1985

Development and formative evaluation of educational products using data base management systems in a college of education

Chad Lee G. Grabow
Iowa State University

Development and formative evaluation of educational products using data base management systems in a college of education

by

Chad Lee C. Grabow

A Dissertation Submitted to the

Graduate Faculty in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Department: Professional Studies in Education

Major: Education (Higher 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

1985

TABLE OF CONTENTS

INTRODUCTION

Data processing has been generally adopted by institutions of higher education. New technologies and economies of scale have made it possible to utilize the computer, whether it be a mainframe, mini-computer or micro-computer, in the educational and administrative processes of colleges and universities. As a concept, a data base is a part of the contemporary data processing design of the 1980s. A data base is a collection of related data, about an enterprise, with multiple uses. The software (computer programs that enable data to be stored, retrieved, modified, and deleted) and procedures that manage the data base comprise a Data Base Management System (DBMS). A Data Base Management System makes it possible to access integrated data that crosses operational, functional, or organizational boundaries within an enterprise (Atre, 1980). In higher education, the enterprise to which Atre refers may be the university, a college, department, or any functional or organizational subdivision of the institution. Automating the data management function consists of data collection, organization, storage, retrieval, and manipulation using commercial Data Base Management Systems. The approach taken by colleges and departments towards the automation of data management has been slow to non-existent (Plourde, 1981).

One such organizational unit within a college or university is the teacher preparatory program within a College of Education. That program may span departments within the College. The program may include those of elementary and secondary education as well as related disciplinary departments within teaching fields such as English, Mathematics or Political Science. Likewise, a functional area such as teacher training may involve the placement office, the teacher certification unit of the state education agency and a host of cooperating schools where student teaching internships may take place. Such a functional area of higher education serves as the environment for this research and development project.

Since 1979, there has been growing interest by state legislators on the competency of teachers. Sandefur (1981) reported that by October, 1980, at least 29 states initiated legislative action or set up fact-finding committees regarding competency assessment of teachers. This action included regulation of entry into the profession and control of the certification process. The need for better tools for teacher assessment, was advocated by Denemark and Nelli in 1968. Denemark stated that teaching is an complex, demanding task of knowing, doing, and being. Such a prismatic view of teaching requires a multidimensional approach to teacher assessment. An approach that supports the use of paper and pencil measures when appropriate, but requires, as well, more complex measures of performance. Kniker's (1981) review of the literature proposed that the goal of teacher competency programs should be to provide students with regular feedback on their performance regarding the published competencies. Students should be appraised of their strengths and needs and whenever possible, given suggestions for resources. Kniker (1981) also suggested that the capabilities of computers be explored in support of teacher assessment.

Two of the factors advanced in favor of using computers in the commercial work place are the ability to store large quantities of data and the speed in which the computer can transform data into meaningful information (Mandell, 1982). Other factors include obtaining rapid responses to questions and the knowledge that one is in on the "state of the art" technology, an important part of keeping up future trends.

Attempts to use the computer in teacher assessment have met with various levels of success. The majority of past applications were simulations (Zuckerman, 1979). Simulations presented an educational experience to the student. The student may modify the environment and observe the result. If undesirable, the student may restore the simulation to its original environment and try another path or option. Other applications focused on drill and practice or an automated page turning capability. These approaches only replicate a book or pencil and paper with a computer screen. For the most part, the above applications were implemented on micro-computers, due to the high costs associated with the procurement of larger computer systems and labor intensive programming efforts.

The administration of Iowa State University's College of Education created the PRO*FILE Task Force in 1981 to identify elements that could be incorporated into a PERSONAL PROFILE ANALYSIS. In other words, the PRO*FILE Task Force was to seek out those knowledge skills, attitudes and dispositions which have been identified as essential for students in teacher education programs. Additionally, the PRO*FILE Task Force was to begin suggesting how knowledge, skills, attitudes and dispositions might be incorporated into the Iowa State program, along with a battery of diagnostic tools, that would aid each student in the College of Education's program to learn what his or her strengths and needs are. The PRO*FILE Task Force was co-chaired by Drs. Charles R. Kniker and Joan C. Breiter. The PRO*FILE Task Force presented their Initial Study on a Profile System in July,

1981. The report included a summary of findings from the literature, an analysis of any local responses, and recommendations for future study. One of the results of the study suggested that portions of the teacher assessment process could be automated using a computer.

Statement of the Problem

Computer architecture is a term used to describe a computer system consisting of physical computer components (hardware) and programs (software) designed to solve a particular information problem. Today's computer architectures offer new approaches to the management of students and their education. This study will help determine how Data Base Management Systems technology may be used in the improvement and/or assessment of potential teachers and graduates in a College of Education. A specific application from Iowa State University's College of Education's ongoing research project, referred to as PRO*FILE, will be used to investigate the use of the computer and software as a prescriptive diagnostic processor. PRO*FILE will assist teacher education candidates during their undergraduate years by providing English-like software for individualized instruction on the basic concepts of teacher competencies.

Need

The purpose of the PRO*FILE System is to increase the already high level of competency of Iowa State University graduates as beginning teachers. The goal of the PRO*FILE System is its use as a prescriptive diagnostic processor assuming the legal and privacy issues of storing personal data can be resolved by federal and state governments. The amount of data required to create a record for one student and corresponding updates to monitor a student's progress is above average in magnitude. In the Spring of 1983, there were more than 500 students in teacher education curricula at Iowa State University. Additionally, the PRO*FILE Task Force has identified over 100 Performance Elements, each containing five to 15 pages of text. Data are dynamic in nature and require continuous updating to keep them relevant.

Based on the College of Education's requirement for an interactive PRO*FILE System, the PRO*FILE Task Force decided to automate the system using a computer. It was the intent of the PRO*FILE Task Force that administrators, faculty and students have access to the data base via on-line terminals. The utility of the PRO*FILE System became self-evident as the research effort continued. The computer architecture at Iowa State University allowed campus-wide access and updating. Campus-wide access freed students from the physical constraints of closed buildings or facilities not available due to inclement weather. A student could use the PRO*FILE System for self-evaluation and research 24 hours a day, seven days a week.

Purpose of the Project

An educational research and development approach was used to evaluate an application using Data Base Management Systems' theories and methodologies. This application represented a computerized prototype of a teacher assessment prescriptive diagnostic system. Borg and Gall (1979) stated that educational research and development (R&D) is a process to develop and validate educational products. It consists of studying research findings pertinent to the product to be developed, developing the product based on these findings; field testing it in the setting where it will be eventually used; and revising it to correct the deficiencies found in the field-testing stage. The 10 major steps required to develop an educational product for dissemination and distribution are:

1. Research and information collecting;
2. Planning;
3. Developing preliminary form of product;
4. Preliminary field testing;
5. Main product revision;

6. Main field testing;
7. Operational product revision;
8. Operational field testing;
9. Final product revision; and
10. Dissemination and distribution.

This study concentrated on the first five steps ending with the main product revision. The fifth step was the revision of the product as suggested by the preliminary field-test results.

The first step of the educational R&D cycle was divided into three major areas. The first two areas were generated from a review of selected literature. The first area described the theoretical relationships of Data Management to Data Base Management Systems. In the second area, the basic concepts of Data Base Management Systems methodology and general design were investigated. The third area focused on the design and implementation of the prototype using an integrated Data Base Management System that supported the PRO*FILE research. As such, the R&D cycle allowed for testing, evaluation and refinement of the prototype to determine the applicability at the systems (technical) level, the administration, faculty and student level. The revised prototype will be used in the on-going PRO*FILE research project.

Codd (1970), a researcher for the International

Business Machine (IBM) Corporation, published a paper on the relational approach to data base management. By pointing out the similarities between the data processing concept of a flat file (a 2-dimensional array of data items) and the notion of a mathematical relation (a 2-dimensional array of data items in normalized form) taken from discrete mathematics, Codd provided a rigorous theoretical foundation for the design and manipulation of data structures for Data Base Management Systems.

This study, using Codd's research as a conceptual framework as well as Iowa State University's policy and philosophy statements, was used to design and implement a prototype data base. The system hereinafter was referred to as the PRO*FILE System. It consisted of two major groups of data: (1) Student Records; and (2) Performance Elements. A typical student record contained historical personal data, past class work, evaluation results, and statements of advisement. A Performance Element summarized what knowledge skills or attitudes a senior ought to have. Performance elements were essentially reviews of the culminating knowledge/skills that students should have achieved. Following the R&D cycle, the initial prototype was evaluated by administrators and faculty from the College of Education and graduate research assistants from the Iowa State University's Research Institute for Studies in Education.

Evaluation by interview followed the Instructional Design Interview procedure developed by Richardson, Nartens, Fisk, Okun, and Thomas (1982). The procedure was used at the community college level and developed at Arizona State University. The methodology used the developed procedure at five meetings to acquire, confirm and ascertain if the preceived ideal was acquired for an instructional design. The procedure was used with four institutional age groups. The results of the study indicated that it was feasible to carry out multiple interviews and that the process yielded useful data. This processs and corresponding procedure can be used to evaluate the PRO*FILE System by diverse groups such as administrators, faculty and students.

The software development evaluation followed basic criteria published by the Federal Information Processing Standards Publication (FIPS PUB 99). The purpose of the federal study was to provide a structure that could be used as a basis for the analysis and classification of software development tools. The target population of this publication was all federal agencies. The methodology provided a framework for evaluation through the use of a taxonomy of tool features. While FIFS PUB 99 noted that software development is an emerging technology and any set of criteria must take into consideration circumstances peculiar to the local environment, the federal standard guideline was designed to evaluate data base management systems as a software development tool.

The data base design evaluation was founded upon the procedures developed by Atre (1980). Atre provided practical steps in quality data base design and operations. Much of the material was developed at the IBM Systems Research Institute and targeted for industrial use. The methodology provided a set of procedures in developing a data base that is predictably sound and able to satisfy the strictest performance criteria. The procedures were widely accepted by the computer industry. At the time of publication, over 60 universities were using Atre's procedures for classroom instruction. The procedures developed by Atre provided this study with a design aid and framework for the computer implementation of the PRO*FILE System.

From the evaluation results, the prototype was revised and refined as part of the on-going PRO*FILE research project.

Significance of the Study

Since there have been very few studies that automate the data management function for the improvement and/or assessment of potential teachers, this study investigated how Data Base Management Systems may be used as a prescriptive diagnostic processor. This study helped establish the applicability of the PRO*FILE design and determination of the PRO*FILE System's ability to meet the requirements of Iowa State University's College of Education.

Outline of Subsequent Chapters

Chapter 2 reviews the literature, defines the concepts of Data Management, Data Base Management Systems, Information Management, and presents the criteria that were used to evaluate the prototype. In addition to an overview of Iowa State University's hardware and software configuration, environmental and situational factors within the College of Education are examined in Chapter 3 to define the constraints under which the prototype was developed and evaluated. The data base design methodology and strategies are described in Chapter 4. A detailed analysis of the model and supporting software is presented in Chapter 5. The findings of the project based on the criteria used to evaluate the product is discussed in Chapter 6. Chapter 7 presents the recommendations of this developmental research project and final summary. Appendixes contain the details in the design of the educational product and methodologies used in the study. Although technical in content, the reader may find the details highly informative in any replication study or future consideration of using a Data Base Management System.

LITERATURE REVIEW

In order to determine how Data Base Management Systems' technology may be used in the improvement and/or assessment of potential teachers and graduates in a College of Education, it was necessary to obtain an understanding of data management. An understanding of the conceptual framework of data management made it possible to examine the concepts of data base management: normalization, information management, design methodology, and future trends. These four areas combined provided the necessary background information for this study. Also, the review examined teacher education assessment and performance monitoring.

Data Management

Descriptions of phenomena are referred to as data. Data correspond to discrete, recorded facts about phenomena from which we gain information about the world. Information is an increment of knowledge that can be inferred from data (Langefors, 1977).

The word "datum" comes from Latin and, literally interpreted, means a fact. Data do not always correspond to concrete or actual facts. Sometimes they describe things that have never happened or are imprecise at best. Tsichritzis and Lochovsky (1982) defined data as corresponding to descriptions of any phenomenon or idea that a person considered worth formulating and recording. Data will be of interest if they are worth not only interpreting in some manner, but also worth recording in a somewhat precise manner.

Data Management refers to the computer technology necessary for data collection, organization, storage, retrieval, and manipulation (Cardenas, 1979). The smallest unit of storage is a data item, also referred to as a field. A field may be a person's social security number or the color of hair. A collection of fields constitute a logical record. Associated with a record is a particular group of fields that together represent characteristics about a person, place, or thing. For example, a record in inventory may have one or all of the following fields; part number, part name, quantity on hand, and reorder point. Cardenas (1979) progressed to the next level and defined a file as a collection of occurrences of the same record type; for example, a file of employee records.

The American National Standard Vocabulary for Information Processing (ANSI, 1970) defined a computer program as a series of instructions or statements, in a form acceptable to a computer, prepared in order to achieve a certain result. During the early 1960s, programming languages (COBOL, FORTRAN) were used to create computer programs. These computer languages used the computer's file access system to physically move data from a storage device to and from the computer's memory for processing (Davis, 1981).

The COBOL implementations of the early 1960s were found to need additional capabilities beyond the fundamental COBOL reporting feature. Flynn (1974) stated that these additional capabilities started the evolution of data base management systems. In a later publication, Fry and Sibley (1976) substantiated Flynn's historical account. Supplemental COBOL capabilities included a SORT package and report writer. Cardenas (1979) defined these capabilities. A SORT package is a specialized module implementing some algorithm for sorting the records of a large file, and perhaps sorting and merging together multiple files. A report writer is a program utility that provides many facilities for editing output, tallying, formatting, and other related tasks needed to generate more complex reports; these facilities go beyond the primitive ones defined as part of the standard of a language. By the mid 1960s, COBOL was revised to include a SORT verb in the programming language. It was also during this time period that so-called report program generators were accepted. The primary one was RPG (Report Program Generator) and was highly successful (Shelly & Cashman, 1976).

In the mid and late 1960s, generalized file management systems were developed. These systems integrated various facilities of COBOL, report writers, report program generators, and SORT packages with a number of other useful data management facilities into a self-contained system (Flynn, 1974). Generalized file management systems operate like an application program on a host programming and operating system, and use the available access methods. One of the most important developments of these systems was that those users who could not write COBOL or FORTRAN code, could specify retrieval of records on the basis of practically any logical expression; they could use any data items of the record structure, without the limitations, burden and concern of its particular sequencing or particular single access key (Informatics, 1974).

In the late 1960s, the need for integration of large files into a data base and the need for application programs to access it effectively and efficiently led to the early data base management systems efforts (Fry & Sibley, 1976). Cardenas (1979) found that conventional file structures were inadequate for such demanding environments, and resulted in burdens in processing time as well as excessive auxiliary storage requirements. Cardenas (1979) further stated that generalized file management systems (GFMS), limited by the conventional single-file structures of their host operating systems, were also indequate.

In the late 1960s, a group of representatives from computer and software manufacturers, users in the Federal government and industry, and university researchers embarked in producing the Data Base Task Group (DBTG) Report published in early 1971 (DBTG, 1971). This proposed standard for generalized data base management systems (GDBMS) became the basis for a growing number of systems. The main functional relationships of data management systems in the 1970s and into the 1980s are summarized below (Cardenas, 1979):

1. Application programs in a conventional procedural programming language communicate with the GDBMS or data base/data communication systems via an appropriate language interface provided by the system.
2. Application programs in a high-level GFMS language communicate with the GDBMS via an appropriate interface usually provided with the GFMS.
3. A very high-level query language particular to each GDBMS or data base/data communication system is generally available for fast, low-volume data access.

A GDBMS system permitted the use of transaction oriented languages, as well as of a nonprocedural English-flavored query language of its own.

Starting in the early 1960s and over a 20 year period, the requirement for programming languages that were easier to use and that offered more capabilites for report generation and query by computer users was developed. However, computer users designing complex business applications found that additional capabilities were required to more fully benefit from the processing of relationships between data.

Data Base Management

Objectives of Data Base Management Technology

An understanding of the major concepts and objectives provided the basis for determining when and under what circumstances data base technology would be used. Date (1977) provided a comprehensive list of concepts and objectives for data base designers. These concepts and objectives were also validated by the works of Martin (1976), Cardenas (1979) and Bradley (1983). Each concept and objective is presented in Table 1.

TABLE 1. Objectives in Using a Data Base Management System

---

1. Data independence - Denotes independence or insulation of application programs or users from a wide variety of changes in the specific logical organization, physical organization, and storage considerations of the computerized data base.

2. Data shareability and non-redundancy of data stored - Enables applications to share an integrated data base containing all the data needed by the applications and thus eliminate as much as possible the need to store data redundantly.

3. Relatability - The ability of defining relationships between records or entities at the local level just as conveniently as defining the records themselves.

4. Integrity - The coordination of data accessing by different applications; propagation of updated values to other copies and dependent values; and the preservation of a high degree of consistency and correctness of data.

5. Access flexibility - The ability to access any part of the data base on the basis of any access key(s) and logical qualification, via a high-level non-procedural query language. For browsing through the data base, or via input/output statements issued from programs written in conventional procedural programming languages.

6. Security - A proper mechanisms to assign, control, and remove the rights of access of any data items or defined subset of the data base. A data item must be fully protected from unauthorized intrusion, be it accidental or malicious.

7. Performance and efficiency - In view of the size of the data base, and of demanding data base accessing needs, good performance and efficiency are major requirements.

Table 1 (Continued)

8. Administration and Control - The functions of data base design, administration, and control reside with the Data Base Administrator (DBA). The DBA is an experienced and highly qualified individual charged with such responsibility and other responsibilities that must be lifted away from any one user for the overall good.

Relationships

Atre (1980) defined a relationship as a mapping or linkage between two sets of data (entity). It can be a "one-to-one", "one-to-many" or "many-to-many". Cardenas (1979) added one other relationship: a "many-to-one". For example, a student is assigned a student identification number and that number uniquely defines the student (one-to-one relationship). Likewise, a student may be assigned to one advisor but the advisor may have many students (one-to-many relationship). On the other hand, many students may be assigned to one instructor for a particular course (many-to-one relationship). Finally, a student has many instructors and instructors have many students (many-to-many relationship). The relationship is as important and as definable as any attribute or record. These relationships were the foundation for developing the logical organization of a data base in the next sub-section.

Logical Organizations

Data base technology introduced powerful logical data base structures (models) made up of interconnected records. These were published by Martin (1977) as:

1. Tree or hierarchic model;
2. Network model; and
3. Relational model.

Cardenas (1979) stated that the foundation of the above three models is based on the concept of a flat file. He defined a flat file as one in which each record instance has a similar number of fields. For example, a fixed length logical file structure without repeating groups is a flat file. Relationships between flat files sometime exist and for that reason application software should be able to define and utilize those relationships in processing data.

During the late 1960s, the early data base management systems were based upon the tree or hierarchic model. These systems included IBM's Information Management System (IMS) and MRI's SYSTEM 2000. Knapp & Leben (1978) presented procedures for modeling data using the "one-to-one" and "one-to-many" relationships only. MRI (1975) also argued that data could be represented using the same relationships to form a tree or hierarchic model.

It was also during the late 1960s that business, government, and the computer industry formed the Data Base Task Group (DBTG) under the supervision of the Conference on Data Systems Languages (CODASYL). A result of this group was the publication of the DBTG Report (DBTG, 1971); the report recommended implementation of the Network model. This Network model became known as the CODASYL model (Olle, 1978). The CODASYL model uses all four relationships previously defined. Its major consideration was the use of the "many-to-many" relationship. Throughout the 1970s, DBMS systems that were based upon the DBTG standards were made available in the commercial market for distribution. These included IDMS (Perron, 1977) and DMS-1100 (UNIVAC, 1978). These and other similar DBMS systems had the capability to model data in a flat file, hierarchic or network representation.

The ability to model data in a hierarchic or network representation also carried considerable overhead in the design and programming effort. This overhead largely included the training needed by analysts and programmers to understand and use the data base structure when considering the four relationships involved. The U.S. Government (FIPS PUB 77) cautioned its agencies on the development effort using hierarchic or network based DBMS's due to the complexity of the design and implementation. DBMS users were becoming overburdened by complexity and sought simpler solutions.

While the DBTG was defining the CODASYL networking model in the late 1960s, a researcher at International Business Machines (IBM) Corporation was preparing a paper on an approach that was based on mathematical set theory. Codd (1970) argued that any group of data elements could be broken down into a series of 2-dimensional relations (files) whereby a unique key would identify all data elements in the tuple (record). Codd's research provided the framework for the fourth DBMS logical model referred to as relational. Prior to Codd's presentation, there were a number of implementations of relational data bases (Levein & Maron, 1967), (Childs, 1968), (Ash & Sibley, 1968), (Feldman & Rovner, 1969). However, from an historical point of view Codd received the most credit for the development of the relational model. IBM built a prototype called System R (Martin, 1977) that validated the relational data base approach but numerous implementation problems precluded a successful entry into the market. Bradley (1983) stated that the relational approach was not widely used in the early 1980s because it took a very long time to bring a comprehensive relational system to the market. However, Bradley (1983) projected that the difficulties would disappear as time went on, and expected that the relational approach would be commonly used (if not the most commonly used) in business by the second half of the 1980s.

Normalization

Cardenas (1979) defined normalization as the process by which any nonflat data structure, such as a COBOL, network, or tree data base, can be transformed by a data base designer into a set of normalized relations, that is, a set of flat relations that have no repeating groups. An unnormalized relation has at least one domain which is in reality another relation. A normalized relation has only simple domains, that is, domains that are not in turn another file.

In attempting to lay out the relationships between data items, the designer should be concerned with which attributes are dependent on which others. Martin (1977) summarized the phrase "functionally dependent" as stating that a given data item (B) is functionally dependent on a given data item (A) is equivalent to saying that A identifies B. In other words, if one instant in time the value of A is known, then the value of B is determined.

Codd (1970), of IBM, outlined the concepts of 1st Normal Form (1NF), 2nd Normal Form (2NF) and 3rd Normal Form (3NF), pointing out that 3NF relations were without dependencies that would cause updating difficulties.

However, further research with compound key relations indicated that the 3NF was not the end to the normalization process since it was possible to have subkey fields functionally dependent on other fields. This was discovered by Codd and another researcher (Boyce), and the Boyce-Codd Normal Form (BCNF) was invented to describe relations that were in 3NF but did not have these undesirable subkey field dependencies (Bradley, 1983). Finally, Fagin (1977) discovered binary join dependencies and the 4th Normal Form (4NF) was conceived to describe relations without binary join dependencies and without undesirable functional dependencies. Date (1981) defined the 5th Normal Form (5NF) relations to eliminate a more subtle type of dependency called a join dependency; however, the dependency involved is so contrived and uncommon that it is unlikely ever to occur in a practical data base design situation. Bradley (1983) suggested that designers may safely forget about 5NF relations.

Bradley (1983) summarized the normalization process as:

1. Given all conceptual files, select all relations without repeating groups. The 1NF relations.
2. Select relations where no nonkey field is functionally dependent on a subkey field. The 2NF relations.

3. Select relations where no nonkey field is functionally dependent on another nonkey field. The 3NF relations.
4. Select relations where a subkey field is dependent on any nonkey field. The BCNF relations.
5. Select relations with no binary join dependencies other than those that are functional dependencies. The 4NF relations.

Herrick (M. A. Herrick, Margann Associates, Cambridge, Mass., August 18, 1983) summarized the normalization process for the lay reader as nonkey attributes depend upon the key (1NF), the whole key (2NF), and nothing but the key (3NF).

Evaluation Tools

Evaluation tools for DBMS's are divided in two broad categories: Tools that aid in the selection of a DBMS; and, tools that aid in the performance and optimization of an installed DBMS.

Cagan (1973) proposed a list of evaluation variables in the investigation and evaluation of commercial DBMS packages for immediate or future acquisition. Cagan's list included: Cost, equipment, languages, file structure requirements, file formats, program conversion, operating efficiency, changeover, benchmark, report formats, training, maintenance, modification, and user experiences. A more comprehensive list, that included Cagan's variables, was provided by Cardenas (1979). Cardenas argued that evaluation must consider the incorporation of attractive features and strong points of competing data base systems. Each vendor of a commercial DBMS offered evaluation criteria but they pointed out the benefits of their own particular DBMS (Software AG of North America [SAG], 1975). The U.S. Government, faced with the prospect of purchasing DBMS's throughout its agencies, published a guideline for the evaluation and comparison of software development tool (U.S. Department of Commerce, 1983). This guideline provided a taxonomy of tool features for evaluation and comparison. In addition, the guideline proposed a sequence of events for the acquisition of tools.

Once the DBMS was installed, the vendor's reference manuals offered criteria on how to evaluate the performance of DBMS applications and optimization techniques (MRI Systems Corporation [MRI], 1975 & SAG, 1975). The U.S. Government (U.S. Department of Commerce, 1980) stated that there were no pertinent international, national, or Federal standards, therefore all commerical DBMS's are unique. In order to facilitate planning for DBMS applications in its agencies, the U.S. Government published a guideline for planning and management of database applications (U.S.

Department of Commerce, 1980). The guideline pointed out that there was no government-wide recommendation as to DBMS suitability for specific application needs.

Information Management

Mandell (1982) defined a Management Information System (MIS) as a formal network that extends computer use beyond routine reporting and into the area of management decision-making; its goal is to get the correct information to the appropriate manager at the right time. Murdick & Ross (1975) further stated that an MIS identified people, computer equipment and computer programs to manipulate the data. To fully support an MIS, an integrated data base that spans organizations is required (Murdick & Ross, 1975). Data Base Management Systems support an integrated data base that in turn supports the MIS concept (Adams, Wagner, & Boyer, 1982).

MIS's support managers throughout the organization in both the batch and on-line modes (Murdick & Ross, 1975). The on-line mode uses a telecommunications network whereby individual users process data via a computer terminal. The network can be hard wired into the computer or may use a variety of telecommunications capabilites, such as dial-up, micro wave and satellite connections (Sherman, 1981). IBM (1982) has chartered a great deal of research on computer user's on-line productivity in various computing environments. IBM (1982) reported that studies proved user productivity increased as response time decreased. The phenomenon is similar to an individual's attention span. IBM's research on response time also has benefits in an MIS environment.

One concern of public officials, as reported by Grady (1981), is how to control computer costs. Grady stated that the in-house development of sophisticated software may be the largest computer-related waste. With few exceptions, software development, outside of research, made poor economic sense. Markle (1983) argued that whether in higher education or industry, developing a chargeback system may minimize users costs and, more importantly for the MIS organization, may substantiate their portion of the overall increase in data processing costs of the corporation or institution.

Although very few management functions have been automated, advances in information retrieval, processing, and display technologies have led to significant computer applications that help people perform management functions (Alter, 1980). Alter stated that since the purpose of these systems is to support managers responsible for making and implemanting decisions rather than to replace them, these applications are often called decision support systems (DDS). Gessford (1980) also contended that the facts must be relevant and newsworthy to justify the system. Alter characterized DDS's as being actively used: line, staff, and management activities; oriented toward overall effectiveness; focused on the present and future; and emphasis on flexibility and ad-hoc utilization. Microcomputer technology, when combined with decision support technology, can provide a powerful problem-solving tool for administrative use in institutions of higher education (Brown & Droegemuller, 1983).

In a 1981 study, Russell (1982) found that the development of computer-based information systems in American higher education increased substantially in the past decade. Russell further stated that the higher education community may find that the development of computer-based information systems is no longer simply an ideal, but has become a necessity. In a comprehensive study of administrative computing at leading institiutions of higher education, Neiheisel (1981) reported that file oriented structures rather than data base structures were the predominant organizational basis: Only one institution identified a data base structure as being currently utilized. Neiheisel further stated that student-oriented applications were primarily integrated. One of the most consistent and major problem areas identified in the

Neiheisel study was the attraction and retention of systems analysts and/or the skills associated with such positions. In a survey of many institutions of higher education, Plourde (1981) reported that some institutions acquired DBMS because it was fashionable, but there was a desire for developing integrated data bases as a means of developing management information systems. Finally, Steingraber and Kunkel (1982), in a report on the management perspective of an on-line/data base system at Washington State University, stated that implementation of on-line systems on campus is raising the productivity of the staff.

Design Methodology

Despite the wealth of literature about information systems and their uses, there is very little good literature on how to put a system together (Orr, 1977). In the late 1960s, data processing management recognized the need for better ways of analyzing, designing, and implementing automated systems. This need arose from the increasing complexity of computer software and the absence of any simple, coherent, workable methodology. Flowcharting tools were the earliest development aids for the higher level languages, such as COBOL and FORTRAN (Stern, 1975). Other methodologies (Awad, 1979; Thieraul & Reynolds, 1980), offered sound design tools for systems using traditional programming languages. Structured methodology started at the programming level and worked its way up to design. Orr (1977) provided a structured methodology using Warnier Diagrams. At the time of this writing, popular design methodologies used by the information industry were procedures design by DeMarco (1979) and Yourdon & Constantine (1979).

As more DBMS's were brought into computer centers, design methodologies were developed that considered DBMS technology (Wetherbe, 1979). Vendors of DBMS's offered specific design procedures particular to their product (Cullinane, 1977). Vendor procedures often overlooked the benefits of the normalization process no matter what logical data base model was to be used (Atre, 1980). DBMS structured design procedures, developed by Atre, were generic to any DBMS. These procedures offered the benefits of normalization and processes to decompose relations to fit any DBMS logical model.

At the time of this writing, some of the DBMS vendors offered 4th Generation Languages (4GL) (SAG, 1983). Sholtys (1983) considered 4GL as being user friendly, having capabilities that conventional programming languages have, work on-line, and usually require heavy commitment of computer resources. Sholtys further stated that in designing information systems, 4GL can be used to quickly develop a prototype system, which is revised and expanded as the user clarifies his or her requirements. Using 4GL to prototype all or a portion of an information system is now considered one of the many structured design tools available (U.S. Department of Commerce, 1983).

Teacher Education Performance Monitoring

Efforts to improve teacher education in the United States have evolved for the past 150 years. Concern about the quality of teaching performance led Samuel R. Hall to open the first private school for teachers in 1823. Hall's book, Lectures_on_School_Keeping, published in 1829, might be considered the first systematic attempt to identify competencies of American teachers. Sandefur (1981), in a study on state reactions to competency assessment in teacher education, reported that by the 1970s, the public and their state legislators perceived that American students were not adequately educated. By 1980, almost 40 states had adopted measures requiring some form of minimum competency examinations for students. Sandefur stated that, by October 1980, at least 29 states had taken some kind of parallel action regarding competency assessment of teachers. Some states sought to regulate entry into the profession, others to control the certification process, and some to do both. Some states and institutions have relied upon already available tests such as the National Teacher Examination. These examinations evaluate on the basis of what courses students have taken and not on how well a future teacher may perform.

Kniker (1982) contended that a goal of teacher competency programs should be to provide students with regular feedback on their performance regarding the published competencies. Students should be appraised of their strengths and needs and whenever possible, given suggestions for resources.

Kauchak and Eggen (1978) studied the use of written simulations in the measurement of teaching competencies. Results of the study indicated that the use of written simulations may help to ascertain understanding of skills before students are asked to demonstrate these skills in micro-teaching or classroom settings. Casteel and Gregory (1975) investigated the degree to which skills may be learned and practiced through microsimulation and then used under microteaching conditions. The results of this study indicated that teachers may acquire, practice, and learn to use a cluster of technical teaching skills.

In a study to determine the relative effectiveness of written and audiotaped feedback to students, Moore (1977) reported that teachers found audiotape responses less time consuming than written responses. Additionally, Moore stated that students had a more positive attitude toward tape-recorded feedback than written feedback. Kniker (1982) argued that teacher assessment requires a multidimensional approach that supports the use of paper and pencil measures when appropriate, but requires, as well, more complex measures of performance. Kniker was convinced that some of these activities can be successfully converted to a computer format.

Adams (1978) developed a simulation of an illustrative model for evaluation of teacher education graduates. The variety and magnitude of data collected from this evaluation system required the use of computer-assisted data processing, storage, and analyses. Adams stated that for the evaluation model to have impact on teacher education programs, some means must be established to communicate the evaluation outcomes to teacher educators. Smith and Shallwani (1978) implemented a computer simulation on various aspects of the supply and demand for teacher personnel. The success of the simulation indicated that users preferred the interactive mode of operation and found the system relatively easy to use.

A series of computer programs designed to provide a dynamic simulator for interactive teaching was developed and tested at the City University of New York (Confessore, 1974). The results of this study indicated that the greatest promise of the computer lay in the opportunity for educators to practice selected options using dynamic interactive teaching simulators. Finally, future studies were recommended for additional programs to make possible the specific man-machine interactions desired and the development of an adequate data base. Sitko, Semmel & Olson (1974) developed a prototype computer-assisted teaching training system to help train special education personnel. The prototype indicated that the computer was a versatile and comprehensive delivery system. Analysis indicated, that with creative application, a similar computer-assisted system may assist in the accomplishment of training objectives for competency or performance-based training programs in teacher education.

Formative Evaluation and Educational Products

Scriven (1967) stated the purpose of formative evaluation generally is to help develop a new program. Anderson & Ball (1980) included such activities as appraisal of the competencies of the program staff and other aspects of the delivery system, well as examination of program content. Borg & Gall (1979) stated that the function of formative evaluation was to collect data about educational programs while they are still being developed.

Research-based development, referred to as educational research and development (R&D), within the formative evaluation concept offers a framework for development of educational products. Educational R&D appears to be the most promising strategy we now have for improving education (Borg & Gall, 1979). The educational research and development (R&D) cycle is a process used to develop and validate educational products.

Evaluation of educational R&D products requires analyzing learning outcomes. Gagne' and Briggs (1974) stressed the importance of analyzing learning outcomes since each type of learning outcome requires the use of different instructional techniques. Richardson, Martens, Fisk, Okun and Thomas (1982) developed instructional design interview procedures. Results of the pilot study indicated that it was feasible to carry out multiple interviews and that the process yielded useful data.

Summary

Although a selected review of the literature showed that there is information available on data management, data base management, information management, design methodology, and teacher education performance monitoring, there is limited information on data base management systems (DBMS) supporting teacher education performance monitoring in colleges of education. Therefore, research on DBMS's for teacher education performance monitoring would be valuable to colleges of education, and to colleges and universities offering teacher education curricula.

ENVIRONMENTAL AND SITUATIONAL FACTORS

This chapter described those environmental variables which conditioned the development of the educational R&D project during the years 1981 to 1983. These variables included the University, the state certification process, computer hardware and software, the College of Education, governance of teacher education, and the PRO*FILE project. The study was subject to the available hardware and software at Iowa State University. The purpose of this composite of information was to provide the reader adequate background to visualize the environmental constraints in designing and implementing the prototype.

Iowa State University

Iowa State University is a land-grant institution located in Ames, a community of 50,000 population just 30 minutes north of Des Moines, Iowa's capital. The University enrollment in 1981 was over 23,000 of which approximately 3,500 are graduate students (Iowa State University 1981'82 Graduate Students, Information for Prospective Graduate Students).

Iowa State offered facilities for study and research in agriculture, design, education, engineering, home economics, sciences and humanities, and veterinary medicine. The University was accredited by the North

Central Association of Colleges and Secondary Schools and other accrediting agencies.

State Certification Process

The Iowa Professional Certificate was recommended for those who hold a bachelor's degree from Iowa State, who desired careers as teachers, and who completed the following:

1. All requirements of an approved teacher education program, including the human relations requirement.

2. A minimum of 42 semester hours in courses designed to serve the general needs of college students. Credits listed were minimum requirements:

Credit		Subject Group
9	I.	Biological science, physical science, and mathematics
9	II.	Social sciences
6	III.	Humanities
9	IV.	Communication skills

1	V.	Health, dance, physical education, safety

34		
8		Additional credits in above areas

42		

3. As part of a total educational program, the prospective teacher needed to complete certain studies related directly to the profession of teaching. All students in teacher education took the following courses:

Credit	Subject
3	The School in American Life
1	Instructional Media
3	Educational Psychology
2	Multicultural Awareness and Non-sexism in the Classroom

Additional courses required by specific teaching areas include:

a. Elementary Education

b. Prekindergarten-Kindergarten Education

c. Secondary Education

d. Professional Courses in Areas of Specialization

4. For full-time teaching in secondary schools an approved subject matter concentration of at least 30 semester hours was required. A second subject matter area of at least 20 semester hours was possible, but not required. Requirements differed by department. For example, Elementary Education required 47 semester hours and Industrial Education required 43 semester hours in the subject area (General Catalog, 1981'83).

Computer Hardware and Software

Computer facilities were centrally located in the Iowa State University Computation Center. The Center provided research and educational computing services to the university community. There were two major hardware suites; a suite of hardware is the combination of the computer's physical components. The main computing system was the National Advanced Scientific (NAS) A/S 6. This system could be accessed from various sites on campus and Ames area terminals. The second computer system was a suite of four Digital Equipment Corporation (DEC) VAX 11/780 computers. This system ran under MVS and was used primarily for instructional interactive computing by the Computer Science and Engineering Departments. Both computer systems were in use 24 hours a day, seven days a week, except for maintenance periods.

A list of computer hardware located in the Computation Center is provided in Table 2. The hardware list includes only relevant computer components that were contemplated in designing the prototype.

TABLE 2. Iowa State University's Computer Hardware

Main Digital Facilities
- 1 NAS AS/6 Model 1 (IBM 370 compatible computer)
- 4 million bytes of high-speed memory
- 3 7330-10 Itel disk drives (100 megabytes per drive) with control unit
- 10 8650 STC drives (635 megabytes per drive)
- 3 STC 3670 9-track magnetic tape units (1600/6250)
- 500 timesharing terminals
- 1 Memorex 1270 Communications Controller
- 2 Vadic Data Stations (phone lines: 44 for AS/6, 25 for VAX systems)
- 1 STC 1200 printer (1200 lpm, Student Services)

Instructional Interactive System (VAX)
- 4 Digital Equipment Corporation (DEC) VAX 11/780
- 4 million bytes of high-speed memory on each VAX
- 2 million byes of high-speed shared memory
- 1 DEC RM05AC disk drive (300 megabytes unformated)
- 11 SI 9766 disk drives (300 megabytes unformated)
- 1 DEC LP05 (300 lpm)
- 1 DEC TE16 tape drive (1600 BPI)
- 440 timesharing terminals

Iowa State University offered a comprehensive library of software products for academic computing. Tables 3 and 4 provide an overview of software capabilities for both hardware suites. Definition of selected acronyms are presented in the Glossary of Terms, Appendix A. Further information and detailed descriptions of hardware, software, and services are described in the _User's Brochure_, Iowa State University Computation Center, Ames, Iowa (Spring, 1983).

TABLE 3. Iowa State University Software Capabilities On The AS/6 Computer

---

Program Libraries
- International Mathematical and Statistical Libraries
- Locally written ISU Program Library
- SHARE Library

Programming Languages

- APL	- MOS65	- PDP-11	- SNOBOL
- ASSEMBLER	- PASCAL	- PL/1	

Programming Packages and Other AS/6 Software
- GEAR (Differential Equations)
- Linear Programming and Simulation
  - . GPSS
  - . SIMSCRIPT 11.5

Plotting

- AUTOPLOT	- SIMPLOTTER

Text Processing (Formatting)

- ISUTHESIS	- SYSLABEL	- SCRIPT
- SYSPAPER	- SYSPUB	

Utilities

- FLOWCHART	- IOPROGM	- SYNCSORT

- LABELS and SLABLES
- MATCHUP, SNAP78, and UPDATE

WYLBUR
- ORVYL is a timesharing monitor that provides the capability of executing user programs in an interactive environment
- SPIRES (Stanford Public Information REtrieval System) is a generalized data base management system. It is designed to handle bibliographic and text applications, as well as general data management.
- WYLBUR is a timesharing system for manipulating various kinds of text by providing online interactive text editing capabilities.

Statistical

- SAS	- SAS/GRAPH	- SPSS

TABLE 4. Iowa State University's Software Capabilities On The VAX Computer System

---

VAX Software

Program Libraries
- PORTLIB (Bell Labs Portable, Outstanding, Reliable, and Tested Library)
- Locally written and supported software in the PUBLIC directory
- Games in the GAMES directory
- User-written or maintained programs in the CLASSLIB directory

Programming Languages
- BASIC
- Dimension Author Language (DAL)
- FORTRAN-77
- PASCAL
- Six Graphing Packages

College of Education

The College of Education provides degree programs leading to certification in elementary education, industrial education, and physical education as well as a professional sequence of courses for all students at Iowa State seeking a teaching certificate (General Catalog, 1981'83). The College of Education's organizational chart is presented in Figure 2.

The teacher education program at Iowa State University is accredited by the National Council for Accreditation of Teacher Education. The major fields of study for teacher certification are presented in Table 5.

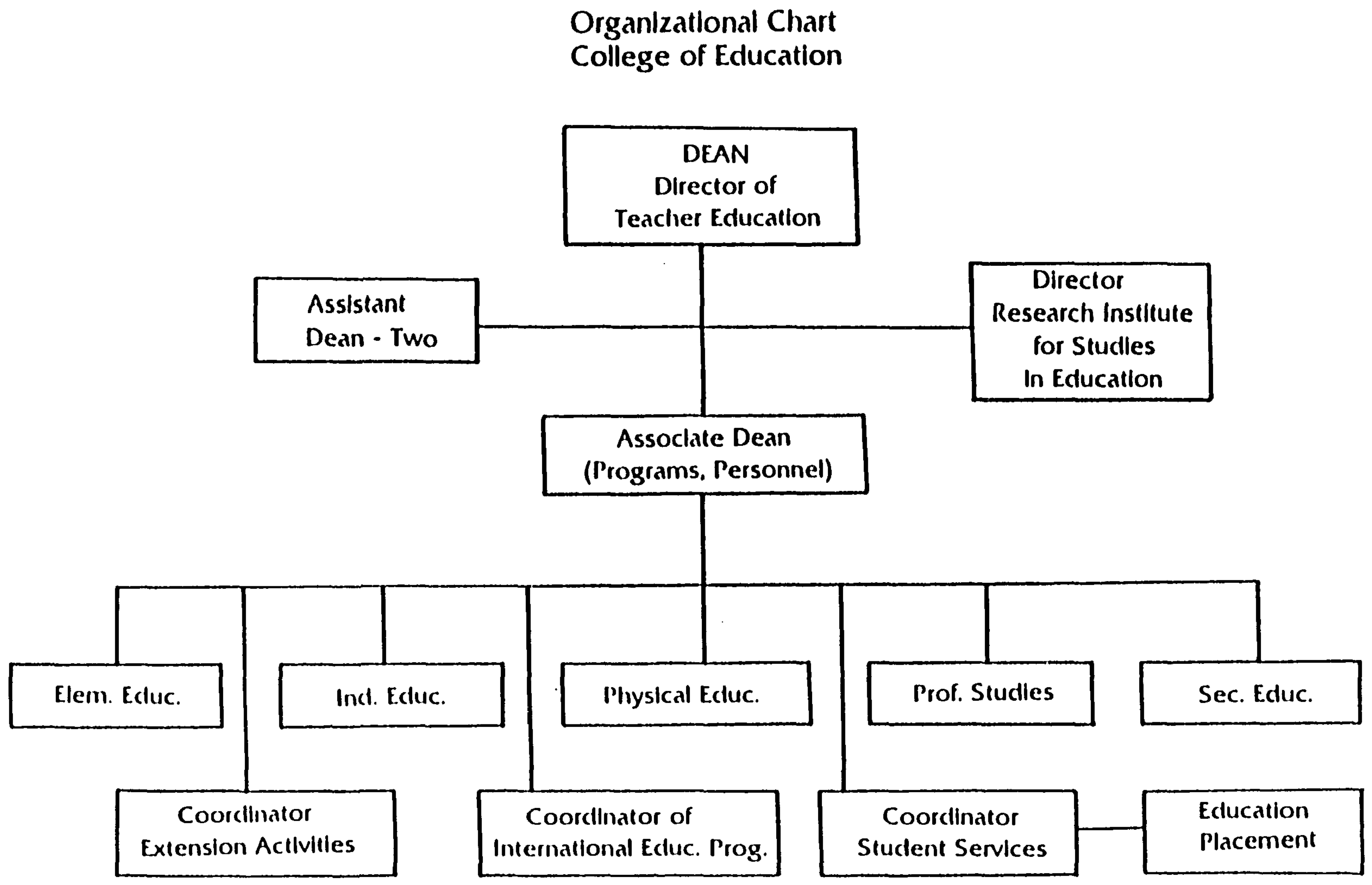


FIGURE 2. College of Education Organization Chart

TABLE 5. Major Fields of Study for Teacher Education at Iowa State University

College	Major field
College of Agriculture	- Agricultural Education
College of Design	- Art Education
College of Education	- Elementary Education
	- Industrial Education and Safety Education
	- Physical Education and Health Education
College of Home Economics	- Home Economics Education
	- Teaching Prekindergarten-Kindergarten Children
College of Sciences and Humanities	- Biology
	- Chemistry
	- Earth Sciences
	- English
	- Foreign Languages & Literatures
	- General Science
	- Journalism and Mass Communications
	- Mathematics
	- Music
	- Physical Science
	- Physics
	- Social Studies
	- Speech

Governance of Teacher Education

At Iowa State University, the University Teacher Education Committee admitted students wishing to enter the Teacher Education Program. The committee was comprised of a chairman (Associate Dean, College of Education), a secretary (Director, Educational Placement, College of Education), and representative members from each of the five colleges: Agriculture, Education, Home Economics, Science and Humanities, and Design. Each college had a Teacher Education Committee. Members of the Teacher Education Committee represented each area of specialization within the respective college. Within each area of specialization, there resided a Selection Committee. The Selection Committee was the beginning level of the approval structure and initiated the process for a student to be admitted into the Iowa State Teacher Education Program.

Iowa State University students, who desired to acquire a teaching certificate, were required to be admitted to one of the teacher certification programs. Students concurrently enrolled in the college and departments of their major study. A student seeking admission to the Teacher Education Program must be accepted by a selection committee for the specific program which he or she seeks to enter. Factors considered in evaluating applications included scholarship, interest in teaching, character, and physical and mental health. Recommendations by the selection committees were forwarded to the respective colleges' Teacher Education Committee. Likewise, recommendations by a colleges' Teacher Education Committee needed confirmation by the University Teacher Education Committee before admittance to the program in teacher education is granted. A 2.3 quality-point average was required for full admission to the Teacher Education Program, and this minimum average was required of students through graduation. Further information and details are described in the brochure Teacher_Education_Admissions Policies_and_Procedures, College of Education Quadrangle, Iowa State University, 1982. Once students were admitted, an automated system might track the progress of these students and report composite information to the various committees. Such a capability would help in the quality control of students by discipline.

PRO*FILE

The purpose of the PRO*FILE System was to increase the already high level of competency of Iowa State University graduates as beginning teachers. PRO*FILE was a specific application from Iowa State University's College of Education's ongoing research project to investigate the use of the computer and software as a prescriptive diagnostic processor. PRO*FILE assisted teacher education candidates during their undergraduate years by providing English-like software for individualized instruction on the basic concepts of teacher education. The first goal of the PRO*FILE System was to provide teacher education candidates during their undergraduate years with individualized work in selected generic performance elements associated with teaching. Another goal of the PRO*FILE System was to provide guidance to students from advisers and faculty via regular adviser contact plus Admission, Interim, and Exit Interviews. The last goal was student self-help in areas of student interest, as well as needs, through the computer-based Performance Elements independent study materials.

As a part of the PRO*FILE System, each student received a PRO*FILE Notebook containing a record of undergraduate experiences and evaluations. The notebook was used by the student and faculty throughout the student's program of study. The need for the PRO*FILE Notebook was to help in the evaluation of the students progress and planning the program of study. The notebook helped in tracking the proficiency of students in order to maximize the level of competencies attained for professional teaching. Yet to be resolved was the issue of who can enter or change data in a student's record. The student could be responsible for this task or a combination of people could maintain the record, such as students, faculty and advisers that have been granted access to certain portions of the record.

The PRO*FILE Process consisted of 10 steps. A student planning a teaching professional program would follow the ten step sequence as presented in Table 6. The rationale for using this sequence of steps was that it provided the foundation, based on the student's past performance and desires, to formulate a program of study to meet the individual needs of the student. This process allowed for periodic evaluation and adjustments in the program of study to ensure that the basic competencies are acquired. At the time of graduation, the student's PRO*FILE Notebook contained a history of his/her teacher preparation experience. The Final Assessment Battery identified strong and weak areas. Through the use of independent study or formal course work, the prospective teacher expanded their knowledge based upon professional needs.

Prospective employers may desire the candidate for a teaching position to bring the notebook to their interview. This presented two issues. First, can an employer require review of the PRO*FILE Notebook prior to the final employment decision? And secondly, would students be

TABLE 6. The 10 Step PRO*FILE Process

STEP 1. Pre-admissions course work, early in-school experiences, and background personal data file development.

STEP 2. Initial Assessment Battery (IAB)

STEP 3. Admissions Interview

STEP 4. Work with Performance Elements reflecting personal interests and recommendations by faculty and/or adviser.

STEP 5. Progress through coursework and Teacher Education Program, working on strengths and needs through the independent study materials available for each Performance Element.

STEP 6. Interim Interview with faculty to review progress and to receive further direction and guidance.

STEP 7. Continue work with Performance Elements, as in Step 5.

STEP 8. Student teaching experience--a time to reinforce and refine elements, concepts and skills previously introduced through coursework and short-term in-school experiences.

STEP 9. Final Assessment Battery--the post-test form of the IAB (Step 2) is taken. Student also does a final update on his PRO*FILE Notebook in preparation for the Exit Interview.

Step 10. Exit Interview. This combines student self-evaluation and faculty evaluation of the student's readiness for classroom teaching. The Exit Interview also includes student evaluation of the Teacher Education Program.

willing to enter marginal data for review, or on the other hand, even willing to show the notebook to the prospective employer? These two potential issues needed resolution prior to defining the scope of the PRO*FILE System.

Taken together, the 10 steps of PRO*FILE provided the student, by the time of graduation, a comprehensive description and analysis of his/her academic abilities, teaching skills and professional attitudes.

Performance Elements mentioned in Step 4 were divided into seven broad areas. Details within each area can be found in Appendix I. The Master-list of Performance Elements identified by the PRO*FILE Task Force are found in Table 7. The Performance Elements contained the information on teacher competencies that had a standardized format and were centralized for ease of use. Students studied the Performance Elements as required by faculty or for their personal development. A typical Performance Element was comprised of five sub-units. While the structure of the format offered flexibility, as a minimum the following modules were included:

a. A paragraph introducing the module;

b. A list of module objectives;

c. The ISU courses which were relevant to the module topic;

d. Resources on the topic, these included

TABLE 7. Master List of Performance Elements

Area I.	Knowledge of Education (4 sub-areas consisting of 24 modules)
Area II.	General Teaching Skills (3 sub-areas consisting of 10 modules)
Area III.	Self-Concept and Goals in Education (3 sub-areas consisting of 13 modules)
Area IV.	Planning Skills (5 sub-areas consisting of 18 modules)
Area V.	Implementing Instructional Plans (6 sub-areas consisting of 19 modules)
Area VI.	Evaluation and Diagnosis (5 sub-areas consisting of 13 modules)
Area VII.	Management (3 sub-areas consisting of 8 modules)

books, journals, films, and tapes;

e. Several activities that helped the student check out his or her new comprehension or skill; and

f. Persons at ISU or in the surrounding area who had expertise in this topic.

Some modules could included a short pre-test and post-test. Additionally, modules could take advantage of the branching capability via menus. The broad nature of a particular performance element module suggested that there be several distinct objectives and that not every student user need complete every objective.

Summary

This chapter discussed the environmental factors which influenced the PRO*FILE Systems design during the years 1981 to 1983. The chapter began with an overview of Iowa State University. The State of Iowa's certification process was presented, detailing the requirements for teacher certificaion. An overview of Iowa State University's hardware and software capabilities was discussed. The fields of study within the College of Education were identified next. The governance of the teacher education was presented in order to acquaint the reader with the overall process. Finally, the PRO*FILE

System goals and components were discussed. Iowa State University's computer hardware and related software could support the PRO*FILE System prototype. However, the design would be restricted to university systems software only.

DESCRIPTION OF THE DESIGN

Introduction

This chapter presented the procedures for designing the computerized PRO*FILE Systems' prototype. The design was guided by the first two steps of the educational research and development (R&D) cycle described in Chapter 3. They were:

1. Research and Information Collecting, and:
2. Planning.

A review of selected research on Data Base Management Systems design methodology deduced a conceptual framework for the PRO*FILE project. The design of the prototype to support the ongoing PRO*FILE research project applied objectives and methodologies inherent to Data Base Management Systems.

Research and Information Collecting

Historical_Perspective

In May 1981, Professors Joan C. Breiter and Charles R. Kniker were asked by the Dean of the College of Education to study teacher competence during the First Summer Session, 1981. Carole Schneider, a graduate student, was also assigned to help with the project thus forming the PRO*FILE Task Force.

The PRO*FILE Task Force presented to the Dean of the College of Education the results of a feasibility study on the Personal Profile Analysis (C. R. Kniker, Secondary Education, ISU, July 16, 1981). The study indicated that a Personal Profile Analysis should include competencies, experiential locations, diagnosis and remediation, and graduate follow-ups. The Task Force recommended that research continue and that the use of the computer be explored for storing data.

Prior to this study, a previous attempt to computerize portions of the PRO*FILE System took place in the summer of 1982. The PRO*FILE Task Force acquired the services of a junior computer programmer to design and program the software using a higher level programming language called FORTRAN (Formula Translation). A memorandum from the Task Force submitted to the Dean of the College of Education and the Director of RISE reviewed the progress of that attempt to computerize PRO*FILE (C. R. Kniker, Secondary Education, ISU, July 16, 1982). The memorandum concluded that the use of the higher level language (FORTRAN) did not succeed due to the complexity of the system and the inherent short-comings of the computer language used. FORTRAN was inappropriate for the PRO*FILE System due to the flat file nature of the data structure, hard coded program requirement and inflexiblity to change. FORTRAN was better suited for mathematical problem solving, and not text manipulation and handling.

On July 21, 1982 the PRO*FILE Task Force and the Director of RISE met with the Dean of the College of Education and the Associate Dean. The Task Force and Director both recommended that a systems analyst be hired who could design and program a student record retrieval system during the Fall Semester, 1982 in support of the ongoing PRO*FILE research effort. Additionally, the Task Force requested that the Director of RISE research computer resources and develop alternatives for computerizing portions of the PRO*FILE System (C. R. Kniker, Secondary Education, ISU, July 21, 1982).

The PRO*FILE Task Force presented an overview of the PRO*FILE System to members of the Department of Public Instruction, State of Iowa, at Iowa State University on October 12, 1982. This author, a graduate research assistant for RISE and in attendance, discussed with the Director of RISE the presentation and made two basic recommendations concerning PRO*FILE. The first recommendation was that a Data Base Management System (DBMS) be used to implement PRO*FILE, rather than a higher level language such as FORTRAN. The DBMS offered PRO*FILE the ability to change requirements quickly, the use of an ad-hoc retrieval and update language, and nonflat file data structures. Secondly, it was recommended that this author be assigned to the ongoing PRO*FILE research effort in support of the computer-based research. The use of SPIRES to implement portions of the PRO*FILE System was fully concurred in a meeting with the Director of RISE, the Director of the ISU Computation Center and the Associate Director of User Services of the ISU Computation Center. (C. G. Maple & R. Lanbert, ISU Computation Center, ISU, October 14, 1982).

The Director of RISE and this author proposed to the Dean of the College of Education that portions of the PRO*FILE System be automated using the University computer. It was proposed that the Educational R&D cycle be used as the process to computerize portions of the PRO*FILE System. Educational R&D is a procedure used to develop and validate educational products. The PRO*FILE Task Force agreed that this was a valid approach, in principle, for the ongoing PRO*FILE research project. The Dean of the College of Education approved the assignment and approach based upon the Director of RISE recommendation. Additionally, the Dean desired a systems' demonstration within one month to verify the approach. The systems analysis effort formally began on October 14, 1982.

Data Base Management Systems' Objectives

Institutions of higher education and the business industry, in the past, created information systems along organizational lines. For example, grades were maintained by the Office of the Registrar, counseling records were maintained by the department or professor, and course objectives by the individual instructor. Some of the aforementioned information systems were automated using a computer while others remained manual. Of benefit would be a computer information system that would allow data to be stored in a central location from the various organizational units (an integrated data base) and would allow access to any portion of data base. A data base management system, in its most general form, is a software system capable of supporting and managing an integrated data base. The PRO*FILE System with its data from distinct organizational components represents an integrated data base by definition.

The main objectives of data base management systems' technology were defined by Cardenas in 1980. His comprehensive list of objectives were collaborated within varying degrees by the works of Martin (1976) and Date (1977). Cardenas contended that achieving the objectives, presented in Table 1, page 21, were an invaluable and essential asset toward developing and supporting modern integrated information systems.

The educational R&D cycle allowed for evaluation at various steps and revision of the prototype. The R&D cycle supported the goal of achieving and maximizing the DBMS objectives.

Selection of the Data Base Management System

A review of Iowa State University's computer facilities and software resources identified that a generalized data base management systems (DBMS), referred to as SPIRES, was the only comprehensive DBMS resident on the university academic computer system. Although there was only one DBMS on the university computer system, the selection of SPIRES was based upon the criteria in the Federal Information Processing Standards Publication (FIPS PUB 77), Guideline For Planning and Management of Database Applications, 1980. The criteria used in the selection of SPIRES are presented in Table 8.

SPIRES' documentation indicated that the DBMS could be used in two modes of operation. For large, time-consuming applications, the batch mode was available. The batch mode is the most economical. For example, a common use of the batch mode would be the processing of large numbers of updates. Some users have the requirement for immediate response to an ad-hoc query or report request. This type of user could use the second mode of operation on-line.

TABLE 8. Criteria Used in the Selection of a Data Base Manage System

---

1. Database Definition. Includes requirements on data element names and characteristics: data structures and relationships; Data Definition Language; types of data; operational aids, as for editing and searching the schema.

2. Data Manipulation. Includes query languages; report formatting statements; common programming language interfaces for data access and processing; subschema database description language, if any.

3. System and Integrity Control. Includes storage allocation and management; access control; transaction logging; backup and recovery; data validation; file dumping and data conversion; performance monitoring; usage monitoring and accounting.

4. Performance, Quality, and Other Requirements. Includes quantitative performance targets and benchmark testing; applicable standards; pertinent hardware constraints, such as available memory and multiple system compatibility.

5. Support. Includes installation; user training; documentation; continued technical assistance in database design and system tuning; design work for enhancements and future maintenance.

The on-line mode allows the user to converse, through the use of a computer terminal, directly with the computer. The batch and on-line mode of operation with the computer's components to solve information needs is depicted in Figure 2. ORVYL is a timesharing monitor that provides the capability of executing user programs in an interactive environment. WYLBUR is a timesharing system for manipulating various kinds of text by providing on-line interactive text editing capabilities. Job Control Language (JCL) is a language that serves as the communication link between the programmer and the operating system. High-level languages (HLL) are programming languages, such as COBOL, FORTRAN or BASIC, that use English-like symbols to stand for computer operations and memory addresses and in which a single statement instruction stands for multiple machine instructions. The combination of computer resources and facilities needed to process the PRO*FILE information requirements is referred to as the PRO*FILE Systems' Architecture.

```
                 AS/6 Computer System
                ---------------------------
                !   DBMS          ! W ! O !
-----------     ! --------        ! Y ! R !   ----------
!---------!     ! !S     !--------! L ! V ! --! ON-LINE!
-----------     ! ! P    !        ! B ! Y !   !TERMINAL!
!         !     ! !  I   !  ---   ! U ! L !   ----------
!  DATA   !-----!-!   R  !  !H!  ! R !   !
! STORAGE !     ! !    E !  !L!  --------!
!         !     ! !     S!--!L!---!  JCL  !-
!  DISK   !     ! !------!  ---   --------! -
!   AND   !     !------------!-------------  -
!  TAPE   !                  !                -
-----------                  !                 -
                             !                 ---------
                      -------------            ! BATCH !
                      !  BATCH    !            ! INPUT !
                      !  OUTPUT   !            ---------
                      !  REPORT   !
                      -------------
```

FIGURE 2. PRO*FILE Sytems Architecture

Data_Base_Management_Systems_Design_Methodology

A review of selected literature was undertaken to identify design procedures in developing applications using a Data Base Management System. There was no standard terminology thoughout the literature. In this review, therefore, terms were cross-referenced to increase the clarity of the discussion. Two books that offered a industry-wide terminology base for future reference were by James Martin, 1977, Computer_Data-Base_Organization,_2nd Edition, and T. William Olle, 1978, The_CODASYL_Approach_to Data_Base_Management.

There was a consensus among Martin (1977), Atre (1980) and Herrick (1983) that the data model is the underlying structure for a DBMS design. Martin's (1977) definition stated that "a data model represents the inherent structure of that data and hence is independent of individual applications of the data and also of the software or hardware mechanisms which are employed in representing and using the data". The process used to develop the data model started with the creation of the conceptual (or business) model of the enterprise. The conceptual model as defined by Atre (1980) "represents the entities of the enterprise and the relationships between them". The first step in creating the conceptual model was data analysis, which provides information about the data elements and the relationships between them. Natural groupings of data that belong together form entities. An example of an entity would be data about a course. The entity's data elements might be comprised of course number, course description and credit. Another entity might be professor with its data elements representing facts about a particular faculty member. There is a relationship between the two entities' course number and professor. The relationship is that a professor could teach none, one or many courses. One of the tasks of the DBMS designer is to analyze the information requirements of the organization (enterprise) and create a conceptual model that embodies all the information needs (entities and their relationships).

Each Data Base Management System allows users to define entities and relationships in various representations. These representations were defined by Date (1977) as Flat File, Hierarchical, Network, and Relational. The terms were discussed in Chapter 2, page 23. The four representations were verified by works published by Martin (1977) and Olle (1978) and are referred to as the logical_models in DBMS terms.

The developers of Data Base Management Systems uniquely program their systems to physically represent a flat file, a Hierarchical, a Network, or a Relational variation on a storage device, such as magnetic disk. The

DBMS term that defines how data is physically represented on a particular Data Base Management Systems is the physical_model.

Therefore, the review of selected literature to identify design procedures deduced the following sequence: A conceptual_model is created that will satisify all information needs of the organization; this model contains entities and corresponding relationships; the conceptual model would be transposed into a logical_model that a particular DBMS supports; the logical model would then be mapped to the actual storage device through the physical model. The physical model is then the underlying data model as defined previously by Martin (1977). It was apparent that a rigorous design procedure must be followed throughout the design process.

One of the design procedures used by the business industry and over 60 universities was developed by Atre (1980). Cardenas (1979) and Herrick (1983) developed similar procedures but Atre's provided the most comprehensive and rigorous method. Table 9 outlines Atre's procedure for DBMS design. In 2.1 of the Table, the three models, relational, hierarchical and network were defined in Chapter 2, page 23. In addition, 3.1 refers to an internal model. Internal_model cross-references with physical model. Finally, 2.2 and 4.2 refer to an external model. _External model_ refers to a representation, or portion thereof, of the internal model needed to generate a physical report that can be read. An example of this is found in Chapter 5, page 129.

Planning

PRO*FILE's Functional Specifications

Table 10 presents the initial set of objectives and contents, referred to as functional specifications, desired for the computerized portion of the PRO*FILE System by the PRO*FILE Task Force. Upon formal review of the specifications by this author, it was recommended that only a portion be used to develop an abbreviated prototype of the PRO*FILE System to demonstrate the capabilities of a DBMS. The Director of RISE recommended that five hypothetical students be entered for the demonstration. The rationale of this approach would shorten the systems development effort since any use of human data at Iowa State University was governed by the Human Subjects Review Board and was subject to review and approval before actual use.

TABLE 9. Procedure for Data Base Design

---

1. Design a conceptual model of a data base.

    1.1 Study the environment, and document assumptions for it.
    1.2 Determine the data elements referenced in every report individually.
    1.3 Determine the relationships between the data elements, such as identifying the primary key data elements and the nonkey data elements.
    1.4 Develop third normal form relations for each set of data elements. Where this is not possible for individual reports, merge data from reports to establish third normal form relations.
    1.5 Draw a conceptual model on the basis of the third normal form relations.

2. Design a logical model of a data base.

    2.1 Draw a logical model based on the conceptual model for a data base management system using:
    a. A relational data model.
    b. A hierarchical data model.
    c. A network data model.
    2.2 Draw external model for the reports. Transaction on the basis of the logical model above.

3. Design a physical model of a data base.

    3.1 Draw an internal model (also called a physical model) on the basis of the logical model from step 2.1.

4. Evaluate the physical model of a data base.

    4.1 Develop space estimates for the internal model above (as in step 3.1). Develop input/output probabilities for the internal model above (as in step 3.1).
    4.2 Draw external models for the reports. Transaction on the basis of the internal model above.

TABLE 10. PRO*FILE Capabilities and Functional Specifications

---

1. Design a program which permits administrative staff, advisers, and teacher education faculty to view, or add to, the following items in each student's "folder":
    a. Name
    b. Student ID
    c. High school rank
    d. GPA (college)
    e. Background information (birthdate, hometown)
    f. ACT or SAT scores (composit and subscale)
    g. Record of performance in a teaching/learning experience
    h. Record of contacts with adviser
    i. Requests by adviser to see student
    j. Curriculum sheet
    k. Adds, drops, transfer, incompletes
    l. other letters (of recommendation)
    m. 204--Report of Writing sample
    n. Admission form (to enter ISU)
    o. Admission to T.E. Report
    p. Record of Interim Interview
    q. Exit Interview Report
    r. Initial Assessment Battery Report
    s. Final Assessment Battery Report
    t. Progress report on Performance Elements
    u. Transfer credit and evaluation
    v. English writing sample
    w. Math placement test scores
    x. Copy of degree program

2. Design a program, with appropriate security checks, which permits students access to the above information.

3. Design a program which permits students to try out elements of the Initial Assessment Battery.

    a. Personal Profile
    b. Professional Strengths and Needs
    c. List of Teaching/Learning Experiences
    d. Knowledge of Education (Test-Sample Items)
    e. Reading List
    f. Philosophy of Education
    g. Basic Skills

Design of the Demonstration Prototype

A search for organizational units using SPIRES at Iowa State University revealed that the Office of Institutional Research was using SPIRES for daily processing. An interview with the Assistant Director indicated that the response time for on-line queries was averaging 3 to 5 seconds. Additionally, he stated that office personnel were discouraged with the development time and effort required to implement a small system. Finally, he thought that computer charges seemed excessive considering the amount of storage.

Based on the interview and actual demonstration of a SPIRES data base application at the Office of Institutional Research, three conclusions were evident by the interviewer. The 3 to 5 second response time was adequate based on the guidelines in FIPS PUB 77 (1980). An experienced data base analyst should design the PRO*FILE System in order to shorten the development time and lessen the level of effort required by functional personnel such as administrators, faculty and students. Finally, separate computer accounts should be established to divide the development charges from the data entry effort and data storage.

An abbreviated PRO*FILE Systems' prototype was designed for SPIRES using the hierarchical logical model presented in Figure 3. The initial design was reviewed by Mr. Joe Struss, a staff member of the Iowa State University Computation Center. Mr. Struss had previously assisted in the design of a SPIRES application for the Iowa State University's Office of Institutional Research. It was thought that a review of the PRO*FILE design by Mr. Struss would help identify any design flaws and/or areas of omission. Mr. Struss regarded the initial design adequate based on criteria from SPIRES documentation and FIPS PUB 99, but warned about ambitious research projects using SPIRES due to the limited knowledge of the software product on campus and the problems experienced in the Office of Institutional Research.

Demonstration of the Abbreviated Prototype

Within a two week period, the abbreviated prototype data base was created and loaded with data on the five hypothetical students. The demonstration to the Dean of the College of Education and PRO*FILE Task Force concentrated upon the basic PRO*FILE Systems' capabilites of:

1. What is available in a student file;
2. How many student records are in the file;
3. How is a student's record retrieved?
4. Examining or browsing information available on a student's record; and

5. Preparing a simple report and/or obtaining summary statistical information.

The demonstration of the prototype displayed:

1. Retrieval of records;
2. Sorting of records by Major and Name;
3. Information on High School Rank, SAT Verbal, Average of SATV, and Standard Deviation for SATV.

The demonstration verified that SPIRES's capabilites could support the ongoing PRO*FILE research effort. The Dean and PRO*FILE Task Force agreed that during the systems development effort, the Director of RISE should continue researching DBMS capabilities. In November, 1982 the Director of RISE and this author presented an updated proposal to computerize the PRO*FILE System to the Dean of the College of Education and the PRO*FILE Task Force. The proposal cited that the Educational R&D Cycle remain as the process to computerize portions of the PRO*FILE System. The Dean and Task Force agreed that this was a valid approach in principle for the ongoing PRO*FILE research project. Being the only comprehensive DBMS on the Iowa State University academic computer system, SPIRES was recommended as the software product that would be used to implement the prototype.

```
                              STUDENT
                           --------------
                           ! NAME         !
                           ! SOC-SEC-NO   !
                           ! SEX          !
                           ! CURRICULUM   !
                           ! COLLEGE      !
                           ! MAJOR        !
                           ! SAT SCORES   !
                           ! ACT SCORES   !
                           !    . . .     !
                           --------------
                                  !
                                  !
          ---------------------------------------------
          !                !               !           !
          !                !               !           !
       IN ! ETRY     EXPER ! IENCE     SEM ! YR    OUT ! ETRY
      ----------     ----------      ----------   -----------
      ! INITIAL!     ! STUDENT-!     !  PROG  !   !  FINAL   !
      ! ASSESS-!     ! TEACHING!     !   OF   !   ! ASSESS-  !
      !  MENT  !     ! PHIL ED !     ! STUDY  !   !   MENT   !
      !  ...   !     !   ...   !     !  ...   !   !   ...    !
      ----------     ----------      ----------   -----------
                                           !
                                           !
                                           !
                                       CRS !
                                     ----------
                                     ! DEPT    !
                                     ! CRS NUM!
                                     ! GRADE   !
                                     !  ...    !
                                     ----------
```

FIGURE 3. PRO*FILE Hierarchical Model

Initial Planning

The initial planning consisted of three efforts. The first was a detailed review of SPIRES capabilities in support of the PRO*FILE functional requirements. This was followed by establishing computer accounts for the development effort. Finally, student data were developed to test the prototype.

A comprehensive review of the SPIRES documentation was initiated to identify SPIRES systems' capabilities that would support the PRO*FILE Systems implementation. Discussions with staff members of the Computation Center (J. P. Hauck, Computation Center, ISU, October 18, 1982) verified that users could use SPIRES throughout the campus. This included not only the computation center and College of Education's Computer Laboratory, but also the other colleges and dormitories that had on-line computer facilities. Discussions with the Computer Services Staff also verified that a dial-up capability for off campus processsing through the use of a modem was available.

Separate computer accounts were established for systems design, data entry, data storage, and faculty/student use in order to keep track of the computer resources and funds being expended.

In reviewing Iowa State University's regulations for using human subjects, the Director of RISE recommended that

12 hypothetical students be created for the Preliminary Form of the Product. This data represented actual scenarios of education students. The 12 students represented seven departments and three year levels (sophomore through seniors). This method of presentation followed the corresponding categories of students pursuing degrees in the College of Education proportionally across year levels and disciplines.

Summary

The description of the prototype design was presented in this chapter. An abbreviated prototype was implememted using a data base management system within the educational R&D cycle to validate the approach. The data base management system design methodology that will transform PRO*FILE's information needs into a physical data base was selected. The next chapter used the design methodology to implement the PRO*FILE System prototype.

DESCRIPTION OF THE PROTOTYPE

Introduction

This chapter presents the process used to develop the computerized PRO*FILE Systems' prototype. The prototype development was guided by the first five steps of the Educational Research and Development (R&D) Cycle. These steps include:

1. Research and Information Collecting;
2. Planning;
3. Develop Preliminary Form of Product;
4. Preliminary Field Testing; and
5. Main Product Revision.

The previous chapter discussed Steps 1 and 2. This chapter continues the description following the educational R&D cycle, presenting Steps 3, 4 and 5.

Development of a Preliminary Form of the Product

The underlying data base design procedure for this step of the educational R&D cycle was adopted from Atre's book _Data_Base_Structured_Techniques_for_Design,_Performance,_and_Management_, 1981. The rationale of this procedure for data base design was presented in the previous chapter. Atre's procedure provided a rigorous methodology and was followed in detail.

Figure 4 lists the types of reports needed from the information in the PRO*FILE System. The reports themselves are shown in Figures 5 to 12.

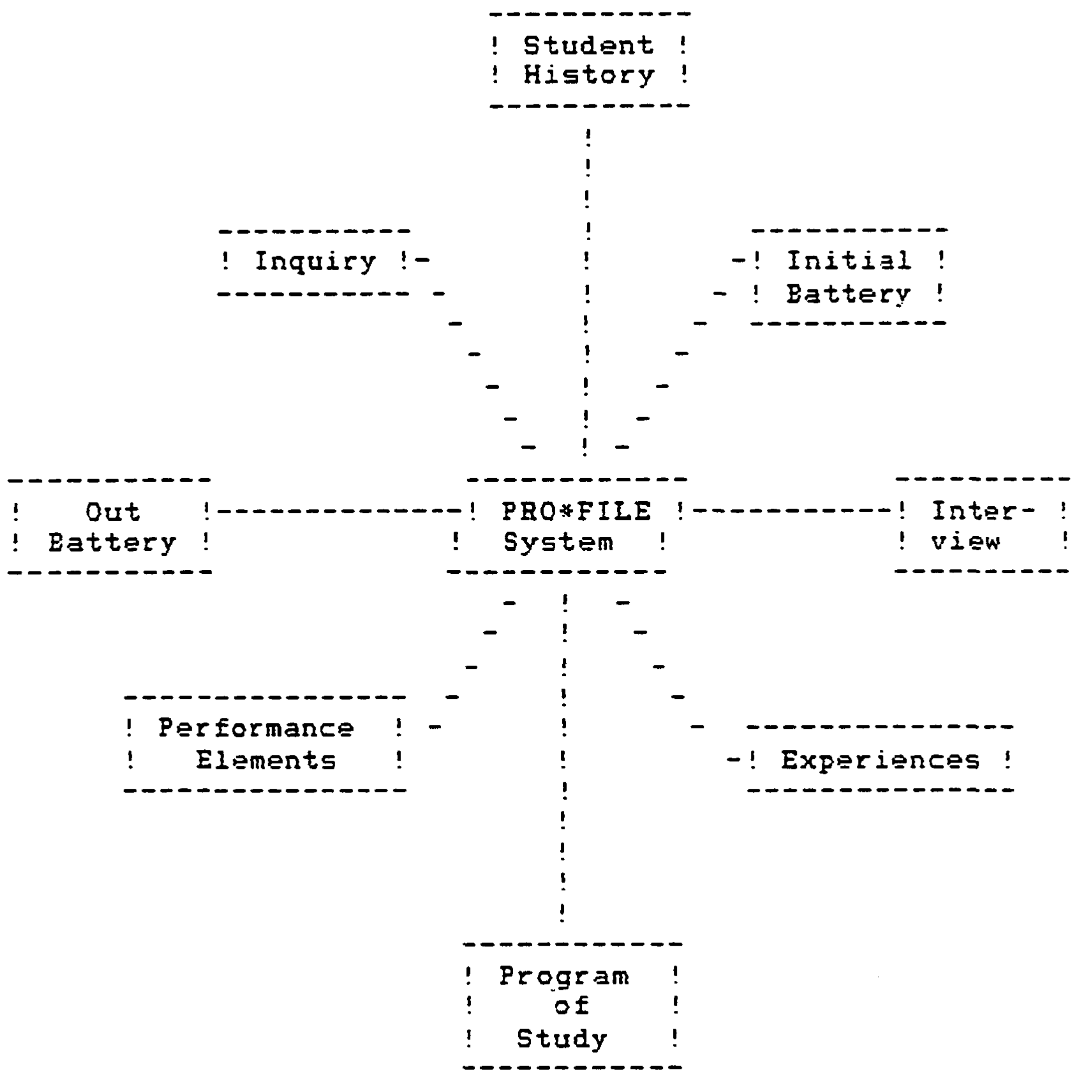


FIGURE 4. Information Needs for the PRO*FILE System

Step I Design of a Conceptual Model for a Data Base

Step I.1 Study The Environment and Document Assumptions For It   The PRO*FILE Systems' architecture is provided in pictorial overview in Figure 2; refer to page 69. The PRO*FILE System used Iowa State University's AS/6 and the timesharing system (MILTEN/WLYBUR/ORVYL), as previously discussed on page 68. The data base management system used for the prototype was SPIRES. SPIRES was used only in the on-line mode. Data were stored on Direct Access Storage Devices (DASD), commonly referred to as disk, with magnetic tape utilized for backup and recovery. Finally, the system supported a mainframe to micro link for downloading files.

Student History (Figure 5)   Certain data about a student was static in nature. However, such data provided advisers with the basic information about the aspirations of a student and his/her foundations in educational subjects. This foundation included such data elements as gender (sex), major, ACT, SAT scores from high school preparation, to the GPA and number of transfer credits from other institutions.

Student History

Date: November 28, 1982

Element Name	Element Value
NAME:	NELSON, NANCY SUE
SOCIAL SECURITY NUMBER:	971574018
SEX:	F
MAJOR:	EL ED
MINOR:	SP
CURRICULUM:	EL ED
COLLEGE:	D
TYPE:	F
CURRENT GPA:	2.96
TEACHING LEVEL:	K-6
SEMESTER STUDENT TAUGHT:	S 82
ENTRANCE GPA:	3.29
SEMESTER ADMITTED:	S
YEAR ADMITTED:	82
TRANSFER CREDIT:	0
HIGH SCHOOL RANK:	8
ACT:	23
ACT - ENGLISH:	22
ACT - MATHEMATICS:	22
ACT - SOCIAL STUDIES:	23
ACT - NATIONAL SCIENCE:	24
SAT - VERBAL:	0
SAT - MATHEMATICS:	0

FIGURE 5. Student History

Initial Battery (Figure 6) Each student entering a discipline for teacher education was required to take an initial battery examination. This examination was written by the respective departments' faculty to assess the students preparation in the subject area. There were more than one initial battery depending on the areas of specialization and information requirements of the department. The format of the report allowed faculty to address specific questions and needs of the student.

Initial Battery

Date: November 28, 1982

Name:	NELSON, NANCY SUE
Social Security Number: . . .	971574018
Test Identification Number: . .	3456
Date of Test:	10/22/1982
Number of Correct Answers: . .	28

Question Number	Actual Answer	Student's Response	Marked Wrong
1	T	F	X
2	F	F	
3	D	D	
4	A	B	X
.	.	.	
.	.	.	
.	.	.	
40	E	E	

FIGURE 6. Initial Battery

Interview (Figure 7) Various interviews took place throughout the educational preparation of a student. These included interviews with advisers, faculty, potential employers, administrators, and other students. This report provided students the abiltity to store the dialogue of such interviews for future reference.

```
                         Interview

                                        Date: November 28,
1982

Name:  . . . . . . . . .   NELSON, NANCY SUE

Interview Type: . . . . .  INITIAL
Interview Date: . . . . .  4/15/1982
Interviewer: . . . . . .   DR. R.J. SMITH

Dialogue:

     Nancy has a strong desire to become an elementary
school teacher.  Her progress at the university indicates
that she has good study habits and is a performer in the
classroom.  Nancy would like to teach at the first or
second year level. This interviewer has encouraged her to
declare the major and apply for an advisor.
```

FIGURE 7. Interview

<u>Experiences (Figure 8)</u> Educational experiences provided potential teachers with information that could be used throughout their tenure in the teaching profession. Student teaching was an example of an educational experience that a student would desire to document for future reference. This report provided students with the capability to recall educational experiences throughout their educational preparation and beyond.

Experiences

Date: November 28, 1982

Name:	NELSON, NANCY SUE
Experience Type:	FIELD TRIP
Experience Date:	5/12/81

Composition:

During a recent track meet at Clinton, Iowa I was informed that the local library had a few books on the 1933 Olympic Games held in Chicago. Being a history buff about the Olympics, I went down to the library when the track meet was over. The library was closed for the remainder of the day. I must return to the Clinton library and browse the books.

FIGURE 8. Experiences

Program_of_Study_(Figure_9) Students completed courses every semester. This report provided a listing of all course work taken to include grades and quality points. In addition, it provided the student an area to plan the remainder of course work by semester until he/she graduated.

Program of Study

Date: November 28, 1982

Name:	Nelson, Nancy Sue
Curriculum:	EL ED
College:	D
Major:	EL ED
Minor:	SP
Teaching Level:	K-6
Year Admitted to Teacher Ed: .	82

Sem Year	Depart	Crs Num	Credit	Course Name	Grade	Quality Points
SP 82	EL ED	345	3	STRATEGIES IN TCHG	B	9.00
	EL ED	375	4	TCHG OF READING	B	12.00
	EL ED	468	2	PRACTICUM IN TCHG	B	6.00
	EL ED	301	1	INSTRUCTIONAL MEDIA	B	3.00
	H S	105	2	EMERG HEALTH CARE	C	6.00
F 82	EL ED	445	4	TCHG LANG-SOC STDS	B	12.00
	EL ED	446	4	TCHG MATH SCIENCE	C	8.00
	EL ED	447	3	TCHG IN KINDERGRTN	B	9.00
	HIST	370	3	HISTORY OF IOWA	A-	11.21

FIGURE 9. Program of Study

Performance (Figure 10) As students completed required course work, they were able to review what skills they should have mastered. At their own convenience, students took a performance test on a specific performance element. By reviewing the results of the examination, they located areas of weakness that needs additional study and/or formal course work.

Performance

Date: November 28, 1982

Name:	NELSON, NANCY SUE
Social Security Number:	971574018
Test Identification Number:	8145
Date of Test:	6/23/1982
Number of Correct Answers:	33

Question Number	Actual Answer	Student's Response	Marked Wrong
1	T	F	X
2	F	F	
3	D	D	
4	A	B	X
.	.	.	
.	.	.	
.	.	.	
40	E	E	

FIGURE 10. Performance

Out_Battery_(Figure_11) This report assessed the cumulative knowledge of a student for a particular discipline. This report showed a potential teacher the status of his/her mastered knowledge required for teaching. Additionally, it identified any weak areas so that students could continue to take formal or informal study in those areas.

Out Battery

Date: November 28, 1982

Name:	NELSON, NANCY SUE
Curriculum:	EL ED
College:	D
Major:	EL ED
Minor:	SP
Current GPA:	2.96
Semester Student Taught:	S 82
Teaching Level:	K-6
Test Identification Number:	9768
Date of Test:	7/12/1982
Number of Correct Answers:	38

Question Number	Actual Answer	Student's Response	Marked Wrong
1	T	F	X
2	F	F	
3	D	D	
4	A	B	X
.	.	.	
.	.	.	
.	.	.	
40	E	E	

FIGURE 11. Out Battery

Inquiry Transaction (Figure 12) A student. faculty member or administrator, from time to time, may desire to examine, extract, or modify the contents of a specific student's record or groups of records. The social security number may be used to acquire specific information on a particular student. Other information may be gathered by major, minor, sex or a host of other criteria. The versatility of the on-line query language allowed a user to view data in an ad-hoc mode. A student may desire to view department-wide data such as grade point average, names, social security numbers and other data that might compromise a student's identification. Such data would not be available and furthermore is locked out by security measures.

Assumptions about the Environment of the PRO*FILE System

The PRO*FILE Task Force and this author held a meeting to establish the assumptions under which the prototype will work (C. R. Kniker, Secondary Education, ISU, October 19, 1982). The assumptions of the conceptual model were:

1. The social security number uniquely would identify the student, that is, the name of the student, the major discipline, the minor discipline, and so on.

Inquiry Transaction

Input consists of:

(Social Security Number) Department or Name

Transaction type:

Inquiry

Output consists of:

Per ad-hoc request -

```
i.e., NAME, SOCIAL SECURITY NUMBER, MAJOR, GPA, SATV
       !     !                      ----------------
       !     !                      !
       !     !                      !---By individual
       !     !                          value
       !     !
       !     !---Not provided per security restrictions
       !     !
       !--------All last names will be provided
```

```
Sample:  Simple Inquiry  -  FIND SOC-SEC-NUM = 971574018
                            TYPE MAJOR, CURRENT-GPA

         Complex Inquiry -  FIND DEPARTMENT = EL ED AND
                                 NAME GT NELSON
                            TYPE NAME, ACT, MINOR
```

FIGURE 12. Inquiry Transaction

2. Certain data elements must be password protected to secure the identity of a student. These would include name, social security number, curriculum, and semester the student taught.

3. Blocks of data elements must be password protected to limit the ad-hoc browsing of sensitive or confidential data.

4. The administrative staff and faculty would enter the static password protected data elements.

5. Students would update and maintain the data elements as their educational experience grows.

6. Students would access the PRO*FILE System from any terminal on campus by either direct line with the computer system or via the dial-up capability.

7. Students may access all data elements on information that is stored about them. Any password protected data element may be changed by an administrative staff representative upon verification of the data and appropriate authorization.

8. Students periodically would retrieve copies of all reports, to include in their personal PRO*FILE folder.

The PRO*FILE Systems' design was in compliance with federal privacy regulations. The Privacy Act of 1974 was designed to protect the privacy of individuals who have information about themselves maintained by the federal government. An extension of this Act was the Educational Privacy Act that protected individuals' privacy by regulating access to private and public school's computer-stored records of grades and evaluations of behavior. Many state laws that regulate government record-keeping practices are patterned after the Privacy Act of 1974. Many state laws contain the provision that require publication of notices describing the records that each government agency maintains; provide for the collection and storage of only data that is relevant, timely, and accurate; and prohibit unauthorized disclosure of data relating to individuals. Although the issue of privacy and confidentially were not formally resolved, the PRO*FILE System assumed the restrictions of the Privacy Act of 1974.

Step_I.2__Determine_the_Data_Elements_Referenced_in Every_Report_Individually A list of all data elements referenced in the reports from Figure 4, page 84, are in alphabetical order and found in Table 11. Table 12 presents a cross-reference table; it shows the data elements and the reports in which the data elements are used.

TABLE 11. Data Elements In Alphabetical Order

ACT	Achievement Test - Composit.
ACT-ENGLISH	Achievement Test - English.
ACT-MATH	Achievement Test - Mathematics.
ACT-NATIONAL-SCI	Achievement Test - National Science
ACT-SOC-STUDIES	Achievement Test - Social Studies.
COLLEGE	College of the university.
COMPOSITION	Essay on educational experiences.
COURSE-NAME	Course name.
COURSE-NUMBER	Course number.
CREDIT	Credit hours per course.
CURRENT-GPA	Cumulative grade point average.
CURRICULUM	Teaching area of specialization.
DEPARTMENT	Department of the college.
DIALOGUE	Written discussion of interview.
ENTRANCE-GPA	GPA when admitted to Teacher Education.
EXPERIENCE-DATE	Date of educational experience.
EXPERIENCE-TYPE	Type of educational experience.
GRADE	Grade received by a student for a course.
HIGH-SCHOL-RANK	High school rank.
INTERVIEWER	Person interviewing student.
INTERVIEW-DATE	Date of interview.
INTERVIEW-TYPE	Type of interview.

TABLE 11 (Continued)

IN-IND-RESPES	Initial battery individual responses.
IN-NUM-CORRECT	Number of correct answers on the Initial Battery.
IN-TEST-QUESTS	Answers to questions on the Initial Battery.
IN-TEST-ID	Identification number of Initial Battery.
IN-TEST-TIME	Date of Initial Battery examination.
MAJOR	Major discipline.
MINOR	Minor discipline.
NAME	Name of student.
OUT-DATE-TIME	Date and time of Out Battery exam.
OUT-IND-RESPES	Out Battery individual responses.
OUT-NUM-CORRECT	Number of correct answers on the Out Battery.
OUT-NUM-QUESTS	Answers to questions on the Out Battery.
OUT-TEST-ID	Identification number of Out Battery.
PERF-DATE-TIME	Date and time of Performance exam.
PERF-IND-RESPES	Performance exam individual responses.
PERF-NUM-CORRECT	Number of correct answers on the Performance exam.
PERF-NUM-QUESTS	Number of questions on the Performance exam.

TABLE 11 (Continued)

PERF-TEST-ID	Identification number of Performance exam.
QUALITY-POINTS	Cumulative quality points.
SAT-MATH	Standard Achievement Test, Mathematics.
SAT-VERBAL	Standard Achievement Test, Verbal.
SEMESTER-ADMITTED	Semester admitted to Teacher Education.
SEMESTER-YEAR	Year admitted to Teacher Education.
SEM-STD-TAUGHT	The semester of student teaching.
SEX	Sex of student.
SOC-SEC-NUM	Social Security Number of student.
TEACHING-LEVEL	Teaching level(s) of student.
TRANSFER-CREDIT	Credits transferred from another college.
TYPE	Teacher Education group identification.
YEAR	Year admitted to Teacher Education.

TABLE 12. Cross-Reference Table Between Data Elements and Reports

	Reports							
Data Elements	Student History	Int Btry	Inter view	Exper ience	Prog of Study	Perf	Out Btry	I n q
NAME	X	X	X	X	X	X	X	X
SOC-SEC-NUM	X	X						
SEX	X							X
CURRICULUM	X				X			X
COLLEGE	X				X		X	X
YEAR	X				X			X
CURRENT-GPA	X						X	X
SEMESTER-ADMITTED	X							X
TYPE	X							X
ENTRANCE-GPA	X							X
TRANSFER-CREDIT	X							X
SEM-STD-TAUGHT	X						X	X
TEACHING-LEVEL	X				X		X	X
MAJOR	X				X		X	X
MINOR	X				X		X	X
HIGH-SCHOOL-RANK	X							X
ACT	X							X
ACT-ENGLISH	X							X
ACT-MATH	X							X
ACT-SOC-STUDIES	X							X
ACT-NATIONAL-SCI	X							X
SAT-VERBAL	X							X
SAT-MATH	X							X
IN-TEST-ID		X						X
IN-TEST-TIME		X						X
IN-TEST-QUESTS		X						X
IN-NUM-CORRECT		X						X
IN-IND-RESPES		X						X
INTERVIEW-TYPE			X					X
INTERVIEW-DATE			X					X
INTERVIEWER			X					X
DIALOGUE			X					X
EXPERIENCE-TYPE				X				X
EXPERIENCE-DATE				X				X
COMPOSITION				X				X
SEMESTER-YEAR					X			X
DEPARTMENT					X			X
COURSE-NUMBER					X			X

TABLE 12 (Continued)

	Reports							
Data Elements	Student History	Int Btry	Inter view	Exper ience	Prog of Study	Perf	Out Btry	I n q
CREDIT					X			X
COURSE-NAME					X			X
GRADE					X			X
QUALITY-POINTS					X			X
PERF-TEST-ID						X		X
PERF-DATE-TIME						X		X
PERF-NUM-QUESTS						X		X
PERF-NUM-CORRECT						X		X
PERF-IND-RESPES						X		X
OUT-TEST-ID							X	X
OUT-DATE-TIME							X	X
OUT-NUM-QUESTS							X	X
OUT-NUM-CORRECT							X	X
OUT-IND-RESPES							X	X

<u>Step I.3 Determine the relationships between the data elements</u> In this step, the primary key, data elements and the non-key data elements were identified.

<u>Step I.4 Develop third normal form relations for each set of data elements</u> Where this was not possible for individual reports, data was merged from reports to establish third normal form relations.

<u>Review of Terminology and Concepts</u> In Chapter 2, Review of Literature, Data Base Management Systems' terms and concepts were discussed. A brief review of those definitions and ideas, in lay terms, follows in order to clarify the data base design procedure.

A fact about something is referred to as data. For example, a person's first name is data and is contained in a data element. A group of data elements makes up a tuple (record) and a group of tuples define a relation (file).

Data elements can be defined as key or non-key. Key simply means that if one knows the contents of the key data element, the other data elements in the relation can be accessed. For example, social security number uniquely identifies a person and could be defined as a key. Knowing the person's social security number, one could access other data elements about the person such as name, age or sex. Knowing the contents of a non-key data element only means that one knows the fact but may not have access to the other data elements in the tuple. A key can be simple or compound.

A simple key has only one data element, whereas a compound key contains two or more data elements. An example of a compound key might be COURSE-NUMBER*SECTION. A primary key could be either simple or compound, but it represents the minimum number of data elements that uniquely identifies a corresponding group of data elements in a tuple.

Normalization is a three step process where a primary key uniquely identifies a tuple. Bradley (1983) identified five normal forms but stated that the fourth normal form was conceived to describe relations without binary join dependencies and without undesirable functional dependencies. Bradley further stated that the fifth normal form is so contrived and uncommon that it is unlikely ever to occur in a practical data base design situation. Therefore, this study will not consider the fourth and fifth normal forms. A relation is in the first normal form if the relation depends on the key to identify the tuple. The relation is in the second normal form if the relation depends on the whole key. That is, if the key is compound all parts of the key (the whole key) must be known to uniquely identify the tuple. Finally, the relation is in the third normal form if the tuple depends on nothing but the key. That is, knowing the contents of one data element does not mean you know the contents of another data element in the tuple. For example, consider two data elements: credit-hours and college-status. In a given tuple, by knowing the number of credit-hours earned one already know what the college-status of the student is. A student with 15 credit-hours is a freshman. This situation is referred to as transitive dependency. During the normalization process, if any of the above is not true, the relations are split into smaller units and the process recycles until a group of relations are defined, each containing tuples that are uniquely identified by the primary key. The normalization process was discussed in Chapter 2, page 26.

Student History The data elements representing the entities of this report are shown in Figure 13.

```
---------------------------------------------------
! NAME, SOC-SEC-NUM, SEX, CURRICULUM, COLLEGE,    !
! YEAR, CURRENT-GPA, SEMESTER-ADMITTED, TYPE,     !
! ENTRANCE-GPA, TRANSFER-CREDIT, SEM-STD-TAUGHT,  !
! TEACHING-LEVEL, MAJOR, MINOR, HIGH-SCHOOL-RANK, !
! ACT, ACT-ENGLISH, ACT-MATH, ACT-SOC-STUDIES,    !
! ACT-NATIONAL-SCI, SAT-VERBAL, SAT-MATH          !
---------------------------------------------------
```

FIGURE 13. Student History Data Elements

The relationships between the data elements from Figure 13 are:

SOC-SEC-NUM <<---> NAME, SEX, CURRICULUM, COLLEGE, YEAR, CURRENT-GPA, SEMESTER-ADMITTED, TYPE, ENTRANCE-GPA, TRANSFER-CREDIT, SEM-STD-TAUGHT, TEACHING-LEVEL, MAJOR, MINOR, HIGH-SCHOOL-RANK, ACT, ACT-ENGLISH, ACT-MATH, ACT-SOC-STUDIES, ACT-NATIONAL-ACI, SAT-VERBAL, SAT-MATH. For a given SOC-SEC-NUM, there is only one NAME (of the student) and one Major (major discipline of the student). This is true for all elements of the report. For a given CURRICULUM there can also be many students. There can also be many students with the same major and minor disciplines. These considerations among data elements represent a one to many relation (mapping), pictorially represented as <<---> or one ---> many. A detailed discussion of the use of the above relationships was presented in Chapter 2, page 22.

In Figure 14, the primary key is underlined (<u>SOC-SEC-NUM</u>). Relation 1 is in the third normal form, because the non-key data elements (MAJOR, SAT-MATH, etc.) from this relation require the full key for their identification. There is no transitive dependency between the non-key elements as well. In other words, there is no way to find the value of a non-key value by knowing the value of any other non-key value. The third normal form relation for the end user's view regarding the Student History is provided in Figure 14.

```
-------------------------------------------------------
! 1  SOC-SEC-NUM <<---> NAME, SEX, CURRICULUM.     !
!                       COLLEGE, YEAR, CURRENT-GPA!
!                       SEMESTER-ADMITTED, TYPE,   !
!                       ENTRANCE-GPA,              !
!                       TRANSFER-CREDIT,           !
!                       SEM-STD-TAUGHT,            !
!                       TEACHING-LEVEL, MAJOR,     !
!                       MINOR, HIGH-SCHOOL-RANK,   !
!                       ACT, ACT-ENGLISH, ACT-MATH!
!                       ACT-SOC-STUDIES,           !
!                       ACT-NATIONAL-SCI,          !
!                       SAT-VERBAL, SAT-MATH       !
-------------------------------------------------------
```

```
-------------------------------------------------------
! 971574018 <<---> NELSON, NANCY SUE,  F, EL ED,  !
!                  D, 3, 2.96, S, F, 3.29, 0, S 82!
!                  K-6, EL ED, SP, 8, 23, 22, 22, !
!                  23, 24, 0, 0                   !
-------------------------------------------------------
```

FIGURE 14. Third Normal Form Relation for the end user's view from Figure 13 and corresponding values

Initial_Battery The data elements representing the entities of this report are shown in Figure 15.

```
--------------------------------------------------------
!  NAME, SOC-SEC-NUM, IN-TEST-ID, IN-TEST-TIME,       !
!  IN-TEST-QUESTS, IN-NUM-CORRECT, IN-IND-RESPES      !
--------------------------------------------------------
```

FIGURE 15. Initial Battery Data Elements

Relations 2 and 3 are in the third normal form. Question numbers can be generated in the report program. Actual test answers can be acquired from a table identified by the Test Identification Number (IN-TEST-ID). Responses marked wrong on the report are generated in the computer program by a comparison of the table response and the student's response. The third normal form relations for the end user's view regarding the Initial Battery are provided in Figure 16.

```
--------------------------------------------------------
!  2  SOC-SEC-NUM  <<--->  NAME                        !
!                                                      !
!  3  SOC-SEC-NUM*IN-TEST-ID*IN-TEST-TIME              !
!     <<---> IN-NUM-CORRECT,  IN-IND-RESPES            !
--------------------------------------------------------

--------------------------------------------------------
! 971574018 <<---> NELSON, NANCY SUE                   !
!                                                      !
! 971574018*4356*10/22/1983 <<---> 28, F,F, ... E!
--------------------------------------------------------
```

FIGURE 16. Third Normal Form Relations for the end user's view from Figure 15 and corresponding values

__Interview__ The data elements representing the entities of this report are shown in Figure 17.

```
-------------------------------------------------
! NAME, INTERVIEW-TYPE, INTERVIEW-DATE,         !
! INTERVIEWER, DIALOGUE                         !
-------------------------------------------------
```

FIGURE 17. Interview Data Elements

The relationships between the data elements from Figure 17 are:

4. __SOC-SEC-NUM__ <<---> NAME

5. __SOC-SEC-NUM__ <<--->> DIALOGUE

Relation 4 is in the third normal form. In considering Relation 5, a student may have many interviews (dialogues) and interviews can vary in type. This is a many-to-many mapping, represented as <<--->> or --->> many <<---. But relation 5 is not even in the first normal form, because the mapping is many to many. Relation 5 can be transformed into a third normal form relation if the primary key is further qualified, that is, if the primary key is further compounded with INTERVIEW-TYPE, INTERVIEW-DATE and INTERVIEWER. Relation 5 is now in the third normal form. A given student (SOC-SEC-NUM), for a given interview type on a specific date and by a specific interviewer, will have a specific dialogue. The third normal form relations for the end user's view regarding the Interview are provided in Figure 18.

```
-------------------------------------------------------------
!  4  SOC-SEC-NUM <<---> NAME                               !
!                                                           !
!  5  SOC-SEC-NUM*INTERVIEW-TYPE*                           !
!     INTERVIEW-DATE*INTERVIEWER <<---> DIALOGUE            !
-------------------------------------------------------------
```

```
-------------------------------------------------------------
! 971574018 <<---> NELSON, NANCY SUE                        !
!                                                           !
! 971574018*INITIAL*4/15/1982*DR. R. J. SMITH               !
!           <<---> Nancy has a strong desire to             !
!                  become an elementary teacher.            !
!                  Her progress at the university           !
!                  indicates ... apply for an               !
!                  adviser.                                 !
-------------------------------------------------------------
```

FIGURE 18. Third Normal Form Relations for the end user's view from Figure 17 and corresponding values

__Experiences__ The data elements representing the entities of this report are shown in Figure 19.

```
-----------------------------------------------------
!  NAME, EXPERIENCE-TYPE, EXPERIENCE-DATE,          !
!  COMPOSITION                                      !
-----------------------------------------------------
```

FIGURE 19. Experiences Data Elements

The relationships between the data elements from Figure 19 are:

6. SOC-SEC-NUM <<---> NAME

7. SOC-SEC-NUM <<--->> COMPOSITION

Relation 6 is in the third normal form. However, Relation 7 requires a compounded key to be represented in the third normal form. In order to fully qualify a specific composition, the compounded key will require SOC-SEC-NUM, EXPERIENCE-TYPE and EXPERIENCE-DATE. The third normal form relations for the end user's view regarding the Experience are provided in Figure 20.

```
-----------------------------------------------------
!  6  SOC-SEC-NUM <<---> NAME                        !
!                                                    !
!  7  SOC-SEC-NUM*EXPERIENCE-TYPE*                   !
!     EXPERIENCE-DATE <<---> COMPOSITION             !
-----------------------------------------------------
```

```
-----------------------------------------------------
! 971574018 <<---> NELSON, NANCY SUE                 !
!                                                    !
! 971574018*FIELD_TRIP*5/12/81                       !
!           <<---> During a recent track meet at!
!                  Clinton, Iowa I was informed !
!                  that the local library had a !
!                  few books on the 1933 Olympic!
!                  games ... browse the books.  !
-----------------------------------------------------
```

FIGURE 20. Third Normal Form Relations for the end user's view from Figure 19 and corresponding values

Program of Study The data elements representing the entities of this program of study report are shown in Figure 21.

```
-----------------------------------------------------
!  NAME, CURRICULUM, COLLEGE, YEAR,                 !
!  TEACHING-LEVEL, MAJOR, MINOR, SEMESTER-YEAR,     !
!  DEPARTMENT, COURSE-NUMBER, CREDIT,               !
!  COURSE-NAME, GRADE, QUALITY-POINTS               !
-----------------------------------------------------
```

FIGURE 21. Program of Study Data Elements

The relationships between the data elements from Figure 21 are:

8. <u>SOC-SEC-NUM</u> <<---> NAME, CURRICULUM, COLLEGE, MAJOR,MINOR, TEACHING-LEVEL, YEAR

9. <u>SOC-SEC-NUM</u>*<u>SEMESTER-YEAR</u>*<u>COURSE-NUMBER</u>* <u>DEPARTMENT</u> <<---> GRADE, QUALITY-POINTS

10. <u>DEPARTMENT</u>*<u>COURSE-NUMBER</u> <<---> CREDIT, COURSE-NAME

The three relations 8,9 and 10 are in the third normal form with the data elements underlined as the primary key. The third normal form relations for the end user's view regarding the Program of Study are provided in Figure 22.

```
------------------------------------------------
! 8  SOC-SEC-NUM <<---> NAME, CURRICULUM,      !
!                          COLLEGE, MAJOR,     !
!                          MINOR, YEAR,        !
!                          TEACHING-LEVEL      !
!                                              !
! 9  SOC-SEC-NUM*SEMESTER-YEAR*                !
!    COURSE-NUMBER*DEPARTMENT <<---> GRADE     !
!                          QUALITY-POINTS      !
!                                              !
!10  DEPARTMENT*COURSE-NUMBER <<--->           !
!                          COURSE-NAME, CREDIT!
------------------------------------------------
```

```
------------------------------------------------
! 971574018 <<---> NELSON, NANCY SUE, EL ED, D!
!                  EL ED, SP, 3, K-6           !
!                                              !
! 971574018*SP_82*345*EL_ED <<---> B, 9.00     !
!                                              !
! EL_ED*345 <<---> STRATEGIES IN TCHG, 3       !
------------------------------------------------
```

FIGURE 22. Third Normal Form Relations for the user's view from Figure 21 and corresponding values

<u>Performance</u> The data elements representing the entities of this report are shown in Figure 23.

```
---------------------------------------------------
!  NAME, SOC-SEC-NUM, PERF-TEST-ID,               !
!  PERF-DATE-TIME, PERF-NUM-QUESTS,               !
!  PERF-NUM-CORRECT, PERF-IND-RESPES              !
---------------------------------------------------
```

FIGURE 23. Performance Data Elements

The relationships between the data elements from Figure 23 are:

11. <u>SOC-SEC-NUM</u> <<---> NAME

12. <u>SOC-SEC-NUM</u>*<u>PERF-TEST-ID</u>*<u>PERF-DATE-TIME</u>
 <<---> PERF-IND-RESPES

Relations 11 and 12 are in the third normal form. The test question numbers were generated by the report program. The correct test answers were acquired by a table identifed by the PERF-TEST-ID. Any answer marked wrong was generated by a comparison of the table entry and the corresponding student answer. The third normal form relations for the end user's view regarding Performance are provided in Figure 24.

```
-----------------------------------------------------------
!  11  SOC-SEC-NUM <<---> NAME                           !
!                                                        !
!  12  SOC-SEC-NUM*PERF-TEST-ID*                         !
!      PERF-DATE-TIME <<---> PERF-IND-RESPES             !
!                            PERF-NUM-CORRECT            !
-----------------------------------------------------------


-----------------------------------------------------------
! 971574018 <<---> NELSON, NANCY SUE                     !
!                                                        !
! 971574018*8145*6/23/1982 <<---> F, F, ... E,           !
!                                 33                     !
-----------------------------------------------------------
```

FIGURE 24. Third Normal Form Relation for the end user's view from Figure 23 and corresponding values

__Out Battery__ The data elements representing the entities of this report are shown in Figure 25.

```
-------------------------------------------------
!  NAME, CURRICULUM, COLLEGE, CURRENT-GPA,      !
!  SEM-STD-TAUGHT, TEACHING-LEVEL, MAJOR,       !
!  MINOR, OUT-TEST-ID, OUT-TEST-DATE,           !
!  OUT-NUM-QUESTS, OUT-NUM-CORRECT,             !
!  OUT-IND-RESPES                               !
-------------------------------------------------
```

FIGURE 25. Out Battery Data Elements

The relationships between the data elements from Figure 25 are:

```
13.  SOC-SEC-NUM <<---> NAME, CURRICULUM, COLLEGE,
                        CURRENT-GPA, SEM-STD-TAUGHT,
                        TEACHING-LEVEL, MAJOR, MINOR

14.  SOC-SEC-NUM*OUT-TEST-ID*OUT-DATE-TIME
                 <<---> OUT-NUM-QUESTS,
                        OUT-NUM-CORRECT,
                        OUT-IND-RESPES
```

Relations 13 and 14 are in the third normal form. Question numbers for the report will be generated by the report program. The correct answers to the test will be acquired for a table identified by the OUT-TEST-ID. Answers marked wrong will be accomplished by a comparision between the table answers and the student's corresponding answers. The third normal form relation for the end user's view regarding the Out Battery are provided in Figure 26.

```
--------------------------------------------------
!  13  SOC-SEC-NUM <<---> NAME, CURRICULUM,      !
!                         COLLEGE,               !
!                         CURRENT-GPA            !
!                         SEM-STD-TAUGHT,        !
!                         TEACHING-LEVEL,        !
!                         MAJOR, MINOR           !
!                                                !
!  14  SOC-SCE-NUM*OUT-TEST-ID*                  !
!      OUT-DATE-TIME <<--->                      !
!                         OUT-NUM-QUESTS,        !
!                         OUT-NUM-CORRECT,       !
!                         OUT-IND-RESPES         !
--------------------------------------------------
```

```
--------------------------------------------------
! 971574018 <<--> NELSON, NANCY SUE, EL ED, D,!
!                 2.96, S 82, K-6, EL ED, SP  !
!                                             !
! 971574018*9768*7/12/1982 <<---> 40, 38,     !
!                                 F, F, ... E !
--------------------------------------------------
```

FIGURE 26. Third Normal Form Relations for the end user's view from Figure 25 and corresponding values

The resulting third normal form relations are presented in Table 13. The final set of third normal form relations are presented in Table 14.

Step_I.5__Draw_a_conceptual_model_based_on_the_third normal_form_relations_from_step_I.4 Relations I to VIII are represented in a pictorial format as follows:

A. A relation for which the primary key consists of only one data element represents an entity. Relation I represents the entity STUDENT. All entities of this type are placed on Level 1. In Figure 27 the box STUDENT on Level 1 represents the entity in Relation I. The data elements were written inside the box. The primary key of the entity is underlined. If a box cannot hold all the data elements representing an entity, the missing data elements are shown by three dashes, - - -.

B. Relations with a primary key consisting of two data elements are placed on the second level. For example, in Relation VI, the primary key consists of two data elements. The box for Relation VI is drawn on Level 2. The primary key in box VI is a compound key and is underlined. The compound key of Relation VI is COURSE-NUMBER*DEPARTMENT. This key represents the relationship between two entities COURSE-NUMBER and DEPARTMENT. There is no box for the entity COURSE-NUMBER

TABLE 13. The Resulting Third Normal Form Relations

```
------------------------------------------------------------
!                                                          !
!   I. Relations  2, 4, 6, and 11 are identical:           !
!      SOC-SEC-NUM- <<---> NAME                            !
!  II. The key of Relation 1, 8 and 13 are identical to    !
!      Relation 2, therefore combining Relation 2 with     !
!      Relations 1, 8 and 13 results in:                   !
!      SOC-SEC-NUM <<---> NAME,SEX,CURRICULUM,COLLEGE,     !
!                          YEAR,CURRENT-GPA,               !
!                          SEMESTER-ADMITTED,TYPE,         !
!                          ENTRANCE-GPA,                   !
!                          TRANSFER-CREDIT,SEM-STD-TAUGHT!
!                          TEACHING-LEVEL,MAJOR,MINOR,     !
!                          HIGH-SCHOOL-RANK,ACT,           !
!                          ACT-ENGLISH                     !
!                          ACT-MATH,ACT-SOC-STUDIES,       !
!                          ACT-NATIONAL-SCI,SAT-VERBAL,    !
!                          SAT-MATH                        !
!                                                          !
! III. The composite key in Relation 3 is unique:          !
!      SOC-SEC-NUM*IN-TEST-ID*IN-TEST-TIME                 !
!      <<---> IN-NUM-CORRECT,IN-IND-RESPES                 !
!                                                          !
!  IV. Likewise with Relation 5:                           !
!      SOC-SEC-NUM*INTERVIEW-TYPE*INTERVIEW-DATE*          !
!      INTERVIEWER <<---> DIALOGUE                         !
!                                                          !
!   V. Likewise with Relation 7:                           !
!      SOC-SEC-NUM*EXPERIENCE-TYPE*EXPERIENCE-DATE         !
!      <<---> COMPOSITION                                  !
!                                                          !
!  VI. Likewise with Relation 9:                           !
!      SOC-SEC-NUM*SEMESTER-YEAR*COURSE-NUMBER*            !
!      DEPARTMENT <<---> GRADE,QUALITY-POINTS              !
!                                                          !
! VII. Likewise with Relation 10:                          !
!      DEPARTMENT*COURSE-NUMBER <<---> COURSE-NAME,        !
!      CREDIT                                              !
!                                                          !
!VIII. Likewise with Relation 12:                          !
!      SOC-SEC-NUM*PERF-TEST-ID*PERF-DATE-TIME             !
!      <<---> PERF-IND-RESPES                              !
!                                                          !
!  IX. Likewise with Relation 14:                          !
!      SOC-SEC-NUM*OUT-TEST-ID*OUT-DATE-TIME               !
!      <<--->OUT-NUM-QUESTS,OUT-NUM-CORRECT,               !
!            OUT-NUM-RESPES                                !
------------------------------------------------------------
```

TABLE 14. Final Set of Third Normal Form Relations

I.	SOC-SEC-NUM	<<--->	NAME, SEX, CURRICULUM, COLLEGE, YEAR, CURRENT-GPA, SEMESTER-ADMITTED, TYPE, ENTRANCE-GPA, TRANSFER-CREDIT, SEM-STD-TAUGHT, TEACHING-LEVEL, MAJOR, MINOR, HIGH-SCHOOL-RANK, ACT, ACT-ENGLISH, ACT-MATH, ACT-SOC-STUDIES, ACT-NATIONAL-SCI, SAT-VERBAL, SAT-MATH
II.	SOC-SEC-NUM*IN-TEST-ID*IN-TEST-TIME	<<--->	IN-NUM-CORRECT, IN-IND-RESPES
III.	SOC-SEC-NUM*INTERVIEW-TYPE*INTERVIEW-DATE* INTERVIEW-DATE	<<--->	DIALOGUE
IV.	SOC-SEC-NUM*EXPERIENCE-TYPE*EXPERIENCE-DATE	<<--->	COMPOSITION
V.	SOC-SEC-NUM*SEMESTER-YEAR*COURSE-NUMBER* DEPARTMENT	<<--->	GRADE, QUALITY-POINTS
VI.	DEPARTMENT*COURSE-NUMBER	<<--->	COURSE-NAME, CREDIT
VII.	SOC-SEC-NUM*PERF-TEST-ID*PERF-DATE-TIME	<<--->	PERF-IND-RESPES, PERF-NUM-CORRECT
VIII.	SOC-SEC-NUM*OUT-TEST-ID*OUT-DATE-TIME	<<--->	OUT-NUM-QUESTS, OUT-NUM-CORRECT, OUT-IND-RESPES

or DEPARTMENT. To establish the relationship between the entities COURSE-NUMBER and DEPARTMENT on Level 2, new entity relations for COURSE-NUMBER and DEPARTMENT are created on Level 1. These relations are created because no unique relationship in Figure 27 is defined by number of course (COURSE-NUMBER) or department (DEPARTMENT). The single-headed and double-headed arrows between Relations I and IV from Figure 27 mean that, for a given student, many experiences may take place, and a given experience may take place in many dates.

C. The procedure for Level 2 is repeated for Level 3 and so on.

The resulting diagram for relations I to VIII is shown in Figure 27.

The resulting third normal form relations are based on the assumptions made regarding the specific PRO*FILE environment; refer to page 95. Since the assumptions are subjective, there is always the chance that someone may challenge them. If the assumptions are changed, the third normal form relations change. The third normal forms are dependent upon the understanding of the environment in which they were designed.

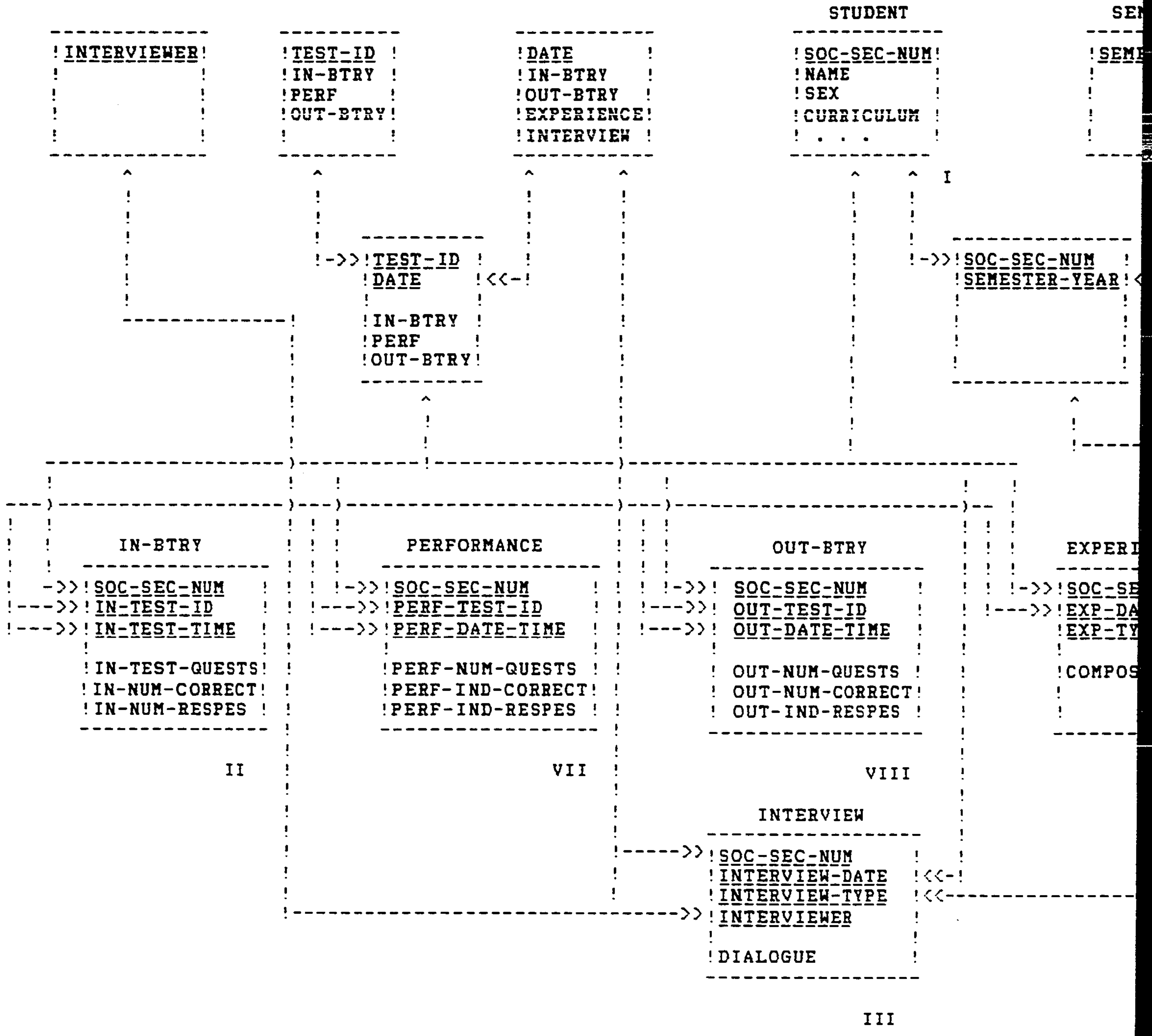
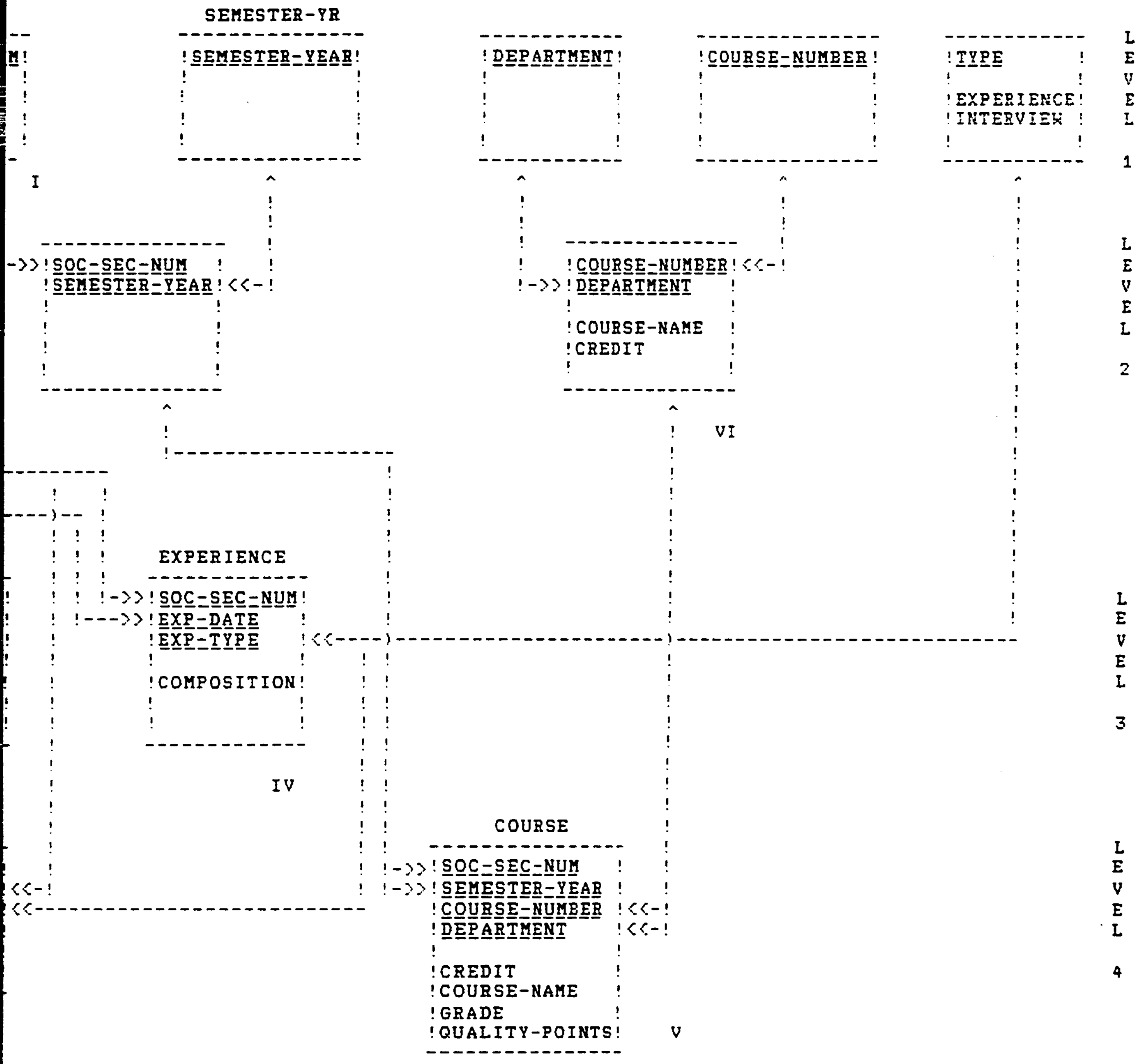


Figure 27. Conceptual Model for the PRO*FILE System



SEMESTER-YR
SEMESTER-YEAR
DEPARTMENT
COURSE-NUMBER
TYPE
EXPERIENCE
INTERVIEW
LEVEL 1
I
SOC-SEC-NUM
SEMESTER-YEAR
COURSE-NUMBER
DEPARTMENT
COURSE-NAME
CREDIT
LEVEL 2
VI
EXPERIENCE
SOC-SEC-NUM
EXP-DATE
EXP-TYPE
COMPOSITION
LEVEL 3
IV
COURSE
SOC-SEC-NUM
SEMESTER-YEAR
COURSE-NUMBER
DEPARTMENT
CREDIT
COURSE-NAME
GRADE
QUALITY-POINTS
V
LEVEL 4

Step_II__Design_of_a_Logical_Model_of_the_Data_Base

Step_II.1__Draw_a_logical_model_based_on_the conceptual_model_from_step_I.5_for_a_DBMS_using_a hierarchical_data_model

Step_II.1.1__Map_a_hierarchical_data_model

A.__Derive_a_hierarchical_model_without regard_for_a_particular_DBMS

A.1__Eliminate_transitivity In the conceptual model of Figure 27 (refer to page 122), the boxes represent segments, and the arrows <<--->, represent parent-child relationships between these segments. For example, a segment called PROFESSOR might be defined as a parent and another segment called STUDENT might be defined as a child. The parent-child relationship, PROFESSOR <<---> STUDENT, would mean that a Professor would have many Students assigned to him/her. Transitivity exists in the conceptual model if the relationship between two segments can be removed without any loss of essential information. Considering all the relationships between segements in the PRO*FILE conceptual model, there are no relationships between any segments that can be removed without losing information in Figure 27.

A.2 Derive parent-child relationships

This step verifies that all relationships between segments are defined. When subsets A.1 and A.2 are applied to the conceptual model of Figure 27, (refer to page 122) the intermediate result of Figure 28 is obtained. The intermediate result presented in Figure 28 concludes that all segments are necessary and all relationships between segments are defined. It is intermediate in the sense that the diagram is not a hierarchical data model as there may be multiple parents for any given child.

A.3 Resolve multiple parentage

It is possible that a child could have a relationship with two or more parents. In the conceptual model, as presented in Figure 27, there are several segments with two or more parents. In a hierarchical data model, a child can have only one parent. The resolution of this problem is to combine segments and create compound keys. In Figure 27, the relationships among STUDENT, TEST-ID, DATE and TEST-ID*DATE with IN-BTRY are removed. The data elements are absorbed in the IN-BTRY segment and the keys are combined to create a compound key. Likewise, the relationship between TEST-ID and DATE with PERFORMANCE, OUT-BTRY, EXPERIENCE, and INTERVIEW. The relationships among DEPARTMENT and COURSE-NUMBER are combined with COURSE. By resolving multiple parentage to the intermediate

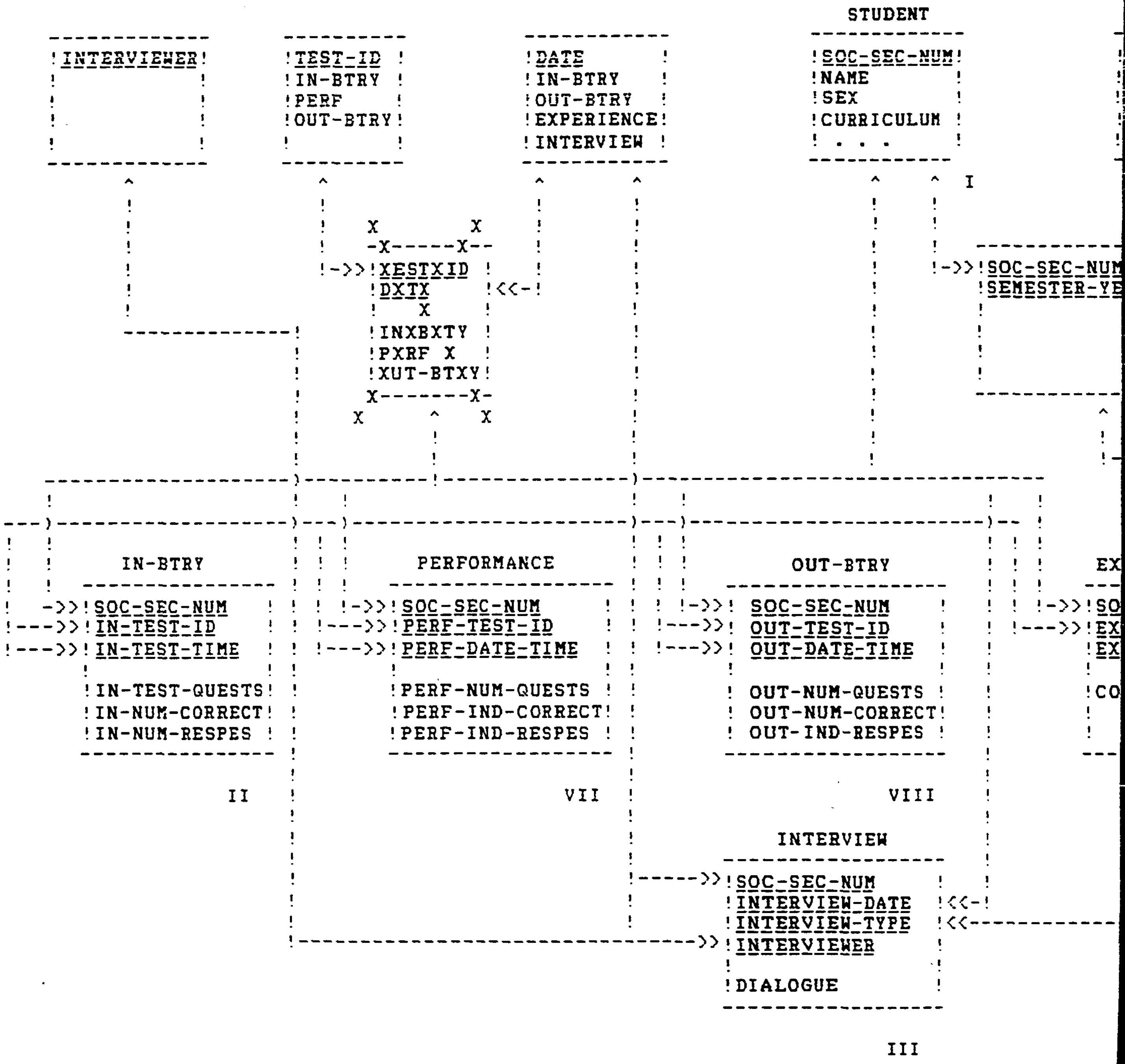
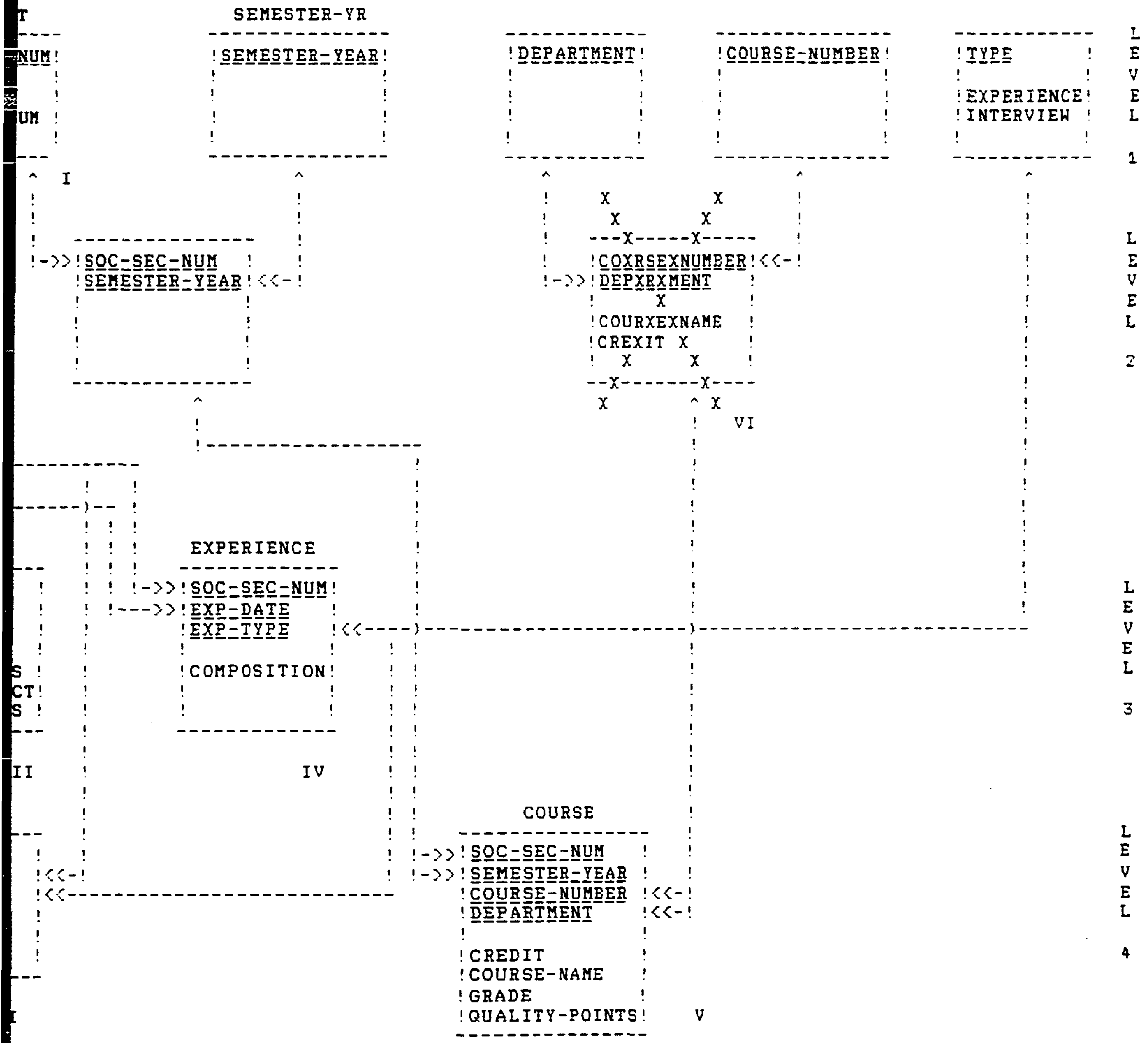


Figure 28. Resolve Multiple Parentage from the Conceptual Model of the PRO*FILE Environment


SEMESTER-YR
SEMESTER-YEAR
DEPARTMENT
COURSE-NUMBER
TYPE
EXPERIENCE
INTERVIEW
LEVEL 1
I
SOC-SEC-NUM
SEMESTER-YEAR
COXRSEXNUMBER
DEPXRXMENT
COURXEXNAME
CREXIT
LEVEL 2
VI
EXPERIENCE
SOC-SEC-NUM
EXP-DATE
EXP-TYPE
COMPOSITION
LEVEL 3
II
IV
COURSE
SOC-SEC-NUM
SEMESTER-YEAR
COURSE-NUMBER
DEPARTMENT
CREDIT
COURSE-NAME
GRADE
QUALITY-POINTS
V
LEVEL 4

result in Figure 28, the result is Figure 29, the hierarchical model that can be physically mapped to SPIRES.

B. Modify the hierarchical data model to eliminate conflicts with the rules of the DBMS to be used The selected DBMS is SPIRES. SPIRES fully supports the hierarchical logical model in Figure 29, therefore, no modifications are required.

C. Refine the modified data model according to "obvious" performance considerations Parents having only one child are candidates for combination with their children. The trade-off is between data redundancy and performance. The hierarchical model in Figure 29 requires no modification. Although it is possible to combine the segments SEMESTER and COURSES, the amount of redundant data would cancel any possible benefit of performance as each course would have to store data on the semester and year taken. Additional refinements may become obvious as the prototype is implemented and quantitative information becomes available.

D. Add relationships that exist between the data The refined logical model in Figure 29 satisfies the functional data requirements. At this point the logical design is considered complete. Future researchers, however, may want to strengthen the logical design by adding some intrinsic data relationships.

```
                         STUDENT
                     ------------------
                     !NAME            !
                     !SOC-SEC-NUM     !
                     !SEX             !
                     !CURRICULUM      !
                     !ENTRANCE-GPA    !
                     !MAJOR           !
                     !MINOR           !
                     !HIGH-SCHOOL-RANK!
                     !ACT-ENGLISH     !
                     !ETC.   .  .  .  !
                     ------------------
                             !
                             !
        ----------------------------------------------
        !        !       !       !         !         !
        !        !       !       !         !         !
    IN-ETRY      !   EXPERIENCE  !       OUT-ETRY    !
---------------- ! ------------- ! ----------------- !
!IN-TEST-ID    ! ! !EXPERIENCE-! ! !OUT-TEST-ID    ! !
!IN-TEST-TIME  ! ! ! TYPE      ! ! !OUT-DATE-TIME  ! !
!IN-NUM-QUESTS ! ! !EXPERIENCE-! ! !OUT-NUM-QUESTS ! !
!IN-NUM-CORRECT! ! ! DATE      ! ! !OUT-NUM-CORRECT! !
!IN-IND-RESPES ! ! !COMPOSITION! ! !OUT-IND-RESPES ! !
---------------- ! ------------- ! ----------------- !
                 !               !                   !
                 !               !                   !
          INTERVIEW           PERF-ELMT            SEMESTER
    ----------------      ------------------     -----------
    !INTERVIEW-TYPE!      !PERF-TEST-ID    !     !SEMESTER-!
    !INTERVIEW-DATE!      !PERF-DATE-TIME  !     ! YEAR    !
    !INTERVIEWER   !      !PERF-NUM-QUESTS !     !         !
    !DIALOGUE      !      !PERF-NUM-CORRECT!     !         !
    !              !      !PERF-NUM-RESPES !     !         !
    ----------------      ------------------     -----------
                                                      !
                                                      !
                                                COURSES
                                           ----------------
                                           !DEPARTMENT    !
                                           !COURSE-NUMBER !
                                           !CREDIT        !
                                           !COURSE-NAME   !
                                           !QUALITY-POINTS!
                                           ----------------
```

FIGURE 29. Mapping of the Conceptual Model from Figure 27 to a Hierarchical Data Model, using SPIRES - PRO*FILE Environment

Step_II.2__Draw_external_model_for_the_reports_in Figures_5_to_12_on_the_basis_of_the_logical_model_from Figure_29 External model refers to a representation, or portion thereof, of the internal model needed to generate a physical report that can be read. The external model for the Program of Study uses three segments and their corresponding parent-child relationships. The external model for Student History uses only one segment. The remaining external model representing the other reports all use two segments and one parent-child relationship. The external models are shown in Figure 30.

Step_III__Design_of_a_Physical_Model_of_the_Data_Base

STEP_III.1__Draw_an_internal_model_(also_called_a "physical"_model)_on_the_basis_of_the_logical_model_from step_II.1 The following segments (or record) types are considered: STUDENT, IN-ETRY, INTERVIEW, EXPERIENCES, SEMESTER, COURSE, PERFORMANCE-ELMT, AND OUT-ETRY. The segment sizes are estimated in Table 15. The procedure estimates the size of each data element and each data element is assigned to a segment. In a hierarchical Data Base Management System, the path between segments are referred to as "pointers". There are typically three pointers: up, down and right for each segment. The path from a parent to a child would follow the down pointer.

```
   STUDENT
   HISTORY
-----------      -----------                   -----------
!SOCSECNUM!      !SOCSECNUM!                   !SOCSECNUM!
!---------!      !---------!                   !---------!
!NAME     !      !NAME     !                   !NAME     !
!ACT-MATH !      !ACT-MATH !                   !ACT-MATH !
! .  .  . !      ! .  .  . !                   ! .  .  . !
-----------      -----------                   -----------
          I           !                             !
                      !                             !
                      !                             !
                      !                             !
                 -----------                   -----------
          INBTRY !TEST-ID  !  II       PROGRAM !SEMESTER-!
                 !-------  !              OF   !-------- !
                 !TEST-TIME!--          STUDY  !  YEAR   !
                 !---------! !                 !  ----   !
                 ! .  .  . ! !                 !         !
                 ----------- !--               -----------
         INTERVIEW !INTER-   ! !  III               !   VI
                   !  VIEWER ! !                    !
                   ----------- !--                  !
          EXPERIENCE !EXP-DATE ! !  IV              !
                     ! .  .  . ! !             -----------
                     ----------- !--           !CRS-NUM  !
           PERFORMANCE !TEST-ID  ! !  VII      !-------  !
                       ! .  .  . ! !           !DEPT     !
                       ----------- !           !----     !
                OUT-BTRY !TEST-ID  !   VIII    ! .  .  . !
                         ! .  .  . !           -----------
                         -----------                    V
```

FIGURE 30. External Model for all Reports Based on the Logical Model from Figure 29

TABLE 15. Estimate of Field and Segment Size

Segment	Field	Length (bytes)
STUDENT	NAME	23
	SOC-SEC-NUM	4
	SEX	1
	CURRICULUM	5
	COLLEGE	1
	YEAR	4
	CURRENT-GPA	4
	SEMESTER-ADMITTED	4
	TYPE	1
	ENTRANCE-GPA	4
	TRANSFER-CREDIT	4
	SEM-STD-TAUGHT	4
	TEACHING-LEVEL	4
	MAJOR	5
	MINOR	5
	HIGH-SCHOOL-RANK	4
	ACT	4
	ACT-ENGLISH	4
	ACT-MATH	4
	ACT-SOC-STUDIES	4
	ACT-NATIONAL-SCI	4
	SAT-VERBAL	4
	SAT-MATH	4

		105
IN-BTRY	IN-TEST-ID	4
	IN-TEST-TIME	10
	IN-NUM-QUESTS	4
	IN-NUM-CORRECT	4
	IN-IND-RESPES	55

		77

TABLE 15. (Continued)

Segment	Field	Length (bytes)
INTERVIEW	INTERVIEW-TYPE	25
	INTERVIEW-DATE	4
	INTERVIEWER	23
	DIALOGUE	72

		124
EXPERIENCES	EXPERIENCE-TYPE	25
	EXPERIENCE-DATE	4
	COMPOSITION	72

		101
SEMESTER	SEMESTER-YEAR	5

		5
COURSE	DEPARTMENT	5
	COURSE-NUMBER	4
	CREDIT	4
	COURSE-NAME	19
	GRADE	2
	QUALITY-POINTS	4

		38

TABLE 15. (Continued)

Segment	Field	Length (bytes)
PERFORMANCE-ELMT	PERF-TEST-ID	4
	PERF-DATE-TIME	10
	PERF-NUM-QUESTS	4
	PERF-NUM-CORRECT	4
	PERF-IND-RESPES	55

		77
OUT-ETRY	OUT-TEST-ID	4
	OUT-DATE-TIME	10
	OUT-NUM-QUESTS	4
	OUT-NUM-CORRECT	4
	OUT-IND-RESPES	55

		77

Likewise, the path from a child to the parent would follow the up pointer. Finally, the path from a given segment to the next adjacent segment follows the right pointer. It is assumed that, on average, three pointers are stored in every segment, that is, 12 bytes have to be added to each segment.

The following assumptions were made regarding segment frequencies (C. R. Kniker, Secondary Education, ISU, October 19, 1982):

1. There are 600 students.

2. Each student will take one In Battery examination, on the average.
3. Each student would average 10 interviews.
4. Each student would average 10 experiences.
5. Each student would have enrolled in an average of nine semesters.
6. Each semester a student would enroll for an average of five courses.
7. Each student would average 10 performance element examinations.
8. Each student would average one Out Battery examination.

These assumptions are used to determine the frequencies that appear in Figure 31. The logical model in Figure 29, page 127, is reflected with the respective lengths (L) and average frequencies (F) relative to the parents in Figure 31.

In order to evaluate the physical model in the next step, the blocksize and padding requirements were estimated. The prototype follows IBM paging alignment recommendations for the blocksize estimate. Padding is a DBMS term that defines the percentage of a block that is left unfilled for future use. The padding percentage is determined by the amount of additions to the data base. As data is added, it is beneficial to have newly created segments that are related to other segments in the same block, if at all possible. Internally, the computer is able to read the segments while minimizing the input/output count.

To calculate the space and time estimates, a blocksize of 4096 bytes (4 x 1024, i.e., 4K) with FSB (free space blocks) and FSW (free space within a block) values of 10% each.

The internal model represents one physical data base. There are two physical relationships interconnecting the segments:

1. STUDENT is the physical parent of IN-BTRY, INTERVIEW, EXPERIENCES, SEMESTER, PERF-ELEMT, and OUT-BTRY; and
2. SEMESTER is the physical parent of COURSES.

This step designed the physical model of the data base. The analysis started by considering segment size, followed by estimating the frequencies based upon assumptions. Finally, a blocksize and padding factor were chosen.

```
                           STUDENT
                         -----------
                         !NAME     !
                         !SOCSECNUM!
                         !SEX      !
                         !SAT-MATH !
                         ! -  -  - !
                         -----------
                              !  F = 600
                              !  L = 101 + 12
                              !
                              !
      -------------------------------------------------
      !       !       !       !       !               !
      !       !       !       !       !               !
      !       !       !       !       !               !
   IN-BTRY    !  EXPERIENCE   !    OUT-BTRY        SEMESTER
-----------   !  -----------  ! -----------     -----------
!TEST-ID  !   !  !EXP-TYPE !  ! !TEST-ID  !     !SEMESTER-!
!TEST-TIME!   !  !EXP-DATE !  ! !DATE-TIME!     !  YEAR   !
!TEST-    !   !  !COMPOSIT-!  ! !NUM-     !     !         !
!  QUESTS !   !  !  ION    !  ! !  QUESTS !     !         !
! -  -  - !   !  !         !  ! ! -  -  - !     !         !
-----------   !  -----------  ! -----------     -----------
F = 1         !  F = 10       ! F = 1           F = 9!
L = 77 + 12   !  L = 101 + 12 ! L = 77 + 12     L = 5!+ 12
              !               !                     !
          INTERVIEW      PERF-ELMT                  !
         -----------    -----------                 !
         !INT-TYPE !    !TEST-ID  !                 !
         !INT-DATE !    !DATE-TIME!                 !
         !INTER-   !    !IND-     !                 !
         !  VIEWER !    !  RESPES !                 !
         !DIALOGUE !    ! -  -  - !                 !
         -----------    -----------                 !
          F = 10        F = 10                      !
          L 124 + 12    L = 77 + 12                 !
                                                 COURSES
                                               -----------
                                               !DEPT     !
                                               !CRS-NUM  !
                                               !CREDIT   !
                                               !GRADE    !
                                               ! -  -  - !
                                               -----------
                                               F = 5
                                               L = 38 + 12
```

FIGURE 31. One physical data base for SPIRES: There are seven relationships connecting the segments

Step_IV__Evaluation_of_the_Physical_Model_of_the_Data_Base

Step_IV.1__Develop_space_estimates_and_input/output probabilities_for_the_internal_model_above_(as_in_Step III.1) The data base designer uses estimates to determine the amount of storage the data base may require and internal input/output (I/O) activity. Estimating the size of the data base is important to ensure that sufficient disk space is available on the computer system. Likewise, estimating the internal I/O activity gives the designer an idea of the required internal processing required to find and move data for processing. The process starts with estimating the number of bytes (characters) required for the hierarchical structure consisting of all segments and corresponding frequencies.

Space_Estimates Estimate the number of bytes (characters) required for the hierarchical structure consisting of all segments and corresponding frequencies.

PRO*FILE Data Base - STUDENT Segment (S)

S IN-ETRY = 89 bytes

S INTERVIEW = 136 bytes

S EXPERIENCE = 113 bytes

S SEMESTER = 17 bytes

S COURSES = 60 bytes

S PERF-ELMTS = 89 bytes

S OUT-BTRY = 89 bytes

S STUDENT = 113 + (89 * 1) + (113 * 10) + (17 * 10) +

((60 * 5) * (17 * 9)) + (89 * 10) +

(89 * 1) = 48381 bytes.

The above process estimates that a typical student's hierarchical structure requires 48,381 bytes (characters).

Segment Btyes There were approximately 600 education students in the Teacher Education Program in 1982. Therefore, the estimate for the hierarchical structure (48,381 bytes) is multiplied by the number of students (600). The estimated size of the data base consisting of pure data is 29,028,600 bytes (characters).

Effective blocksize This process calculates the most effective blocksize for internal processing. The blocksize is used to maximize the use of memory, I/O buffers and disk management routines. There are four criteria used in the calculation:

1. Frame alignment blocksize = 4096 bytes.
2. Wasted space (W)/block = the biggest segment size 136 (Interview Segment) - 1 = 135 bytes.
3. Free space within (FSW); fraction of each block left free at load. This was previously discussed as a padding factor. For this application, 10%

is used.

4. Free space blocks (FSB); fraction of whole blocks left free at load. For this application, 10% is used.

Effective blocksize = (((1-FSW * BLK) - W) * (1-FSB)

= (((1 - 0.1) * 4096) - 135) * 0.9

= 3196 bytes

The most effective blocksize is estimated as 3,196 bytes (characters).

__Block_Bytes__ The designer may now calculate the estimated size of the data base.

Block bytes = Segment Bytes * blocksize/ effective blocksize

= 29,028,600 * 4096 / 3196

= 37,203,111 bytes

The total number of bytes (characters) to store the data base is 37,203,111 bytes. This is equivalant to approximately 9 million computer words.

__Input/Output_(I/O)_Probabilities__ Physical (I/O) is the movement of data from disk storage through buffers to memory and return. One of the biggest time consumers in the execution of data base calls is physical I/O activity (Atre, 1981). In the hierarchy of a physical data base, the length of every segment type, the expected frequency of occurrence of every segment type relative to its parent, and the sequence of segment accesses for a specific application can be used as predictors for finding the I/O activity necessary to access the required segments.

The probability that the parent and the first occurrence of the child segment under the parent are not in the same block (called "control interval" for virtual storage access method: VSAM) is "PCIO" (physical child input/output). According to Atre (1980), the assumption made is that all occurrences of the child segment type are likely to be selected equally. The probability that a segment and its twin are not in the same block is "PTIO" (physical twin input/output). Finally, the probability that the segment and its parent are not in the same block is "PPIO" (physical parent input/output). The equations used to calculate these probabilities are found in Table 16. The PRO*FILE data base probabilities are presented in Table 17.

TABLE 16. Equations for Input/Output Probabilities

```
-----------------------------------------------------------

    PC(a,b)    = length of B + all subtrees to left of B



                    --------------------------------------
    PCIO(a,b) =     !                PC(a,b)             !
               min. !1, ---------------------------------!
                    !   (((1-FSW) * BLK) - W) * (1 - FSB)!
                    --------------------------------------



                                   ------------------------
                                   !          S(a)        !
    PTIO for segment type A = min. !1, -------------------!
                                   !   effective blocksize!
                                   ------------------------



                         -------------
                         ! f(b)      !
    PP(b,a)    = L(b) +  ! ---  * S(b)!  + all the subtrees to
                         !  2        !    left of B
                         -------------

                where L(b) = length of segment b
                      f(b) = average frequency of
                             occurrence of B under its
                             parent A
                      S(b) = average subtree size of B



                      ----------------------------------
    PPIO(b,a) = min.  ! 1,            PP(b,a)          !
                      !    ----------------------------!
                      !    (((1-FSW)*BLK)-W) * (1 - FSB)
                      ----------------------------------

-----------------------------------------------------------
```

TABLE 17. PRO*FILE Data Base Input/Output Probabilities

Segments	Min	Cum	PCIO	PTIO	PPIO
IN-BTRY	1	89	.03	.03	.04
INTERVIEW	1	225	.07	.04	.28
EXPERIENCES	1	1562	.49	.04	.64
SEMESTER	1	2596	.81	.01	.84
COURSE	1	2782	.87	.02	.89
PERF-ELMT	1	4846	1.52	.03	1.66
OUT-BTRY	1	5736	1.79	.03	1.81

Preliminary Field Testing

The PRO*FILE System was implemented using the SPIRES Data Base Management System and loaded with 12 hypothetical student records. The implemented system was tested and modifications made to ensure that the reports as depicted in Figures 5 to 12, were accurate. The on-line queries verified that the English-like language could retrieve information based upon the desire of the response required. These tests also included verifying the PRO*FILE Systems' functions such as COUNT, AVERAGE and STANDARD DEVIATION.

The Preliminary Field Test consisted of the actual use of the PRO*FILE System by two representative groups. The PRO*FILE Task Force represented the view of the administration and faculty, while RISE Graduate Research Assistants represented the view of students. Over a period of one week, the two groups evaluated the prototyped system. The evaluation started with a system overview, followed by a demonstration of capabilities, and finally, the actual use of the PRO*FILE system to input data and retrieve information. Responses, comments and recommendations were recorded by this author using an evaluation recording form (Appendix J).

The criteria used to evaluate the Preliminary Field Test consisted of the following four areas:

1. Functional Requirements met (Appendix B);

2. Ease of use;
3. Response time; and
4. Utility

The groups were first presented with the PRO*FILE Systems' Architecture. Evaluators reviewed the hardware and software integration that was depicted in Figure 2, page 69. This presentation was important in order to establish the network environment whereby users of PRO*FILE could access the system in the batch mode or interactively through the use of a terminal.

The PRO*FILE Systems' prototype, demonstrated by this author, showed that the basic functional requirements were met using the ad-hoc query language based upon the criteria in FIFS PUB 77 (1980). The capabilities of the PRO*FILE System allowed administrative staff, advisers, teacher education faculty, and students the ability to view, add, modify and delete items of data from a student's folder. Sample reports from Figures 5 through 12 were generated. The PRO*FILE Task Force recommended that the system be expanded to add the capability to retrieve information on performance elements and administer self-testing with immediate feedback directing the student to the correct answer and appropriate reference material.

The requirement for security to restrict access to sensitive or confidential data was demonstrated.

Evaluators asked to see data that were password protected using the on-line query language. Evaluators found that sensitive data were omitted from the output, or they were notified they did not have authorization to view the data. Although security was proven adequate to protect student data, the PRO*FILE Task Force was unwilling to consider submitting an application to the Human Subjects Review Board for the use of real student data until further research was provided for application. In addition, evaluators expressed misgivings about the redundancy of maintaining student data on both the Academic and Administrative computer system.

Once past the initial learning phase of how to use the PRO*FILE System, both groups considered the ease of use requirement satisfactory. The RISE Graduate Research Assistants, all of whom used computer terminals in their assistantships, found the English-like query language easy to understand and use. The PRO*FILE Task Force, with little or no experience using terminals or query languages, at first expressed apprehension on using the terminal and English-like query language. However, all evaluators found that it was easy to learn and follow to obtain ad-hoc information using the English-like language.

Response time averaged five seconds or less per query. The average response time may be somewhat misleading.

Recall that there are only 12 hypothetical students in the prototype data base. This amount of data can be stored internally on two computer pages. Therefore, once the data base was brought up and made available to the evaluators, the data were accessible via buffer pools. Performance considerations, such as buffer pool statistics, I/O times and I/O wait were therefore minimum.

The utility of the PRO*FILE Systems' prototype demonstrated potential beyond the physical limits of the campus. The evaluation demonstrated that through the university telecommunication network, potential users could have access via direct lines to the computer, remote terminals and even long distance via telephone lines using the dial-up capability. The ease of use and ability to format information in the ad-hoc request mode presented an automated capability for teacher education evaluation.

Main Product Revision

Performance Elements encompassed within PRO*FILE may be called competencies. There are approximately 100 performance elements as identified by the PRO*FILE research effort. The inclusion of the Performance Elements represented an additional data base with the PRO*FILE System. In keeping with the desire of the PRO*FILE Task Force for ease of use, an automatic logon procedure was developed. When a user logged onto the Iowa State University computer system, a menu would be presented asking the user which option s/he wanted to use:

1. Performance Elements;
2. PRO*FILE Student Records; or
3. Logoff System.

The student would simply follow the instructions and enter the corresponding number. A sample of this procedure is found in Appendix G.

The development of the Performance Elements (PERF) System followed the same steps as previously described in Development of a Preliminary Form of the Product, page 83. During the Fall of 1982 a similar system had been developed by the staff of the university computation center called the "The Iowa State University Student Information System". Although this system was not fully operational, the methodology of providing students and faculty information about the university was a one to one relationship along with providing information about performance elements. The Director of RISE requested that the software and documentation be made available to the ongoing PRO*FILE research effort. Due to the fact that the university computational center had the software and corresponding documentation under copyright, the Director also requested permission to modify the software as required.

The main revision of the PRO*FILE System included the PERF subsystem. The overall system is menu driven from the initial logon. The following list of Appendices will provide detail information about the enhanced system:

Appendix C - physical data base definition

Appendix D - menu.driver

Appendix E - menu instructions

Appendix F - display generation

Appendix G - system generation

Appendix H - building and maintaining performance element frames

Appendix I - master list of performance elements

The final PRO*FILE System, at the end of the 5th Step, Main Product Revision of the Educational R&D Cycle, is depicted in Figure 32. This product and corresponding documentation was turned over to the on-going PRO*FILE Task Force Research Project for main field testing and the remainder of the Educational R&D Cycle.

```
          AREAS                  STUDENT                  FRAME
   -----------------      -----------------      -----------------
   !INSTRUCTION     !     !NAME            !     !OBJECTIVES      !
   !EVALUATION      !     !SOC-SEC-NUM     !     !RESOURCES       !
   !MANAGEMENT      !     !SEX             !     !ACTIVITIES      !
   !ETC.   .  .  .  !     !CURRICULUM      !     !ECT.   .  .  .  !
   !                !     !ENTRANCE-GPA    !     !                !
   !                !     !MAJOR           !     !                !
   !                !     !MINOR           !     !                !
   !                !     !HIGH-SCHOOL-RANK!     !                !
   !                !     !ACT-ENGLISH     !     !                !
   !                !     !ETC.   .  .  .  !     !                !
   -----------------      -----------------      -----------------
                                   !
                                   !
           --------------------------------------------
           !        !       !        !       !        !
           !        !       !        !       !        !
       IN-ETRY      !  EXPERIENCE    !     OUT-ETRY   !
    --------------- !  ------------  !  ------------- !
    !IN-TEST-ID    ! ! !EXPERIENCE-! ! !OUT-TEST-ID    ! !
    !IN-TEST-TIME  ! ! ! TYPE      ! ! !OUT-DATE-TIME  ! !
    !IN-NUM-QUESTS ! ! !EXPERIENCE-! ! !OUT-NUM-QUESTS ! !
    !IN-NUM-CORRECT! ! ! DATE      ! ! !OUT-NUM-CORRECT! !
    !IN-IND-RESPES ! ! !COMPOSITION! ! !OUT-IND-RESPES ! !
    --------------- !  ------------  !  --------------- !
                    !                !                  !
                    !                !                  !
           INTERVIEW          PERF-ELMT            SEMESTER
        ---------------   -----------------      -----------
        !INTERVIEW-TYPE!  !PERF-TEST-ID     !    !SEMESTER-!
        !INTERVIEW-DATE!  !PERF-DATE-TIME   !    ! YEAR    !
        !INTERVIEWER   !  !PERF-NUM-QUESTS  !    !         !
        !DIALOGUE      !  !PERF-NUM-CORRECT!     !         !
        !              !  !PERF-NUM-RESPES  !    !         !
        ---------------   -----------------      -----------
                                                      !
                                                      !
                                                   COURSES
                                              ---------------
                                              !DEPARTMENT    !
                                              !COURSE-NUMBER !
                                              !CREDIT        !
                                              !COURSE-NAME   !
                                              !QUALITY-POINTS!
                                              ---------------
```

FIGURE 32. The Final PRO*FILE Logical Model

Summary

This chapter presented the process used to develop the computerized PRO*FILE Sytems' model. The model was guided by the first five steps of the Educational R&D Cycle. Using this procedure, the prototype was designed, implemented, reviewed, and revised. The next chapter presents the findings from the R&D process used to develop the PRO*FILE prototype and evaluates the prototype.

FINDINGS

This chapter presents the findings of the study. First, the interview process used to evaluate the prototype is described. The observations are noted and associated costs for the development are identified next. Finally, data base size estimates and input/output probabilities are discussed.

Interview

The interview process followed a procedure developed by Richardson et al. (1982) for Instructional Design Interviews. This procedure provided a framework for the PRO*FILE Systems' prototype evaluation.

The interviewer of the PRO*FILE Systems' prototype was a doctoral candidate in Education specializing in Higher Education and also an Assistant Professor of Computer and Management Science. Additionally, the interviewer was the principal researcher and designer of the PRO*FILE Systems' prototype.

Procedures

Within the context of the educational R&D cycle the PRO*FILE Systems' prototype was evaluated at various stages of development. During the Fall Semester, 1982, an abbreviated prototype was developed to demonstrate the capabilities of a Data Base Management System. At the end of that development, a pilot demonstration and interview process was conducted to test the procedures and materials developed for the PRO*FILE Systems' prototype evaluation. Results of the pilot demonstration and interview indicated that it was feasible to carry out multiple interviews with different groups and that the process would yield useful data on the evaluation of the prototype. As a result of the pilot study, the following procedures were used in the evaluation of the prototype:

1. Three groups were provided with a demonstration and corresponding interview. The groups represented the administration, faculty and students.
   a. The College of Education Administration - The Dean, Associate Dean and Director of RISE
   b. Faculty - PRO*FILE Task Force
   c. Students - Graduate Research Assistants. RISE
2. An overview of the meeting and PRO*FILE design activities was provided.
3. PRO*FILE Systems' objectives concerning computerization were provided.
4. PRO*FILE Systems' objectives were confirmed.

5. Demonstration and interaction was completed.
6. Perceived Ideal was ascertained.

Materials

The PRO*FILE Systems' prototype provided the vehicle for the demonstration and corresponding interview. The interview process used the functional specifications of the PRO*FILE System (see Appendix B) to a) list the objectives; b) determine the activities undertaken to meet each objective; and c) show how attainment of each objective was assessed. During the demonstration and corresponding interview, the interviewer recorded findings on a predetermined form that followed the format outlined on the previous page. See Appendix J for detailed sample of the form.

Results

Educational Research and Development Cycle

The rationale for using the educational R&D cycle was that it was a process used to develop and validate educational products. The PRO*FILE System was considered as a potential educational product. This study used the first five of the 10 major steps, the process provided a smooth transition from the initial idea to a computerized prototype.

Planning The planning used to identify computer resources and facilities in conjunction with PRO*FILE Systems' Functional Specifications provided the foundation for the systems design. The PRO*FILE Functional Specifications requiring access flexibility, integrity, data independence, and security met the objectives for using a DBMS as detailed in Table 1, page 21. The DBMS environment also offered the PRO*FILE Task Force non-redundance of data, relatability, and administration and control. The PRO*FILE Task Force and Director of RISE thought that the use of 12 hypothetical students, representing actual scenarios of education students, allowed for ample systems' testing in a timely progression. However, the planning did not consider establishing a long range strategic plan for the PRO*FILE System, that is, what should the PRO*FILE System be able to do for the College of Education five to seven years hence.

Data_Base_Design The actual design and implementation of the PRO*FILE Systems' prototype followed the procedure provided by Atre, 1981 in her book Data_Base -_Structured_Techniques_for_Design,_Performance,_and Management. This procedure, as presented in Table 9, page 74, took the prototype through a structured analysis, design and implementation that provided the required information in the most efficient and effective manner. The normalization process and decomposition of structures described in the analysis and design of the prototype was implemented on the selected Data Base Management System, SPIRES; normalization and decomposition of structures reduced redundancy and took advantage of the benefits of Data Base Management Systems technology. Data Base Management theories and methodologies, as presented in Chapter 2, page 20, validated the PRO*FILE Systems design.

<u>SPIRES</u> At the time of this study, the only Data Base Management System available on the university computer was SPIRES. However, SPIRES did meet the criteria used in the selection of a DBMS, as presented in Table 8, page 67. Specific criteria met by SPIRES included database definition, data manipulation, system and integrity control. To a lesser degree the criteria of performance, quality and support were met. The evaluators found the prototype, with its English-like language, was easy to use, as per FIPS PUB 77 Guideline For Planning and Management of Database Applications (1980). Considerable programming effort by the designer was necessary to make the prototype appear "user friendly" to the novice. From the evaluator's and user's viewpoints, SPIRES not only met the FIPS PUB 77 criteria for adequacy but was sought for other applications

once its capabilities were revealed.

This writer/designer of the PRO*FILE Systems' prototype found SPIRES lacking in flexibility, facilities and documentation, as per FIPS PUB 99 Guideline: A Framework For The Evaluation and Comparison of Software Development Tools (1983). A networking or a relational data base logical model was not allowed using SPIRES. Restricting the logical model to a flat file or hierarchical model represents 15 year old technology (Cardenas, 1979). If one accepts the premise that computer technology turns over once every five to seven years (Adams, Wagner, and Boyer, 1983), SPIRES is outdated. Although the PRO*FILE System has the capability to download data from a mainframe to a micro-computer, it requires exiting the SPIRES environment. This in turn requires using the university's main computer system and telecommunications network to accomplish the task. Current (state of the art) Data Base Management Systems include inherent mainframe to micro links, graphics, and statistical analysis capabilities integrated with the Data Base Management System, as in FOCUS (Information Builders Inc., 1983). In addition to the above capabilities, FOCUS offers a personal computer (PC) version of the DBMS that interfaces with the FOCUS DBMS residing on the mainframe computer. At the time of this publication, PC FOCUS was priced at approximately $1,500.00 per copy. These expanded facilities would allow administrators, faculty and students to use the PRO*FILE System for analysis and planning in an efficient, effective manner and friendly mode.

Finally, the quality of SPIRES documentation is bulky and hard to use. When searching for a discussion on or clarification of capabilities, many cross-references lead back to a discussion of the original capability. Many hours of research were required to find how to accomplish a simple task that used a capability or series of capabilities.

PRO*FILE_Functional_Specifications The evaluators were provided a demonstration of the system and its capabilities. They were also offered the opportunity to log onto the system and actually use it as would a prospective administrator, faculty member or student. Each group took the PRO*FILE Systems' prototype through the functional specifications as required by the PRO*FILE Task Force. SPIRES fulfilled the requirements with its ad-hoc English-like language and its automatic report writer feature. Evaluators could see the benefits of prototyping and offered suggestions for systems additions and modifications. One such suggestion coming from the first review was to add the Performance Elements capability to the PRO*FILE System. One of the major benefits of using a

Data Base Management Systems is that it offers the ability to change the structure without having to modify current programs and data (Date, 1977). Future modifications and enhancements could be added with little impact, based on the SPIRES data base structure and ad-hoc commands. This includes adding new data elements, declaring keys, adding new segments, and new hierarchical structures, if necessary.

Sample_Size The PRO*FILE Systems prototype was loaded with 12 hypothetical students. Each student had representative data corresponding to typical students within the College of Education. These data were used to test the system and provide data for demonstrations and evaluations. Although these data were sufficient for a prototyping mode, actual student data should be used for the remaining five steps within the educational and research development cycle. Due to the small sample size, estimates on the data base size and performance were not realized.

Support_Staff Evaluators expressed concern about the technical support available once the PRO*FILE Systems' prototype was turned over to the PRO*FILE Task Force for field testing. SPIRES is not a well-known or commercially accepted Data Base Management System by industry. Therefore, finding experienced and well-trained programmer/analysts will continue to be a problem. Additionally, the university's computer support staff does not have a qualified resident software expert in SPIRES, nor should they have, based on the use of SPIRES on campus. The PRO*FILE Task Force during its evaluation determined that if PRO*FILE is to remain on SPIRES, or on any other commercial Data Base Management System, adequate funding must be made available to hire and train qualified support personnel. Additionally, their analysis stated that to continue the development of PRO*FILE without addressing this problem and resolving it will only cause serious delays and even the end of PRO*FILE itself. For the PRO*FILE System to meet its full potential, qualified technical staff must be available to maintain and enhance the system.

The interview process and corresponding evaluations determined that SPIRES met the functional specifications for the PRO*FILE System. Additional DBMS capabilities such as statistical, graphic, and a mainframe to micro link would enhance the PRO*FILE System. Evaluators expressed concern about a) the level of technical expertise needed to keep the PRO*FILE System running, b) how the technologist communicates to management and c) estimates of future costs. Evaluators also stated that the issue of using actual student data should have been agreed upon from the onset. Such an agreement might have resolved the problem of storing redundant data on the PRO*FILE System and the administrative grading system.

Development Costs

Analyzing development costs is an important factor in establishing the cost of the effort and also in inferring future costs associated with the continued use of the system. This discussion analyzed the actual computer charges for the PRO*FILE Systems' prototype and loading the data. Four accounts were used to establish the cost of the following categories:

1. Systems work;
2. Storage;
3. Data entry: and
4. Faculty/Student use.

For each account, the cost included various charges for particular subsystems of the university computer facility. These charges included:

1. Time Sharing
    a. MILTEN Line Time
    b. WYLBUR CPU Time

c. WYLBUR Disk Excp

d. ORVYL CPU Time

e. ORVYL File Usage

2. Computer Charges

a. Lines Out

b. Pages Out

It was not the intent of this analysis to determine each category cost, but to give the reader an overall view of the costs associated with the development of the prototype and the loading effort. An overview of the costs are depicted in Table 18.

Detailed analysis of the associated costs revealed that the development and disk storage of the PRO*FILE Systems' prototype was less than $1,000. It was projected at the onset that development costs would be approximately that same figure. As of October, 1983, the cost of data entry was almost equal to the systems development effort. Considering that only 12 of 600 students were entered and 20 of 100 Performance Element Frames, the approximate cost to fully load the data base could reach $8,000 plus $500 per month thereafter for storage. As more users logon to the system, the monthly charge for access to the PRO*FILE System by administrators, faculty and students could rise above $1,500 per month. Based on FIPS PUB 77 (1980)

TABLE 18. Development Costs for the PRO*FILE System

MONTH YEAR	FIXED SYSTEMS ACCOUNT	MONTHLY STORAGE ACCOUNT	CUMULATIVE DATA ENTRY ACCOUNT	CUMULATIVE FACULTY STUDENT ACCOUNT	COMMENT
JUL82		6.24	26.95	17.02	- Summer -
AUG82		9.87	95.36	38.55	! Programming!
SEP82		17.40	305.14	168.34	- Effort -
OCT82		15.75	454.04	207.01	- Initiation -
NOV82		14.70	497.50	236.52	! of !
DEC82		6.30	499.32	248.13	- PRO*FILE -
JAN83		2.85	501.12	253.65	-Modification-
FEB83		5.52	503.04	255.52	! of !
MAR83		11.30	504.72	256.93	! PRO*FILE !
APR83		17.58	506.58	258.33	! Student- !
MAY83		16.92	508.26	259.59	- Record -
JUN83		26.31	511.08	261.71	- Initiation -
JUL83		25.25	570.82	290.41	! of !
AUG83		24.56	830.98	314.36	-Perf/Element-
SEP83		32.04	871.75	474.76	- Loading -
OCT83		44.56	927.18	563.54	-Perf/Element-
TOTAL	$650.00	$277.15	$927.18	$563.54	

criteria for on-line use, the cost for on-line access by administrators, faculty and students would be $2,000 per month or $24,000 per year. Of course, these are only projections.

The administration of the College of Education and the PRO*FILE Task Force may have to justify the benefits versus cost. One recommendation from the author as an alternative to the College of Education paying for the PRO*FILE System would be to have a base charge per student to help defray the cost; another method of cost recovery would be to institute a chargeback to departments. The benefit of developing a chargeback system includes factors that are understandable, controllable, and consistent with the users' environment; users are in a position to minimize their costs (Markle, 1983). Another alternative would be to recapture costs by selling the PRO*FILE service to other institutions.

The cost projections up to this point did not consider the cost of an adequate technical support staff. Depending upon the level of acceptance of the PRO*FILE System, support could range from part-time programming to full-time positions.

Estimates

Data_Base_Size

In the previous chapter, the data base size was estimated to grow to appromixately 37,000,000 bytes or equivalent to 11,500 blocks of data sometimes referred to as pages. The prototype did not verify this estimate and will not until the data base is at least 25% loaded. At that time, a review of the storage would give an indication of the total data base size.

Input/Output_Probabilities

Table 17, on page 141, presented the input/output probabilities for the various segments in the PRO*FILE Systems' prototype. These estimates were based upon a fully loaded data base. This exercise was designed for the analyst who will be working on the PRO*FILE System when it is loaded.

Summary

This chapter presented the findings and results from the evaluation of the PRO*FILE Systems' prototype. It consisted of demonstrations and corresponding interviews as well as a detailed analysis of the development cost and performance considerations. Finally, a discussion on estimating the data base size and input/output probabilities was presented. The next chapter will present the final summary and recommendations.

SUMMARY AND RECOMMENDATIONS

Summary

This research was undertaken to determine how Data Base Management Systems' technology could be used in the improvement and/or assessment of potential teachers in a College of Education. Such technology was potentially useful to administrators and educators for advising and tracking the progress of students. For students, this technology was potentially useful for self-evaluation and career/major decision-making. Most previous studies had investigated the use of educational simulation, focused primarily on drill and practice applications, or the use of an automated page turning capability. This study was designed to investigate the use of the computer and Data Base Management Systems' software as a prescriptive diagnostic processor. Specifically, the study would determine if the computerized PRO*FILE System, based on a Data Base Management System could assist teacher education candidates during their undergraduate years by providing English-like software for individualized instruction on the basic concepts of teacher competencies.

This study was based on the on-going PRO*FILE research effort by the College of Education at Iowa State University and supported by the Director, Research Institute for

Studies in Education (RISE) for technical assistance. The educational research and development cycle was used to evaluate a prototype using Data Base Management Systems' theories and methodologies. This prototype represented a computerized teacher assessment prescriptive diagnostic system. The prototype was loaded with 12 hypothetical students each containing representative data on a typical student and program of study. Following the educational research and development cycle, the initial prototype was evaluated by selected administrators, faculty, and graduate research assistants from Iowa State University's Research Institute for Studies in Education. Modifications to the PRO*FILE System were based upon the preliminary field test evaluation. Interviews with selected evaluators, analysis of statistics on development time, devlopment cost, and performance considerations resulted in:

1. The use of the educational research and development cycle as a model for the computerization of the PRO*FILE System as an educational product was a valid approach. The cycle allowed for review after each step which validated previous design and programming efforts. The prototype was modified at various steps to reflect the true desires of the PRO*FILE Task Force.
2. The data base design model provided by Atre (1981)

was an appropriate design methodology. The level of detail and formalized structure provided a smooth transition from the initial idea, through a rigorous design and finally to a computerized prototype using a Data Base Management System.

3. The Data Base Management System called SPIRES was used as the foundation for the application software. Although SPIRES proved to be adequate, there were shortcomings that could have been eliminated with a more technically advanced Data Base Management System. SPIRES supported only the hierarchical data model which placed restrictions on the data base design. The quality of documentation provided for SPIRES was inadequate. Like most computer vendor's manuals, detailed explanation and examples were missing and in some cases cross-references did not lead to a meaningful discussion of capabilities. A relational Data Base Management System with adequate documentation might have resulted in a better design.
4. The costs associated with designing and implementing the prototype in SPIRES were well within the original estimates. As the PRO*FILE data base continued to grow, the cost of storage

would increase proportionally. For on-line users, the most costly factor was updating.

5. Data Base Management Systems' theories and methodologies did support the information needs of admininstrators, faculty and students in a College of Education. However, finding technical support personnel to maintain a Data Base Management System based application would be a major obstacle with an older Data Base Management Systems such as SPRIES.

6. The sample size used to evaluate the PRO*FILE System was insufficient to make meaningful comparisons to the estimated data base size and performance considerations such as input/output buffers. This limitation could have been resolved by an agreement between the universiy administration and the PRO*FILE Task Force on on the issue of confidentiality guided by the Privacy Act of 1974.

7. The prototyped PRO*FILE System was turned over to the PRO*FILE Task Force for main field testing and the remaining steps in developing a product using the educational research and development cycle. Preliminary modifications and field testing indicated that the PRO*FILE System would be a

viable educational product.

Recommendations for Further Study

Final testing of the PRO*FILE System within the educational research and development cycle should use actual student data. This approach would verify the sizing estimates previously calculated. Researchers would be able to evaluate the internal performance of the data base, such as input/output buffer utilization, and also verify the adequacy of response time.

SPIRES was found not to be a current (state of the art) commercial Data Base Management System. Although SPIRES proved adequate, it did not offer the expanded integrated functionalities that current commercial Data Base Management Systems offer such as graphics, statistical analysis, and a download capability to a micro-computer. It would be most advantageous to prototype the PRO*FILE Systems using a current relational Data Base Management System containing the expanded facilities noted above and compare the design, performance and added capabilities.

A viable Data Base Management System should have a micro-to-mainframe link inherent to the system. It would be beneficial for users to have the ability to download their student record or performance element frames for future reference. This would decrease the dependency on the university's main computer system for users who are reviewing or working with their own dynamic data. Downloaded data could be stored on a student's personal floppy disk.

The College of Education should formulate a Strategic Business Plan for the development, implementation and use of the PRO*FILE System over the next seven years. The design of the final PRO*FILE System as an educational product must consider the information needs of the administration, faculty, and students in the future.

It is imperative that a knowledgeable technical staff be hired and trained to maintain the final system. This of course requires a commitment from the administration for adequate funding of salaries and support training. Without this commitment, the PRO*FILE System will not reach its full potential.

Any replication of this effort is welcome. It must be noted that any comparisons would be questionable unless the same hardware, data base management system and computer environmental mix is identical.

Finally, the PRO*FILE System offers great potential as a educational product that could be used over a nation-wide network or exported (sold) to other Colleges of Education. The PRO*FILE System could be designed for flexibility so other institutions could enter their own particular criteria such as performance elements. The prototype design in this study would have to consider mainframe utilization, telecommunications and a support staff large enough to maintain adequate service to its subscribers. Iowa State University has an opportunity to be the first major univeristy to bring a teacher evaluation prescriptive diagnostic processor on-line for nation-wide use.

BIBLIOGRAPHY

Adams, D. R., Wagner, G. E. & Boyer, T. J. (1983). Computer_information_systems:__An_introduction. Cincinnati, Ohio: South-Western Publishing.

Adams, R. D. (1978). Presentation_and_simulation of_an_illustrative_model_for_the_evaluation_of teacher_education_graduates. Paper presented at the annual meeting, American Association of College Teacher Education, Chicago, Illinois. (ERIC ED 152 774)

Alter, S. L. (1980). Decision_support_systems:__Current practice_and_continuing_challenges. Reading, Mass.: Addison-Wesley.

American_National_Standard_Vocabulary_for_Information Processing. (1970). American National Standards Institute Publication Number ANSI X3.12-1970.

Anderson, S. B. & Ball, S. (1980). The_profession_and practice_of_program_evaluation. San Francisco, CA: Jossey-Bass.

Ash, W. & Sibley, E. H. (1968). TRAMP: An interactive associative processor with deductive capabilities. Proceedings,_ACM,_23rd_National_Conference. New York: ACM.

Atre, S. (1980). Data_base:__Structured_techniques for_design,_performance,_and_management. New York: John Wiley.

Borg, W. R. & Gall, M. D. (1979) Educational_research: An_introduction, 3. New York: Longman.

Bradley, J. (1983). Introduction_to_data_base_management in_business. New York: Holt, Rinehart and Winston.

Brown, K. G. & Droegemuller, L. (1983, July). Microcomputer use in administrative decision support systems. CAUSE/EFFECT, 6(4), 12-18.

Cagan, C. (1973). Data_management_systems. Los Angeles, CA: Melville Publishing.

Cardenas, A. F. (1979). Data_base_management_systems. Boston, Mass.: Allyn and Bacon.

Casteel, J. D. & Gregory, J. W. (1975). _A cluster of technical teaching skills -- Acquisition through microsimulation and evaluaton through microteaching_. Department of Secondary Education and Institute for Development of Human Resources, University of Florida. (ERIC ED 107 645)

Childs, D. L. (1968). Description of a set-theoretic data structure. Proceedings FJCC, American Federation of of Information Processing Societies (AFIPS). New York: AFIPS.

CODASYL, data base task group (DBTG) report. (1971, April). New York: Association for Computing Machinery.

Codd, E. F. (1970, June). A relational model of data shared data banks. _Communications of the ACM_, _13_(6), 377-387.

Confessore, G. J. (1974). _A computer supported simulator for analyzing the relationships between actual and intended teaching behavior_. City University of New York, Columbia University, CUNY, Teachers College. (ERIC ED 112 826)

Date, C. J. (1977). _An introduction to database systems, 2_. Reading, Mass.: Addison-Wesley.

Date, C. J. (1981). _An introduction to database systems, 3_. Reading, Mass.: Addison-Wesley.

Davis, W. S. (1981). _Information processing systems_ (2nd ed.). Reading, Mass.: Addison-Wesley.

DeMarco, T. (1979). _Structured analysis and system specification_. New York: Yourdan Press.

Denemark, G., & Nelli, E. (1968). Quality teacher education: A context for competency assessment. In S. G. Boardman & M. J. Butler (Eds.), _Competency assessment in teaching education_ (pp. 1-17). Lexington, KY: American Association of College Teachers for Education. (ERIC ED 206 570)

Fagin, R. (1977). A new normal form for relational data bases. _ACM Transitions on Database Systems_, _2_, _3_. New York: ACM.

Feldman, J. A. & Rovner, P. D. (1969, August). An algol-based associative language. Communications_ACM._12._8. New York: ACM.

Flynn, R. L. (1974, August). A brief history of data base management. DATAMATION, 20(8), 71-77.

Fry, J. P. & Sibley, E. H. (1976, March). Evaluation of data base management systems. Computer_Surveys, 8(1), 7,42.

Gagne', R. M. & Briggs, L. J. (1974). Principles_of instructional_design. New York: Rinehart & Winston.

General Catalog. (1981'83). Iowa_State_University Bulletin. Ames, Iowa: Iowa State University.

Gessford, J. E. (1980). Modern_information_systems: Designed_for_decision_support. Reading, Mass.: Addison-Wesley.

Graduate Catalog. (1981'83). Iowa_State_University Bulletin. Ames, Iowa: Iowa State University.

Graduate students. (1981'83). Iowa_State_University. Ames, Iowa: Iowa State University.

Grady, D. B. (1981, May). How elected officials can control computer costs. CAUSE/EFFECT, 4(3), 26-28.

Herrick, M. A. (1983). Data_analysis:__Building the_enterprise_data_model_-_the_key_to_data_base design. Cambridge, Mass.: Margann Assoc.

IBM. (1982). The_economic_value_of_rapid_response_time (GE20-0752-0). White Plains, N.Y.: IBM.

Informatics. (1974, June). MARK_IV_systems,_technical systems_description. No. Sm-7406-T76A. Canoga Park, CA.

Information Builders, Inc. (1983). FOCUS_user's manual,_Rel._4.0, 3. New York: Author.

Kauchak, D. & Eggen, P. (1978). The_use_of_written simulations_in_the_measurement_of_teaching competencies. (ERIC ED 151 350)

Knapp, D. & Leben, J. F. (1978). IMS_programming techniques. New York: Van Nostrand Reinhold.

Kniker, C. R. (1981, July). Personal_profile_analysis. Paper presented to the Dean, College of Education, Iowa State University, Ames, Iowa.

Kniker, C. (1982). [A comparison of conventional and computer-assisted techniques to assess three competencies of teacher education students]. Unpublished raw data. Secondary Education, Iowa State University, Ames, Iowa.

Langefors, B. (1977). Information systems theory. Information_Systems, 2, 207-219.

Levein, R. E. & Maron, M. E. (1967). A computer system for inference execution and data retrieval. Communication_ACM, 10, 11. New York: ACM.

Mandell, S. L. (1982). Computers_and_data_processing,_2. Saint Paul, Minn.: West Publishing.

Markle, E. N. (1983, March). MIS chargeback. CAUSE/EFFECT, 6(2), 4-7.

Martin, J. (1976). Principles_of_data-base_management. Englewood Cliffs, New Jersey: Prentice-Hall.

Martin, J. (1977). Computer_data-base_organization,_2. Englewood Cliffs, New Jersey: Prentice-Hall.

Moore, G. E. (1977). Providing_instructional_feedback to_students_in_education_classes. Purdue University. (ERIC ED 173 309)

MRI Systems Corporation. (1975). SYSTEM_2000_reference manual. Austin, Texas: Author.

Murdick, R. G. & Ross, J. E. (1975). Information_systems for_moderan_management, 2. Englewood Cliffs, New Jersey: Prentice-Hall.

Neiheisel, S. R. (1981, July). A comprehensive study of administrative computing at leading institutions of higher education. CAUSE/EFFECT, 4(4), 8-25.

Olle, T. W. (1978). The_CODASYL_approach_to_data_base management. New York: John Wiley & Sons.

Orr, K. T. (1977). Structured systems development. New York: Yourdan Press.

Perron, B. (1977). IDMS concepts and facilities. New York: Cullinane Corporation.

Plourde, P. J. (1981, July). User experiences with data base management systems in higher education. CAUSE/EFFECT, 4(2), 14-25.

Richardson, R. C., Jr., Martens, K. J., Fisk, E. C., Okun, M. A. & Thomas, K. J. (1982). A report on literacy development in community college: Technical Report. National Institute of Education, Washington, DC. (NIE No. 400-78-0061)

Russell, R. E. (1982, November). Administrative information systems in higher education: The 1981 study. CAUSE/EFFECT, 5(6), 24-27.

Sandefur, J. T. (1981). State reactions to competency assessment in teacher education. In S. G. Boardman and M. J. Butler (Ed.), Competency assessment in teacher education: Making it work, (pp. 21). Washington, D.C.: AACTE and ERIC SP 018 202.

Scriven, M. (1967). The methodology of evaluation. Perspectives of Curriculum Evaluation. American Educational Research Association, Monograph Series on Curriculum Evaluation, No. 1. Chicago: Rand McNally.

Shelly, G. B. & Cashman, T. J. (1976). Computer programming RPG II. Fullerton, CA: Anaheim Publishing.

Sherman, K. (1981). Data communications: A users guide. Reston, VA: Reston Publishing.

Sholtys, P. (1983). Systems prototyping with fourth generation tools. CAUSE/EFFECT, 6(6), 8-23.

Sitko, M. C., Semmel, M.I. & Olson, J. (1974). The effectiveness of a computer-assisted teacher training system (CATTS) in generating specific teacher behaviors in a preservice college teaching environment. Center for Innovation in Teaching the Handicapped, Indiana University. (ERIC ED 111 436)

Smith, G. R. & Shallwani, I. (1978). Computer_simulation of_the_supply_and_demand_for_teachers_in_Michigan public_schools. Paper presented to the American Educational Research Association, Toronto, Canada. (ERIC ED 160 059)

Software AG of North America (SAG). (1975). ADABAS_users manual. Reston, VA: Author.

Software AG of North America (SAG). (1983). ADABAS_users manual. Reston, VA: Author.

Sperry UNIVAC. (1978). DMS-1100_reference_manual. St. Paul, Minn.: Author.

Stanford University. (1982). A_guide_to_data_base development_--_a_SPRIES_primer. Standford University, Center for Information Technology, Document No. 69.

Steingraber, J. K. & Kunkel, D. F. (1982, July). An integrated university online/data base system: A reality. CAUSE/EFFECT, 5(3), 8-24.

Stern, N. (1975). Flowcharting:__A_tool_for understanding_computer_logic. New York: John Wiley & Sons.

Teacher education admissions policies and procedures. (1982). College of Education, Iowa State University, Ames, Iowa.

Tsichritzis, D. C. & Lochovsky, F. H. (1982). Data_models. Englewood Cliffs, New Jersey: Prentice-Hall.

User's Brochure. (1983, Spring). Iowa State University Computation Center, Ames, Iowa.

U.S. Department of Commerce/National Bureau of Standards. (1980). Guideline_for_planning_and_management_of database_applications. (FIPS PUB 77). Washington, DC: U.S. Government Printing Office.

U.S. Department of Commerce/National Bureau of Standards. (1983). Guideline:__A_framework_for_the_evaluation and_comparison_of_software_development_tools. (FIPS PUB 99). Washington, DC: U.S. Government Printing Office.

U.S. Marine Corps. (1983). M3S_prototyping_standard. Albany, GA: U.S. Government.

Wetherbe, J. C. (1979). _Systems analysis for computer-based information systems_. St. Paul, Minn.: West Publishing.

Yourdan, E. & Constantine, L. L. (1979). _Structured design: Fundamentals of a discipline of computer and systems design_. Englewood Cliffs, New Jersey: Prentice-Hall.

Zuckerman, R. A. (1979). Simulation helps preservice students acquire pragmatic skills. _Journal of Teacher Education_, 30(4), 14-16.

ACKNOWLEDGEMENTS

The writer would like to express appreciation to all members of his committee: Dr. James Ratcliff, chair of the committee, Dr. Richard Warren, Dr. J. Stanley Ahmann, Dr. Rex Thomas, and Dr. Dale Grosvenor, for their assistance in developing the research design, for their suggestions with the evaluation, for their extensive help in editing, and for their guidance throughout the entire study.

Special acknowledgement is given to Dr. Charles Kniker and Dr. Joan Breiter who co-chaired the PRO*FILE Task Force, all the ISU personnel who participated in the on-going PRO*FILE project for their efforts, and to all my college professors, to my colleagues in the RISE Office, to all my friends, for their caring, encouragement, and support.

Special acknowledgement is also given to my wife and children for their love and patience. Because of this, I was able to accomplish my goals at Iowa State University.

APPENDIX A -- GLOSSARY OF TERMS

attribute. A property or characteristic of one or more entities, for example, color, weight, sex.

batch. A job that is grouped with other jobs as input to a computer system. Constrast with interactive.

COBOL. Common business-oriented language. An English-like programming language designed for business data processing applications.

CODASYL. Conference on Data Systems Languages.

computer. A functional unit that can perform substantial computation, including numerous arithmetic operations or logic operations, without intervention by a human operator during a run.

computer architecture. The specification of the relationships between the parts of a computer system.

computer system. A functional unit, consisting of one or more computers and associated software, that uses common storage for all or part of a program and also for all or part of the data necessary for execution of the program; executes user-written or user-designed programs; performs user-designated data manipulation, including arithmetic operations and logic operations; and that can execute programs that modify themselves during their execution.

data base management system (DBMS). A software system facilitating the creation and maintenance of a data base and the execution of computer programs using the data base.

data structure. The syntactic structure of symbolic expressions and their storage allocation characteristics.

entity. An object or an event about which information is stored in a data base, for example, a person, or a train departure time.

flowchart. A graphical representation of the definition, analysis, or solution of a problem in which symbols are used to represent such thing as operations, data, flows, and equipment.

FORTRAN(formula translation). A programming language primarily used to express computer porgrams by arithmetic formulas. A programming language primarily designed for applications involving numeric computations.

hardware. Physical equipment used in data processing, as opposed to programs, procedures, rules, and associated documentation. Constrast with software.

input/output(I/O). Pertaining to a device or to a channel that may be involved in an input process, and, at a different time, in an output process.

integrated data base. A data base which has been consolidated to eliminate redundant data.

interactive. Pertaining to an application in which each entry calls forth a response from a system or program, as in an inquiry system or an airline reservation system. An interactive system may also be conversational, implying a continuous dialog between the user and the system.

inquiry transaction. A transaction that does not update a data base.

key. One or more characters, within a set of data that contains information about the set including its identification.

microcomputer. A computer system whose processing unit is a microprocessor. A basic microcomputer includes a microprocessor, storage, and an input/output facility, which may or may not be on one chip.

modem(modulator-demodulator). A device that modulates and demodulates signals transmitted over data communication facilities.

MVS. Multiple virtual storage.

on-line. Pertaining to a user's ability to interact with a computer. The term "on-line" is also used to describe a user's access to a computer via a terminal.

operating system. Software that controls the execution of programs; an operating system may provide services such as resource allocation, scheduling, input/output control, and data management.

plotter. An output unit that presents data in the form of a two-dimensional graphic representation.

processor. In a computer, a functional unit that interprets and executes instructions.

program. A set of actions or instructions that a machine is capable of interpreting and executing.

program library. A collection of available computer programs and routines.

query. The process by which a master station asks a slave station to identify itself and to give its status. In interactive systems, an operation at a terminal that elicits a response from the system.

RPG(report program generator). A commercially oriented programming language specifically designed for writing application programs that meet common business data processing requirements.

software. Programs, procedures, rules, and any associated documentation pertaining to the operation of a computer system. Constrast with hardware.

SPIRES. Standford Public Information REtrieval System.

structure. A hierarchical set of names that refers to an aggregate of data items that may have different attributes.

system. In data processing, a collection of men, machines, and methods organized to accomplish a set of specific functions.

telecommunication. Communication over a distance, as by telegraph or telephone.

telecommunication network. The assembly of fuctional units that establishes data circuits between data terminal equipments.

terminal. A point in a system or network at which data can either enter or leave. A device, usually equipped with a keyboard and a display device, capable of sending and receiving information over a link.

text processing. In word processing, information for human comprehension that is intended for presentation in a two-dimensional form, for example, printed on paper or displayed on a screen. Text consists of symbols, phrases, or sentences in natural or artificial languages, pictures, diagrams, and tables.

time sharing. Pertaining to the interleaved use of time on a computer system that enables two or more users to execute computer programs concurrently.

utilities. A computer program in general support of the processes of a computer; for instance, a diagnostic program, a trace program, a sort program.

VSAM. Virtual storage access method.

APPENDIX B -- FUNCTIONAL SPECIFICATIONS FOR THE PRO*FILE SYSTEM

Functional_Specifications_for_the_PRO*FILE_System

---

1. Design a program which permits administrative staff, advisors, and teacher education faculty to view, or add to, the following items in each student's "folder":

    a. Name
    b. Student ID
    c. High school rank
    d. GPA (college)
    e. Background information (birthdate, hometown)
    f. ACT or SAT scores (composit and subscale)
    g. Record of performance in a teaching/learning experience
    h. Record of contacts with adviser
    i. Requests by adviser to see student
    j. Curriculum sheet
    k. Adds, drops, transfer, incompletes
    l. other letters (of recommendation)
    m. 204--Report of Writing sample
    n. Admission form (to enter ISU)
    o. Admission to T.E. Report
    p. Record of Interim Interview
    q. Exit Interview Report
    r. Initial Assessment Battery Report
    s. Final Assessment Battery Report
    t. Progress report on Performance Elements
    u. Transfer credit and evaluation
    v. English writing sample
    w. Math placement test scores
    x. Copy of degree program

2. Design a program, with appropriate security checks, which permits students access to the above information.

3. Design a program which permits students to try out elements of the Initial Assessment Battery.

    a. Personal Profile
    b. Professional Strengths and Needs
    c. List of Teaching/Learning Experiences
    d. Knowledge of Education (Test-Sample Items)
    e. Reading List
    f. Philosophy of Education
    g. Basic Skills

APPENDIX C -- PHYSICAL DATA BASE DEFINITION

```
FILE = E1.MAR.PROFILE/AUTHOR CHAD GRABOW, RISE, 4-9413
GOAL ENTRY
KEY ID/SLOT
REQUIRED
ELEM NAME,NAM,N/NAME/LEN 23/SINGLE/INDEX/PRIV-TAG 1
OPTIONAL
ELEM SOCIAL-SECTY-NUM,SOC-SEC-NUM,SSNUM/INT/SINGLE/INDEX/+
     PRIV-TAG 1
ELEM SEX,SE,S/TEXT/LEN 1/SINGLE/PRIV-TAG 2/MSG THE SEX+
     MUST BE EITHER A, M, F OR U (UNKNOWN),+
     PLEASE RE-ENTER.
ELEM CURRICULUM,CURR/TEXT/LEN 5/SINGLE/PRIV-TAG 1
ELEM COLLEGE,COLL/TEXT/LEN 1/SINGLE/PRIV-TAG 2/MSG CHECK+
     FOR THE PROPER 1 DIGET COLLEGE CODE, PLEASE RE-ENTER.
ELEM YEAR,YR/INT/SINGLE/PRIV-TAG 2
ELEM CURRENT-GPA,CURR-GPA/DEC 4.2/SINGLE/PRIV-TAG 2
ELEM SEMESTER-ADMITED,SEM-ADMIT,SAD/TEXT/LEN 4/SINGLE/+
     PRIV-TAG 2
ELEM TYPE,T/TEXT/LEN 1/SINGLE/PRIV-TAG 2
ELEM ENTRANCE-GPA,ENTR-GPA,EGPA/DEC 4.2/SINGLE/PRIV-TAG 2
ELEM TRANSFER-CREDIT,TRANS-CREDIT,TCR/DEC 3.1/SINGLE/+
     PRIV-TAG 2
ELEM SEM-STD-TAUGHT,SST/LEN 4/SINGLE/PRIV-TAG 1
ELEM TEACHING-LEVEL,TLV/TEXT/LEN 4/SINGLE/INDEX/PRIV-TAG 2
ELEM MAJOR,MAJ/TEXT/LEN 5/SINGLE/INDEX/PRIV-TAG 2
ELEM MINOR,MIN/TEXT/LEN 5/SINGLE/PRIV-TAG 2
ELEM ACT/INT/SINGLE/PRIV-TAG 2
ELEM ACT-ENGLISH,ACT-ENGL,ACTE/INT/SINGLE/PRIV-TAG 2
ELEM ACT-MATH,ACTM/INT/SINGLE/PRIV-TAG 2
ELEM ACT-SOC-STUDIES,ACTSOCSTDY,ACTSS/INT/SINGLE/+
     PRIV-TAG 2
ELEM ACT-NATIONAL-SCI,ACTNATSCI,ACTNS/INT/SINGLE/+
     PRIV-TAG 2/RANGE 0,35/MSG THE ACT SCORE MUST BE+
     BETWEEN 0 AND 35, PLEASE RE-ENTER.
ELEM SAT-VERBAL,SAT-VERB,SATV/INT/SINGLE/PRIV-TAG 2
ELEM SAT-MATH,SATM/INT/SINGLE/PRIV-TAG 2
ELEM IN-BTRY/STRUCTURE/PRIV-TAG 2
     ELEM IN-TEST-ID,INTID/INT/SINGLE/PRIV-TAG 2
     ELEM IN-TEST-TIME,INDT/TEXT/LEN 10/SINGLE/+
          PRIV-TAG 2
     ELEM IN-NUM-QUEST,INNQ/INT/SINGLE/PRIV-TAG 2
     ELEM IN-NUM-CORRECT,INNC/INT/SINGLE/PRIV-TAG 2
     ELEM IN-IND-RESPES,ININDRES,IIR/TEXT/LEN 55/+
          SINGLE/PRIV-TAG 2
END
ELEM INTERVIEW/STRUCTURE/PRIV-TAG 2
     ELEM INTERVIEW-TYPE,INT-TYPE,IT/TEXT/LEN 25/+
          SINGLE/PRIV-TAG 2
     ELEM INTERVIEW-DATE,INT-DATE/DATE/SINGLE/+
          PRIV-TAG 2
          ELEM INTERVIEWER/NAME/LEN 23/SINGLE/PRIV-TAG 2
          ELEM DIALOGUE/TEXT/LEN 72/PRIV-TAG 2
     END
     ELEM EXPERIENCES/STRUCTURE
          ELEM EXPERIENCE-TYPE,EXPT/TEXT/LEN 25/SINGLE
          ELEM EXPERIENCE-DATE/DATE/SINGLE
          ELEM COMPOSITION/TEXT/LEN 72
     END
     ELEM SEMESTER,SEM/STRUCTURE/PRIV-TAG 2
          ELEM SEMESTER-YEAR,SEM-YR/TEXT/LEN 5/SINGLE/+
               PRIV-TAG 2
          ELEM COURSE,CRS,CRS-DATA/STRUCTURE/PRIV-TAG 2
               ELEM DEPARTMENT,DEPT/TEXT/LEN 5/SINGLE/+
                    PRIV-TAG 2
               ELEM COURSE-NUMBER,CRS-NUM,NUM/TEXT/LEN 4/+
                    SINGLE/PRIV-TAG 2
               ELEM CREDIT,CR/INT/SINGLE/PRIV-TAG 2
               ELEM CORUSE-NAME,CRS-NAME,CRSNAME/TEXT/+
                    LEN 19/SINGLE/PRIV-TAG 2
               ELEM GRADE,GD/TEXT/LEN 2/SINGLE/PRIV-TAG 2
               ELEM QUALITY-POINTS,QTY-PTS,Q-PTS/DEC 4.2/+
                    SINGLE/PRIV-TAG 2
          END
     END
     ELEM PERFORMANCE-ELMT/STRUCTURE/PRIV-TAG 2
          ELEM PERF-TEST-ID,PERFTID/INT/SINGLE/PRIV-TAG 2
          ELEM PERF-DATE-TIME,PERFDT/TEXT/LEN 10/SINGLE/+
               PRIV-TAG 2
          ELEM PERF-NUM-QUESTS,PERFNO/INT/SINGLE/PRIV-TAG 2
          ELEM PERF-NUM-CORRECT,PERFNC/INT/SINGLE/PRIV-TAG 2
          ELEM PERF-IND-RESPES,PERFINDRESS,PIR/TEXT/LEN 55/+
               SINGLE/PRIV-TAG 2
     END
     ELEM OUT-BTRY/STRUCTURE/PRIV-TAG 2
          ELEM OUT-TEST-ID,OUTTID/INT/SINGLE/PRIV-TAG 2
          ELEM OUT-DATE-TIME,OUTDT/TEXT/LEN 10/SINGLE/+
               PRIV-TAG 2
          ELEM OUT-NUM-QUESTS,OUTNQ/INT/SINGLE/PRIV-TAG 2
          ELEM OUT-NUM-CORRECT,OUTNC/INT/SINGLE/PIRV-TAG 2
          ELEM OUT-IND-RESPES,OUTINDRES,OIR/TEXT/LEN 55/+
               SINGLE/PRIV-TAG 2
     END
     SUBFILE STUDENT.MASTER/ACCOUNTS = E1.MAR
     SUBFILE STUDENT.INPUT/ACCOUNTS = E1.CLG
     SUBFILE STUDENT.BROWSE/ACCOUNTS = E1.MAY/NOS 1/NOU 1,2/+
             CON 1
     END
```

```
FILE = E1.MAR.PEM.FILEDEF
COMMENTS = THIS IS A MODIFIED VERSION OF JOHN P. HAUCK'S
COMMENTS = FILEDEF FOR THE ISU INFORMATION SYSTEM.  THE
COMMENTS = ORIGINAL FILEDEF IS COPYRIGHT BY IOWA STATE
COMMENTS = UNIVERSITY.  THEREFORE, THE BELOW FILEDEF IS
COMMENTS = AN ABBREVIATED FORM OF THE DEFINITION.
RECORD-NAME = AREAS;
  REQUIRED;
     KEY = AREA;
  OPTIONAL;
     ELEM = STORED-STR;
     ELEM = POINTER;
     ELEM = MASTER;
  STRUCTURE;
     REQUIRED;
        KEY = STORED;
        ELEM = SHORT.NAME;
        ELEM = LONG.NAME;
RECORD-NAME = AUTH;
     KEY = ACCOUNT;
  REQUIRED;
     ELEM = AUTHORIZATION;
RECORD-NAME = REC01;
     ELEM = ACCOUNT
     ELEM = DATE.ADDED;
     ELEM = TIME.ADDED;
     ELEM = DATE.UPDATED;
     ELEM = TIME.UPDATED;
     ELEM = REVIEW.DATA;
  REQUIRED = KEY;
     ELEM = AREA;
  OPTIONAL;
     ELEM = TITLE;
     ELEM = TEXT;
     ELEM = MENU;
     ELEM = YESNO;
     ELEM = DONE.BRANCH;
     ELEM = SUBJECT;
  VIRTUAL;
     ELEM = MAJOR.AREA;
     ELEM = SUB.AREA;
     ELEM = LONG.MAJOR.AREA;
     ELEM = REVIEW;
  STRUCTURE = MENU;
     REQUIRED;
        KEY = ITEM.NUMBER;
        ELEM = ITEM.TEXT;
        ELEM = ITEM.BRANCH;
  STRUCTURE = YESNO;
     REQUIRED;
```

```
          ELEM = YESNO.TEXT;
          ELEM = YES.BRANCH;
          ELEM = NO.BRANCH;
RECORD-NAME = ZIN02;
   REQUIRED;
       KEY = SUBJECT;
   OPTIONAL;
       ELEM = PTR-STRUCT;
   STRUCTURE - PTR-STRUCT;
          KEY = POINTER;
RECORD-NAME = ZIN03;
       KEY = DATE.ADDED;
       OPTIONAL;
          ELEM = PTR-STRUCT;
   STRUCTURE = PTR-STRUCT;
          KEY = POINTER;
RECORD-NAME = ZIN04;
       KEY = ACCOUNT;
       OPTIONAL;
          ELEM = PTR-STRUCT;
   STRUCTURE = PTR-STRUCT;
          KEY = POINTER;
RECORD-NAME = ZIN05;
       KEY = BRANCH;
       OPTIONAL;
          ELEM = PTR-STRUCT;
   STRUCTURE = PTR-STRUCT;
          KEY = POINTER;
RECORD-NAME = ZIN06;
       KEY = AREA;
       OPTIONAL;
          ELEM = PTR-STRUCT;
   STRUCTURE = PTR-STRUCT;
          KEY = POINTER;
RECORD-NAME = ZIN07;
       KEY = MAJOR.AREA;
       OPTIONAL;
          ELEM = PTR-STRUCT;
   STRUCTURE = PTR-STRUCT;
          KEY = POINTER;
RECORD-NAME = ZIN08;
       KEY = SUB.AREA;
       OPTIONAL;
          ELEM = PTR-STRUCT;
   STRUCTURE = PTR-STRUCT;
          KEY = POINTER;
RECORD-NAME = ZIN09;
       KEY = DATE.UPDATED;
       OPTIONAL;
          ELEM = PTR-STRUCT;
```

```
  STRUCTURE = PTR-STRUCT;
         KEY = POINTER;
RECORD-NAME = ZIN10;
     KEY = REVIEW.DATE;
     OPTIONAL;
         ELEM = PTR-STRUCT;
  STRUCTURE = PTR-STRUCT;
         KEY = POINTER;
GOALREC-NAME = AREAS;
  PTR-ELEM = STORED;
  INDEX-NAME = AREAS;
  SEARCHTERMS = XX;
     QUAL-ELEM = LONG.NAME;
         SEARCHTERMS = LONG.NAME;
     QUAL-ELEM = SHORT.NAME;
         SEARHTERMS = XXXX;
GOALREC-NAME = REC01;
  PTR-ELEM = POINTER;
  INDEX-NAME = ZIN04;
     SEARCHTERMS = ACCOUNT.ACCT;
INDEX-NAME = ZIN06;
  SEARCHTERMS = AREA;
  PTR-GROUP = PTR-STRUCT;
INDEX-NAME = ZIN07;
  SEARCHTERMS = MAJOR.AREA, MA;
  PTR-GROUP = PTR-STRUCT;
INDEX-NAME = ZIN08;
  SEARCHTERMS = SUB.AREA. SA;
INDEX-NAME = ZIN05;
  SEARCHTERMS = BRANCH, BR;
```

APPENDIX D -- MENU DRIVER

The MENU.DRIVER was originally written by the staff of the Iowa State University (ISU) Computation Center as part of the ISU Student/Faculty Information System. It is the property of ISU, all rights reserved. Permission to copy, modify and use the MENU.DRIVER was granted to RISE in May, 1983.

The information stored in the PERF System is provided to inquirers via a series of "information frames". Each frame is essentially a piece of information and one or more choices for the inquirer of where to go after he/she is finished with the frame. The choices link the frames together and allow the inquirer to move through the system from one frame to another. The MENU.DRIVER is a command program that drives the user friendly menus. A discussion of the process is provided in Appendix H.

APPENDIX E -- MENU INSTRUCTIONS

```
*
* "UES" IS AN INVALID RESPONSE HERE
*
*
*
* Do you need instructions about how to use the system?
: Please enter YES or NO -  yes


                                        (INSTRUCTION/1)

INSTRUCTIONS FOR USING THE ISU PERFORMANCE ELEMENT
INFORMATION SYSTEM



The ISU Performance Element Information System porvides
information about various aspects of performance elements
via a series of information "frames" linked together by
multiple choice "menus" and yes/no type questions.  The
menus and questions allow you to jump from one frame to
another depending on the specific subjects about which you
wish information.

The system begins by displaying its first frame, called the
TOP, and asking you to choose the first subject from the
menu provided.  You respond to the system with your choice
of subject by typing the response then hitting the RETURN
key.  The RETURN key causes your response to be sent to the
system; you have not responded until this key is hit.
(Sometimes the RETURN key is labelled as ENTER or
CARRIAGE-RETURN; in any event, use whatever your terminal
calls the "RETURN" key to terminate and send your
response).

:When done reading, type DONE.  (Type STOP to leave the
System) - stop
END OF SESSION
```

APPENDIX F -- DISPLAY GENERATION

The DISPLAY.GENERATION was originally written by the staff of the Iowa State University (ISU) Computation Center as part of the ISU Student/Faculty Information System. It is the property of ISU, all rights reserved. Permission to copy, modify and use the DISPLAY.GENERATION was granted to RISE in May, 1983.

The MENU.DRIVER, as discussed in Appendix D, uses this program to display the information the user has requested on a terminal screen. The program formats the screen automatically; the process is transparent to the user. Samples of the formatted terminal screens are presented in Appendix H.

APPENDIX G -- SYSTEM GENERATION

CREATING A NEW DATA BASE ON SPIRES

ON AS/6 WYLBUR

```
LOGON
USE <File Name>                        { Name of file that your data
                                         base defn is stored on.
```

IN SPIRES

```
CALL SPIRES
ENTER FILE DEFINER
INPUT ACTIVE                           { SYNTAX Check on Data Base Defn
GENERATE CLE
RETURN
SELECT    FILEDEF
ADD
CALL SPICOMP
COMPILE   <SPIRES FILE NAME>           { Creates
           (ie Perf. Elmts.)             Object
                                         Data Base Defn
EXIT
```

```
LOGOFF
```

* CAUTIONS ------------

1. The account you logged on with will be the account the data base files will be stored on.

2. While in the INPUT ACTIVE Mode, if an error occurs, do an EXIT, edit the Data Base Defn and start process over.

3. Once the COMPILE Mode is complete, use the ATTACH<SUBFILE>NAME vice SELECT until overnight processing is complete.

ORVYL Files Created by SPICOMP

1.	filename. MSTR	contains an encoded version of the file definition. (Record names, element names, occurrence and length attributes of elements, etc.)
2.	filename. DEFQ	contains records that are added to the file during SPIRES sessions. (Deferred querre)
3.	filename. RES	contains records that, for various reasons, are not stored directly in the goal record data set(s) or indexes.
4.	filename.record.name	contains one or more goal record data sets and index record data sets as described by the file definition.
5.	filename.SYSTEM.CHKPT	In case of a system crash, the contents of the checkpoint file are written to SYSTEM.CHKPT.
6.	filename.SYSTEM.LOG (History of updates)	Used by SPIRES consultants to determine why the data was not processed.

MAKING CHANGES TO A DATA BASE DEFN

```
SET UPLOW

SET LENGTH  235

CALL SPIRES

SELECT FILEDEF

TRA      E1.ADS.CARD3             <FILE NAME>
        (make changes)

UPDATE  E1.ADS.CARD3

CALL  SPICOMP

RECOMPILE   E1.ADS.CARD3


CALL SPIRES   ⟶  Check Results


Select FileDef
Show Search Terms
Find Account Subfile Resources
```

DESTROY A DATA BASE

```
Call SPICOMP
ZAP FILE  E1.ADS.CARD3
Call SPIRES
SELECT FILEDEF
ZAP FILE  E1.ADS.CARD3
Remove E1.RDW. <Subfile Name>
```

REMOVE SUBFILE NAMES

```
CALL SPIRES

SHOW SUBFILES                          RECORD.83
                                       GRANTS.83

SELECT FILEDEF

FIND SUBFILE    RECORD.83
     3 Files

TYPE
     E1.RDW.RISEGTS
     E1.RDW.RGRANTS          Each had a Subfile
     E1.RDW.GRANTS              names Records.83

REMOVE E1.RDW.RISEGTS

REMOVE E1.RDW.RGRANTS

FIND SUBFILE RECORD.83
     3 Files                will not show one file
                            until overnight processing

TYPE
     E1.RDW.GRANTS
```

UNLOADING and UPLOADING

UNLOADING:

    Select Subfile Name

    For Tree

    In Active Clear Display All

Change "****" 1 to "ADD:" in all nolist

Save dsname on volume

UPLOADING:

    Call Spibild

    Batch <Subfile Name>

{ Note the builds Indexes for immediate processing

Also Check Scan ⟶ Scan also puts in the active file

* If you omit call SPIBILD

Then the data will be entered in the overnight processing mode

CREATING AN INPUT FORMAT

1. Creat Input Format in WYLBUR File

2. Adding a New Input Format

```
Call SPIRES
Select Formats
USE  File Name of Input Format
ADD
CALL SPICOMP
FORMAT  <FORMAT ID>  <------ ie, ORV.E1.ADS.CARD3IN
                                 PERFELMTS.ACTYOUT
```

3. Updating an Input Format

```
Select Formats
TRA E1.WAR PERFELMTS.PEOPLEOUT
TRA    *   FORMAT ID            ie., *CARD3IN
      (make changes)
UPDATE      * <Format ID>
CALL SPICOMP
REFORMAT    * <Format ID>
```

4. Destroying an Input Format

```
Select Formats
ZAP format   Key     *  FORMAT ID    Source
                    ie., *CARD3IN
```

ADDING RECORDS USING INPUT FORMATS

1. ON LINE ADD (Tables built in demand processing -- EXPENSIVE$)

Call SPIRES	
Select <Subfile Name>	ie., Select CARD383
Show Formats	
Call SPIBILD	
Use <Data File Name>	
Set Format <Format ID Name>	ie., CARD3IN
Batch <Subfile Name>	ie., CARD383

2. BATCH ADD (Overnight Processing)

Call SPIRES	
Select <Subfile Name>	ie., select CARD383
Show formats	
Use <Data File Name>	
Set Format <Format ID Name>	ie., CARD3IN
Batch <Subfile Name>	ie., CARD383
OR Batch <DSN> on <Volume>	
Show Batch Request	(to get Batch Number)
Dequeue Batch Request <Num>	(Purge JOB)

$\boxed{\text{BUILD ENTRY COMMANDS}}$ (In Spires)

```
  Call Spires                                   Perf Elmt System
                                                       |
? Select Entry Commands                                ↓
                                                *△ E1.MAR
? Collect Clear                                 Get MENU.DRIVER
      1.   ? *△E1.MAR                           XEQ
      2.   ? SET△XEQ                            Return
      3.   ? ..MENU.DRIVER
      4.   ? Return
      5.   ? (atten)

? ADD
```

$\boxed{\text{UPDATE ENTRY COMMANDS}}$

```
Select Entry Commands

TRA *△ E1.MAR
        (update)

UPDATE
```

$\boxed{\text{BUILD EXEC}} \longrightarrow$ (WYLBUR ONLY)

```
Set Uplow                          M.U3326.START IT

Set Length 235

Call Spires

CLE EXE
______________________
<SAVE STARTIT>
______________________
EXEC FROM STARTIT
```

BUILD ENTRY COMMANDS

```
     Call SPIRES

?    Select Entry Commands

?    Collect Clear                                       *E1.CLG

          1.   ? *△E1.MAR  ! set NOECHO                  *E1.MAY
          2.   ? Get MENU.DRIVER
          3.   ?XEQ
          4.   ?  RETURN
          5.   ?  (ATTEN)

?    Add
```

UPDATE ENTRY COMMANDS

```
          Select Entry Commands  OR  Remove Entry

               TRA   *△E1.MAR              Select Entry Commands
                     <changes>             Remove *△E1.MAR

          UPDATE
```

BUILD EXEC FILE IN WYLBUR

```
     Col

          1.   ?  Set Uplow
          2.   ?  Set Length 235
          3.   ?  Call Spires
          4.   ?  Cle EXE                          M.U3326.STARTIT
          5.   ?  (atten)

     Save START

     EXEC from STARTIT
```

```
Please enter A,B,C,D, or M
ENTER SYSTEM ID: M
VAX A WILL BE DOWN FROM 2-3PM TODAY FOR HARDWARE SYSTEM WORK.


IOWA STATE COMPUTATION CENTER  AC003-03C  14:09:56 07/12/83

USER? E1.MAY
PASSWORD?
ACCOUNT I5197?
LAST LOGOFF AT 16:08 07/11
NO LAST PASSWORD CHANGE TIME

-WELCOME TO SPIRES-3 ... IF IN TROUBLE, TRY 'HELP'
* Which Data Base Do You Desire To Use?
*
* 1.  PERFORMANCE ELEMENTS
* 2.  PRO*FILE STUDENT RECORDS
* 3.  LOGOFF SYSTEM
*
:Please enter 1, 2, or 3 ->  2
* You are now in the STUDENT DATA BASE, enter your SPIRES commands,
* when done EXIT the Data Base and LOGOFF the terminal.
->
```

1. LOGON TO SPIRES

```
LOGON
 ACCOUNT NUMBER
 PW
CALL SPIRES
```

2. LOGOFF SPIRES

```
EXIT
LOGOFF
```

3. GET ACCESS TO A SPIRES DATA BASE

```
CALL SPIRES
SELECT   SUBFILE NAME
```

4. COMMANDS FOR GENERAL INFORMATION

```
SHOW INDEXES
SHOW ELEMENTS
SHOW FILES
SHOW SUBFILES
BROWSE  Index Name
SHOW SUBFILE SIZE
```

```
9.     SEARCHING

          FIND  <Index Name>     {Unique Value}

                  AND  ⎫
                  OR   ⎬ <INDEX Name>  {Unique Value}
                  ALSO ⎭

          FIND     <Index >      < Relational >     {Unique }
                    Name            Operator         Value

                  a)  Relational Operators

                        Equality        = or to   From ... To

                        Range           Before, After, Between ... AND

                        Inequality      ¬=,  >,  >=,  <,  <=

                        Content         Prefix, Suffix, Word,
                                        String, Having, MASK, WITH

                  b)  Logical Operators

                        AND       OR           AND NOT

                  c)  Compound Searching

                        FIND  {A}    OR    {B}

                        (   )  Overrides Natural Process

10.    EXAMINE RECORD AFTER SEARCH

          TYPE

          TYPE   <Element>

          TYPE   <Element List>

11.    SORTING

          SEQUENCE  <Element>   <Element> (A)
                                          (D)

                         {Defaults to Ascending Order}
```

12. TABULAR OUTPUT

```
FIND DEPT PRO. STUDIES  ⟶ qualifies records
SEQUENCE NAME

DEFINE TABLE NAME       <Element list>
TYPE

ENDTABLE
```

System Functions ---

.MIN

.MAX

.SUM

.COUNT

.AUG

.STD

i.e. DEFINE TABLE UCOST .AVG UCOST

TABLE OF CONTENTS

UPDATING

I. ADDING NEW RECORDS

A. ADDING COMPLETE RECORDS ON LINE

```
set format $prompt
ADD
ADD
```

B. ADDING PARTIAL RECORDS ON LINE

```
Set format $prompt   <Element List, Structure>
ADD
ADD
```

C. RELOADING A DATA BASE ON LINE

See tab DB Creation/Changes
Refer to Uploading and Downloading

D. ADDING USING AN INPUT FORMAT (Card, Tape, Disk)

See tab INPUT format

E. ADD PROCESSING

Once you ADD your records they will automatically be submitted to a quere for overnight batch processing. The index values and data will be processing overnight.

However, if you need to use the data and indexes immediately, enter the following command:

```
CALL SPIBILD
PROCESS <Perf. Elmt>     Expensive $$$$
BATCH <Subfile Name>
```

Also refer to Tab DB Creation/Changes
SEE UPLOADING

II. REMOVING RECORDS

Remove <Slot Number> OR <Key Value>

III. UPDATING RECORDS

A. ADDING/REMOVING/MODIFYING ELEMENTS

(1) Transfer <Key Value> / Slot Number

- update elements using WYLBUR
- update (edit) commands ie MOD_, delete, insert, collect.

UPDATE

– OR – (2) Set FORMAT $PROMPT <Element List, Structure> optional

Merge <Slot Number>

– OR – (3) Transfer Key Value / Slot Number

- Update elements, add, delete using SYLBUR commands

MERGE using <line$_N$/line$_N$> <Key Value>

*CAUTION --- be sure to indent, place your elements in proper sequence and put a semi colon (;) at the end.

✳ Refer to SPIRES Searching & Updating -- Doc. No. 40, pg. 143

B. <u>ADDING/REMOVING/MODIFYING OCCURANCES (Structures)</u>

(1) <u>Adding occurances</u>

```
set format $prompt   <structure>

Merge <Slot number>

        if no  previous occurances, enter data

  ELSE--on first occurance enter

          /AS  { this tells SPIRES to ADD
               { a new occurance bypassing
               { all others
        then follow prompt and enter data
```

(2) <u>Removing Occurances</u>

```
Collect Cle

        1.  EXPERIENCES (-3);   { "Neg" removes
        2.  EXPERIENCES (-4);   { occ #4
Merge  <Slot No>
```

(3) <u>Modifying Occurances</u>

```
Set format $prompt(n)  <-------  occ you want
                                 to modify
Merge  < Slot Number>  <-
                          \
        update as prompted  \    if unknown you'll
                                 have to page thru
                                 the occurances
```

```
Call SPIRES

Select Students

Set Format $Prompt (Brief) Phil-of-Educ

Display 1

Generate Format (Display) Philout
      [PE RELMTS.PEOPLEOUT]
      [E1.WAR.PRO.FILE.philout]

Select Formats

Add

Call SPICOMP

Format Pro.file.Philout
```

USING REPORT WRITER

{Prepare your report on an 80 or 132 column output form FIRST. You will be prompted for column and variable info.

1. Generating a report:

Logon

Call SPIRES

Define Report (TERSE)

(Follow Instruction)

Generate Report <Report Name>

2. Executing a Report Writer

Generate Report <Report Name> {May select and sequence before.

3. Modification of Report Write

Review Report <Report Name>

(Display, Move, Delete, Add, Insert, Change, END)

4. Destroying Reports

Select TRG.SPI.Reports

Remove <Report Name>

(Full Name ie., E1.MAR.PROFILE.SUM)

Show Names

5. Reference: Spires Terminal Report Write, Report, Oct. 1, 1980

BACKUP & RECOVERY

There are three basic ways to back-up and recover a SPIRES data base:

1. Everynight the computer center backups all ORV Files. This backup is kept for 7 days (approx). It does not always work and there is no guarantee that a backup will be created.

)isk to Tape]

   You must notify the Computer Center immediately if you need a backup.

2. Thre is a ORV copy command that you can use to copy the six Data base ORV Files to another account. NOW THE BAD NEWS....All file def's or any other code that references the account must be changed in the active file and updated before the copy. ALL Reportwriters, Protocols, Formats, etc., must be transferred and update separately.

   isk to Disk]

3. The Best Way is to use the ORVDUMP utility to copy the complete Database to tape and also to recovery.

   :sk to Tape]

* Back and Recovery is your responsibility !!!

BACKUP & RECOVERY

1. BACKUP Student Records & Perf. Elmts. Data bases

   Use Backup Cle
   Run UNN

2. Verify DSN on Tape (Optional)

   Use TAPE INFO CLE
   Run UNN

3. Restore Student Records and Perf Elmts Data Bases

   A. Student Records
      Erase PROFILE.DEFQ
      Erase PROFILE.MSTR
      Erase PROFILE.REC1
      Erase PROFILE.REC2
      Erase PROFILE.RES
      Use BKUPSR
      Run Unn

   B. Perf Elmts
      Erase Perf.Elmts.DEFQ
      Erase Perf.Elmts.MSTR
      Erase Perf.Elmts.REC1
      Erase Perf.Elmts.REC2
      Erase Perf.Elmts.RES
      Use BKUPPE
      Run Unn

```
          1
           1.     //D225 JOB E1.MAP,GRABOW
           2.     //S1 EXEC ORVDUMP,VOL=PROFIL
           3.     //SYSIN  DD  *
           4.     DUMP NAME=PROFILE.DEFO
           5.     DUMP NAME=PROFILE.MSTR
           6.     DUMP NAME=PROFILE.REC1
           7.     DUMP NAME=PROFILE.REC2
           8.     DUMP NAME=PROFILE.RES
           9.     DUMP NAME=PERF.ELMTS.DEFO
          10.     DUMP NAME=PERF.ELMTS.MSTR
          11.     DUMP NAME=PERF.ELMTS.REC1
          12.     DUMP NAME=PERF.ELMTS.REC2
          13.     DUMP NAME=PERF.ELMTS.RES
          14.     //
      >
```

```
  use tapeinfo cle
> 1
    1.     //D225 JOB E1.MAP,GRABOW
    2.     //TAPEINFO EXEC TAPEINFO,PARM='NODUMP,LISTVOL',TAPE=PROFIL
    3.     /*
>
```

```
  use bkuppe cle{
> 1
    1.     //D225 JOB E1.MAR,GRABOW
    2.     //S1 EXEC ORVDUMP,VOL=PROFIL
    3.     //SYSIN  DD  *
    4.     RESTORE NAME=PERF.ELMTS.DEFO,SEQ=6
    5.     RESTORE NAME=PERF.ELMTS.MSTR,SEQ=7
    6.     RESTORE NAME=PERF.ELMTS.REC1,SEQ=8
    7.     RESTORE NAME=PERF.ELMTS.REC2,SEQ=9
    8.     RESTORE NAME=PERF.ELMTS.RES,SEQ=10
    9.     //
>
```

```
  use bkupsr cle
> 1
    1.     //D225 JOB E1.MAR,GRABOW
    2.     //S1 EXEC ORVDUMP,VOL=PROFIL
    3.     //SYSIN  DD  *
    4.     RESTORE NAME=PROFILE.DEFO,SEQ=1
    5.     RESTORE NAME=PROFILE.MSTR,SEQ=2
    6.     RESTORE NAME=PROFILE.REC1,SEQ=3
    7.     RESTORE NAME=PROFILE.REC2,SEQ=4
    8.     RESTORE NAME=PROFILE.RES,SEQ=5
    9.     //
>
```

LOG & ARCHIVE FILES

1. The Performance Element Data Base has the logging capability in effect.

   The file definition has STATISTICS = 2 and LOG = 1 (for each Subfile)

   Log = 1 $\longrightarrow$ Every user's selection and deselection of the data base is logged.

   Statistics = 2 $\longrightarrow$ Information in the Log is appended to the ARCHIVE FILE, and the Log is erased.

2. Reference: File Definition Doc. #35
   Chapter B.11 (p. 185)

3. How to obtain output (on screen)

   ```
   El.MAR
   Call SPIRES
   Select PERF FRAMES
   Show Subfile LOG
   Show ARCHIVE
   ```

4. How to Read

   See pg 191 of Reference

SPIRES SECURITY

1. Issues:	On Line	Batch
A. Access to Computer	Acct. Numbers Password (Key)	Acct Number (Key)
B. Access to SPIRES	NONE	NONE

C. File Access (Attributes for defining access priviledges)

MASTER - Authority to do anything; no restrictions. SEE Visual access to the entire SPIRES File.

UPDATE - Update file

PROCESS - Change authority; write

Public/urs/ Private File Names — COPY - Copy file to their account

Accounts — ACCOUNT - Valid access accounts

D. Indexes — SECURE SWITCH(Browse, Delete, Update)

E. Subfiles — ACCOUNTS - Valid access accounts (Browse, Delete, Update)

SECURITY
- Mask - Value hidden
- Constraint - Limit users accts
- No Update - No updates allowed
- No Search - No Browsing
- Switches - 1-16

used with Priv Tags

Edit; update, logical operators, Global, conversion, upp→lower Defn your own editing

SPIRES SECURITY (con't)

F.	Record	- Security for a particular record using a Bit Map - Examine, browse, update, delete - Record level data dependent security
G.	Element	- used with Priv Tags Browse, update, visual display - Down to a specific element
H.	User Proc's	- Procedures (Small programs or segments of code) - Security switches - Variables - Processing values - Editing - Conversion - Browse, update, delete

$\boxed{\text{MOVING FILES FROM VAX TO AS/6}}$

$\boxed{\text{Logon to VAX}} \left\{\begin{matrix} A \\ B \\ C \end{matrix}\right\}$

RUN $\triangle$ PUBLIC: VAXRFE

<u>Follow Prompt's enter</u> $\left\{\begin{matrix} \text{Computer} \\ \text{PW} \\ \text{Files} \end{matrix}\right.$

$\boxed{\text{Logoff}}$

This is used to transfer the Pre/Post Perf Elements Ecoms to the As/6 System. The files can then be used with the inpur format

$\boxed{\text{COMPILING VGROUPS}}$

```
CALL SPIRES

GET ORV.E1.MAR.FRAMES.VARS

SELECT VGROUPS

ADD

CALL SPICOMP

SELECT VGROUPS

COMPILE FRAMES.VARS

ZAP VGROUPS   *FRAMES.VARS   SOURCE
```

$\boxed{\text{COMPILING PROTOCOLS}}$

```
CALL SPIRES

GET  ORV.E1.MAR. MENU.DRIVER

SELECT SYS△PROTO

COMPILE MENU.DRIVER
```

APPENDIX H -- BUILDING AND MAINTAINING PERFORMANCE ELEMENT FRAMES

The College of Education Performance Element Information System

Building and Maintaining the Performance Element Frames

Prepared by Dr. Richard Warren & Chad Grabow
Research Institute for Studies in Education[1]

Document No. 1, Version 1.0

July 16, 1983

The College of Education Performance Element Information System, referred to as the PERF System in this document, is a data base system implemented at Iowa State using SPIRES (the Stanford Public Information Retrieval System). The system is designed to provide users with information about Performance Elements. The information is provided to inquirers via a series of "information frames". Each frame is essentially a piece of information and one or more choices for the inquirer of where to go after he is finished with the frame. The choices link the frames together and allow the inquirer to move through the system from one frame to another. The purpose of this document is to explain to you, the designer, how to enter and maintain the information frames in the PERF System.

The Information Frames

General Description: In general, an information frame is lines of text to be displayed to an inquirer and a link to other frames in the system. The information text is made up of two different parts: 1) the title and 2) the body, or text, of the information. The title is a concise description of the information in the frame while the text is a narrative presenting the desired information.

The link to the other frames in the system is a "What next?" prompt at the end of each frame. The "What next?" prompt at the end of the frame can be one of three methods for allowing the inquirer to choose what he will see next. The three methods are: the "menu", the "yes/no question", and the "done branch".

The "menu" consists of a list of numbered items and the names of the frames which contain the information about these items. The inquirer is presented with this "multiple choice question" and asked to pick the next item he wishes to see. After a selection is made, the frame associated with that choice is displayed and the process continues with the next frame. A frame with a "menu" might look like this:

Example 1

[1] Permission obtained to use and modify by the Systems Group, ISU Computation Center. Adopted from "The ISU Student/Faculty Information System," by John P. Hauck, Systems Group, ISU Computation Center.

```
THIS IS THE BEGINNING OF THE PERFORMANCE ELEMENT SYSTEM.
You will always return to this point when you
type TOP.
To locate information, select one of the major information
categories listed below.
1. Knowledge of Education.
2. General Teaching Skills.
3. Self-Concept and Goals in Education.
Please enter item number here. (Type STOP to leave the system) ->
```

The "yes/no question" is made up of a question that can be answered with either "yes" or "no" and the names of two frames--one to be displayed if the answer to the question is "no" and one to be displayed if the answer is "yes". The question is displayed at the end of the frame, and the user chooses where he will go next by answering "yes" or "no" to the question. For example, the inquirer might see:

Example 2

```
KNOWING JOB INTERVIEW TECHNIQUES
4.0 Resources/Activities
Numerous printed and filmed materials...
      .
      .
      .
Do you wish to see information on any resources listed above?
Please enter YES, OK, or NO. (Type STOP to leave the system) ->
```

If the inquirer answers "yes" to the question, he will get more information about requirements; if he answers "no", he will be shown the frame designated as the "no" response frame.

The "done branch" is made up of a single choice and essentially only allows the inquirer to pause before he goes on. For example, a "done" frame might look like this:

Example 3

```
SELECTING AND GENERATING INSTRUCTIONAL OBJECTIVES

2.0 Objectives
    ----------

The objectives of this module are:

2.1 Students will distinguish between a general objective...
        .
        .
        .
When done reading, type DONE. (Type STOP to leave the system) ->
```

When the inquirer types "DONE", the frame designated as the "done branch" will be displayed.

It is important to remember that only one of the three methods of choosing "what next" can be used in any one frame.

The Elements That Make Up a Frame: An information frame, referred to as a "record" in SPIRES terminology, is made up of individual pieces of information called elements. Each element has certain restrictions placed on it such as: the element must appear in the record, the element is optional in the record, the value of the element must be only so many characters in length, only so many occurrences of the element are allowed, etc. Below is a description of the elements that make up a record in the PERF System and a list of the restrictions imposed on the values of each.

1. **ACCOUNT** - holds the timesharing account of the designer entering the record. It has the form gg.uuu, for instance, HI.XXX. It need not be entered by the designer since it is automatically generated by the system when a record is added to the file.

2. **DATE-,TIME-/ADDED,UPDATED** - hold the dates and times when the record was added to the system and when it was last updated. These four elements are also generated automatically so they need not be entered.

3. **KEY** - holds the name of the frame being added to the system. This value cannot be more than 128 characters in length and can contain only the characters A-Z, the numbers 0-9, and the special characters ./_-. Blanks are not allowed in this value. This element is required in the frame.

   It is strongly recommended, but not required, that each KEY be prefixed with one of the area designations shown in Appendix 1. For instance, general information about the Planning Skills would be prefixed with "PLAN" while information about Knowledge of Education would use "FOUND". This prefixing scheme will be useful in the management and maintenance of the data base.

   SPIRES requires that the KEY of each record in a file have a value which is unique throughout all records in the file, i.e., only one record in a file will have a given value for its KEY. SPIRES checks each KEY for uniqueness when a new record is added to a file, so the designer need not worry about adding franes which conflict with existing frames.

4. **AREA** - holds the identifier of the general area to which the information in the frame refers. For instance, the identifier for information about Planning Skills would be PLAN. This element is required in the record and its value must be one of the values shown for AREA in Appendix 1.

5. **TITLE** - holds the title of the information in the frame. The words in the title should be a concise description of the information in the frame since the individual words will be used as "subject" identifiers for the frame (see SUBJECT below). TITLE is limited to 80 characters in length and can contain only the characters A-Z, blanks, the numbers 0-9, and the special characters '"¢/:,<{[(|)]}>&¬=;@!*$%.+#-_?. This element is optional in the frame but, if it occurs, it can occur only once in a frame.

6. **TEXT** - holds the textual information to be displayed by this frame. Each occurrence of TEXT carries the same valid character and length restrictions as does TITLE. TEXT may occur more than once per frame since each occurrence of TEXT represents one line of text to be displayed at the terminal. TEXT is an optional element although it usually occurs one or more times in a frame.

7. "Menu" items - The next three elements are the components of one item in a "menu". There will be one set of these elements for every individual choice in the menu. If there is to be no menu, none of these three elements should be present in the record. The menu elements are optional in the record but all three must occur for each item in a menu.

   a. **ITEM.NUMBER** - holds the internal name of this choice. At present, this name is not used in the system so the designer may enter anything here as a place holder.

   b. **ITEM.TEXT** - holds the text to be displayed for this particular menu item. When a menu is displayed, SPIRES will generate a number for each item and display that number in front of the text as shown in Example 1. The item text is limited to 75 characters in length and has the same valid character restrictions as does TEXT.

   c. **ITEM.BRANCH** - holds the KEY of the frame to branch to if this item is chosen by the inquirer. Length and valid character restrictions are the same as those for KEY.

8. The "yes/no question" - The next three elements make up the "yes/no question" method for asking the inquirer what he wants to see next. These elements are optional in the record but all three must occur if a "yes/no question" is used in the frame.

   a. **YESNO.TEXT** - holds the text of the yes/no question to be displayed after the text of the frame. Length and character restrictions are the same as those for ITEM.TEXT.

   b. **YES.BRANCH** - holds the KEY of the frame to branch to if a response of "YES" is given to the yes/no question. Length and valid character restrictions are the same as those for KEY.

   c. **NO.BRANCH** - holds the KEY of the frame to branch to if a response of "NO" is given to the yes/no question. Length and valid character restrictions are the same as those for KEY.

9. **DONE.BRANCH** - holds the KEY of the frame to branch to if "DONE" is typed in response to the "When done reading, type DONE" prompt. Length and valid character restrictions are the same as those for KEY. DONE.BRANCH is optional in the frame.

10. **SUBJECT** - holds subject or "keyword" values which describe the information in the frame. The total set of subject words for a frame is made up of the individual words in the title of the frame, the individual words in the long form of the AREA of the frame (see Appendix 1), and any words specified in the

SUBJECT element. Any words which describe the subject matter of a frame that do not appear in the title or area designation, should be specified under SUBJECT. See Examples 5 and 9. This element is optional and several words may be specified in one occurrence of the element (as in Example 5). These additional words may be separated by blanks or commas.

The logic of the "menu", the "yes/no question", and the "done branch": Only one type of prompt is allowed in a given frame. If a certain frame is to have a "menu" at the end, then the frame will contain one or more sets of the elements ITEM.NUMBER, ITEM.TEXT, and ITEM.BRANCH--one set for each possible way to go next. If the frame carries a "yes/no question", then the frame will contain one set of the elements YESNO.TEXT, YES.BRANCH, and NO.BRANCH thus giving the inquirer two ways to go next. If the frame is to have a "done branch", then the frame will contain only one branching value, the value in the DONE.BRANCH element, thus giving the inquirer only one way to go from this frame. Since the designer has the capability of giving the inquirer many, only two, or only one way out of a frame, consideration should be given as to what method of "What next?" is appropriate for the frame.

Adding Frames to the System

Adding frames to the PERF System involves calling SPIRES, selecting the file to to add information, setting up a prompting program, and initiating an add operation. The add operation invokes the prompting program and SPIRES starts asking for the values of the elements in the frame. Examples of this are shown in Examples 4 & 5. After the ADD command is entered, SPIRES begins to ask for values for the elements in the record by prompting with a colon (:) followed by the element name. The designer enters the value for the element, presses the return key, and SPIRES goes on to prompt for the next element. This process continues until values for all of the elements in the record have been entered. Note that SPIRES prompts for the proper number of occurrences for each element and checks each value for proper length, valid characters, etc. If any error conditions are detected, SPIRES rejects the value and asks for the value again.

There are several control sequences used during the prompting operation to control the form of the text entered. They are: '//', '/   text', and 'text//'. The double slash, '//', indicates that a null value is to be entered; this gives blank lines for output. The leading slash, '/   text' allows values to begin with blanks, and the trailing double slash, 'text//', allows the value to be continued on the next input line. The return key, shown as **<return>** in the following examples, can also be considered a control sequence since it is used to terminate prompting for multiple occurrences of an element and to skip optional elements. An example of all of these sequences will be shown in the following examples.

Two examples of adding frames are shown below. The text shown in bold is text typed by the designer. The text lines along the left that are not bold are the prompts issued by the system while the text to the right of the bold text is explanatory notes for this document. All lines typed by the designer are terminated by pressing the return key. The symbol **<return>** indicates that the return key was pressed at that point while **<break>** indicates that the break key was pressed.

Example 4

```
COMMAND? CALL SPIRES <-- call SPIRES
-WELCOME TO SPIRES-3 ... IF IN TROUBLE, TRY 'HELP'
-> SELECT PERF FRAMES <-- select the proper file
-> SET FORMAT $PROMPT <-- set prompting program
-> ADD <-- initiate an add operation
: KEY               GEN$1000 <-- invalid character ($) in key
Invalid character(s) in value--only letters, numbers, and _/-.*allowed
: KEY               GEN1000
: AREA              UNIV
: TITLE             THIS IS THE BEGINNING OF THE INFORMATION SYSTEM.
: TEXT(1)           You will always return to this point when you
: TEXT(2)           type TOP.
: TEXT(3)           //  <-- 2 slashes give a blank line on output
: TEXT(4)           To locate information, select one of the major information
: TEXT(5)           categories listed below.
: TEXT(6)           <return> <-- indicates no more text elements

:    ITEM.NUMBER       Q1
:    ITEM.TEXT         Knowledge of Education. <-- 1st item in menu
:    ITEM.BRANCH       FOUND1000 <--  KEY of frame to branch to if
                                          this item chosen

:    ITEM.NUMBER       Q2
:    ITEM.TEXT         General Teaching Skills. <-- 2nd item
:    ITEM.BRANCH       TEACH1000 <--  branch if 2nd item chosen

:    ITEM.NUMBER       Q3
:    ITEM.TEXT         Self-Concept and Goals in Education <--  3rd menu item
:    ITEM.BRANCH       PERSONAL/1 <--  branch if 3rd item chosen

:    ITEM.NUMBER       <return> <--  indicates no more menu items


:    YESNO.TEXT        <return> <--  no yes/no question

: DONE.BRANCH       <return> <--  no done branch
: SUBJECT(1)        <return> <--  no additional subject words
->
```

When displayed to an inquirer, the above frame looks like this:

```
     THIS IS THE BEGINNING OF THE INFORMATION SYSTEM.

     You will always return to this point when you
     type TOP.

     To locate information, select one of the major information
     categories listed below.
     1. Knowledge of Education.
     2. General Teaching Skills.
     3. Self-Concept and Goals in Education.
     Please enter item number here. (Type STOP to leave the system) ->
```

Next is an example of adding a frame that contains a "yes/no question" to the system. Since we are already in SPIRES, have a file selected, and have the prompting routine set, all that is necessary is to initiate another add by typing ADD as shown:

Example 5

```
-> ADD
: KEY               FOUND1414
: AREA              FOUND
: TITLE             KNOWING JOB INTERVIEW TECNIQUES
: TEXT(1)           4.0  Resources/Activities
: TEXT(2)
: TEXT(3)           The subcategories listed below contain suggestions
: TEXT(4)           valuable to individuals seeking assistance in
: TEXT(5)           the interview process.
: TEXT(6)           //   <-- Blank line on output
: TEXT(7)           A.  Pamphlets/Brochures
: TEXT(8)           B.  Films
: TEXT(9)           C.  Periodicals
: TEXT(10)           <return> <-- end of TEXT

:    ITEM.NUMBER       <return> <--  indicates no menu items


:    YESNO.TEXT        Do you wish to see information on// <-- continue input
                                                              on next line
     more-->           on any of the above subcategories?
:    YES.BRANCH        FOUND1416  <-- branch to this frame if inquirer responds YES
:    NO.BRANCH         FOUND1410 <-- branch to here if response is NO

: DONE.BRANCH       <return> <--  no done branch
: SUBJECT(1)        basic requirements <-- additional subject words for this frame
->
```

When displayed, the above frame looks like this:

```
     KNOWING JOB INTERVIEW TECHNIQUES

     4.0 Resources/Activities

     The subcategories listed below contain suggestions valuable to
     individuals seeking assistance in the interview process:
     A.  Pamphlets/Brochures
     B.  Films
     C.  Periodicals
     Do you wish to see information on Basic Program requirements?
     Please enter YES, OK, or NO. (Type STOP to leave the system) ->
```

General comments about errors in element values: SPIRES will detect most errors in the elements as they are entered and will reject the element, issue an error message, and reprompt for the element when an error is detected (see Example 4). If an error is detected in the line currently being entered, it can be corrected by either: 1) backspacing to the error and retyping or 2) pressing the break key in which case SPIRES will ask for the element again (see the first part of Example 6). If an error is discovered in an element that has already been entered or the designer decides to make a change in the frame, it is usually easiest to complete the frame and correct it later. The method for correcting errors in existing frames will be discussed in the next section. In some instances though, it may be better to terminate the add altogether, and start fresh. This is done by pressing the break key in response to an element prompt and responding with "YES" to the "Do you wish to cancel ..." question. SPIRES will terminate the add and it can be started again (see the second part of Example 6). Using the break key produces results like these:

Example 6

To correct a mistake in an element-

```
:TEXT(3)              the txet in this farme<break> <-- break pressed after text entered
:TEXT(3)              the text in this frame
```

To terminate the add operation-

```
:DONE.BRANCH          <break> <-- break key pressed immediately in response
:Do you wish to cancel this request? (YES/NO/HELP) YES        to prompt
-UPDATE ABORT. CODE=S2      <-- ADD terminated
->
```

Displaying Frames

Any frame in the system can be displayed at the terminal if the key of the frame is known. This is done by using the DISPLAY command. For instance, to display frame FOUND1416, the designer would type:

```
-> DISPLAY FOUND1416
```

Updating Existing Frames

Material in frames that already exist in the file can be changed by retrieving the particular frame from the file, changing the information using WYLBUR editing commands, and putting the changed record back into the file. An existing record can be retrieved from the file by using the TRANSFER command. For instance, to retrieve the record with the key FOUND1416, the designer would type:

```
-> TRANSFER FOUND1416
```

This will place the record in the active file where it can be edited using WYLBUR editing commands. This record is shown below:

Example 7

```
-> LIST
```

```
 1. ACCOUNT = E1.MAR;
 2. DATE.ADDED = 12/19/82;
 3. TIME.ADDED = 13:42:36;
 4. DATE.UPDATED = 03/07/83;
 5. TIME.UPDATED = 15:00:24;
 6. KEY = FOUND1416;
 7. AREA = Foundations of Education;
 9. TITLE = KNOWING JOB INTERVIEW TECHNIQUES;
11. TEXT = To locate information, select one of the Resources/Activities;
12. TEXT = subcategories listed below;
13. ITEM.NUMBER = Q1;
14.   ITEM.TEXT = Pamphlets/Brochures;
15.   ITEM.BRANCH = FOUND1414.1;
16. ITEM.NUMBER = Q2;
17.   ITEM.TEXT = Films;
18.   ITEM.BRANCH = FOUND1414.2;
19. ITEM.NUMBER = Q3;
20.   ITEM.TEXT = Periodicals;
21.   ITEM.BRANCH = FOND1414.3;
22. ITEM.NUMBER = Q4;
23.   ITEM.TEXT = Books;
24.   ITEM.BRANCH = FOUND1414.4;
```

To change information in the record, simply edit the information on the right side of the '='s and then type:

```
-> UPDATE
```

to put the changed record back into the file. This tells SPIRES to update the old copy of the frame with the revised copy in the active file.

If it becomes necessary to add information to a frame (more text, another "menu" item, etc.), the elements should be added in a form similar to that shown above. The element name is typed, followed by an equals sign (=), followed by the element's value, ended with a semicolon (;). A list of proper element names is shown in Table 1. Most elements also have an alias, a shorter form of the name, that is valid for use when entering new values (see Example 9). For example, to add a new "menu" item as the third item in the "menu", the three elements which make up an item would be inserted between lines 18 and 19 as shown below.

Example 8

```
-> TRANSFER FOUND1416
-> COLLECT 25.1
     25.1  > ITEM.NUMBER = Q5; <-- NOTE:  all element values are terminated
     25.2  >   ITEM.TEXT = Audio Tape;             with a semicolon (;)
     25.3  >   ITEM.BRANCH = FOUND1416.5;
     25.4  > <break>  <-- break key pressed
-> UPDATE  <-- put changed record back into file
```

If a frame is to be converted from one form of prompt to another, the elements that make up the existing prompt have to be deleted and the new elements need to be added. To convert the frame in Example 7 from a "menu" to a "yes/no question", lines 13 through 24 have to be deleted and the proper elements for a "yes/no question" added. Additional subject words will also be added to the frame. Again, elements should be added in a format similar to that shown making sure that each element is terminated with a semicolon (;). For example,

Example 9

```
-> TRANSFER FOUND1416          NOTE: Element name aliases used here
-> DELETE 8/25   <-- delete "menu"
-> COLLECT 8
     13.  > YTEXT = Do you wish to see more?; <-- ends with a ';'
     14.  > YBR = FOUND1414.1;
     15.  > NBR = FOUND1410;;
     16.  > <break> <-- break key pressed
-> UPDATE
```

If new elements have to be added to a record, a list of valid element names and their aliases is available in Table 1 or one can be obtained by entering the command SHOW ELEMENT NAMES when the file is SELECTed, i.e.,

```
-> SELECT INFO FRAMES
-> SHOW ELEMENT NAMES
```

As can be seen from Example 9, either element names or aliases thereof can be used to enter elements in a record.

Table 1: Element Names for the PERF System Frames

ACCOUNT, ACCT	NO.BRANCH, NBR
AREA	SUBJECT, SUB
DATE.ADDED, DA	TEXT, T
DATE.UPDATED, DU	TIME.ADDED, TA
DONE.BRANCH, DBR	TIME.UPDATED, TU
ITEM.BRANCH, IBR	TITLE
ITEM.NUMBER, INUM	YES.BRANCH, YBR
ITEM.TEXT, ITEXT	YESNO.TEXT, YTEXT
KEY	

Entering the PERF System

Designers can enter the INFO System to display and evaluate newly added or updated frames by issuing the command:

```
-> ..MENU.DRIVER
```

Note the two periods (..) in front of MENU.DRIVER; these periods are part of the command and have to be entered.

MENU.DRIVER is the "driver" program for the PERF System. The above command invokes this program and puts the designer in the PERF System where he can display and evaluate frames using the inquiry facilities of the system. The designer may exit the PERF System by pressing the break key in response to any prompt and responding with "OUT" to the next menu prompt. For example,

Example 10

```
-> ..MENU.DRIVER <-- invoke driver for PERF System

     the designer displays frames as inquirer would

:Please enter item number here. (Type STOP to leave the system) -> <break> <-- press
break key
      .
      .
      .
Do you wish to:
1. Terminate this session and leave the system.
2. Go to the first menu (TOP) in the system.
3. Get instructions about how to use the system.
4. Return to previous frame.
:Which option do you want? Please enter number here -> OUT <-- return to SPIRES
->
```

The option of responding "OUT" at the above prompt is restricted to designers of the system (persons with HI.xxx or HE.xxx accounts); inquirers using the system do not have this option.

Removing Frames From the PERF System

A frame can be removed from the system by using the SPIRES command REMOVE. For instance, to remove the frame PLAN9999 from the system, the designer would type:

```
-> REMOVE PLAN9999
```

Since the PERF System is a inter-connected structure, removing a frame from the system may leave a "hole" in the structure. This means that whenever a change is made to any KEY in the system, such as removing a frame or changing a KEY, all of the frames which contain branches to that frame need to be changed also. The section below called "Searching the File..." will explain how to find all the frames that reference the changed/removed frame.

Changing the KEY of a frame

Since the value of the KEY of a frame is the unique, identifying element for that frame, changing the KEY essentially means that a new record is created. Since the new frame is identical to the old one, except for the KEY, changing the KEY can be accomplished by:

Example 11

```
-> TRANSFER PLAN/CURRENTKEY <-- get record whose KEY is to be changed
-> REMOVE PLAN/CURRENTKEY <-- remove "old" KEY record from file

        use editing commands to change value of KEY

-> ADD <-- add "new" record to file
```

As with removing a frame from the file, changing a KEY value implies that other frames need to be changed to reflect the new KEY value in the changed frame. See "Searching the File..." for details about how to find the frames that need to be changed.

Searching the File and Retrieving Particular Frames--Using Indexes

Indexes and How to Use Them: An INDEX is a facility provided by SPIRES that allows the designer to search through all of the records in the system and retrieve those records which meet a specified criterion. Indexes are useful when it becomes necessary to find, for instance, all of the records which were added to the system since a given date, all of the records added by a particular designer, all of the records which deal with a particular area, etc.

Several indexes have been defined for the PERF System to allow for searching like that described above and for the easier and more efficient building and maintenance of the system. A list of these indexes is shown in Table 2 or one can be obtained by entering the command SHOW INDEXES when the file is SELECTed.

Table 2: Index Names for the PERF System Frames

ACCOUNT, ACCT	DATE.ADDED, DA
AREA	DATE.UPDATED, DU
BRANCH, BR	SUBJECT, SUB

Indexes are accessed via the FIND command. The general form of the FIND command is:

Example 12

FIND index-name relational-operator search-value where:

- *index-name* is one of the names shown in Table 2

- *relational-operator* is one of the following:

=	BEFORE	¬=
(blank)	AFTER	>
		>=
		<
		<=

- *search-value* is the criterion specified by the designer

For example, a designer might wish to find all of the records added to the system in March of 1983. The command to do this would look like:

```
-> FIND DATE.ADDED = MARCH 83
        or
-> FIND DATE.ADDED MARCH 83  <--The blank is equivalent to the "="
```

The command to find all of the frames updated after January 20, 1983 would be:

```
-> FIND DATE.UPDATED AFTER 01/20/83
```

If a designer wished to find all of the frames that he entered, he would use the ACCOUNT index and FIND all of the frames with his account in the ACCOUNT element. For instance,

```
-> FIND ACCT HI.XXX
-RESULT: 51 FRAME(S)
->
```

SPIRES responds with the "-RESULT:" line that tells how many frames have a value for ACCOUNT of HI.XXX. Note that ACCT instead of ACCOUNT was used in the FIND command. This can be done since ACCT is an alias for the ACCOUNT index. Any index name or alias thereof can be used in a FIND command (see Table 2).

Displaying the Search Result: After a result is obtained, the frames in the result can be listed in whole, or in part, at the terminal. The frames can be listed using the TYPE command. If the designer issues a TYPE command after a result is obtained, all of the information in each frame will be displayed at the terminal. If the designer wishes to see only a part of the information, such as the KEY and AREA, he can issue a command of the form:

```
-> TYPE KEY AREA
```

and only the KEY and AREA values from each record in the search result will be displayed. Any valid element names (see Table 1) can be specified on the TYPE command and only those values from each record will be displayed.

Retrieving Frames Using the Search Result: Another instance in which index searching is valuable is when a frame has been removed from the system or a KEY value has been changed and all of the frames which reference the removed/changed frame must be changed. These frames can be found by searching the BRANCH index for the KEY value of the removed/changed frame. The BRANCH index holds all of the branch values used in the system, so the command

```
-> FIND BR PLAN1416
```

will find all of the frames that contain a branch to the frame PLAN1416. This search result can then be used to retrieve these frames for editing.

The easiest method of updating the frames in the search result to reflect the change made to PLAN1416 is to step through the frames in the result, correcting and updating the frames one at a time. This is done by issuing the following series of commands:

Example 13

```
-> FIND BR PLAN1416 <-- get the frames to change
-RESULT: 10 FRAME(S)
-> FOR RESULT <-- start processing the search result
+> TRANSFER <-- bring the first frame from the search result
                        into the active file

              edit the frame in the active file

+> UPDATE <-- put the edited version of the first frame back into the file
+> TRANSFER <-- bring the second frame into the active file

              edit the frame

+> UPDATE <-- put the edited version of the 2nd frame back in the file
+> TRANSFER <-- get the third
    etc.   <-- repeat the process until "-END OF GLOBAL FOR" message appears
-END OF GLOBAL FOR
->
```

When the "-END" message appears, all of the frames in the search result have been updated.

APPENDIX 1

Area Designations and Suggested Key Prefixes

AREA	SHORT FORM	LONG FORM
FOUND	Found. of Educ.	Foundations of Education
TEACH	Gen. Teach. Skills	General Teaching Skills
GOAL	Concept Goals	Self-Concept and Goals in Education
PLAN	Plan. Skills	Planning Skills
INSTR	Imp. Instr. Plans	Implementing Instructional Plans
EVAL	Eval. Diag.	Evaluation and Diagnosis
MGMT	Mgmt.	Management

Technical Note 1.01 for Document No. 1

Using the Quote (") and the Semicolon ( ) in Text|

The quote (") and the semicolon(;) are treated as "magic" symbols by SPIRES. Therefore, some care should be taken when these symbols have to be used in text in a frame. Note that the quote as mentioned here is the single character symbol ("), not two adjacent apostrophes (').

When the designer is using the $PROMPT format to enter text for a frame, the quote (") and the semicolon (;) may be entered directly and SPIRES will handle them just as it does other text.

If a frame is being updated, after a TRANSFER command, and the designer wishes to add a quote or a semicolon to a line, the entire line must be enclosed in quotes and each quote within the line must be doubled. For instance, the input line

```
TEXT = "This is a line with ""quotes"" and a semicolon; in it.";
```

would be displayed as

```
This is a line with "quotes" and a semicolon; in it.
```

Technical Note 1.02 for Document No. 1

Logon Procedure for Performance Element Data Base

```
ENTER SYSTEM ID: M

IOWA STATE COMPUTATIONAL CENTER  AC003-031  13:42:16 07/12/83

USER? E1.CLG
PASSWORD? XXX
ACCOUNT I4049? (RETURN)
LAST LOGOFF AT 13:42

-WELCOME TO SPIRES-3 ... IF IN TROUBLE, TRY 'HELP'
* Which Data Base Do You Desire to Use?
*
* 1.  PERFORMANCE ELEMENTS
* 2.  PRO*FILE STUDENT RECORDS
* 3.  LOGOFF SYSTEM
*
:Please enter 1, 2, or 3 ->  1
* You are now in the PERFORMANCE ELEMENT Data Base for the
* input mode.  The format $PROMPT has already been set.  When you
* are finished using the data base, EXIT from SPIRES and LOGOFF
* the system.
->(enter SPRIES Commands)
->
->
->
->EXIT
?LOGOFF
```

APPENDIX I -- MASTER LIST OF PERFORMANCE ELEMENTS

MASTER LIST OF PERFORMANCE ELEMENTS

I. Pedagogical Knowledge
   A. Learning specific subject matter (6 elements)
   B. Knowing about the school as a social/historical institution (6 elements)
   C. Knowing about the school as a legal/political institution (6 elements)
   D. Knowing about the teaching profession (6 elements)

II. General Teaching Skills
   E. Working with other porfessionals and adults (3 elements)
   F. Working with students (3 elements)
   G. Working in a variety of educational situations (4 elements)

III. Self-Concept and Decision-making Skills
   H. Working on self-development (4 elements)
   I. Involving yourself in the teaching profession
   J. Building an effective learning environment (4 elements)

IV. Planning Skills
   K. Planning lessons and units (5 elements)
   L. Developing curriculum content (4 elements)
   M. Planning activities based on diagnosis (3 elements)
   N. Planning for efficient organization of time, space, materials, and equipment (1 element)
   O. Using resource materials in planning (5 elements)

V. Instructional Planning and Implementation
   P. Applying theories of learning (4 elements)
   Q. Presenting subject matter (1 element)
   R. Modeling basic skills (1 element)
   S. Helping students develop learning/thinking skills (6 elements)
   T. Maintaining student interest (4 elements)
   U. Dealing with unplanned aspects of instruction (3 elements)

VI. Evaluation and Diagnosis
  - V. Analyzing teaching effectiveness (2 elements)
  - W. Incorporating student evaluation techniques (3 elements)
  - X. Encouraging student involvement (2 elements)
  - Y. Designing and implementing evaluation instruments (3 elements)

VII. Management
  - AA. Understanding the theory and application of management techniques (2 elements)
  - BB. Generating positive classroom attitudes (2 elements)
  - CC. Utilizing effective disciplinary techniques (4 elements)

APPENDIX J -- EVALUATION FORM

Evaluators Name ______________

Group __________________________

Objectives - Activities - Assessment Sheet

Objectives	Activities	Assessment
Evaluator will	Logon/Logoff System Determine SAT-MATH Get a Copy of Program of Study Enter Educational Experience . . . * Note	Opinion of Group or Individual

*Note: List activities for the two evaluations

- Pre evaluation for prototype

- Preliminary field test