

The size of the data processing department was represented by 38.8 percent from small departments of less than 20 employees, 36.6 percent from medium size departments, and 24.6 percent from large departments of more than 50 employees.

The survey also provided data of the size of computer programmed by the respondents. There were 50.4 percent representing the large scale computer, 30.9 percent representing the small scale computer, and 18.7 percent representing the medium scale computers. Analysis was also presented of the geographic location of the respondents. There were 63.3 percent from metropolitan Des Moines area, and 36.7 percent from the remainder of Iowa.

Next, the data were organized into three tables of rank order to determine which competencies were identified as most important, which were considered more difficult to perform, and which were used most often. The logic of this form of presentation was that emphasis of curriculum content should reflect those three measurements of each competency.

The analysis of importance for each competency determined that there were twelve competencies essential to job entry level computer programmers. Of the competencies identified as most important; "testing programs" was first followed by "develop personal initiative."

The analysis of difficulty for each competency determined there was a large group of 22 competencies identified as "difficult" to perform, but none were identified as "very difficult" to perform. The competency measured to be most difficult to perform was "determine the impact of alternatives", and "understand computer hardware" was second most difficult.

The analysis of frequency determined that eight competencies were identified as used daily. The competency used most often was found to be "interact with user and management", followed by "develop personal initiative."

Further analysis of the survey results was presented to examine the relationships of importance to difficulty, importance to frequency, and difficulty to frequency. The reason for this form of presentation was that additional curriculum emphasis may be required for those competencies which are both important and difficult and used often. It is also useful in curriculum planning to identify those competencies that are important but not difficult, or important but not used often, or frequently used but not difficult. To identify these relationships, the technique suggested by Dr. Wolins of using scatter diagrams was applied.

First, the relationship of importance to difficulty was analyzed by scatter diagram. There were eight compe-

tencies identified as being both difficult to perform and essential to entry level programmers. The scatter diagram also identified three competencies as being important but not difficult to perform.

The relationship of importance to frequency was analyzed next by scanning the scatter diagram. The scatter diagram identified five competencies as being important and used daily. There were also three competencies identified as being neither frequently used nor important to entry level computer programmers.

Last, the relationship of difficulty to frequency was analyzed by scatter diagram. There were five competencies identified as being both difficult to perform and used daily. The scatter diagram also identified one competency as being used frequently but not difficult, and one competency as not frequent nor difficult to perform.

After the survey findings were presented, the discussion chapter compared the recommended curriculum content with the actual curriculum content. The format of the discussion first identified the criteria to measure the curriculum content. The actual curriculum content was presented in Appendix F. Next, the criteria for recommended curriculum content was identified and presented in Table 12. Then comparison analyses were made for each competency using the assessed and recommended values presented in Table 13. There

were 22 competencies which appeared to have been lacking curriculum content, and 17 competencies which appeared to have received excessive curriculum emphasis.

Conclusion

The first question of the study was:

1. What are the competencies considered to be needed by entry level business computer programmers?

This question was answered by conducting the competency survey. There were 62 competencies considered to be needed by entry level business computer programmers.

The second question of the study was:

2. Does the existing curriculum at Des Moines Area Community College lack emphasis of competencies identified as needed?

This question was answered by the analysis of data in the discussion chapter. The existing curriculum at Des Moines Area Community College lacked emphasis of 22 needed competencies. Those competencies were:

1-a Gather information from user personnel about specific information system problems.

1-b Analyze existing system documentation to become familiar prior to modification.

1-c Determine the impact of alternative solutions upon the total system.

1-e Analyze problems to determine the cause being computer, personnel, procedures, etc.

2-h Establish program function requirements and module logic flow.

3-g Code and debug indexed file processing routines.

3-h Code and debug table-look-up routines.

3-i Code and debug internal sort routines.

4-a Develop test files and records to simulate all conditions encountered in a live file.

4-b Create test teansactions to prove that programs accept a range of variables and combinations of input.

4-c Create testing data to prove that programs respond properly to all possible errors.

5-f Develop easy to follow program logic.

5-g Train user / customer in details of system.

6 Understand computer hardware concepts and capabilities.

6-a CPU physical storage and data representation.

7 Understand computer software concepts and capabilities.

7-b Software utility packages.

9 Develop professional and interpersonal qualities.

9-a Establish job goals and objectives.

9-b Interact with user, co-worker, management, and non-computer personnel in a productive manner.

9-c Develop problem solving techniques.

9-d Develop personal initiative.

The third question of the study was:

3. Does the existing curriculum at Des Moines Area Community College contain excess emphasis of competencies identified as needed?

This question was also answered by the analysis of data in the discussion chapter. The existing curriculum at Des Moines Area Community College contained excess emphasis of 17 needed competencies. Those competencies were:

2 Design computer programs and systems to implement problem solutions.

2-a Develop system flowcharts.

2-b Write system workflow narratives.

2-c Develop program flowcharts.

2-d Write program specifications.

2-f Design output forms and report formats.

2-i Develop library usage and common routine specifications.

3 Write computer programs to implement system specifications.

3-b Code and debug COBOL programs.

3-c Code and debug assembler programs.

3-d Code and debug interactive teleprocessing programs.

3-j Study language manuals to become proficient.

4-d Provide audit data for non-computer personnel.

5 Document system and program elements.

5-c Prepare job stream and linkage control statements.

7-c System library cataloging capabilities.

8-b Business accounting practices.

The fourth question of the study was:

4. Can the evaluation procedure developed and applied facilitate systematic curriculum development?

This question was answered by successfully applying the research model as a decision making tool. Systematic curriculum development was facilitated by identifying the competencies needed by business and then determining whether the curriculum was lacking or exceeding emphasis of competencies. It is suggested that this study be viewed as one step in the ongoing process of curriculum improvement as presented in Appendix G. The analysis of findings and discussion of comparisons with the curriculum provided objective evidence of content adequacy. Although this study may not directly solve the problem of curriculum obsolescence, it has provided creditable information essential to decision making for curriculum design and modification.

Based upon the results of this study, two major conclusions were drawn. First, a systematic comparative analysis can be made to determine the instructional emphasis which will approximate the entry level competencies needed by computer programmers. Also, the Computer Programming

Department at Des Moines Area Community College should readjust instructional time allocations to develop specific competencies with closer approximations to industry need.

Recommendation

A general recommendation is that many career education programs would benifit from additional studies of curriculum content compared to the needs of industry.

Based upon the review of literature and discussions with representatives of the computer programming profession, it is recommended that the community college computer programming department pursue additional research. The areas related to this problem include:

Duration of coursework

Student attrition

Equipment obsolescence

Student scheduling flexibility

Staff development

Furthermore, this study derivied specific recommendations based upon the findings, comparisons, and conclusions:

1. In-service training should be encouraged so that the instructional staff may become aware of the implications of this study.
2. Public relations should be encouraged by providing feedback of the results of this study to business

and industry.

3. Additional research should be pursued to further validate the findings and conclusions of this study.
4. If supported by additional research, the curriculum of the computer programming program at Des Moines Area Community College should be revised to reduce the emphasis on those competencies found to receive excessive content.
5. If supported by additional research, the curriculum of the computer programming program at Des Moines Area Community College should be revised to increase the emphasis on those competencies found to receive insufficient content.
6. Further research is recommended in the definition and integration of the college transfer electives.

REFERENCES

1. Advanced Management Sciences. Complete guide to data processing staff training. Anonymous, Lynfield, Mass. 1971.

2. Amarel, S. Computer science: a conceptual framework foe curriculum planning. Communications of Association for Computing Machinery 14:391-401. 1971.

3. Ammerman, H. L. and Pratzner, F. C. Occupational survey report on business data programmers. Department of Business, Ohio State University. 1974.

4. Ashworth, A. W. and Gebolys, S. Data processing technology curriculum. Department of Computer Science, Central Texas College. 1973.

5. Barnes, B. H. and Gotterer, M. H. Attributes of computer professionals. Communications of Association for Computing Machinery 14:300-301. 1971.

6. Barnett, L. and Davis, L. Careers in Computer Progamming. Henry Walck, Inc., New York. 1967.

7. Bell, D. Programmer selection and programming errors. The Computer Journal 19:202-208. 1976.

8. Berger, R. M. Computer programmer job analysis. American Federation of Information Processing Society, New York. 1974.

9. Berryman, T. E. Business data processing: Hardware and curriculums. Business Education Forum 25:47-48. 1970.

10. Borcher, S. D. and Joyner, J. W. Business data processing occupational survey. Ohio State University. 1973.

11. Borg, W. and Gall, M. Educational research. David McKay Company, New York. 1971.

12. Borgen, J. and Davis, D. Planning, implementing, and evaluating career education preparation programs. McKnight Publishing, Bloomington, Ill. 1974.

13. Brightman, R. W. The computer and the junior college curriculum. American Association of Junior Colleges, Washington, D.C. 1970.

14. Brooks, F. P., Jr. The mythical man-month. Addison-Wesley Publishing, Reading, Mass. 1975.

15. Bruce, H., Jr. and Carpenter, B. Competency-based curriculum Kentucky model. American Vocational Journal 52:58-63. 1977.

16. Bryant, E. An analysis of data processing occupations. Unpublished Ph.D. thesis. Ohio State University. 1975.

17. Bunyan, C. Computing Manpower. Infotech Information, Berkshire England. 1973.

18. Burnett, E. Computers in use: Analyzed by standard industrial classification. Computers and People 24: 26-31. 1975.

19. Carver, L. Survey of junior high school general shop courses of study. Unpublished master's thesis. Iowa State University, Ames, Iowa. 1937.

20. Cashman, T. J. How the data processing industry has failed education. The Compiler 2:2-4. 1975.

21. Couger, J. D. Curriculum recommendations for undergraduate programs in information systems. Communications of Association for Computing Machinery 16: 727-749. 1973.

22. Couger, J. D. Education of Systems Designers. Computerworld 7:18-19. 1977.

23. Davis, S. Your future in computer programming. Richard Rosen Press, New York. 1969.

24. Dodd, G. Position statement on industry reaction to computer science education. Communications of Association for Computing Machinery 17:79-80. 1974.

25. Dooley, A. Annual data processing survey. Computerworld 7:66-68. 1977.

26. Gay, L. R. Educational research: competencies for analysis and application. Charles Merrill Publishing, Columbus, Ohio. 1976.

27. Gloster, E. D. A curriculum study in data processing. Unpublished master's thesis. Appalachian State University, Boone, N.C. 1970.

28. Gruenberger, F. So you are trying to teach computing. Datamation 26:119-124. 1977.

29. Heiker, V. and Galli, D. Is todays systems education adequate? Journal of Systems Management 27:24-29. 1976.

30. Herschbach, D. R. Deriving instructional content through task analysis. Journal of Industrial Teacher Education 14:63-73. 1976.

31. Hertlein, G. C. Computers and society: a current course outline for general education. Computers and People 24:24-26. 1975.

32. Johnson, J. R. A working measure of productivity. Datamation 26:106-112. 1977.

33. Johnson, M. F. Job specifications for the data processing cluster. Unpublished Ph.D. thesis. Temple University. 1976.

34. Jordan, K. F. An analysis of the content of computer technology with implications for technical education. Unpublished Ph.D. thesis. University of Missouri. 1969.

35. Kearney, J. F. Curricula for two year data processing programs: review and recommendations. Communications of Association for Computing Machinery 18:143-147. 1975.

36. Kernighan, B. W. and Plauger, P. J. The elements of programming style. McGraw-Hill Book Company, New York. 1974.

37. Kirkley, J. L. The first twent years. Datamation 26: 63-79. 1977.

38. Langerman, P. D. A skilled needs survey with implications for vocational-technical education in central Iowa. Unpublished Ph.D. thesis. Iowa State University, Ames, Iowa. 1968.

39. Lehman, M. and Belady, L. Programming systems dynamics. Communications of Association for Computing Machinery 14:286-289. 1971.

40. Little, J. C. Curriculum recommendations and guidlines for community and junior college programs in computer programming. Communications of Association for Computing Machinery 19:111-116. 1976.

41. Love, L. T. Relating individual differences in computer programmer performance to human information processing abilities. Unpublished Ph.D. thesis. University of Washington. 1977.

42. Loveday, E. George Stibitz and the Bell Laboratories relay computer. Datamation 26:80-85. 1977.

43. Luftig, M. Computer programmer aptitude test. Arco Publishing, New York. 1969.

44. Lyon, R. L. and Christiansen J. E. A task analysis approach to developing data processing curricula. Journal of Educational Data Processing 13:2-11. 1976.

45. Mader, C. and Hagin, R. Information systems: technology economics, and applications. Science Research Associates, Chicago. 1974.

46. Mager, R. F. and Beach, K. M. Jr., Developing vocational instruction. Fearon Publishing, Belmont, Calif. 1967.

47. Mayer, R. E. Different problem-solving competencies established in learning computer programming. Journal of Educational Psychology 66:725-730. 1975.

48. McLaughlin. R. A. Data processing salary survey. Datamation 25:61-79. 1976.

49. Melching, W. H. and Borcher, S. D. Procedures for constructing and using task inventories. Ohio State University. 1973.

50. Peck, J. A. A proposed computer science curriculum for two-year colleges. Communications of Association for Computing Machinery 11:114-120. 1968.

51. Sackman, H. and Gold, M. M. Time-sharing versus batch processing an experimental inquiry into human problem solving. Systems Development Corporation, Los Angeles, Calif. 1968.

52. Sackman, H., Erikson, W., and Grant, E. Exploring experimental studies comparing online and offline programming performance. Communications of Association for Computing Machinery 11:88-97. 1968.

53. Sammet, J. E. Programming languages: History and fundamentals. Prentice-Hall, Englewood Cliffs, N. J. 1969.

54. Shaw, B. The teaching of programming at the university level. University of Newcastle Upon Tyne, England. 1971.

55. Shapiro, H. P. Customer training in a small corporation: A case analysis. Unpublished Ph.D. thesis. Boston College. 1977.

56. Shelly, G. B. Why industry will not hire your graduates. The Compiler 1:2-4. 1974.

57. Shneiderman, T. Exploratory experiments in programmer behavior. International Journal of Computer and Information Sciences %:128-141. 1976.

58. Snyder, J. N. The teaching of computer science. Computers and People 24:25-26. 1975.

59. Starr, H. and Diffenderfer, J. A system of statewide evaluation of vocational education. Ohio State University. 1972.

60. Storer, D. Unpublished Community College Needs Survey. Kirkwood Community College, Cedar Rapids, Iowa. 1976.

61. Taba, H. Curriculum Development. Harcourt, Brace & World Inc., New York. 1962.

62. Testa, C. J. Behavior factors in information systems. Computers and People 23:13-17. 1974.

63. Toyne, M. C. An evaluation of data processing training in Georgia public schools. Unpublished Ph.D. thesis. Georgia State University. 1974.

64. U. S. Government. Procedure manual for task analysis and curriculum guide development: Report number three. Ohio State University. 1972.

65. U. S. Government. Recommended minimum vocational/technical programmer performance objectives. U. S. Office of Education. Code 16.0401. 1974.

66. Walker, T. M. Computer science curricula survey. Communications of Association for Computing Machinery 16:19-28. 1973.

67. Ware, R. R. Can the university train data processing personnel? Infosystems 18:52-54. 1976.

68. Weinberg, G. J. Programming teaching techniques. American Elsevier Publishing Co., New York. 1972.

69. Weinberg, G. J. The psychology of programming. American Elsevier Publishing Co., New York. 1974.

70. Wier, M. D. A comparative study of business data processing curricula among selected junior colleges within the U.S. Unpublished master's thesis. Oklahoma State University. 1974.

71. Wiersma, W. Research methods in education. Lippincott Co., New York. 1969.

72. Willemsen, E. W. Understanding statistical reasoning. W. H. Freeman and Company, San Francisco, Calif. 1974.

73. Yates, F. Sampling methods for censuses and surveys. Charles Griffin and Co., London. 1960.

74. Yourdon, E. J. Techniques of program structure and design. Prentice Hall, New York. 1975.

ACKNOWLEDGEMENTS

The author wishes to express his sincere appreciation to his major advisor, Dr. William D. Wolansky for the constructive guidance and assistance given in the development of this dissertation and throughout the graduate program.

Appreciation is expressed to Dr. Dwight W. Bensend, Dr. Trevor G. Howe, Dr. Gerald A. Parks, and Dr. Dominick D. Pellegreno for their thoughtful participation on the graduate committee and for the helpful guidance in the development of the proposal and questionnaire.

Appreciation is also expressed to Dr. Robert J. Gelina for the direction given during the preparation of this dissertation.

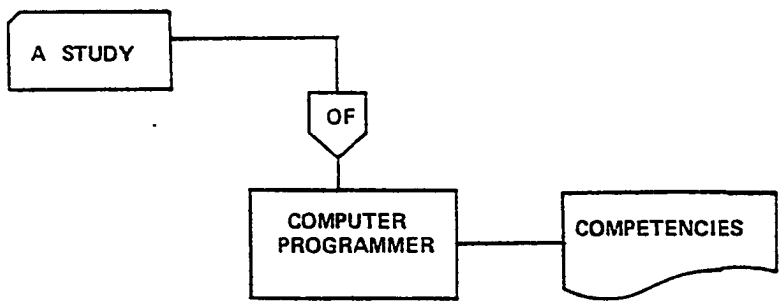
Appreciation is also expressed to Dr. Leroy Wolins, Dr. Richard D. Warren, and Dr. Milton D. Brown for assistance and consultation on the analysis used in this dissertation.

Finally, appreciation is extended to the author's wife, Marilyn, and children, Mark and Marcia, whose patience and understanding made completion of this study possible.

APPENDIX A: CRITERIA FOR COMPETENCY STATEMENTS

1. Construct brief statements to save reading time.
2. Avoid competency statements that are too general.
3. Avoid vauge or ambiguous words.
4. Use clear statements that are easily understood.
5. Use short words rather than long words if possible.
6. Begin statements with an action word.
7. Each statement should be a complete sentence.
8. Omit the period at the end of each statement.
9. Avoid "and/or" and "etc".
10. The competency statement must be ratable in terms of the measurement criteria.
11. Use terminology that is consistent with current occupational practice.
12. Arrange competency statements functionally to facilitate scanning to eliminate duplicate statements.
13. Avoid obviously trivial statements.
14. Avoid multiple verbs in one statement.
15. Competency statements should be constructed independent and distinct.
16. Use abbreviations with caution.
17. The competency statement must be capable of standing alone.
18. If the statement contains a modifier, include all relevant alternatives.

APPENDIX B: SURVEY LETTER AND QUESTIONNAIRE



Des Moines Area Community College
2006 Ankeny Blvd.
Ankeny, Iowa 50021

You have been selected to participate in a survey to identify computer programmer skills needed at the job entry level. This survey is being conducted by the Data Processing Department of Des Moines Area Community College in cooperation with the Industrial Education Department of Iowa State University. The results of this study will be used in computer programmer curriculum development and also as data in a research dissertation for Iowa State University.

This survey is a concise list of entry level business programmer competencies grouped into nine categories to minimize duplication. You may be assured your answers to this questionnaire will be treated confidentially. Each form is identified for auditing and feedback but the responses are analyzed strictly as group statistics.

Your cooperation in completing this questionnaire is sincerely appreciated. You may request information concerning the results of this survey in approximately four weeks.

Sincerely

William D. Wolansky
Professor and Head
Industrial Education
Iowa State University

Thank you very much

Ralph L. Keul
Instructor
Data Processing
Des Moines Area Community College

STATISTICAL GROUPING INFORMATION

Please circle the appropriate codes

Type of business
1 Customer services (banking, insurance, publishing)
2 Consumer goods (manufacturing, retailing, distribution)
3 Public service (government, education, utilities)

Prior training
1 High school
2 Vocational school
3 University
4 On the job
5 Military

sex
M Male
F FEMALE

Age _____

Number of employees in data processing _____

Computer primarily programmed

_____

Name _____

Business address

_____

_____

_____

ISU / DHACC
COMPUTER PROGRAMMER
COMPETENCY SURVEY

Circle the codes that best describe each competency

		IMPORTANCE 1 not used 2 least important 3 important 4 very important 5 essential	DIFFICULTY 1 not used 2 easy 3 somewhat hard 4 difficult 5 very difficult	FREQUENCY 1 not used 2 seldom used 3 used monthly 4 used weekly 5 used daily
1	ANALYZE data processing problems to devise and specify logical solutions	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
1-a	Gather information from user personnel about specific information system problems	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
1-b	Analyze existing system documentation to become familiar prior to modification	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
1-c	Determine the impact of alternative solutions upon the total system	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
1-d	Evaluate the results of system / program modification	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
1-e	Analyze problems to determine the cause being computer, personnel, procedures, etc.	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
1-f	Prepare time and cost estimates	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
1-g	Define system error conditions and corrective action	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
2	DESIGN computer programs and systems to implement problem solutions	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
2-a	Develop system flowcharts	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
2-b	Write system workflow narratives	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
2-c	Develop program flowcharts	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
2-d	Write program specifications	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
2-e	Design data file structures and record formats	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
2-f	Design output forms and report formats	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5

ISU / DVACC
COMPUTER PROGRAMMER
COMPETENCY SURVEY

Circle the codes that best describe each competency

		IMPORTANCE 1 not used 2 least important 3 important 4 very important 5 essential	DIFFICULTY 1 not used 2 easy 3 somewhat hard 4 difficult 5 very difficult	FREQUENCY 1 not used 2 seldom used 3 used monthly 4 used weekly 5 used daily
2-g	Use decision table techniques	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
2-h	Establish program function requirements and module logic flow	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
2-i	Develop library usage and common routine specifications	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
3	WRITE computer language programs to implement system specifications	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
3-a	Interpret the logic of program flowcharts and formal specifications	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
3-b	Code and debug COBOL programs	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
3-c	Code and debug ASSEMBLER programs	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
3-d	Code and debug interactive teleprocessing programs	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
3-e	Develop and catalog program routines into libraries	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
3-f	Code and debug sequential file processing routines	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
3-g	Code and debug indexed file processing routines	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
3-h	Code and debug table-look-up routines	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
3-i	Code and debug internal sort routines	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
3-j	Study programming language manuals to become proficient in the language	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
4	TEST computer programs to prove system correctness	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
4-a	Develop test files and records to simulate all conditions encountered in a live file	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5

ISU / DMACC
COMPUTER PROGRAMMER
COMPETENCY SURVEY

Circle the codes that best describe each competency

		IMPORTANCE 1 not used 2 least important 3 important 4 very important 5 essential	DIFFICULTY 1 not used 2 easy 3 somewhat hard 4 difficult 5 very difficult	FREQUENCY 1 not used 2 seldom used 3 used monthly 4 used weekly 5 used daily
4-b	Create test transactions to prove that programs accept a full range of variables and combinations of input	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
4-c	Create testing data to prove that programs respond properly to all possible errors	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
4-d	Provide audit data for non-computer personnel	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
5	DOCUMENT system and program elements	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
5-a	Prepare computer operator instructions	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
5-b	Prepare and identify most current program compilations	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
5-c	Prepare job stream and linkage control statements	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
5-d	Write user / customer procedures for implementation	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
5-e	Maintain current data formats and content codes	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
5-f	Develop easy to follow program logic	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
5-g	Train user / customer in details of system	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
6	UNDERSTAND computer hardware concepts and capabilities	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
6-a	CPU physical storage and data representation	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
6-b	Tape storage and physical capabilities	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
6-c	Disk storage and physical capabilities	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
6-d	Teleprocessing devices and physical capabilities	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
6-e	Data entry operation and device capabilities	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
7	UNDERSTAND computer software concepts and capabilities	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5

ISU / DHACC
COMPUTER PROGRAMMER
COMPETENCY SURVEY

Circle the codes that best describe each competency

		IMPORTANCE 1 not used 2 least important 3 important 4 very important 5 essential	DIFFICULTY 1 not used 2 easy 3 somewhat hard 4 difficult 5 very difficult	FREQUENCY 1 not used 2 seldom used 3 used monthly 4 used weekly 5 used daily
7-a	Supervisor and interrupt concepts	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
7-b	Software utility packages	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
7-c	System library cataloging capabilities	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
8	UNDERSTAND business orginization and standard procedure	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
8-a	Data processing department and user interaction	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
8-b	Business accounting practices	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
8-c	Business policies, procedures, and standard manuals	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
8-d	Compensation, performance review and promotion paths	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
9	Develop professional and interpersonal qualities	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
9-a	Establish job goals and objectives	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
9-b	Interact with user, co-worker, management, and non-computer personnel in a productive manner	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
9-c	Develop problem solving techniques	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
9-d	Develop personal initiative and persistence	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
	Please list other items you feel should be included			
	_____	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
	_____	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5

APPENDIX C: FOLLOW UP LETTER

des moines area
community college
2006 S. ANKENY BLVD., ANKENY, IOWA 50021
PHONE 964-6200

February 6, 1978

Dear Survey Participant:

We have not yet received your completed questionnaire for the computer programmer competency survey.

As a successful professional, your concerned response to the items in the questionnaire are especially important to this survey. The analysis of entry level computer programmer skills needed by industry and subsequent curriculumn development are essential to the future of the data processing community.

All information in the questionnaire will be treated in a confidential manner, with only group statistics reported. Enclosed is a copy of the questionnaire, which is a joint research effort of Iowa State University and Des Moines Area Community College.

Once again, thank you for your valuable assistance.

Sincerely

William Wolansky

William D. Wolansky
Professor and Head
Industrial Education
Iowa State University

Ralph Keul

Ralph L. Keul
Instructor
Data Processing
Des Moines Area
Community College

APPENDIX D: DES MOINES AREA COMMUNITY COLLEGE COMPUTER PROGRAMMING CURRICULUM COURSE SUMMARY

Course	Course name	Credit
CMPP 310	Introduction to Computer Systems	3
CMPP 312	Computer Operations I	5
CMPP 314	COBOL Programming I	5
CMPP 321	COBOL Programming II	5
CMPP 322	Programming Concepts I	3
CMPP 323	Report Program Generator	3
CMPP 331	COBOL Programming III	5
CMPP 332	Assembler Programming I	5
CMPP 333	Programming Concepts II	3
CMPP 340	COBOL Programming IV	5
CMPP 342	Assembler Programming II	5
CMPP 344	Programming Concepts III	3
CMPP 351	Assembler Programming III	5
CMPP 352	Telecommunication Systems	2
CMPP 353	Business Organization	1
CMPP 354	Systems Analysis I	5
CMPP 361	Systems Analysis II	5
CMPP 362	Application Programming I	3
CMPP 363	Application Programming II	3
CMPP 364	Application Programming III	3
CMPP 371	Application Programming IV	3
CMPP 372	Application Programming V	3
CMPP 373	Application Programming VI	3
CMPP 301	Programming Language One	3

Course	Course name	Credit
BSAD 103	Principles of Accounting I	4
BSAD 105	Principles of Accounting II	4
BSAD 202	FORTRAN Programming	3
	Electives	20

APPENDIX E: DES MOINES AREA COMMUNITY COLLEGE COMPUTER PROGRAMMING CURRICULUM COURSE DESCRIPTIONS

CMPP 310 Introduction to Computer Systems:

Acquaint the student with the history and development of modern data processing. It gives a general overview of computer systems, central processing unit structure, program execution, I/O channel devices, data management and programming systems.

CMPP 312 Computer Operations I

A course structured to give the student an introduction to the computer. The student will gain experience in operating the various peripheral devices. Also included in the course is practical experience unit record devices utilized as a means of input preparation for the computer.

CMPP 314 COBOL Programming I

Basic introduction to American National Standard COBOL emphasizing the design, coding, and debugging of card processing programs.

CMPP 321 COBOL Programming II

Sequential disk and tape programming concepts and emphasis of subscripting techniques involving one, two and three level arrays using American National Standard COBOL.

CMPP 322 Programming Concepts I

Material in this course covers the logic involved in preparing program flowcharts and system flowcharts.

CMPP 323 Report Program Generator

Study of the basic steps of programming. Student learns to design, code, and debug various problems written in the RPG language.

CMPP 331 COBOL Programming III

Material covered involves index sequential file processing, overlay techniques, and the use of the sort function.

CMPP 332 Assembler Programing I

Teaches the student to design, code, and debug card processing programs written in Assembler language code.

CMPP 333 Programming Concepts II

This course presents the concepts and practical applications of the job control language.

CMPP 340 COBOL Programming IV

Advanced COBOL programming techniques utilizing remote job entry.

CMPP 342 Assembler Language II

Will provide the student with alternative programming capabilities using the standard set of instructions. Major emphasis is placed on machine language comprehension, explicit coding notation, and data table organization.

CMPP 344 Programming Concepts III

An introduction into the operating system utilities, control system generation, and sort package.

CMPP 351 Assembler Programming III

Designed to provide an extensive exposure to Assembler language applications. Programs include advanced coding techniques and macro writing associated with index sequential processing.

CMPP 352 Telecommunication Systems

This course presents concepts of teleprocessing to the person who is familiar with computer applications in business but is not conversive on the subject of remote entry transmission techniques.

CMPP 353 Business Organization

A course designed to familiarize the student with computer installations. This is accomplished by visiting local data processing shops and by having guest speakers relate what management expects of its data processing employees.

CMPP 354 Systems Analysis I

Stresses methods and tools used in systems analysis and design. Analyses of information flow and establishing system specification and equipment needs, and planning the implementation of information systems.

CMPP 361 SYSTEMS ANALYSIS II

Advanced concepts in management information systems are studied through application of actual business studies.

CMPP 362 Application Programming I

CMPP 363 Application Programming II

CMPP 364 Application Programming III

CMPP 371 Application Programming IV

CMPP 372 Application Programming V

CMPP 373 Application Programming VI

Individual projects are assigned which require the student to apply the programming knowledge of design and implementation of an assigned business application in increasing degree of complexity.

CMPP 301 Programming Language One

The capabilities of the language are explored, case problems are written to emphasize the features of the language flexibility.

BSAD 202 FORTRAN programming

The language is studied with in-depth coverage of full capabilities. Case studies are written by the student involving card and disk input to produce statistical reports.

BSAD 103 Principles of Accounting I

BSAD 105 Principles of Accounting II

Basic theory of accounting cycles and preparation of statements for organizations.

APPENDIX F: ASSESSMENT OF INSTRUCTIONAL-UNITS BY COMPETENCY WITHIN COURSE OFFERED IN DES MOINES AREA COMMUNITY COLLEGE COMPUTER PROGRAMMING PROGRAM

CMPP 310 Introduction to Computer Systems

Item	Description	Instructional-units
2-c	Develop program flowcharts	1
2-f	Design output forms	1
3	Write computer programs	3
3-b	Code COBOL programs	2
3-f	Code sequential file routines	1
4	Test computer programs	1
5-b	Prepare program compilations	1
6	Understand computer hardware	5
6-a	CPU physical storage	2
6-b	Tape storage capabilities	1
6-c	Disk storage capabilities	1
6-d	Teleprocessing device capabilities	1
6-e	Data entry capabilities	2
7	Understand computer software	4
8-a	Data processing user interaction	1

CMPP 312 Computer Operations I

Item	Description	Instructional-units
1-e	Analyze problem cause	1
1-g	Define system error conditions	3
3-j	Study language manuals	4
5-a	Prepare operator instructions	3
5-c	Prepare job stream statements	2

6	Understand computer hardware	7
6-b	Tape storage capabilities	1
6-c	Disk storage capabilities	1
7	Understand computer software	4
7-a	Supervisor and interrupt concepts	5
7-b	Software utility packages	3
7-c	System library cataloging	2
8-a	Data processing user interaction	1

CMPP 314 COBOL Programming I

Item	Description	Instructional-units
1	Analyze data processing problems	3
1-d	Evaluate results of modification	2
2	Design computer programs	1
2-c	Develop program flowcharts	4
2-f	Design output forms	3
3	Write computer programs	5
3-a	Interpret program flowcharts	4
3-b	Code COBOL programs	5
3-f	Code sequential file routines	2
3-j	Study language manuals	3
4	Test computer programs	5
4-b	Create test transactions	1
5-b	Prepare program compilations	2
5-f	Develop easy to follow logic	5

7	Understand computer software	1
9-c	Develop problem solving techniques	3
9-d	Develop personal initiative	2

CMPP 321 COBOL Programming II

Item	Description	Instructional-units
1	Analyze data processing problems	3
1-d	Evaluate results of modification	2
2	Design computer programs	2
2-c	Develop program flowcharts	4
2-f	Design output forms	2
3	Write computer programs	5
3-a	Interpret program flowcharts	5
3-b	Code COBOL programs	5
3-f	Code sequential file routines	3
3-h	Code table-look-up routines	2
3-j	Study language manuals	4
4	Test computer programs	5
4-a	Develop test files	2
4-b	Create test transactions	1
4-c	Test all possible errors	1
5-b	Prepare program compilations	1
5-c	Prepare job stream statements	1
5-f	Develop easy to follow logic	2
6-b	Tape storage capabilities	3

6-c	Disk storage capabilities	2
9-c	Develop problem solving techniques	2
9-d	Develop personal initiative	1

CMPP 322 Programming Concepts I

Item	Description	Instructional-units
1	Analyze data processing problems	4
1-c	Determine impact of alternatives	3
2-a	Develop system flowcharts	10
2-c	Develop program flowcharts	18
9-c	Develop problem solving techniques	1

CMPP 323 Report Program Generator

Item	Description	Instructional-units
2	Design computer programs	3
2-c	Develop program flowcharts	4
3	Write computer programs	11
3-f	Code sequential file routines	1
4	Test computer programs	4

CMPP 331 COBOL Programming III

Item	Description	Instructional-units
1	Analyze data processing problems	3
1-c	Determine impact of alternatives	1
1-d	Evaluate results of modification	3
2	Design computer programs	3

2-c	Develop program flowcharts	3
2-f	Design output forms	2
3	Write computer programs	5
3-a	Interpret program flowcharts	5
3-b	Code COBOL programs	5
3-g	Code indexed file routines	3
3-i	Code internal sort routines	2
3-j	Study language manuals	2
4	Test computer programs	5
4-b	Create test transactions	1
5	Document system and programs	2
5-c	Prepare job stream statements	1
5-f	Develop easy to follow logic	3
6-c	Disk storage capabilities	2
9-a	Establish job goals	1

CMPP 332 Assembler Programming I

Item	Description	Instructional-units
2	Design computer programs	4
2-c	Develop program flowcharts	2
3	Write computer programs	8
3-a	Interpret program flowcharts	2
3-c	Code Assembler programs	9
3-f	Code sequential file routines	3
3-j	Study language manuals	4

4	Test computer programs	5
4-b	Create test transactions	1
5-c	Prepare job stream statements	2
5-f	Develop easy to follow logic	2

CMPP 333 Programming Concepts II

Item	Description	Instructional-units
3-j	Study language manuals	4
5-c	Prepare job stream statements	20
6-c	Disk storage capabilities	5
7	Understand computer software	2
7-c	System library cataloging	11

CMPP 340 COBOL Programming IV

Item	Description	Instructional-units
1	Analyze data processing problems	2
1-c	Determine impact of alternatives	1
1-d	Evaluate results of modification	2
2	Design computer programs	4
2-c	Develop program flowcharts	2
2-e	Design data file structures	1
2-f	Design output forms	1
2-i	Develop library routines	1
3	Write computer programs	6
3-a	Interpret program flowcharts	2
3-b	Code COBOL programs	5

3-d	Code interactive programs	2
3-e	Develop routine libraries	1
3-f	Code sequential file routines	1
3-h	Code table-look-up routines	1
3-j	Study language manuals	5
4	Test computer programs	5
4-a	Develop test files	2
4-b	Create test transactions	1
5	Document system and programs	1
5-b	Prepare program compilations	1
5-f	Develop easy to follow logic	3
6	Understand computer hardware	2
6-c	Disk storage capabilities	1
6-e	Data entry capabilities	1
7	Understand computer software	1
7-a	Supervisor and interrupt concepts	1

CMPP 342 Assembler programming II

Item	Description	Instructional-units
2	Design computer programs	3
3	Write computer programs	20
3-c	Code assembler programs	18
3-h	Code table-look-up routines	4
3-j	Study language manuals	10
6-a	CPU physical storage	4

CMPP 344 Programming Concepts III

Item	Description	Instructional-units
3-e	Develop routine libraries	5
3-j	Study language manuals	8
5-c	Prepare job stream statements	2
6-c	Disk storage capabilities	4
7	Understand computer software	5
7-a	Supervisor and interrupt concepts	3
7-b	Software utility packages	8
7-c	System library cataloging	10

CMPP 351 Assembler Programming III

Item	Description	Instructional-units
1	Analyze data processing problems	2
2	Design computer programs	6
2-e	Design data file structures	4
2-i	Develop library routines	10
3	Write computer programs	5
3-c	Code assembler programs	20
3-e	Develop routine libraries	2
3-g	Code indexed file routines	5
3-j	Study language manuals	2
5-c	Prepare job stream statements	2
6-c	Disk storage capabilities	2
7	Understand computer software	3

CMPP 352 Telecommunication Systems

Item	Description	Instructional-units
2-e	Design data file structures	2
2-f	Design output forms	3
3-d	Code interactive programs	12
4-a	Develop test files	1
4-c	Test all possible errors	1
6-d	Teleprocessing device capabilities	8
7-a	Supervisor and interrupt capabilities	1
8-a	Data processing user interaction	1

CMPP 353 Business Organization

Item	Description	Instructional-units
8	Understand business procedures	11
8-a	Data processing user interaction	8
8-c	Business policy manuals	4
8-d	Compensation and promotion paths	2
9	Develop professional qualities	1
9-b	Interact with user and management	4

CMPP 354 Systems Analysis I

Item	Description	Instructional-units
1	Analyze data processing problems	5
1-a	Gather information	6
1-b	Analyze existing documentation	2

1-c	Determine impact of alternatives	4
1-d	Evaluate results of modification	8
1-e	Analyze problem cause	5
1-g	Define system error conditions	5
2	Design computer programs	10
2-a	Develop system flowcharts	5
2-b	Write program narratives	3
2-d	Write program specifications	2
2-e	Design data file structures	3
2-f	Design output forms	5
2-g	Use decision tables	1
8	Understand business procedure	3
8-a	Data processing user interaction	3
8-c	Business policy manuals	2
9-a	Establish job goals	2
9-b	Interact with user and management	5
9-c	Develop problem solving techniques	4

CMPP 361 Systems Analysis II

Item	Description	Instructional-units
1	Analyze data processing problems	5
1-a	Gather information	3
1-b	Analyze existing documentation	4
1-c	Determine impact of alternatives	5
1-d	Evaluate results of modification	4

1-e	Analyze problem cause	6
1-g	Define system error conditions	4
2	Design computer programs	8
2-a	Develop system flowcharts	2
2-b	Write system narratives	3
2-d	Write program specifications	2
2-e	Design data file structures	4
2-f	Design output forms	3
4-a	Develop test files	2
4-b	Create test transactions	1
4-c	Test all possible errors	2
4-d	Provide audit data	1
5	Document system and programs	5
5-d	Write user procedures	3
5-g	Train system user	2
9-a	Establish job goals	1
9-b	Interact with user and management	3
9-c	Develop problem solving techniques	2

CMPP 362, 363, 364 Application Programming I, II, III
Offered as a cluster only

Item	Description	Instructional-units
1	Analyze Data processing problems	8
1-a	Gather information	2
1-b	Analyze existing documentation	2
1-c	Determine impact of alternatives	5

1-d	Evaluate results of modification	4
1-e	Analyze problem cause	2
1-g	Define system error conditions	4
2	Design computer programs	5
2-a	Develop system flowcharts	2
2-b	Write system narratives	6
2-d	Write program specifications	5
2-e	Design data file structures	7
2-f	Design output forms	6
2-h	Establish program requirements	5
3	Write computer programs	11
3-c	Code Assembler programs	4
4-a	Develop test files	5
4-b	Create test transactions	6
4-c	Test all possible errors	5
4-d	Provide audit data	6
5	Document system and programs	3
5-d	Write user procedures	9
5-g	Train system user	8
8-a	Data processing user interaction	5
9-a	Establish job goals	3
9-b	Interact with user and management	2

CMPP 371, 372, 373 Application Programming IV, V, VI

Item	Description	Instructional-units
1	Analyze data processing problems	6
1-a	Gather information	4
1-b	Analyze existing documentation	6
1-c	Determine impact of alternatives	7
1-d	Evaluate results of modification	3
1-f	Prepare time and cost estimates	1
1-g	Define system error conditions	3
2	Design computer programs	7
2-a	Develop system flowcharts	4
2-d	Write program specifications	11
2-e	Design data file structures	6
2-f	Design output forms	7
3	Write computer programs	11
3-b	Code COBOL programs	3
4	Test computer programs	5
4-a	Develop test files	6
4-c	Test all possible errors	1
4-d	Provide audit data	3
5	Document system and programs	14
5-a	Prepare operator instructions	1
5-d	Write user procedures	5
5-e	Maintain data formats	1

5-g	Train system user	2
8-a	Data processing and user interaction	3
8-b	Business accounting practices	2
8-c	Business policy manuals	3
9-a	Establish job goals	2

CMPP 301 Programming Language One

Item	Description	Instructional-units
3	Write computer programs	5
3-j	Study language manuals	10

BSAD 103 Principles of Accounting I

Item	Description	Instructional-units
8	Understand business procedures	3
8-b	Business accounting practices	22
8-c	Business policy manuals	1

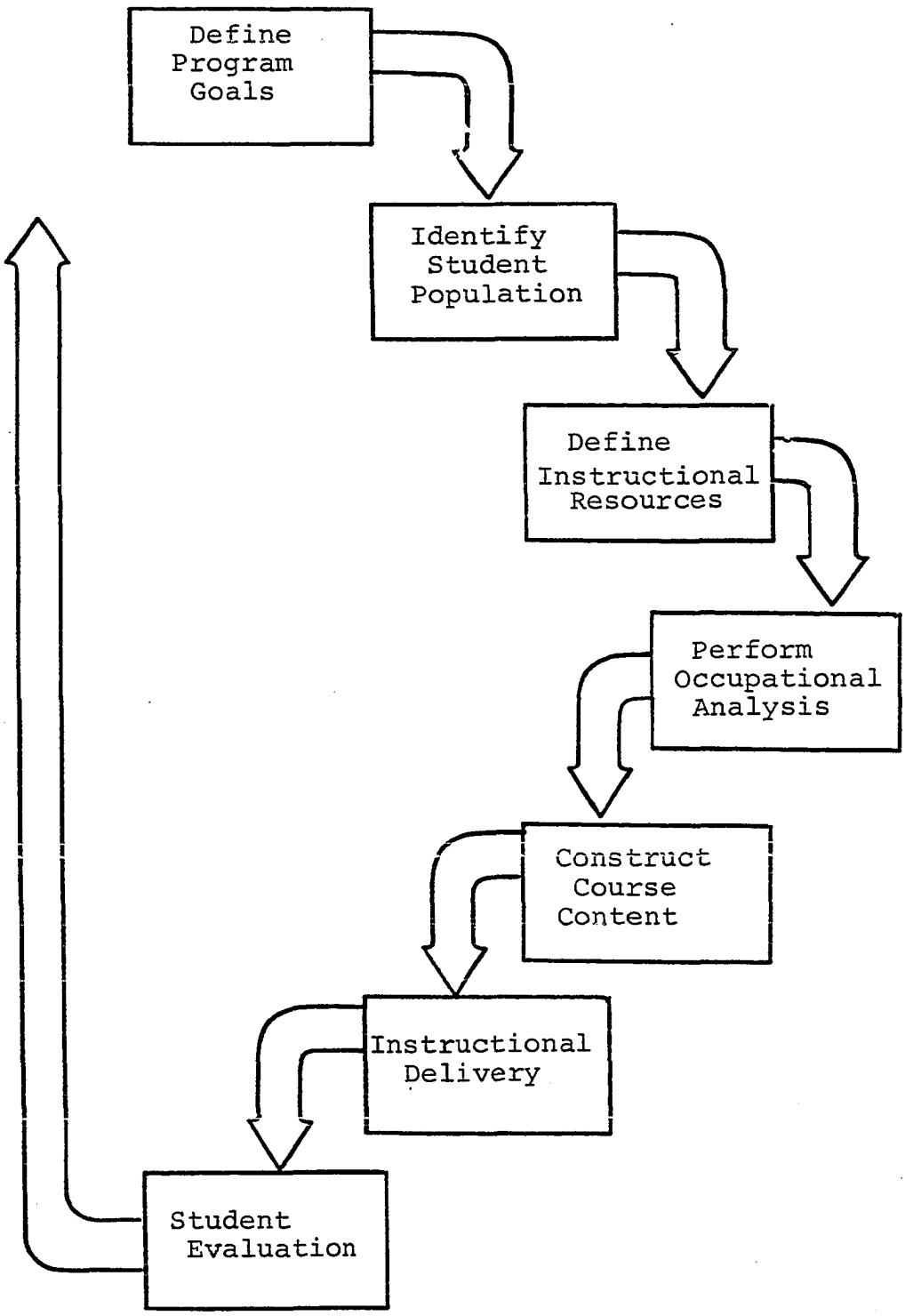
BSAD 105 Principles of Accounting II

Item	Description	Instructional-units
8	Understand business procedures	5
8-b	Business accounting practices	26
8-c	Business policy manuals	3

BSAD 202 FORTRAN Programming

Item	Description	Instructional-units
3	Write computer programs	11
3-j	Study language manuals	5

APPENDIX G: PERSPECTIVE OF CURRICULUM DEVELOPMENT



Perspective of curriculum development