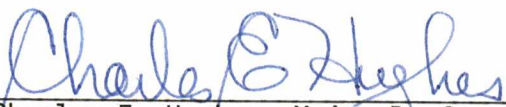
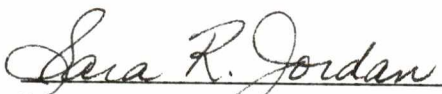
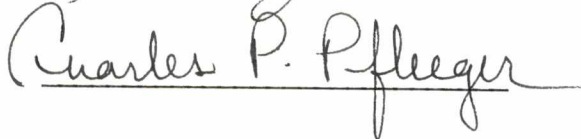


To the Graduate Council:

I am submitting herewith a thesis written by Nancy E. Dowell entitled "RAPTS: Recycle Assembly Parts Traceability System." I recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.


Charles E. Hughes, Major Professor

We have read this thesis
and recommend its acceptance:

Accepted for the Council:


Vice Chancellor
Graduate Studies and Research

X

Thesis

80

.D694

cap 2

RAPTS: RECYCLE ASSEMBLY PARTS TRACEABILITY SYSTEM

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Nancy E. Dowell

June 1980

3041910

DEDICATION

To Mama, whose confidence, encouragement, strength, and common sense I leaned on for thirty-three years, and to my Dad on whom I still lean.

ACKNOWLEDGMENTS

The implementation of this project was requested and funded by the Separation Systems Division (SSD) of Union Carbide Corporation, Nuclear Division. Individuals involved in this project from the SSD area were R. E. Neir, E. L. Newcome, P. S. Roth, and N. I. Cohen.

The author wishes to express thanks to Dr. C. E. Hughes, Dr. S. R. Jordan, and Dr. C. P. Pfleeger for their guidance during this project. Their technical suggestions about the system design, as well as their assistance in developing this report is appreciated.

Several individuals from the Computer Science Division of Union Carbide contributed to the success of this project. Thanks are extended to R. K. Osborn for developing the software needed to implement this project and to C. B. Settles for her assistance in the editorial presentation of this report. The author would also like to acknowledge the contributions of the late W. F. Nolen to her technical and professional growth during the past ten years and to state appreciation for his encouragement toward the completion of this project.

This work was performed at a U. S. Department of Energy installation operated by Union Carbide Corporation, Nuclear Division, under U.S. Government Contract W-7405-eng-26. The computing facility used is located at the Oak Ridge Gaseous Diffusion Plant, Oak Ridge, Tennessee.

ABSTRACT

This paper documents the development cycle of the Recycle Assembly Parts Traceability System (RAPTS). RAPTS maintains historical explosion and implosion traceability of assemblies, as well as key pieces of manufacturing, quality assurance, inspection, and performance data.

This system is being implemented in support of the centrifuge test facility operation at the Oak Ridge Gaseous Diffusion Plant operated by Union Carbide Corporation, Nuclear Division for the Department of Energy.

RAPTS was designed to allow maximum flexibility to the user. Data assembly patterns and data items to be stored are defined to the system by the user. Therefore, the user can alter data storage requirements as needed without computer programmer assistance.

RAPTS was designed using linked lists for actual data storage. Indexes are maintained as data entrance tools.

Although RAPTS was designed to support a specific project, the software package could be used in support of any manufacturing research project due to the degree of control given the user.

TABLES OF CONTENTS

CHAPTER

INTRODUCTION	1
I. RAPS: THE APPLICATION SYSTEM	2
What is RAPS?	2
Gas Centrifuge Test Project	2
Needed Data Accessibility	5
II. SYSTEM DEVELOPMENT CYCLE	12
The "Normal" System Life Cycle	12
RAPS Life Cycle	16
III. DETAIL DESIGN TERMINOLOGY	23
Database Terminology	23
Data Structure Terms	26
IBM/360 Jargon	29
IV. RAPS DETAIL DESIGN	31
Conceptual Model	31
Overview of the Physical Design of the System	36
Detailed File Design	43
Input Form Design	52
Software Design	58
V. RAPS: THE PRODUCTION SYSTEM	62
Customer Support	62
Customer Acceptance	63
RAPS in the Database World	63
Summary	65

CHAPTER

BIBLIOGRAPHY	67
VITA	69

LIST OF FIGURES

FIGURE

1. Interior of a Centrifuge Cutaway	4
2. System Life Cycle	13
3. Roles in Application System Development	14
4. Roles in Construction Business	15
5. RAPTS System Life Cycle	17
6. An Example of Operational Data	24
7. Data Entry List	28
8. Conceptual Model for RAPTS	32
9. Conceptual Schema for RAPTS	33
10. Data Network I	34
11. Data Network II	35
12. Sample Data Network	37
13. RAPTS System Overview Chart	39
14. RAPTS Data Flow Chart	41
15. Assembly Subassembly Record Layout	44
16. Assembly Procedure Frame File	46
17. Procedure Variable Frame Record Layout	48
18. Assembly Transaction Master File Layout	50
19. Procedure Variable Transaction Master File Layout	51
20. RAASM Input Form	53
21. RADEF Input Form	55
22. RADTA Input Form	57
23. Edit Structure Chart	60

INTRODUCTION

Since the 1950's programmer/analysts have been endeavoring to assist corporate management in decision-making by presenting them with computer generated reports. The bulk of the task of successful report generation relies on the existence of a comprehensive, accurate data collection.

Through the years, the basic philosophy of data management has changed as high level languages have been developed, high-density random-access storage devices have become available, and CPU time has become less costly. Commercially written software packages are used more today at least to supplement, if not replace, in-house tailored software. However, when no known commercially available package closely matches a user's needs, a tailored system such as the Recycle/Assembly Parts Traceability System (RAPTS) must be developed.

This thesis documents the design and the implementation phases of RAPTS. Chapter I states the purpose of the application for which RAPTS was designed. Chapter II reviews the steps in an application system's life cycle in general and RAPTS' life cycle in particular. In Chapter III, common ground is established by reviewing currently acceptable database terminology, data structure terms, and IBM/360 jargon. Chapter IV discusses the detailed design of RAPTS. RAPTS' customer support, customer acceptance, and place in the database world are discussed in Chapter V.

CHAPTER I

RAPTS: THE APPLICATION SYSTEM

What is RAPTS?

The Recycle/Assembly Parts Traceability System (RAPTS) is a computer software package designed to manage technical data about serialized production parts. A complete trace of a part and its relevant data is maintained from its piece part stage, through its intermediate subassembly stages, its top assembly residence, and further as it is torn down and its parts reused. RAPTS is concerned with data storage, validation, and retrieval. To be more specific, RAPTS is being developed to manage technical data about gas centrifuge machines.

Gas Centrifuge Test Project

A research and development project in the area of uranium enrichment using the gas centrifuge is well underway at the Oak Ridge Union Carbide operated facilities. The concept of uranium enrichment, increasing the concentration of the isotope uranium 235, using a centrifuge was implemented in 1940 by Jesse W. Beams at the University of Virginia. However, another process, gaseous diffusion, was selected in 1943 to supply the needed enrichment material for the Manhattan Project. Significant technical advancements in the area of gas centrifuge were made by W. E. Groth in Germany in the late 1940's and Gernot Zippe at the University of Virginia in the late 1950's. The Atomic Energy Commission, largely as a result of Zippe's work, authorized a production

facility based on gas centrifuge technology. In the years since 1960, the technology has advanced to the point that today small scale centrifuge test facilities are in existence.

Much of the information about the current Oak Ridge facility is classified. However, an illustration of the basic component makeup of a centrifuge, is depicted in Figure 1. This figure is similar to one published in an August, 1978, Scientific American article.

The concept of centrifuge machine functioning is explained in the following quote from the same article.

The thin-walled rotor is driven by a small electromagnetic motor attached to the bottom of the casing. The top end of the rotor is held in a vertical position by a magnetic bearing and does not touch stationary components. Gas is fed into and withdrawn from the rotor through the stationary center post, which holds three concentric tubes for the feed, the product and the waste. The stationary bottom scoop protrudes into the spinning gas and provides a mechanical means for driving the vertical flow of gas. The top scoop, which serves to remove the enriched product is protected from direct interaction with the rotating gas by the baffle, which has holes allowing the enriched gas to be bled into the area near the scoop. The baffle is needed to keep the top scoop from imposing a vertical flow that would counteract the crucial one generated by the bottom scoop.¹

The operation of a gas centrifuge machine has been compared to an old-time cream separator. Separation is caused by spinning the material at high speed. The heavier $^{238}\text{UF}_6$ material moves to the outside while the lighter $^{235}\text{UF}_6$ remains toward the center.

Enriched material is withdrawn from one centrifuge and cycled to another until the desired level of enrichment is achieved. Many factors

¹Donald R. Olander, "The Gas Centrifuge," Scientific American, Volume 239 (August, 1978), page 39.

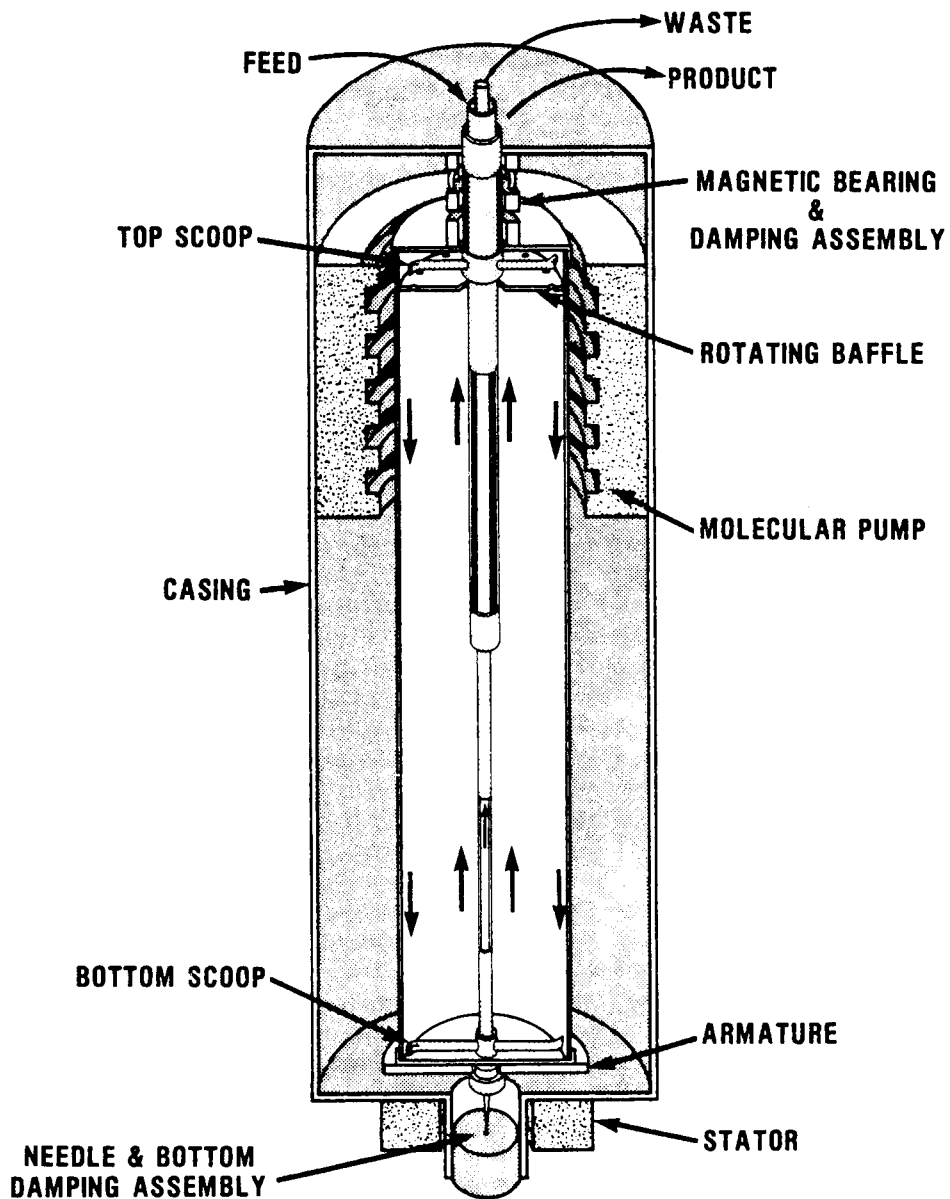


FIGURE 1. INTERIOR OF A CENTRIFUGE CUTAWAY

govern the effectiveness of a particular centrifuge machine. Variations of design, manufacturing techniques, materials, and operating criteria are currently being tested.

Centrifuge technology is of great interest today because of the predicted needs for nuclear fuel and the economic advantages of the centrifuge process over gaseous diffusion. It has been proven that less electricity is needed for the centrifuge process's separative work. However, another major factor is the life expectancy of the centrifuge components. To effectively evaluate the process, not only the "test facility production rates versus power consumption" must be considered but also the initial manufacturing cost for the machines, the expected life of components, and the repair feasibility.

Needed Data Accessibility

In order to effectively evaluate the functioning of the test facility machine assembly, manufacturing and operating records must be available in a readily accessible form.

Currently, the manufacturing data (assembly records, certifications, inspection results, record of waived requirements) is contained in twenty-five standard file cabinets in the Component Preparation Laboratory record center. Most information is contained in packets filed by the machine or subassembly serial number. However, some information is filed serially within a procedure. To service a single request such as "find all machines certified by procedure XYZ between March, 1976, and April, 1977" might require two weeks. A further complication of this request such as doing reliability studies on a subcomponent of these machines based on total run hours might add

another two weeks of clerical search time. Manual searches were made easier in late 1979 due to a massive microfilming effort in the record center. Indexes are available by part serial number and procedure. Yet a scan of a particular reading from a procedure still requires considerable manual effort. To search for trends or correlations hidden in the microfilmed data is still an almost impossible task.

Therefore, faced with the task of designing a Recycle/Assembly (R/A) Facility for the large production centrifuge facility (GCEP) scheduled to be built at Portsmouth, Ohio, the Recycle/Assembly task group were painfully aware of the need for a more usable data collection. Decisions that would effect GCEP's operational effectiveness would have to be made for this large plant based on the information gathered during the operation of current test facilities. The task group cited the following areas of support information needed in order to make sound judgments.

A. PRODUCTION/RECYCLE and SUBASSEMBLY REPAIR

1. A means to facilitate effective Manpower/Materials control for centrifuge assembly and recycle and subassembly repair shall be provided. This requires a means of:
 - (a) determining the materials required for the specific tasks to be performed,
 - (b) relating a cost to each unit of each material to be used in the assembly-recycle-repair tasks,
 - (c) determining the standard manpower-hours by craft for the specific tasks to be performed,
 - (d) relating cost per hour to each craft.

2. A means of providing support for activities that would enhance R/A facility productivity shall be provided. This requires a means of:
 - (a) analyzing variables to determine algorithms that would speed the diagnosis of a non-crash machine failure,
 - (b) analyzing variables to determine algorithms that would speed the diagnosis of failed subassemblies,
 - (c) supporting analyses to estimate the components damaged in a failed subassembly prior to diagnostic testing.
3. The information required for improved scheduling of recycled machines shall be provided. This requires a means of:
 - (a) determining the materials required for the specific tasks to be performed,
 - (b) determining the standard manpower-hours by craft for any given task to be performed,
 - (c) determining what tasks need to be performed on machines pending recycle, and
 - (d) determining current demand for centrifuges by the process buildings.
4. A means for improved scheduling for the repair of damaged subassemblies shall be provided. This requires a means of:
 - (a) determining what type of repair is required for each damaged subassembly,
 - (b) determining the standard manpower-hours by craft for any given task to be performed.

B. QUALITY CONTROL and INSPECTION

1. A means for the storage and retrieval of Quality Assurance/
Quality Control data shall be provided.
 - (a) storing certain data values in a form suited for computation
and analysis.
2. A means to facilitate surveillance of vendors of centrifuge
parts shall be provided. This requires a means of:
 - (a) tracing centrifuge parts to their respective vendors, and
 - (b) storing and retrieving vendor supplied inspection data.

C. SCRAP/WASTE MANAGEMENT

1. A means of monitoring the amount and type of centrifuge parts
scrapped shall be provided. This requires a means of:
 - (a) recording centrifuge parts that are to be scrapped.

D. OPERATIONS ANALYSIS

1. A means for the generation of the required periodic reliability
reports shall be provided. This requires a means of:
 - (a) recording "failure symptoms" (reason for shutdown) for all
centrifuges shutdown, and failure diagnosis for all recycled
centrifuges,
 - (b) determining total run hours and total hours on gas for any
centrifuge machine or subassembly,
 - (c) facilitating the determination of failure rate per machine
operating hours,
 - (d) recording the extent of damage incurred by any particular
failure,

- (e) facilitating analysis of "damage incurred" data for any number of failures,
 - (f) facilitating the analysis of effects of design changes,
 - (g) tracing any centrifuge part to its vendor and/or its drawing/revision number,
 - (h) relating cost to each failure,
 - (i) determining subassembly/machine life history,
 - (j) facilitating the retrieval of functional test variables and operating variables for any failed machine or subassembly,
 - (k) determining total run hours and gas hours since last startup for and centrifuge machine or subassembly,
 - (l) determining and recording life cycle phase during which the failure occurred,
 - (m) reporting repairs made on the machine in the process building.
2. A means of facilitating configuration management shall be provided. This requires a means of:
- (a) determining the subassembly content of each machine assembly,
 - (b) determining the component makeup of each subassembly,
 - (c) determining under which drawing/revision number each subassembly was fabricated/assembled,
 - (d) reporting effective dates of revisions,
 - (e) assisting in the evaluation of the impact of an engineering change.

3. A means of giving notice to the process personnel of machines that might be shutdown to avert imminent failures shall be provided. This requires a means of:
 - (a) analyzing the cost of the predicted failure,
 - (b) analyzing the cost of a machine shutdown, and
 - (c) determining centrifuges which contain subassemblies/ components having drawing/revision numbers or quality control log numbers similar to subassemblies/components that have failed.
4. A means to help perform special studies in a length of time that makes the studies valid shall be provided. This requires a means of:
 - (a) retrieving data and relating specific elements in the data to other specific elements in the data,
 - (b) retrieving data elements having similar values for certain items, and
 - (c) reporting the data that is retrieved in a usable format.

Considering this list of needed information, the fact that a considerable amount of the test facility data had already been collected and the fact that the data to be captured was of such a varying nature, the decision was made to authorize RPTS, a very general user-driven system. RPTS would basically allow the user to:

1. predefine a pattern for manufacturing a subassembly.
2. predefine the procedures from which information was to be stored.
3. predefine the necessary procedures to be input about a particular part-type.

4. input actual raw assembly records according to the pattern referenced in item 1.
5. input actual pieces of raw material information according to the pattern referenced in item 2.

Vast amounts of data could be captured. Studies following a subassembly from machine to machine, analyzing its operating characteristics, analyzing the failure rates of a component received from a certain vendor or manufactured using a particular process could be done. This approach places great responsibility on the user to make a skillful selection of the procedures to store. The reader should note that here the word "procedure" is used quite loosely to mean any collection of facts about an event in a component's life. The system also requires a user rather sophisticated in his knowledge of both the nature of the database content and his own discipline.

CHAPTER II

SYSTEM DEVELOPMENT CYCLE

The "Normal" System Life Cycle

A normal life cycle for a software system is depicted in Figure 2. A request for a "system study" or a "problem study" is submitted by some level of management, hereafter referred to as the "user." A feasibility study is made in which the primary objectives of the system are determined and the merit of the desired request is weighed by the user, analyst, and user's management.

With the priorities of the system defined, a preliminary system design is made by the analyst. Results from this design are reviewed with regard to hardware demands, and additional hardware needs are determined. A dollar estimate and schedule are by-products of the preliminary design. The principal product of the preliminary design is a specification document.

After all affected parties agree on the schedule, estimate, and specifications, a detailed system design can be made. In addition to the file design, system flow diagram, and program design charts, an implementation plan is developed during detailed system design. During the implementation phase, programs are developed in an orderly fashion. That is, after the first program is written, tested against the agreed on test data and its output files validated, the next program can be validated using the prior program's output files.

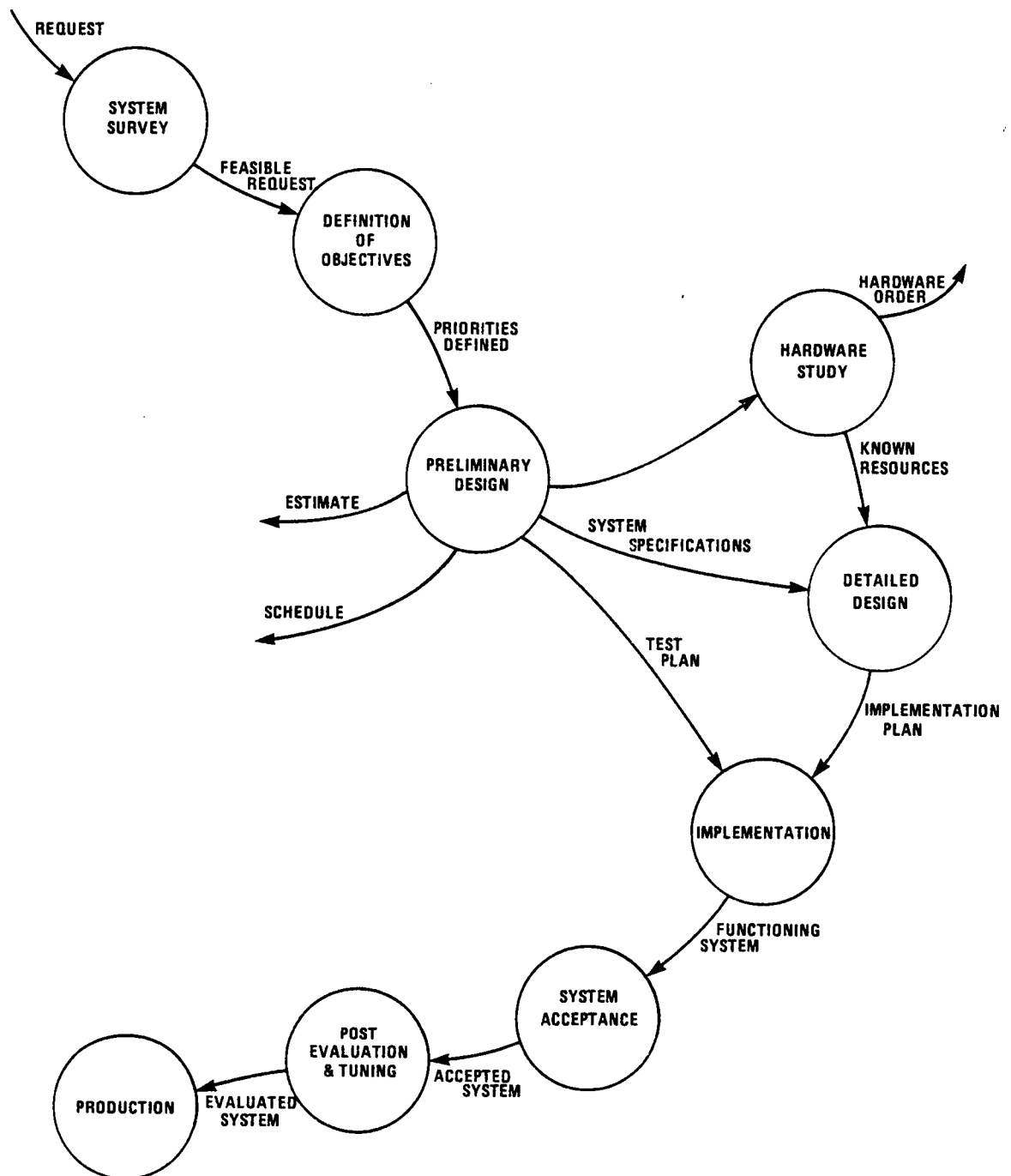


FIGURE 2. SYSTEM LIFE CYCLE

After the system is considered complete, it is necessary to prove to the user's satisfaction that the system is functional. System acceptance is a long, involved process, and depending on circumstances, it may take place along with top-down implementation. After evaluating system performance and attaining a better feel for input data timing and data magnitude, some system tuning may be in order.

After the system is accepted, it is deemed in a production state. Run procedures are written so the regular production runs can be made and monitored by a support group. The programmer/analyst would then be involved only when a system extension is requested by the user or when problems arise which are not covered by the run procedure.

According to a paper presented by Christopher P. Gane at GUIDE 43 in Chicago on November 17, 1976, there are four roles in a system's life cycle. Figure 3 and Figure 4 are taken from his paper.

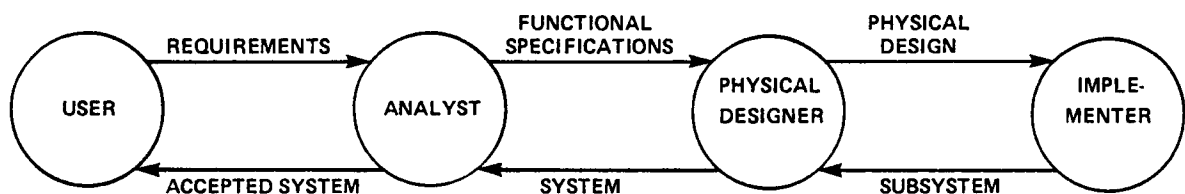


FIGURE 3. ROLES IN APPLICATION SYSTEM DEVELOPMENT

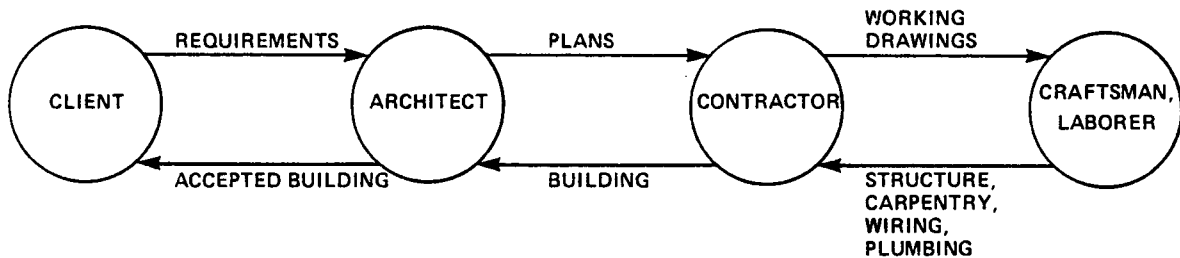


FIGURE 4. ROLES IN CONSTRUCTION BUSINESS

By following the top arrow chain we see the user stating his needs or requirements to an analyst, the analyst passing the functional specifications (inputs, outputs, conceptual schema of the user's area, listing of special users logic rules) to a physical designer who passes the physical design (data flow diagrams, program specifications, file layouts) to the implementer. The bottom arrow chain represents completed products.

C. P. Gane in Figure 4 illustrates the parallel he finds in the construction industry. To complete an office building many participants would be involved in each role. To build a small storage shed all necessary roles could be filled by one person.

Similarly, although there are four distinct roles shown, there are not necessarily four separate individuals involved in a system's development cycle. In the simplest state there might be only one individual filling all the roles. A good illustration of this might be a lone researcher developing a means of analyzing a small collection of data.

This situation has one distinct advantage in that no communication problems arise. In contrast, in the case of developing large sophisticated production data management systems many more than four individuals are involved. Communication problems arise and an overlap of role functions often exists.

Many approaches have been taken toward system development: Team user committees; the problem of who should fill the analyst role — a user based person who must learn computing, a computer oriented person who must learn the user's area, or perhaps a team analyst approach; some computing shops have a distinct regimentation of the design and implementation functions, while others feel the project is best served by having the same individual perform both tasks for a segment of the project; the programming project team concept with a chief programmer has been applied to projects. The list of system development approaches goes on and on. There is no set "correct" approach to system development. Based on the nature of the project, the capabilities of the available personnel in the user's area, the capabilities of the available personnel in the computing department, the availability of outside support services, the budget and time constraints, a system development approach takes form.

RAPTS Life Cycle

A graphic representation of the life cycle of RAPTS is depicted in Figure 5.

A request was received from Separation Systems' Component Preparation Laboratory (CPL) in 1977 to develop a system to computerize the mammoth amount of data then resident in hard copy form in the CPL data center.

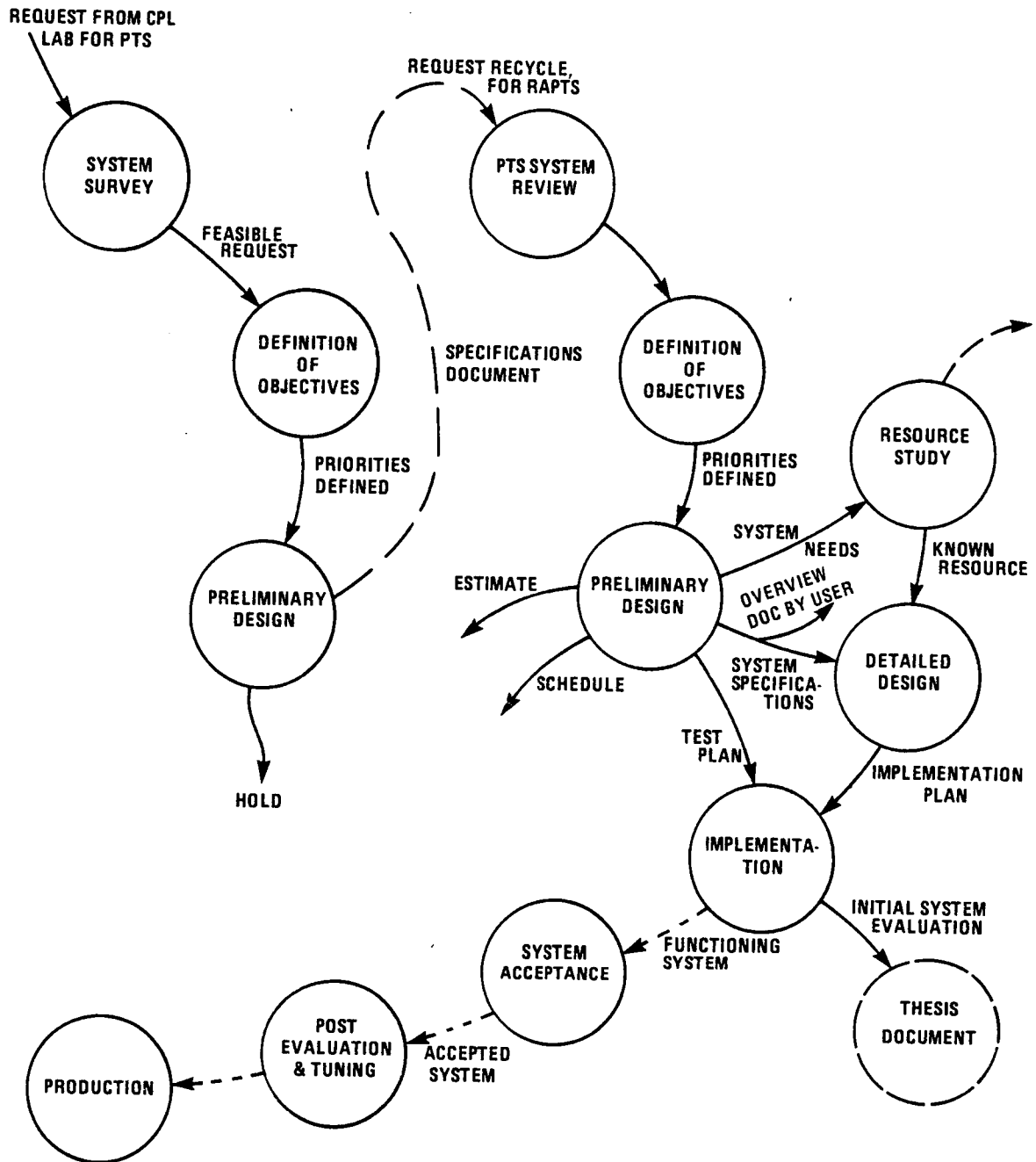


FIGURE 5. RAPT SYSTEM LIFE CYCLE

This request was reviewed in the light of known existing software. The complexity of the data was reviewed by R. E. Neir (CPL), W. F. Nolen, Computer Sciences Production Control Department Head, and the author. This study revealed two basic types of data: assembly records and procedurally obtained technical data. The technical data was of such varied form and the historical records of such volume that a preprinted specialized input form for each procedure was not practical. Past experience had shown that having a separate input form for each new procedure creates a massive program maintenance problem. The problem was approached in a manner to allow the project engineers to define a procedure external to the computer software.

Validation of data was of particular importance; therefore, a mechanism to check the base for complete entry of all serialized parts used in an assembly was merited. It was decided to let the project engineer externally define a parts "bill of material" or "subcomponent list" to the system. Assembly data about a particular serialized part would be rejected if its assembly data did not conform to its defined "bill of material." A mechanism was needed to assure that all mandatory procedures were present for a serialized part. A need was expressed to be able to follow or "trace" a part through all its initial use, its rebuilding, and all its reuses. A complete listing of a part's subcomponent makeup, both current and historical, was needed. A means of tracing a piece part through all the sub-levels to its final assembly, both current and historical, was needed. A means of retrieving a particular technical data reading about a particular part was needed. A tentative design was developed with these operations in mind.

However, because of another study beginning in the application area to develop a comprehensive, integrated database, this project was put in a hold status.

In the meantime, personnel in the Recycle/Assembly Technology group of Separation Systems had realized the need for a similar type system and in their pursuit had uncovered the Parts Traceability System document. Recycle/Assembly justified a very similar system in late 1978 to be utilized in support of a large production recycle/assembly centrifuge facility. This base, filled with historical test stand data, would provide a tool to be used in major policy decision making. The PTS system was reviewed by P. S. Roth, S. A. Herron, E. L. Newcome and the author, and minor modifications were made to service Recycle/Assembly's needs. The system was then dubbed Recycle/Assembly Parts Traceability System.

A more formal specification document was requested by the Recycle/Assembly user group. Table 1 is similar to the schedule contained in the RAPTS specification package. After reviewing the specification document, Roth and Newcome authored a paper entitled An Overview of the Computerized Recycle/Assembly Parts Traceability System that was distributed within their organization.

The Recycle Assembly group lost funding for their database project in 1979. However, the SSD Database group expressed interest in RAPTS and assumed responsibility for the systems implementation.

With the specifications approved, needed resources were reviewed. It was apparent that a large volume of keypunching would be needed.

TABLE 1
RAPTS SCHEDULE

Item	Begin Date	Target and Completion Date	Responsibility
Design Study	11-03-78	12-01-78	ELN,PSR,SAH,NED
Brief Spec Document Written		12-01-78	NED
Spec Document Reviewed and Approved		12-12-78	ELN,PSR
RAPTS Program Redesign and Coding	Item 3 Completed	5-18-79	NED
Data Analyzed for Storage Value	Item 3 Completed	5-18-79	ELN,PSR,SAH
Document Definition Data Forms Filled-In	As Item 5	5-18-79	PSR,SAH,ELN
Responsibility for Data Entry Assumed by SSD-DBA		12-01-79	NIC
Data-Item Input Gathered per Document Definition	As Item 6 Completed	ASAP	CPL Data Center Personnel
RAPTS Functional		6-01-80	NED,RKO
RAPTS Accepted		8-01-80	NIC
Final Polishing of Retrieval System and Documentation		8-01-80	NED

This problem was discussed with CSD's Keying Committee. The disk storage needs were discussed with the appropriate disk space coordinator.

Work on the detailed design of the system would continue on the assumption that the needed resources would be available. The following takes place in detail design: all files (permanent and temporary) are laid out in detail; a system data flow is derived; program overview charts are drawn; backup file considerations are serviced. In RAPTS detailed design all formats are reviewed with personnel currently microfilming the hard-copy procedure, to assure compatability between their computerized indexing system and RAPTS. This included data field lengths and standard data entry rules. A working notebook was established containing the results from the detailed design.

In order to accomplish the task within the given time frame the assistance of another programmer/analyst (R. K. Osborn) was used in implementing RAPTS. This task would have been better served if he had been available to assist in the detail design phase of RAPTS. Not only would this have saved orientation time, but a better design would have resulted from his interaction.

The initial system checkout was conducted with data compiled by the programmer/analyst. A small subset of data will be furnished by the user at a later date to further test the system. The operation of the system with this data will be used as a basis for system acceptance.

RAPTS is nearing completion of the implementation phase of its life cycle. Checkout of the system is now taking place. The user area is defining its data storage philosophy. Once this philosophy is determined and the majority of the historical data is transcribed and stored, the life cycle steps of system acceptance, tuning, and production mode can take place.

CHAPTER III

DETAIL DESIGN TERMINOLOGY

Database Terminology

Although RAPTS was not designed as a true integrated database (modeling the enterprise's total effort in terms of Bauchman charts, utilizing commercially available database packages, considering the serious problems associated with on-line processing of a true database, or striving for software and data independence), much of the currently acceptable database terminology is applicable to the more traditional design of RAPTS. The following definitions were extracted from C. J. Date's An Introduction to Database Systems to establish a vocabulary for the explanation of RAPTS's detailed design.

DATABASE: A database is a collection of stored operational data used by the application systems of some particular enterprise.

ENTERPRISE: A convenient generic term for any reasonably large-scale commercial, scientific, technical, or other operation.

OPERATIONAL DATA: Data about the operation of an enterprise. For example, data about a manufacturing company would include product data, account data, and planning data.

BASIC ENTITIES: Item about which data is recorded in the database.

Examples of basic entities in the product data of a manufacturing company's operational data would be projects, parts, suppliers, warehouses and employees.

ASSOCIATION OR RELATIONSHIP: The connection linking together two basic entities. Examples of relationships that would be found in a manufacturing companies data are:

- (a) suppliers can supply certain parts;
- (b) employee works for particular department.

This terminology can more concretely be fixed by close examination of an example from C. J. Date's An Introduction to Database Systems. In Figure 6 there are seven entries (supplies, projects, warehouses, parts, employees, locations, and departments) depicted in this enterprise's database as rectangles. Associations or relationships are depicted by connecting arrows. We know parts are made by several suppliers, and a supplier produces several types of parts. This two-way relationship is illustrated by the double arrowed pointers.

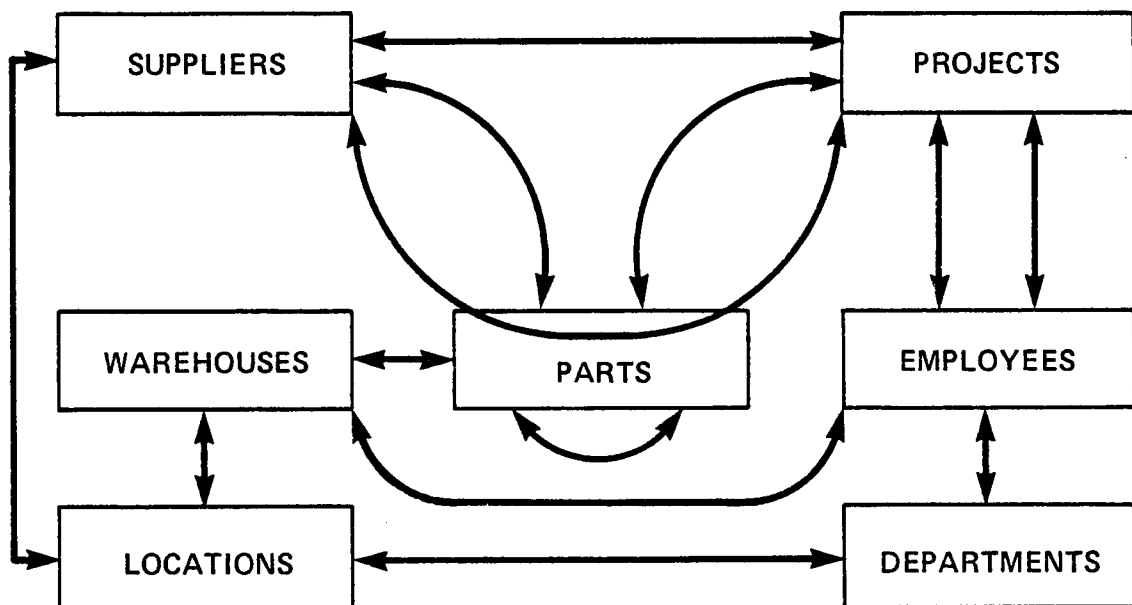


FIGURE 6. AN EXAMPLE OF OPERATIONAL DATA

The arrow between project and part illustrates certain parts are required for a project. However, there is also an arrow from supplier to project going through parts. This relationship must be depicted in order to know exactly which parts were supplied by supplier X for project Y. The arrow from part back to part illustrates the fact that one part can be a component of another part. Note that more than one relationship can exist between two entity types. The fact exists that employees work on a project, as well as the fact that one employee manages a project. Date states that a relationship can be viewed as a special type of entity. His argument follows:

However, an association between entities may itself be considered as an entity. If we take as our definition of entity "an object about which we wish to record information," then an association certainly fits the definition. For example, "part P4 is stored in warehouse W8" is an entity about which we may wish to record information, e.g., the appropriate quantity.²

Since the relationships between entities are a very important part of the operational information about the enterprise, they must be implemented into the database. There are several ways to actually represent these associations in file design: pointer, physical adjacency, etc. In RAPTS a pointer technique is used.

Other definitions from C. J. Date of interest are:

STORED FIELD: Smallest named unit of data stored in the database.

STORED RECORD: Named collection of associated stored fields.

²C. J. Date, An Introduction to Database System (Reading, Massachusetts, 1977), page 6.

STORED FILE: Named collection of all occurrences of one or more types of stored records.

Data Structure Terms

In his book, Techniques for Direct Access, Keith R. London discusses the concept of one record "owning" another record (Date's entity association concept, reference Figure 6), in terms of data structures. Definitions extracted from his narrative follow:

TREE STRUCTURE: File structure in which sub-records are "owned" by the master record. These in turn can "own" further sub-records. One record may own a number of records (possibly none), but each record has only one owner (except the master owner has no owner).

NETWORK STRUCTURE: This means that a record can own a number of other records, and that a record can have a number of owners.

London also discusses three methods for cross-referencing data: tree or network chaining, chain-modified ring structure, and lists. Definitions follow:

CHAIN: A technique for linking logically related records using pointers.

A pointer is a special reference field incorporated into a record which contains a reference to the next logically related record. The pointer of the last record on the chain contains a special End-of-Chain marker. (Note: Other texts refer to this as a linked list.)

CHAIN-MODIFIED RING STRUCTURE (RING): This is a chain which ends not with an end-of-chain marker but with a pointer back to the first record of the chain. This has the distinct advantage that processing may start on any record on the chain and all records will be accessed. (Note: Other texts refer to this as a circularly linked list.)

SIMPLE LIST: An indexing system used when the length of chain through the file is restrictive. The chain is fragmented into a number of shorter ones. Each fragment has its entry point stored in a table of entry. The table of entry forms the list. An example is cited: Supplier/Product data file is chained by product number. Information on a particular product of a particular supplier is needed. An entry for only the first data entity of a supplier is maintained in an index. Subsequent product codes can be reached by walking the chain in the actual database. This limits the number of accesses to the base to, in the worst case, one plus the number of codes on the product chain. (Note: Other texts refer to this as a non-dense index.)

INVERTED LIST: No pointers are maintained within the actual data file. The list contains a list entry for each entry in the file. Associations between records are contained in the list. An example cited supplier codes and their associated product codes, multiple supplier codes (one per product) are in the Inverted list. An illustration of this versus a simple index is Figure 7, taken from London's text. (Note: an inverted index is also referred to as a dense index in other texts.)

Supplier Code	Prime Key (Product Code)
001	1346
002	0056
003	0756
004	3452
005	0785
006	2345
007	4545
008	1008
009	9672
010	0787
011	5845

A Simple List

In this example there is one entry for each supplier. Against each supplier code there is in the product number the first record in the chain for that supplier. The records in the file are chained by supplier.

Supplier Code	Product Codes
001	1346
	1348
	1447
	1895
	1990
002	3076
	3087
	7569
	9981
	0056
	1056
	1347
	2445
	2678
	3879
003	4778
	6799
	9072
	8756
	8757
004	8760
	3452
	3453
	3454
	3455
	3456
	3457

In this example there is one entry for each stock record. The product codes are in sequence within supplier. No pointers are used in the actual data records. All linkages are via the list.

An Inverted List:Stock File

FIGURE 7. DATA ENTRY LIST

IBM/360 Jargon

Some basic IBM System/360 definitions should be stated. The source for the following definitions is an IBM student text Introduction to IBM Direct-Access Storage Defices and Organization Methods.

DIRECT ACCESS STORAGE DEVICES (DASD): A device on which each physical record has a discrete location and a unique address. Thus records can be stored on a DASD in such a way that the location of any one record can be determined without extensive searching. Records can be accessed directly as well as serially.

FILE: An organized collection of related information. It may occupy several physical units or part of one physical unit. Sometimes the term "dataset" is used instead of "file."

LOGICAL RECORD: A logical record can be defined as a collection of data related to a common identifier.

PHYSICAL RECORD (BLOCK): Usually consists of one or more logical records. Also referred to as a page or the amount of records brought into core by one input/output operation (IO). This is also referenced as buffer size.

KEY: Each logical record contains a control field or key that uniquely identifies it.

DIRECT DATASET ORGANIZATION: The record within the dataset, which must be on a direct-access volume, may be organized in any manner you choose. All space allocated to the data set is available for data records. No space is required for indexes. Records are stored and retrieved directly by address. Direct data sets are created by using special Basic Sequential Access Method (BSAM) macros. Records can be accessed by using the Basic Direct Access Method (BDAM).

RELATIVE BLOCK ADDRESS: Way of referring to a record in a direct file in which the user supplies the system with a binary number that indicates the position of a block in relation to the first block of a data set.

CHAPTER IV

RAPTS DETAIL DESIGN

Conceptual Model

Basic entities present in the conceptual model for RAPTS are:

1. Serialized part - unit of assembly having a unique identity. It is composed of a part-type code and the serial number stamped on the part. Both a piece part and the highest level assembly are considered parts. The terms "part," "serialized part" and "assembly" are used interchangeably. The word "subassembly" refers to a part at a stage in its life when it is a subcomponent of another part.
2. Procedure - an event during the life history of a part around which data is recorded. Procedures may be in the form of a record of assembly, certifications, inspections, special material composition, operating records, etc.
3. Procedure Variable - one of several data entries taken from a procedure. From a certification paper, examples of procedure variables would be date of certification, person doing certification test, performance reading, etc.

Relationships or associations present in the model are:

1. One part can be composed of several other parts (subassembly makeup).
2. A rebuild assembly has had a previous subassembly makeup.
3. Certain events (procedures) have occurred during a part's life.
4. Certain data entries (procedure variables) have been recorded about these events.

These entities and relationships are depicted in Figure 8.

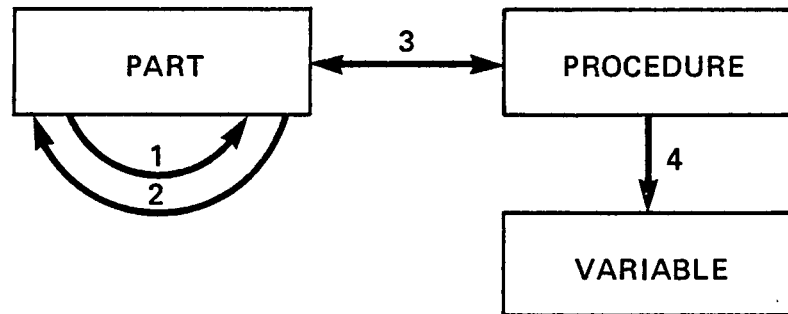


FIGURE 8. CONCEPTUAL MODEL FOR RAPTS

Using Date's graphic representation, in which rectangles are entities and arrows are relationships, we see a one-to-many relationship between a part and its subassembly makeup (relationship 1). Relationship 1 also implies that a part which has subcomponents may also be a subcomponent to another part. Relationship 2 is also a one-to-many relationship between the current part assembly record and all its prior assembly records. Relationship 3 is many-to-many, implying that a part has many procedures and that a particular procedure is used on many parts. Relationship 4 is one-to-many in that a procedure variable is recorded from one procedure, although a procedure may have many variables.

Figure 9 more closely depicts the logical conceptual schema.

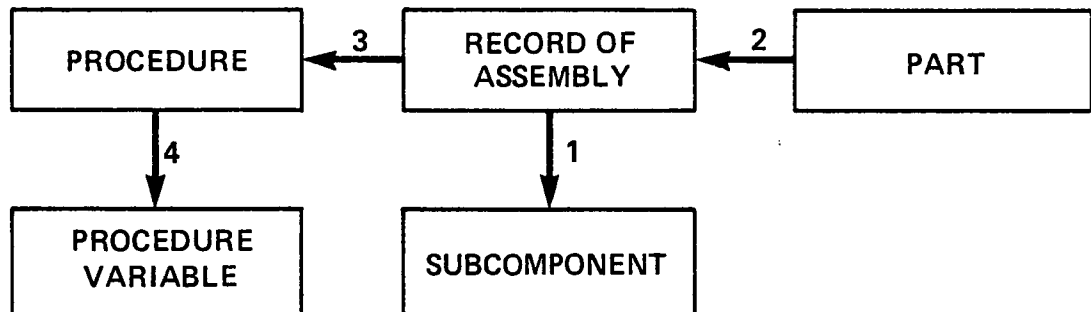


FIGURE 9. CONCEPTUAL SCHEMA FOR RAPT

The arrows here again represent the previously defined relationships. The difference is that here they have been reduced to one-to-many relationships. One-to-many relationships are implementable. They were reduced by entering two new rectangles, Record-of-Assembly and Subcomponent. Part, Record-of-Assembly and Subcomponent are needed in this logical conceptual schema to represent the basic entity Part.

If we assume that the relationships in Figure 9 are represented in our data structure as chains, then we can clearly see that, if chain 2 is traversed for a particular part, then a record of all its historical builds could be obtained. By walking or traversing each chain 1, the subcomponent makeup at the time of a particular build could be obtained. A traversal of chain 3 gives all recorded procedures during that particular assembly build life. Walking chain 4 for a particular procedure gives all data readings (procedure variables) recorded about that event (procedured).

It should be noted that not all relationships exist for each entity. That is, a piece part does not have a chain 1 (its chain 1 is null). By the same token, different relationships exist at different phases of a part's life. To more graphically depict this, let us examine the chain of events necessary to develop the small data network depicted in Figure 10 through Figure 12. Our basic entities are depicted as A_x for serialized parts, P_x for procedures, and V_x for procedure variables. Our relationships are depicted as "____" for relationship 1, composition; "~~xxx~~" for relationship 2, historical build chain; "~~///~~" for relationship 3, procedure chain; and "~~∇~~" for relationship 4, procedure data chain.

The following sequence of events leads to the sample data network.

1. First Part A_3 was shipped using Procedure P_1 to do receiving inspection tests.
2. As P_1 event occurred, variables V_1 , V_2 , and V_3 were recorded from Procedure P_1 ;
3. Procedure P_2 was recorded as a certification record on A_3 ;

Figure 10 illustrates these initial/first events.

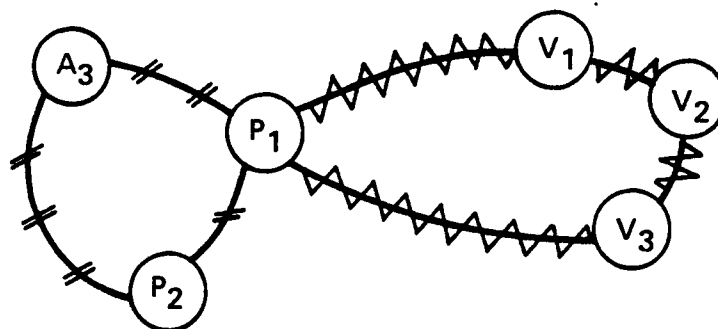


FIGURE 10. DATA NETWORK I

Note that no relationship 1 exists for A_3 since it is a piece part. To continue our data sequence of events:

4. Part A_1 was assembled using A_2 and A_3 ;
5. Part A_4 was assembled using A_5 , A_6 , and A_7 ;
6. Part A_8 was assembled using A_1 and A_4 ;

Figure 11 illustrates through step 6. In reality the chain of events would be much more complex with many more procedures and subassemblies.

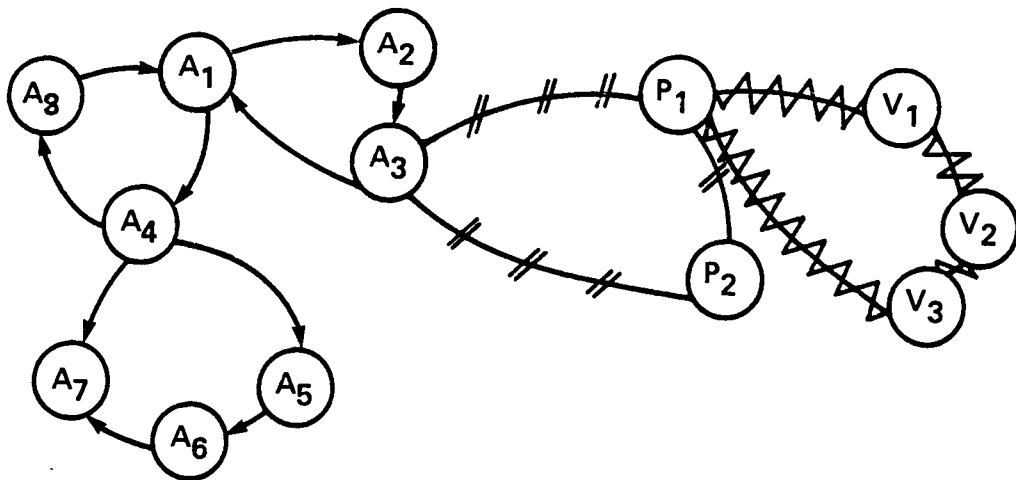


FIGURE 11. DATA NETWORK II

Continuing with our illustration:

7. Part A_8 was torn down and reassembled using A_9 and A_{10} ;
8. Procedure P_1 was used to assemble A_8 's rebuild.
9. V_1 , V_2 , and V_3 were recorded from A_8 's P_1 ;

10. Procedure P_{10} was used to perform certification;
11. Procedure P_4 was used to do a follow-up inspection;
12. Variables V_1 , V_2 , V_3 , and V_4 were recorded from A_8 's P_4 .

Figure 12 illustrates through step 12. All of the database relationships are present. Note that Relationship 2 (historical assemblies) necessitates two part entity occurrences labeled A_8 . In actual implementation this apparent duplication is also present. That is, multiple records having A_8 as a key are stored. However, after consideration this may not be a duplication since a part's existence ends at the time of tear down, and its reappearance (storage) with a new makeup is truly a new part.

Overview of the Physical Design of the System

RAPTS is being implemented as a network type structure. In this system, seven relationships are implemented as chain-modified ring structures. These relationships are:

1. listing of all part types necessary to build an assembly.
2. listing of all procedures that should have data recorded for a part type.
3. listing of all procedure variables to be recorded from a procedure.
4. listing of serialized parts used to build another serialized part.
5. listing of procedures that have data recorded for a serialized part.
6. listing of procedure variable values recorded about a procedure for a particular serialized part.
7. listing of historical builds for a particular serialized part.

Relationships 1, 2, and 3 define a pattern or blueprint necessary for complete data storage. Relationships 4, 5, 6, and 7 are recorded about a particular instance of assembly. Relationships 1 and 4, relationships 2 and 5, and

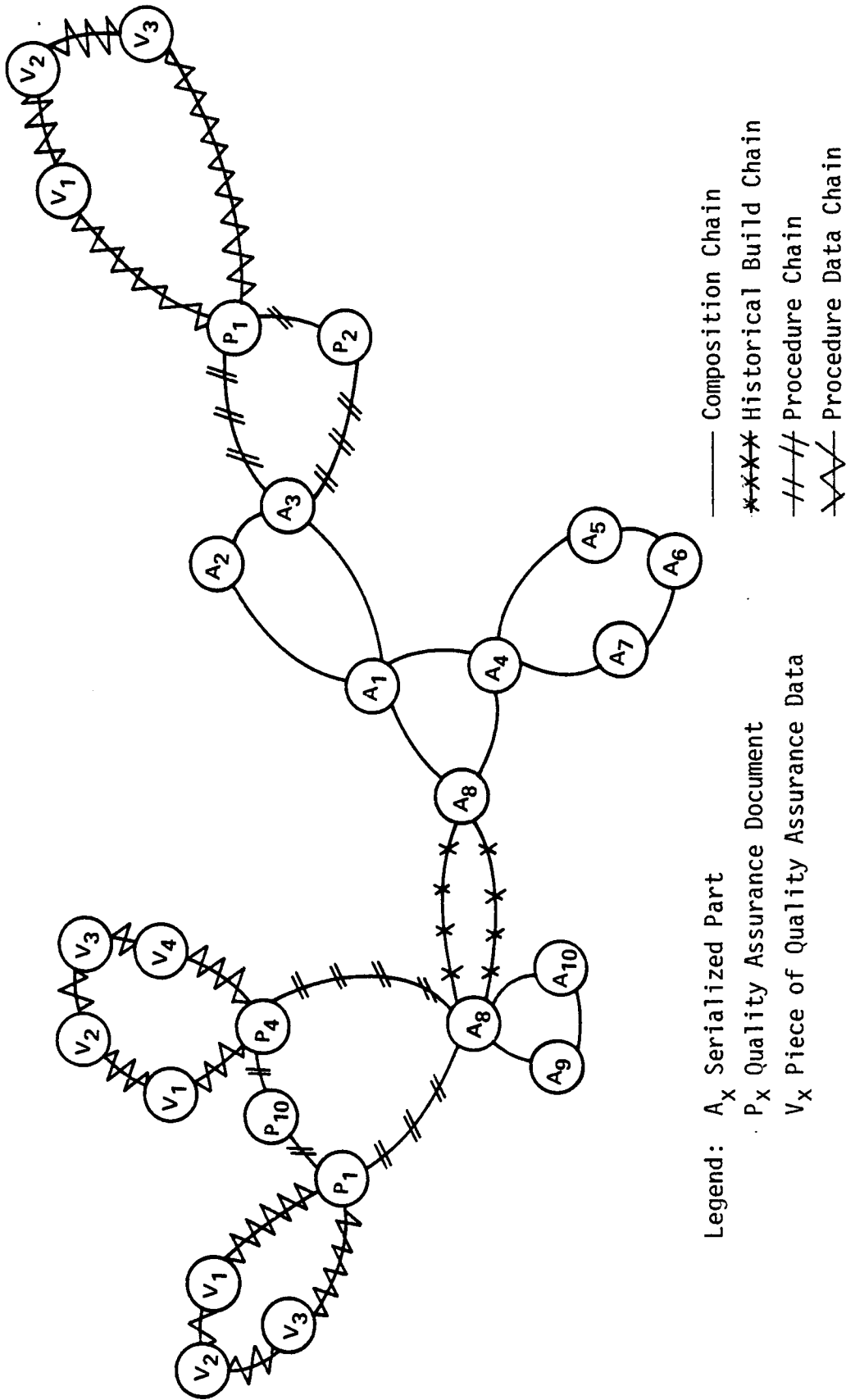


FIGURE 12. SAMPLE DATA NETWORK

relationships 3 and 6 are compared to assure complete data storage for the serialized part as defined in the pattern relationships.

Relationship 1 is maintained in a physical file referred to as the Assembly-Subassembly file. Relationship 2 is maintained in a physical file referred to as the Assembly-Procedure file. Relationship 3 is maintained in a physical file referred to as the Procedure-Variable file. Relationships 4 and 7 are maintained in the Assembly-Master file. Relationship 5 ties physical files Assembly-Master and Procedure-Master together. Relationship 6 is maintained in physical file Procedure-Master.

There are eleven source programs in the RAPTS computer system. They can be broken into four categories: edits, updates, reports, and utilities. Edits examine pieces of input data, checking for valid data entries in certain fields. Edits also format acceptable input data into a form to be used by an update program. Data that does not pass edit tests is written on an error report to be examined by the data entry technicians. An update program uses the output from an edit program. An update has the capability to add a data entry, correct a data entry, or delete a data entry. Report programs extract certain pieces of information from the files for distribution to customers. Utility programs enable smooth system function by reading a file and writing a backup tape, reading a backup tape and rebuilding a system file, and writing and creating an empty file with the proper space allocation and index entries. The programs in the RAPTS system is depicted in Figure 13, RAPTS's System Overview Chart.

As implied from the previous discussion, for the system to function properly the following occurs: data is passed from one program to another on temporary files; permanent files are updated; written reports are

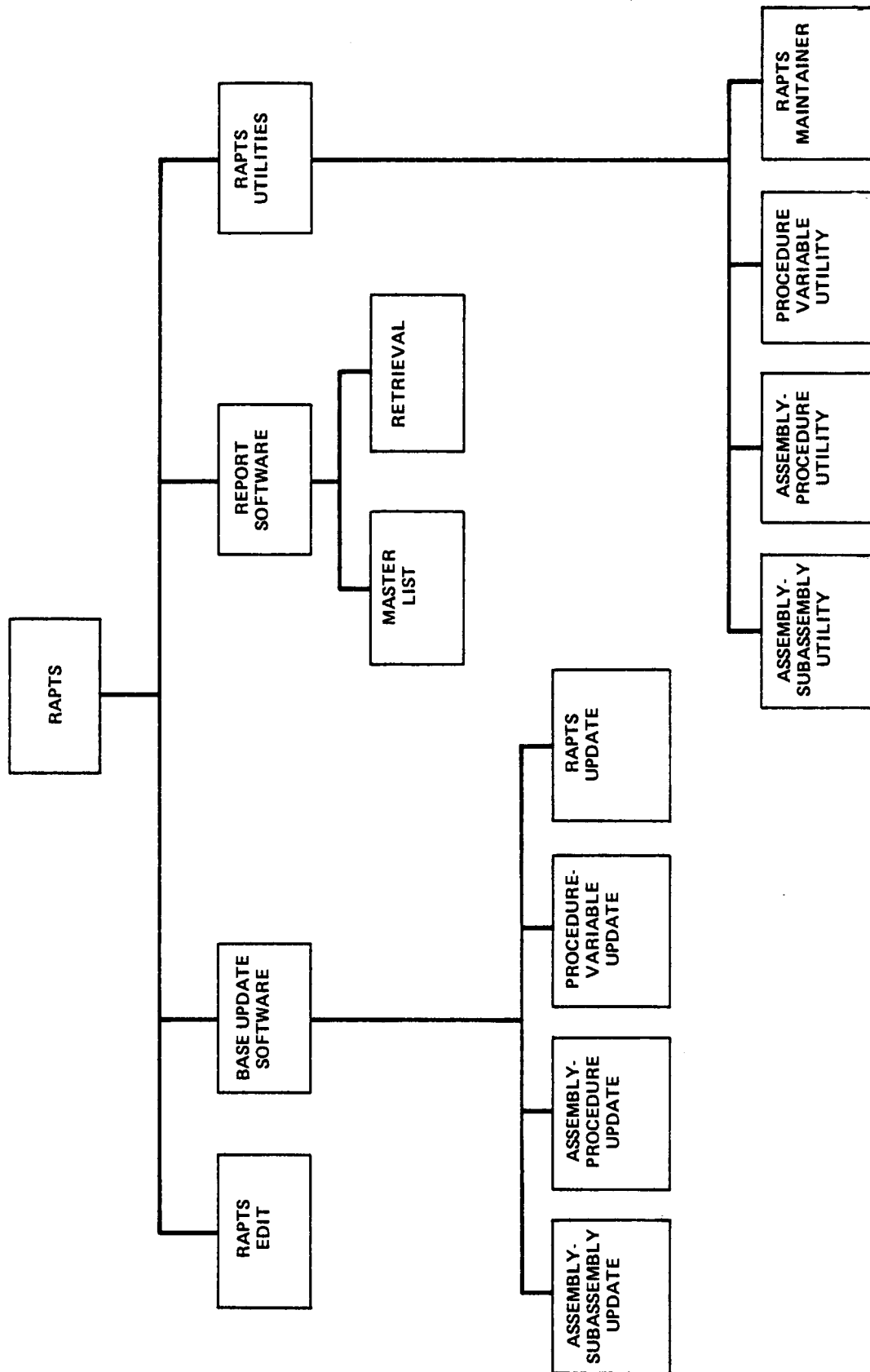


FIGURE 13. RAPTS SYSTEM OVERVIEW CHART

produced; and utility programs must be executed periodically to create backup files as a safeguard. The true data flow for RAPTS is depicted in Figure 14. Another type permanent file, the incomplete file, was not previously mentioned but is depicted in Figure 14. The incomplete file contains data that has passed edit rules but, because of the state of the data, cannot be stored on a permanent file. An example of a situation causing an entry on an incomplete file is a procedure-variable record for a serialized part that has not been identified in the assembly master. Data is maintained on the incomplete file for a predetermined number of system update cycles before being discarded as data in error.

Let us examine the RAPTS System flow diagram more carefully. The RAPTS Edit Program checks data from the Entrex tape for syntax errors. Data items in error are listed. A tape containing "incomplete" data, data that could not be stored because of the system's state during the previous update, is also read by the edit. Data on the incomplete file is aged for ten system cycles before being purged. A listing of the purged incomplete data is produced. Data items input from the "incomplete" tape and edited Entrex data items are passed to one of the four appropriate update programs via temporary files.

Raw data entered via the "SSDASM" input form with zeroes in the serial number is passed by the edit to the RAPTS Assembly-Subassembly Update. This system updates the Assembly-Subassembly Frame file that is later used in RAPTS for assembly input validation.

Data items associating with a part type and its needed procedure, entered via a "PROFLD" form with no field definition entry, are passed to

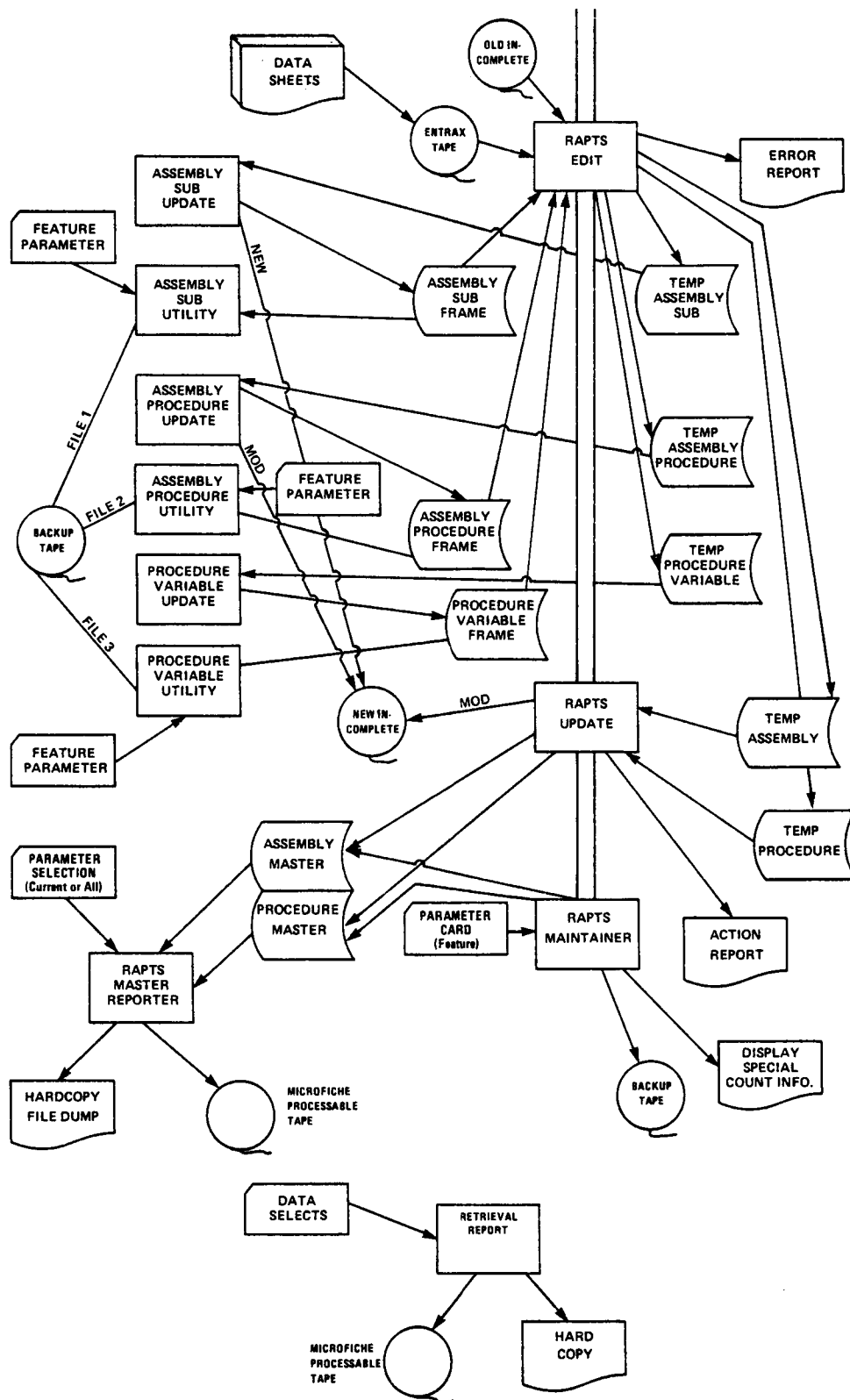


FIGURE 14. RAPTS DATA FLOW CHART

the RAPTS Assembly Procedure Update. This program updates the Assembly-Procedure Frame file. Before data storage, edits are made to insure that the procedure is defined in the Procedure Variable Frame Definition file and that the assembly type has been defined in the Assembly-Subassembly frame file. Data not passing these edits is placed on the incomplete file.

Procedure definitions data is entered to the Procedure Variable Update via a temporary disk file from the edit. This data was initially submitted on the "PROFLD" form.

The true raw data items, (data about serialized parts), entered via "PTSDTA" and "SSDASM" forms, are routed by the edit to the update. A report of the data modifications made by the update is produced. Data items not properly defined according to the Assembly-Subassembly file definition are rejected. Procedure data about a serialized assembly not found in the base are placed on the incomplete file and a message is printed. Procedure data not conforming to the Procedure-Variable frame definition is rejected. Data from a procedure not defined in the Procedure-Variable Definition File is put on the incomplete file, and a message is printed.

A complete listing of the RAPTS Master file is available via a report program. However, this feature is costly. Total database dumping is anticipated only to be used initially as an aid in customer acceptance of the system.

Random retrievals along predefined paths will be serviced via another program. This feature is expected to service the day-to-day needs in lieu of complete file lists.

Four utility programs, one paralleling each update, exist. The utility functions of "backup," "load," and "rebuild" are available for each file. Storage analysis figures to aid in scheduling reorganization are printed each time the "backup" function is performed. Utility programs in RAPTS are the Assembly-Subassembly Utility, the Procedure Utility, the Procedure Variable Utility and the RAPTS Master Utility.

It should be noted that the frame file utilities have a "list" utility feature which prints out their respective files' contents. This hardcopy listing will be used as the guide for input "PTSDTA" and "SSDASM" data to the system.

Detailed File Design

The five permanent files in RAPTS have direct dataset organization. The three frame files use a relative track addressing scheme. The two master transaction data files use a relative record addressing scheme.

In a relative track addressing scheme, the tracks of the direct access device are numbered from zero to n. The placement of a record is determined by a key value. The first four bytes of the key are the relative track address. The remainder of the key is the true-ID of the record. When a store command is executed, the designated track is searched for available space to store the record. If space is not available on the designated track, the track number portion of the key is incremented and the process continued. For data retrieval the process is reversed; the track specified by the first four bytes of the key is searched for a record having an ID equal to the key-ID. If the record is not found, the search continues with the next track.

The recording mode for the frame files is variable. Each of these variable length records consists of a fixed portion of primary information and a varying number of occurrences of secondary information.

As shown in Figure 15, the Assembly-Subassembly framed file record is composed of a delete flag, machine group number, machine model code, part type, revision status switch, build sequence number, effective date for

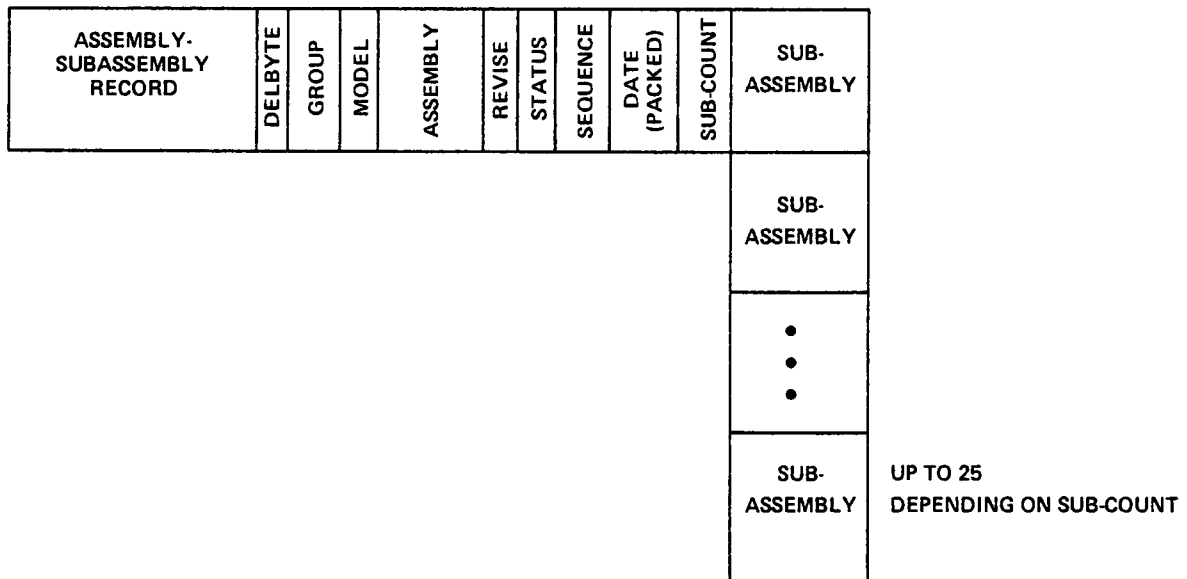


FIGURE 15. ASSEMBLY SUBASSEMBLY RECORD LAYOUT

build definition, a count of the number of subcomponents defined in the build, and a series of defined subcomponent part types. The ID-key for the file consists of group, model, part type, revision, status, and sequence. The variable portion (subcomponent part types) can occur a maximum of twenty-five times. The status switch is set to "C" if it is the currently defined build of the assembly build definition. A status switch of "0" indicates a historical build definition. The sequence number is "00" for the current build. A historical build's sequence number indicates the number of prior builds that have been defined.

A simple randomizing routine is used to select the track on which to store a record. The key field (alphanumeric) is moved to a computational field. This packs (removes zone fields) and converts to binary notation. The results is numeric. This number is then divided by a prime number. The remainder from this division is used as the relative track number. The prime number used as the denominator is the prime number closest to the estimated number of tracks needed for storage capacity. A capacity packing factor of 80 percent to 85 percent is suggested to be used for figuring needed space.³ Even with the best randomizing routines, records will randomize to the same track. This 20 percent of unused space is a sacrifice to offset the excessive IO that would occur if several tracks had to be searched to find an empty storage location. A multiple read (IO) situation would also occur in retrieving a record not stored at its randomized track address.

³"Direct File Processing", IBM System/360 Operating System Full American National Standard COBOL Compiler and Library, Version 3 Programmer's Guide (New York, 1972), page 87.

Figure 16 shows the record layout for the Assembly Procedure Frame file record layout. This variable length record has identification information about a part type in its fixed portion (group, model, part type, status, sequence, effective date and time, number of defined procedures). The variable portion contains a listing of documents valid for the part type defined in the fixed portion. The delete flag is used for file maintenance purposes. The ID-key is composed of group, model, part type, revision, status, and sequence.

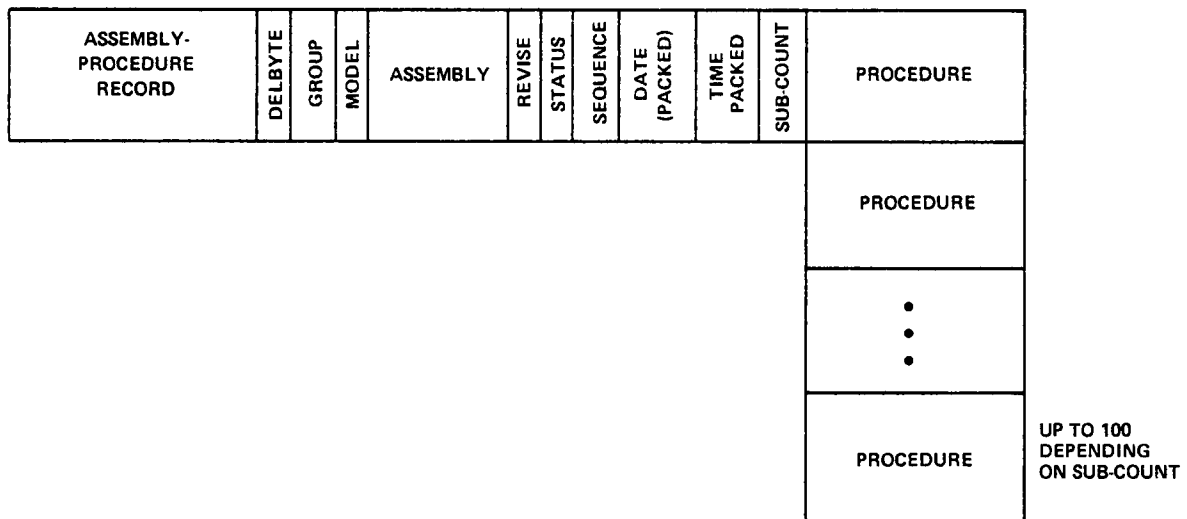


FIGURE 16. ASSEMBLY PROCEDURE FRAME FILE

The comments concerning relative track addressing and randomizing routines made about the Assembly Subassembly Frame file also apply to the Assembly Procedure Frame file and the Procedure Variable Frame file depicted in Figure 17. The concept of the status switch, sequence number, and delete flag also applies to all three files.

The Procedure Variable Frame file consists of a fixed portion describing the data items to be entered from that procedure. Data items on the fixed portion are follow sheet (sheet used to collect data), revision, group-model-part type document defined for, status, sequence, effective date and time, type document, originator of document, an associated procedure number, procedure title, and the number of variables to be recorded. The variable portion data items are variable number, number of readings, recording mode, type reading, number of significant digits, unit of measure, variable definition, and entry form sequence.

The two permanent transaction files in RAPTS have direct data set organization and a relative record addressing scheme. In a relative record addressing scheme the position of a record is determined relative to the first record on the file (record zero). Both files are composed of fixed length records. Since blocked records are not supported by this file processing technique, RAPTS operates in a mock blocked mode using a paging technique.

The first few records on each file contain a directory to the file content. When read into an array they form a SIMPLE LIST data structure. An item can be found in the list and the associated pointer followed to find the piece of data needed.

The pointers used in RAPTS have the format "BBBBBRRRR" where BBBBB represents the Relative Block Address of the record and RRR represents the Logical Record Occurrence number within the Relative Block Address.

DELBYTE	PROCEDURE				ASSEMBLY	STATUS	SEQUENCE	DATE (PACKED)	TIME (PACKED)	TYPE DOC	ORIGINATOR	FOLLOW SHEET	PROCEDURE TITLE	SUB-COUNT
---------	-----------	--	--	--	----------	--------	----------	---------------	---------------	----------	------------	--------------	-----------------	-----------

							VARIABLE DEFINITION
• • •							
							VARIABLE DEFINITION
VAR NO.	OCCURRENCES	NODC	TYPE	SIGFIGS	UNIT OF MEAS.	VARIABLE DEFINITION	ENTRY FORM NUM.

UP TO 100
DEPENDENT ON SUB-COUNT

FIGURE 17. PROCEDURE VARIABLE FRAME RECORD LAYOUT

The Assembly Master file has only one record type. However, several relations are carried on this one type. Pointers present are: the "owner" pointer (points to the part of which this part is a component); the "next same level" pointer (points to the next subcomponent of this part's owner); the "first lower" pointer (points to the first subcomponent of this part); the "Previous History" pointer (points to a previous build header for this part); and the "First Procedure Header" pointer (points to the procedure header in the procedure file associated with this part. Figure 18 illustrates the Assembly Master file layout. Only the current assemblies are listed in the index. Other builds are found by following the history chain. Two status flags are present in the index; these are set to zero (incomplete) or one (complete). One flag indicates whether all subassemblies for the build, as defined in the Part-Subassembly file, are present; the other indicates whether all procedures, as defined in Assembly-Procedure file, are present for the build.

Several special purpose entries are maintained in the index. Part-type "000000" has the address of the next free space in its pointer. Part-type LOST's pointer contains the number of unreachable records in the file. Part type MAXR's pointer contains the count of allocated records.

The Procedures/Variable file contains two records types. One header record is present for each procedure recorded about a serialized part. A detail record is present for each variable reading. This record is variable length because of the unpredictable length of the data variable reading. The pointer for the file is BBBBCCCC where BBBB is the Relative Block Address for the record and CCCC is the beginning character for the record within the Block. Figure 19 illustrates the structure of the Procedure/Variable Master. The pointer "next header" points to the location

GROUP	SERIAL	VENDOR	MODEL REV	DATE ASSEMBLED YYMMDD	OWNER POINTER	NEXT SAME LEVEL	1ST LOWER	PREV HISTORY	FIRST PROCE- DURE HEADER	STATUS
-------	--------	--------	--------------	-----------------------------	------------------	-----------------------	--------------	-----------------	-----------------------------------	--------

LEGEND:

/// → *** ~~~~~ ○○○

DETAIL RECORD

GROUP	SERIAL	POINTER	EDIT STATUS	EDIT STATUS
-------	--------	---------	-------------	-------------

INDEX

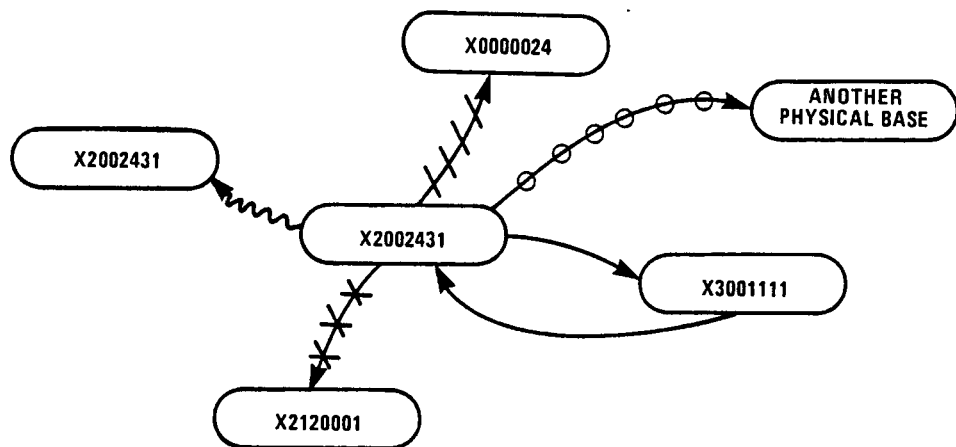


FIGURE 18. ASSEMBLY TRANSACTION MASTER FILE LAYOUT

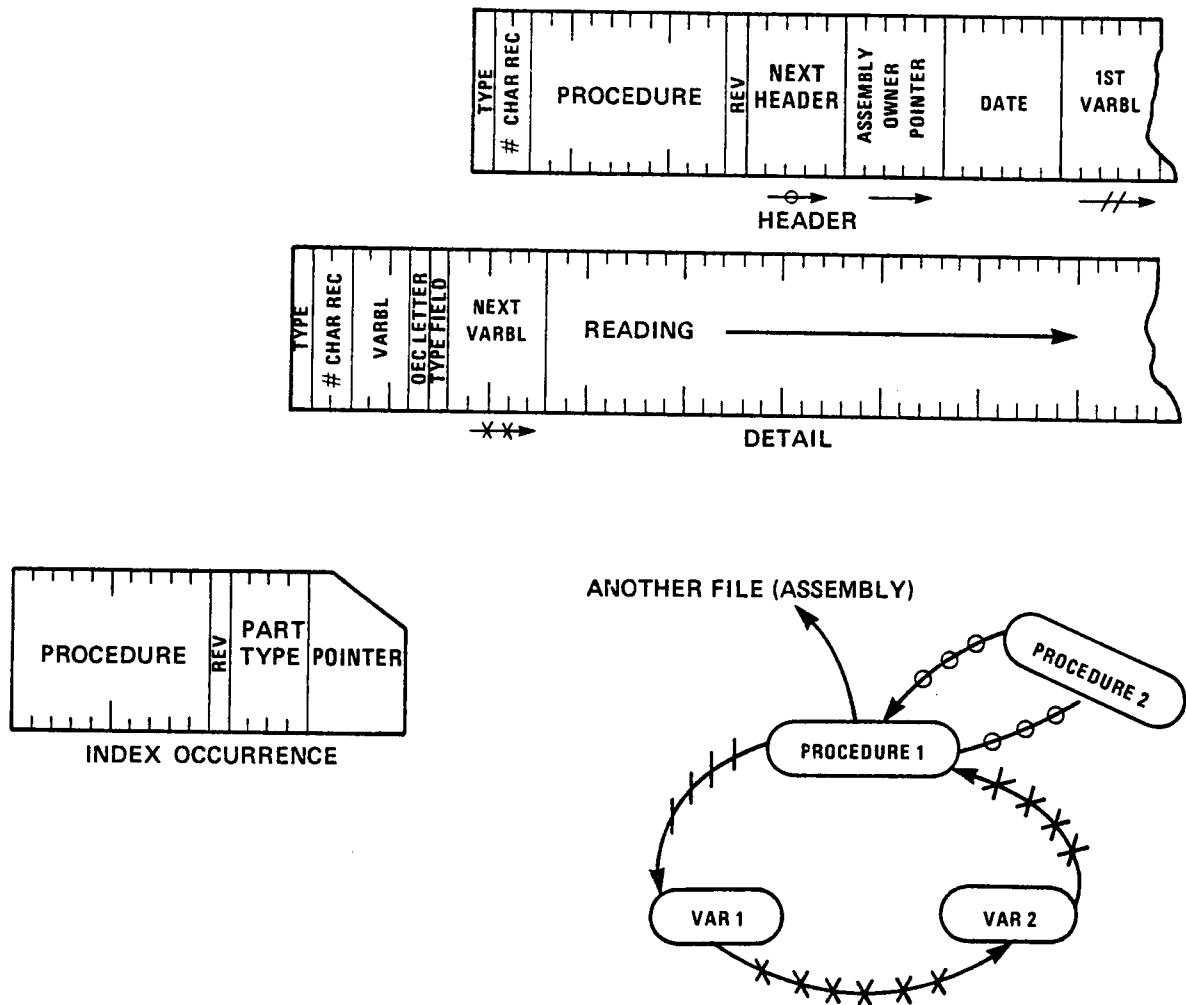


FIGURE 19. PROCEDURE VARIABLE TRANSACTOR MASTER FILE LAYOUT

of the next header record in the block. The pointer "Assembly Owner Pointer" points to the address of the serialized part in the Assembly Master file. The "First Variable" pointer points to the location of the first variable recorded from the procedure. The detail record contains a "next pointer" to the next variable from the procedure or back to the header record. Special purpose pointers are also present in this index. Each procedure header has an entry in the index. The number of characters in the record is given or recorded in the "# Characters" field in the header and detail record.

The format of the intermediate files will not be covered to any extent. Each file is a variable length sequential file. The incomplete file's format is also variable sequential. A purge count field is included in this record.

In blocking the online files, the Relative Block Size is as close to the maximum track size for a 3330 as possible.

Input Form Design

Input forms were needed to serve five purposes:

1. To establish a pattern or listing of subcomponents necessary when entering an assembly transaction.
2. To define the data items to be collected from procedures.
3. To define procedures which are valid to be recorded about a part-type.
4. To input assembly transactions.
5. To input data items collected via procedure follow sheets.

These purposes are serviced using three input forms.

The RAASM form found in Figure 20 is used to establish the pattern of subcomponents to be input about an assembly transaction. For this purpose the header information is input describing the assembly. This header information includes machine design group, machine model, part type, effective data and effective time. The assembly serial is left blank. The part types of the necessary subcomponents are input on the lower half of the form. Subassembly serial numbers are also left blank. The date and time fields on the RAASM form are filled with the date and time after which an assembly transaction must conform to this pattern.

The RAASM form is also used for the purpose of entering assembly transaction data. For this purpose all header fields are entered. The date and time reflect the time of the assembly action. The part-type and serial number of all the subcomponents used in the assembly process are entered on the lower portion of the RAASM form.

The RADEF form shown in Figure 21 is used to define the data items to be collected from a procedure. It should be noted here that the term procedure is used very loosely and could mean any document used to record an event in a part's manufacture, inspection, procurement, or operation.

The header information defines the procedure or document used to collect the data items. The header fields include the follow sheet number or number of sheet to be used for recording data; the revision letter of the follow sheet; the date and time this definition becomes effective for collection purposes; the machine design group, model and part-type for which this definition applies; the type document code (e.g., "01" - inspection, "02" - assembly procedure); the code of the originating source of the document (e.g., "UC" - Union Carbide, "AR" - AiResearch); an associated procedure or document. Note that the procedure number or document number could be used as the primary number (follow sheet number) and the secondary number (procedure) left blank.

The body of RADEF form contains a series of information lines about data items. The pieces of information to be recorded about a data item includes: variable identification number; the number of readings of this type to be recorded (e.g., "2" if measure of diameter is required to be taken twice); the mode of recording data (e.g., "T" - text, "E" - Exponential); type reading (e.g., "O" - Optional, "M" - Mandatory); Digits (number of

significant digits of reading if numeric); unit of measure if applicable; the variable title or definition; and the entry order number on the input form. The entry order number would be "001" for the first data item to be entered on the RADTA form. The entry order number would be "024" for the twenty-fourth item entered on the sheet. This arrangement allows the order of data item entry to vary from procedure definition to definition. That is, an item not defined on procedure revision A (e.g., Variable 423) could be entered on procedure revision B as entry order number "002". By establishing a listing of unique variable number across procedures, the same item can be entered on multiple procedures, yet the fact that the reading is of the same thing is maintained.

The RADTA form shown in Figure 22 is used to define the procedures to be collected about a part type and to record actual data.

To record the valid procedures about a part type, the document number, group, model, and part type are entered. The serial is left blank. No data items are entered.

To input transaction data, the document, revision, group, model, part type, serial, and the date and time of recording must be entered on the RADTA form. Data items are entered in the order defined on the RADEF form. Each data item is separated by a "/". If a data item is missing, the positional slash is still entered. All fractions are converted to their decimal equivalent before entry. A maximum of 950 characters of data point information is allowed per entry. Data entry continues from line to line, ending with the characters "&&".

All three forms support the normal input action codes add and delete. A special action code "K" is allowed on the RADTA form. This allows a particular data item from a procedure to be deleted.

These data forms were designed to be keyed on a card-to-tape system (Entrex) having a maximum record length of 1000 characters. This maximum did not seem to present a problem in reviewing the data to be stored.

Software Design

The programs making up the RAPTS and the data flow through the system was discussed in Chapter IV, "Overview of the Physical Design of the System." As depicted in Figure 13, page 39, eleven programs make up the RAPTS software. These programs can be categorized as one edit program, four update programs, two report programs, and four utility programs.

Each program has been coded in a structured manner. All programs were coded using COBOL. Computer Science Management Systems' guidelines were followed.

R. K. Osborn was the programmer for all software composing RAPTS. Basic structure charts were composed for each program by the author. These charts were further refined by Osborn in developing the actual code. Osborn also contributed to the design concept of the system. The author had originally intended to use a relative record addressing scheme in processing the frame files as well as the transaction files. By changing to this mode the coding was made simpler, and less time and possibly fewer IOs would be necessary to locate the frame record. Core requirements for the update would have been much greater if the indexes for each frame file were maintained in core.

This approach did force limits to be placed on the number of subcomponents making up an assembly, the number of variables defined about a procedure, and the number of legal procedures about a part type. However, these limits did not prove troublesome. This approach also brought the subject of randomizing routines to light in the database system.

During the development process, a limitation of variable length records and relative track addressing was discovered. Once a record is stored, the length of this record cannot be increased in a modification operation. The record must be marked for deletion and the record written at another location. This information was not documented in any readily available literature, and caused some additional work in the frame updates.

The steps taken in developing the software were:

1. Developing Structure Charts to show overall intent of the program.
2. Supplementing portions of the chart with pseudo code, flow charts, or decision tables to further solidify logic.

An example of one of the simpler structure charts is found in Figure 23. Structure charts differ from traditional flow charts in the level of detail depicted. The structure chart is concerned with the major functions to be accomplished. Some order of functions is implied by the placement of the functions on the chart (because of our training we traditionally view the chart from left to right). However, the "loop" and "go to" flow indicators found in traditional flow charting do not exist.

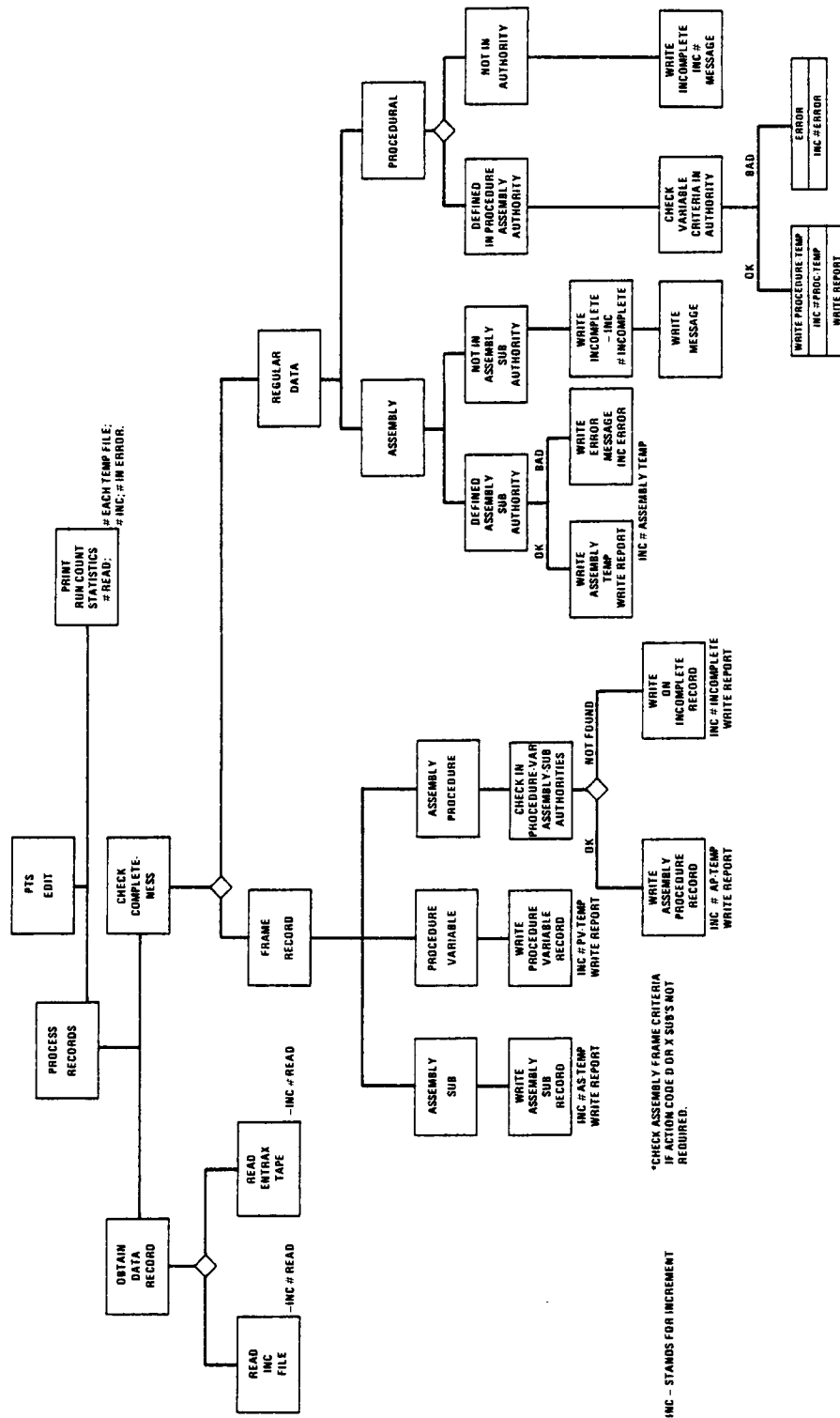


FIGURE 23. EDIT STRUCTURE CHART

To review Figure 23, we see there are two major functions of the edit: to process records and to print run statistics about the execution of the edit. The "process record" function is broken down into two functions "Obtain Data" and "Check for Completeness." "Obtain data" is further broken down into "Read from Incomplete file" and "Read from Entrex file." Under "Check for Completeness" a determination is made as to type of "Frame Record" or "Regular Record." Each of these types of data is further divided as to specific type of input and a process for handling that type data is defined.

As the chart is worked from, more and more detail is defined. This type chart encourages the use of structured code by its nature. Working with a pseudo code or a flow chart about a particular area of a structure chart allows the programmer to concentrate on a particular function. Yet, from the structure chart, its relationship to other aspects of the code is evident.

The structure chart has served its purpose once the code has been completed. A very general "high level" chart could be maintained as part of the program documentation. Indeed a detail level chart would be helpful. However, the maintenance of such charts would be an endless task. The true benefit of the structure chart is that the code constructed from such a chart is easier to follow. By using meaningful names and consolidating related code, a "self-documented" code can be developed.

Osborn developed this system using Time Sharing Option (TSO) on an IBM 370/155. When in production the system will run in a batch mode on an IBM 360/195 because of the security classification of the input data.

CHAPTER V

RAPTS: THE PRODUCTION SYSTEM

Customer Support

A computer system cannot be a true success without strong customer or user support. Only if the customer supports the system can a workable implementation plan be constructed. Total commitment to following the implementation plan must be present. The resources needed to carry out the plan must also be available.

As previously stated, RAPTS has had three sets of customers during its development cycle. Each customer has realized the need for such a system. Faced with the task of storing large amounts of data from documents housed in a number of file cabinets, the idea of such an externally controlled system appealed to the users. The software could be developed while the data definition and collecting task was underway. The users found the task of defining which items of data to collect to be the most difficult task. The task of back entering the data could only be started after the needed data items were identified.

The Recycle Assembly Database Task group was making good progress towards implementing the system when their funding was lost. They had experimented with a tear-sheet data entry method. In this method, a carbon sheet resembling the RADTA input form could be torn from the follow sheet and sent directly to keypunch. This idea was rejected because of security problems since the data was classified.

After the Recycle Assembly Database Task group lost funding, the project was assumed by the SSD Database group. Because of a lack of personnel, this group has made slow progress towards data definition or entry.

Customer Acceptance

The general concepts of the system have been well accepted by the user. These concepts are:

1. Externally controlled definition of stored data.
2. Tracability of assembly history.
3. Useful retrieval of data along predefined paths.
4. Data available for trend or other analysis.

However, the realization of the data entry task is overwhelming.

The system requires a sophisticated user. The system's use must be planned and controlled. If not carefully managed, a useless collection of data would exist. This is perhaps RAPTS's biggest downfall.

RAPTS in the Database World

RAPTS is an effective computer system. This application system was approached not with the idea of creating a database management system, but with the concept of investigating some of the data structure and storage concepts used in a database management system.

RAPTS gives the user much freedom in selecting the data items to be stored. Due to its sophisticated data structures, RAPTS allows retrieval of selected items without searching the entire data collection. The majority of data items can be retrieved with one access (this assumes indexes remain in core during processing). A mechanism exists to reclaim unused space.

Storage use statistics are available. Good control mechanisms exist for monitoring data input. A mechanism for keeping a backup copy of the data also exists.

The RAPTS application system was designed using a networking scheme. A randomizing routine was used to determine the track location of records for the frame files. The data master files use indexes and pointers to locate records.

The concepts used in RAPTS, the data scheme, and even the storage logic are very similar to the processes that would be needed to develop RAPTS using the commercially available database management system IDMS. IDMS uses a networking scheme for storing data. If a hierarchical system such as IMS were used, the data system would take more redesign.

Because of the ground rules established for IDMS databases and the fact that relationships cannot exist within a particular data set, some additional space would have been required for storage using IDMS.

However, many advantages are received from using a commercially available database management system. If a commercial database management system is used for all application systems developed by a department, maintenance becomes an easier task. All personnel are familiar with the networking concept and the documentation practice forced by the package. A commercial package has a staff of top-notch people constantly devoted to upgrading the database management package. Therefore, better randomizing routines are developed, better locking mechanisms exist leading to better data security, better reorganization mechanisms exist, fast load facilities exist, and compatible on-line query systems can be developed.

Developing RAPTS using COBOL and logic to maintain the networking structure was a sizable undertaking. It has proved that to approach a very large system using this conventional approach would be an unwieldy project. If developed, the maintenance of such a system would be an almost impossible task.

Summary

The development of RAPTS has been a true learning experience. Much was learned from surveying literature as well as in the actual design of the system.

The RAPTS project has proven the merits of using a commercial database package in a production application environment to the author. Although the idea of creating a database management package is intriguing, it is not practical.

RAPTS requires a user sophisticated both in his own discipline and in the data entry scheme of the system. However, this aspect of the system is a positive attribute in that this is what allows the user to specify his data storage needs externally.

The need for future system maintenance is not built in. Adding a procedure or changing the data item to be included from a procedure can be changed by the user without programmer intervention. The system is very flexible, yet good control must be exercised by the user to have a reliable system.

The concept of the RAPTS system will ultimately be incorporated with other data collection systems into one integrated data collection

system for SSD. This system will be implemented using IDMS. It is hoped that through the RPTS system, ideas for even a better procedure data collection system can be developed.

RPTS was developed using very general concepts. Because of the flexibility of the system, the software could very easily be used to support data from almost any other research project.

BIBLIOGRAPHY

BIBLIOGRAPHY

- Bassler, Richard A., and Jimmie L. Logan. The Technology of Data Base Management Systems. Arlington, Virginia: College Reading, Incorporated, 1974.
- Cardenas, Alfonso F. Database Management Systems. Boston: Allyn and Bacon, Incorporated, 1979.
- Date, C. J. An Introduction to Database Systems. Reading, Massachusetts: Addison-Wesley Publishing Company, 1977.
- Gane, Christopher P. Structured Systems Analysis and the Training of Systems Analysts. New York: Yourdon, Incorporated, 1976.
- Ghosh, Sahli P. Data Base Organization for Data Management. San Jose, California: Academic Press, 1977.
- House, William C., editor. Data Base Management. London: Madison & Lipscomb Publishers, 1974.
- IBM System/360 Operating System Full American National Standard COBOL Compiler and Library, Version 3 Programmer's Guide. New York: IBM Corporation, Programming Publications, 1972.
- Introduction to IBM Direct-Access Storage Devices and Organization Methods. Poughkeepsie, New York: IBM Corporation, EPD Education Department Publication Services, 1976.
- Kindred, Alton R. Data Systems and Management. Englewood Cliffs, New Jersey: Prentice-Hall, 1973.
- London, Kieth R. Techniques for Direct Access. Surrey, England: Auerbach Publishers, Incorporated, 1973.
- Newcome, E. L., and P. S. Roth. Gas Centrifuge Enrichment Plant Recycle/Assembly Facility Data Base/Management System Baseline for Requirements Analysis. Oak Ridge, Tennessee: Union Carbide Corporation, ORGDP Reproduction Department, 1978.
- Orlander, Donald R. "The Gas Centrifuge." Scientific American, Volume 238, Number 2 (August, 1978), 37-43.
- Tsichritzis, Dionysios C., and Frederick H. Lochovsky. Data Base Management Systems. New York: Academic Press, 1977.

VITA

Nancy E. Dowell was born in Cookeville, Tennessee on December 23, 1946. She attended elementary schools in that city and was graduated from Putnam County Central High School in June 1964. She entered Tennessee Technological University later in June and in August 1967 she received a Bachelor of Science degree in Mathematics.

From September 1967 until August 1969, she was employed by Boeing Aircraft, Space Division, in Huntsville, Alabama as an associate engineer. In September 1969, she accepted a programmer analyst position with Union Carbide Corporation, Nuclear Division in Oak Ridge, Tennessee. Dowell is currently head of the Technical Data Management Section of Management Systems' Production Control Department of Union Carbide's Computer Sciences Division.

Dowell entered the Graduate School of the University of Tennessee, Knoxville, in September 1974. She received the Master of Science degree with a major in Computer Science in June 1980.

The author is a member of the Association for Computing Machinery and the Business and Professional Women's Club.