AN IMPLEMENTATION OF A SOFTWARE ENGINEERING PROJECT MANAGEMENT SYSTEM: A TOOL FOR A PROTOTYPE SOFTWARE ENGINEERING ENVIRONMENT

by

OLIVER BERT CASTLE

B. S., Miami University, Oxford, Ohio, 1973

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1984

Approved by:

[illegible]

Major Professor

ACKNOWLEDGEMENTS

This Master's Report is dedicated to my Lord, who arranged my circumstances and gave me the motivation to see it completed.

A special thanks to Dr. David Gustafson for his detailed review of this report and his guidance and help throughout the course of the implementation. I would also like to thank Dr. William Hankley and Dr. Virgil Wallentine for their time and advice.

I am indebted to my mother for teaching me the joy of learning and the value of education. Also, a sincere thank you to my lovely wife, Janice, for her patience and constant support.

AN IMPLEMENTATION OF A SOFTWARE ENGINEERING PROJECT MANAGEMENT SYSTEM: A TOOL FOR A PROTOTYPE SOFTWARE ENGINEERING ENVIRONMENT

LIST OF FIGURES

LIST OF TABLES

CHAPTER ONE - Overview

1.1 Introduction

There are two major research and development projects which could have a great effect on both the traditional von Neumann computer systems and the software development environment. One is fifth-generation computer systems (FGCS) project and the other the STARS program. The United States and Japan are the two major participants.

The Japanese aim to develop by 1990 a prototype FGCS which will be a knowledge information processing system and processor. The plan for the project has been documented in a series of reports [JIPDC, 1981]. Also nearly every technical publication has had an article of FGCS in the last two years.

The U.S. Department of Defense (DoD) has consistently moved to advance computer technology, such as the VHSIC--very high speed integrated circuit [Sumney, 1982] and Ada* [Carlson-Druffel-Fisher-Whitaker, 1980], [Freeman-Wasserman, 1983] programs. Now the U.S. has begun a program to develop software techniques call STARS--Software Technology supporting the development of Adaptable, Reliable Systems.

* Ada is a trademark of the U.S. Department of Defense.

This Master's report documents the development of the Project Management System, that will provide features for a prototype of a software development environment suitable for either FGCS or the STARS effort introduced in opening paragraphs. The Project Management System is an interactive system that maintains a data base of manhours by project activities. The system calculates cumulative manhours to completion by activity and tracks remaining manhours by activity. In addition to generating an output activity status report, graphic displays are constructed on either a graphic display terminal or a plotter. Graphic displays include a bar chart of manhours allocation against remaining manhours by activity and a bar chart showing PERT time estimates to allocated manhours for each activity.

The remainder of this chapter presents a literature survey of FGCS and STARS. This survey will discuss issues related to the project--Project management, the FGCS project, and the STARS program. The literature survey first discusses the parallel evolution of project management and generations of computer systems. This evolution is depicted both in terms of hardware and application areas. Next surveyed is the the DoD STARS program. The survey addresses the rationale behind the program. Project management is discussed as one of the major technical areas within the STARS program noting Japan's interest in project mangement.

The remaining chapters of this Master's Report address the major areas within the development of the implementation Project Management System. Each chapter focusing on a particular area in the implementation. Chapter Two contains the requirements for the Project Management System implementation. Here we clearly and precisely describe each of the essential requirements of the implementation and the external interfaces. CHAPTER THREE presents the design of the Project Management System. Discussed are design decisions, techniques, data and file organization and I/O (input and output). Finally, the Master's Report concludes with the author's summary and general impressions of the project as a whole.

1.2 Computer Systems Evolution and Project Management

The context in which software has been developed and the need for project management is closely coupled to three generations of computer system evolution. Table 1-1 depicts the evolution of computer-based systems both in terms of application area [Pressman, 1982] and hardware characteristics [Lord, 1983]. Better hardware performance, smaller size, and lower cost have precipitated more sophisticated systems. Computer system generations have moved from slow vacuum tube predecessors to fast microelectronic devices [Oborne, 1979], [Toffler, 1978].

First-Generation (1951-1960)
hardware: characterized by the vacuum tube
application area: batch orientation, limited distribution, custom software

Second-Generation (1960 to mid-1970's)
hardware: characterized by the transistor
application area: multiuser, real-time, product software

Third-Generation (mid-1960's to early 1970's)
hardware: characterized by microminiatur integrated circuit fabrication techniques

Fourth-Generation (mid 1970's -)
hardware: characterized by "plug compatible"
application area: consumer computing, office automation, data collection

TABLE 1-1. Evolution of Computer Systems

During the first-generation, hardware underwent continual change while software was viewed by many as an afterthought. Few systematic methods existed for computer programming. Software development was virtually unmanaged. When projects were managed, it was not until schedules slipped or costs escalated. Software was custom designed for each application and had a relatively limited distribution. Design was an implicit process performed in one's head and rarely controlled or documented. During the first-generation little was learned about project management.

Second-generation computer systems introduced new concepts of human-machine interaction. Multiuser systems, interactive techniques, and real-time systems led to new levels of both hardware and software techniques. The second-generation was characterized by the introduction of product software, i.e. programs developed to be sold in a multidisciplinary market. This increased growth in computer systems software carried with it management problems. Problems associated with how we develop software, how we maintain a growing volume of existing software, and how we can expect to keep pace with a growing demand for more software had to be resolved. Software development was out of control. There was a management problem. This situation led to what has been called the software crisis [Pressman, 1982]. The term "software engineering" was coined during this time too.

Third-generation of computer system was an architectural departure from its predecessors. Microminiature integrated circuit fabrication techniques resulted in the availability of complex logical functions and embedded intelligence at low cost. While advances in hardware increased dramatically, our ability to deal with increasing complexity and still produce software on time and within budget had not kept pace. The software crisis heightened. As a response to the software crisis, project management began to be taken seriously. Fundamental research in systems development began. Software management and development aids were proposed.

Progression to fourth-generation computer systems [Hessinger, 1984], [Cochran, 1983] has been an evolution of, not a departure from, third-generation concepts. The significant advances in what has been termed Very-large-scale intergration, commonly referred to as VLSI is resulting in more densely packaged components and is premitting very dense RAM (random access memories) memories. This microminiaturization permits increasingly more powerful devices to be packaged in smaller physical space. Equally important is the fact that the new technology results in lower costs and hence, lower prices. These lower prices are justification for new software applications with vast market potential [Weil, 1982]. There became an increased use of software engineering techniques and automated software development tools. The transition from a technical to a consumer marketplace demands professionalism that can be accomplished only through project management.

1.3 Fifth-Generation and Beyond

Transition to fifth-generation computer systems and beyond has already begun. In October 1981, Japan's Ministry of International Trade and Industry (MITI) sponsored a conference to announce a new national project. Alongside national projects in supercomputering [Marbach, 1983], [Norrie, 1984] and robotics [Togai, 1984], there would be a effort to develop a new generation of computers known as the fifth-generation computer systems or FGCS [Moto-oka, 1982].

FGCS is being developed predominately for use with knowledge information processing systems. They are expected to come into wide spread utilization in the 1990s [Moto-oka, 1982], [Treleaven-Lime, 1982]. These knowledge information processing systems or KIPS, will be able to reason, learn, associate, and make inferences [Duda-Gaschnig, 1981]. This type of nonnumeric data processing will require a departure from the conventional architectures found in today's computers to new architectures [Moto-oka, 1983]. Computer scientists in Japan have designed a ten year research and development plan in three stages given in Table 1-2.

Stage 1	
	3 years for idea development
Stage 2	
	4 years for prototyping
Stage 3	
	3 years for evaluation

TABLE 1-2. FGCS Project Schedule

The FGCS project involves research and development teams from the academic, industrial, and governmental sectors and deals with several key technologies, including software engineering project management. The need to produce software and hardware of such a large-scale and complex project requires the development of new tools and techniques in software engineering project management. The complexity of the end product, FGCS, has also necessitated a new approach to managing the development process. By 1990, Japan's software engineering technology, project management in particular, may rival that of the U.S. [Kim, 1983].

1.4 The DoD STARS Program

The United States Department Of Defense (DoD) has a development effort called the STARS program (Software Technology for Adaptable, Reliable Systems) [Druffel-Redwine-Riddle (2), 1983]. This program is intended to improve software embedded in mission-critical systems by initiating a coordinated research and development program to improve the software development environment.

Virtually every system in the current and planned military inventory makes extensive use of computer technology. Being an integral component which controls mission-critical functions, software has become an essential element of our whole defense system. References both emphasize the importance of software to the DoD and the difficulties caused by the current state of practice [DoD, 1982], [DoD, 1983].

The need to manage this software as a critical component of defense systems over their life cycle has finally been recognized. A general awareness of this need as an institutional problem requiring special attention within the Office of the Secretary of Defense has intensified as software problems have reached top level DoD management visibility [De Roze-Nyman, 1978]. Consequently, DoD has undertaken the STARS program. The overall objective of the STARS program, to improve software practice by improving the environment, involves three specific objectives [Druffel-Redwine-Riddle (1), 1983]. They are to increase the level of expertise, improve and develop software tools, and increase their use. Edith Martin, deputy under secretary of defense for research and advanced technology, has documented in detail the STARS strategy for accomplishing those objectives [Martin, 1983]. Briefly, the DoD plans to exploit available technology by building on existing methods and techniques, while at the same time, supporting research and development in software engineering, including the area of project management.

Although the STARS program evolved primarily from a DoD need to manage its software, it is highly relevant to the entire software community. Similar to the Japan's fifth-generation computer systems project, it represents a major commitment of resources. These efforts, the establishment of software life cycle management policy and practices (Software Engineering Institute**) and the development and application of new software technology will be a major driving force in software technology for the next five to ten years.

1.5 Project Management and Beyond

Most software projects fail. They are plagued with a number of problems leading to schedule and cost overruns, dissatisfied users, or error-prone production systems. Sometimes these problems are technical, but more often they are managerial in nature [Keider, 1974]. Software engineering technologies have addressed many of the major issues in software production [Kernighan-Plauger, 1976]. A

** Study Report on the DoD software Engineering Institute, Institute for Defense Analysis, Washington, DC. Sept. 1983.

variety of software engineering tools, techniques, and methods, shown in Table 1-3 [Beck-Perkins, 1983], are commonly available for requirements and designs. Improvements and developments in the managerial aspect of software engineering have not kept pace with advances in the technological aspects. An interesting article [Thayer-Lehman, 1977] addresses this by asking the question: "What is the state-of-the-art in software engineering project management today?". Although the need for project management is apparent, research in the area of software engineering has been deficient [Cooper, 1978]. Project management, within the DoD's STARS program is taking the initiative and reflects this in its objectives to enhance project planning, provide improve communications and develop project management expertise [Lubbes, 1984].

A. Requirements	B. Design
HIPO	Jackson Method
SADT	Warnier Method
SREM	Structured Design
PSL/PSA	Abstract Data Types

TABLE 1-3. Software Engineering Tools and Methods

The United States is not alone in realizing the importance of project management. In addition to the highly publicized FGCS project, Japan has been pursing the so called software factory.

The Toshiba Corporation a leading Japanese manufacture in the electric industry, has a software factory which they call SWB which stands for "software workbench system" [Matsumoto, 1980]. SWB is a software factory containing approximately 2000 employees and an integrated software development and test tool system. SWB is characterized by the fact that their software products are application software for use in strict real-time conditions such as nuclear power stations, chemical process plants, and steel rolling mills. Therefore, software engineering project management is a major concern of the factory.

The Hitachi Software Engineering Co., popularly known as HSK, has developed a permanent staff and production methods to expand its world market [Tajima-Matsubara, 1984]. HSK has exported two major software tools to the U.S.: HIDOC, a hierarchical documentation writer, and SHC, a shorthand Cobol. Another software tool HSK is considering for export is STAMPS, a standardized modular programming system. As the leading Japanese software house, HSK is widely reputed for its project management. Its average system consists of 50,000 lines of code. Their application software development is mostly, Cobol-based.

For the project manager to be able to control any given project with many activities, it is essential that he has up-to-date information on the current state of progress at all times. He needs a way of tracking the myriad of details that are found in large projects. Specifically, the project manager needs to know what's happening through monitoring and reporting to successfully managed a project through its life cycle. Project monitoring and reporting involves breaking the project into a series of activities. Then the status of the project could be assessed. It requires maintaining a detailed schedule for all the activities showing interrelationships among them and what impact a problem in completing a particular activity will have on the schedule of the total project. Finally, it involves defining and implementing a system for collecting and updating activity status data on a timely basis, so that problems can be quickly recognized and resolved. A Project Management System is needed. To response effectively to this need, the proposed system must meet certain requirements discussed in the following chapter.

CHAPTER TWO - Requirements

2.1 Introduction

This chapter describes the essential requirements of the Project Management System and the external interfaces. Also, it defines what the system is to do. It does not address the process of producing the software products.

2.2 Requirements

A Project Management System will be developed for the project manager to aid in management and control of a project throughout the software life cycle. It must provide flexibility in the access of and maintenance of information. The Project Management System must be able to access activity data by specific field and also be able to reference individual activities easily.

Software engineering project management has become the center of attention because of the potential penalties that can be incurred due to lack of project management. A survey of software engineering revealed that project success is related to planning, organizing, and control [Thayer-Pyster-Wood, 1980], [Thayer-Pyster-Wood, 1982]. To this end, software teams have used a variety of techniques to manage the development effort. The Program Evaluation

Review Technique, more commonly call PERT, and the Gantt chart are the approaches most often used for this purpose. Therefore, techniques from these two methods will be incorporated into the system. From PERT a form of network analysis will be performed and from Gantt, project management charting; that is, bar charts with vertical bars, the length of which is proportional to time.

The Project Management System must be made available for use interactively, with batch reporting on-request. Since the Project Management System will perform interactive computing, human factors must be considered. All information displayed to the terminal operator should be user-friendly, and concise for ease of use. To ensure this, it must require only minimal response from the terminal operator.

The Project Management System was identified as having two software components, interactive processing and graphic generation. They are discussed in the next two subheadings.

2.2.1 Interactive Processing Requirements

The Interactive processing will be menu driven(see Figure 2-1). The terminal operator will follow a prompt and response format. Clear, concise prompts that require very short responses of the terminal operator will be required. This will assure speed and efficiency of each interactive session. Menu selections are to consist of six functions, they are ADD, UPDATE, DELETE, INQUIRY, PRINT, and EXIT.

```
------------------------------------
P R O J E C T  M A N A G E M E N T
            M E N U

[1] ADD     [2] INQUIRY  [3] DELETE
[4] PRINT   [5] UPDATE   [6] EXIT

     ENTER YOUR SELECTION:
------------------------------------
```

Figure 2-1. Project Management Menu Screen

These functions will provide flexibility in data entry, access, and maintenance of activity data. Each of the six functions will require specific inputs from the terminal operator and will generate specific outputs to the terminal operator.

The ADD function of the menu will be selected to add a unique activity to the system. The terminal operator will be prompted for activity code, activity description, start date, completion date, manhours, PERT times, predecessor activities, and successor activities.

To aid in the following discussion, refer to Figure 2-2, which illustrates a sample activity display from the Project Management System.

```
     p r o j e c t  m a n a g e m e n t
              activity report

code:          1                programmer: smith
activity description :feasibility study

start date:840101      completion date:840131

manhours...
budgeted hrs.   :        150 expended hrs.  :         20
remaining hrs.  :        130 cumulative hrs.:        150

pert times...
optimistic:150 pessimistic:200 most likely:175

-----------------------------------------
   predecessors:    successors:
           0            2
           0            0
           0            0
           0            0
           0            0
```

Figure 2-2. Activity Display

The UPDATE function of the menu screen is for updating individual fields of existing activity in the system. Fields for update are the same as for the ADD function with the exception of the activity code which can not be revised.

The INQUIRY function will display activity data by activity code while PRINT displays the entire activity file to a subfile for a subsequent hardcopy listing.

The DELETE function when given an activity code will display and then delete the activity. Lastly, the EXIT function of the menu screen will terminate the session and output an updated activity file.

2.2.2 Graphic Generation Requirements

The Graphic generation component of the Project Management System will generate two vertical bar charts from the activity Master File. One bar chart will show budgeted manhours(planned) versus remaining manhours by activity. The second vertical bar chart will show budgeted manhours versus PERT time estimates by activity.

2.3 Interfaces: Software/Hardware

Next listed in Table 2-4, are references to software and hardware interfaces for the implementation of the Project Management System.

<u>Software</u>:

- -Berkeley Pascal
- -UNIX***
  Operating System
- -Graphic Facilities on the UNIX Operating System

<u>Hardware</u>:

- -Perkin-Elmer
  8/32 at Kansas State University Computer Science Department
- -Plexus P/85 Computer
- -Retro-Graphics Terminals
- -Soltec Servogor Plotter 281

TABLE 2-4. Interfaces: Software/Hardware

2.4 Validation Criteria

Described in the following are all the relevant objectives to be used as validation criteria at completion of the implementation to recognize a successful implementation.

===

- To allow a project's activity data be modified on the basis of each independent activity as near as possible to the time of its occurrence.
- Process input activity data in any sequence.
- Provide on-line data entry and inquiry.
- Provide for on-request reporting.
- From a behavioral viewpoint, make project manager's job more satisfying, by allowing him to deal with activity data as they occur.
- Provide for ease of use.
- Reduction of voluminous reports

===

TABLE 2-5. Project Objectives

*** Trademark of AT&T Bell Laboratories.

CHAPTER THREE - Design

3.1 Introduction

The chapter describes the design process of the Project Management System.

3.2 Design Decisions

The Project Management System was designed for use under the UNIX operating system at the Kansas State University (KSU) Computer Science Department. The UNIX Operating System, being a de facto standard for minicomputers along with its availability, influenced that design decision.

Berkeley Pascal [Joy-Graham-Haley 1980] was selected as the implementation language. It was chosen because of availability and extensive use at KSU. An additional advantage of Pascal is portability.

3.3 Structured Design Techniques

The Project Management System was designed by using the top-down structured design technique. The overall design was planned using a hierarchy block diagram which depicts information as a series of multilevel blocks organized as a tree structure. Figure 3-3 represents the Project Management System subdivided into two components "graphic generation" and "interactive processing." These two components are then further refined in Figure 3-4 and Figure 3-5.

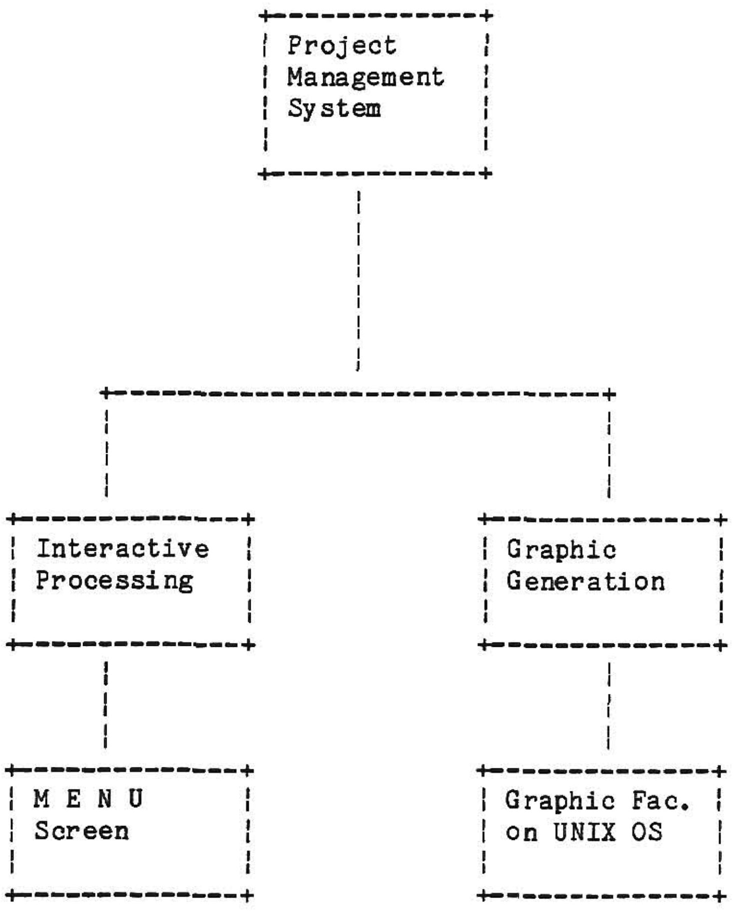


Figure 3-3. Hierarchy Block Diagram of Project Management System

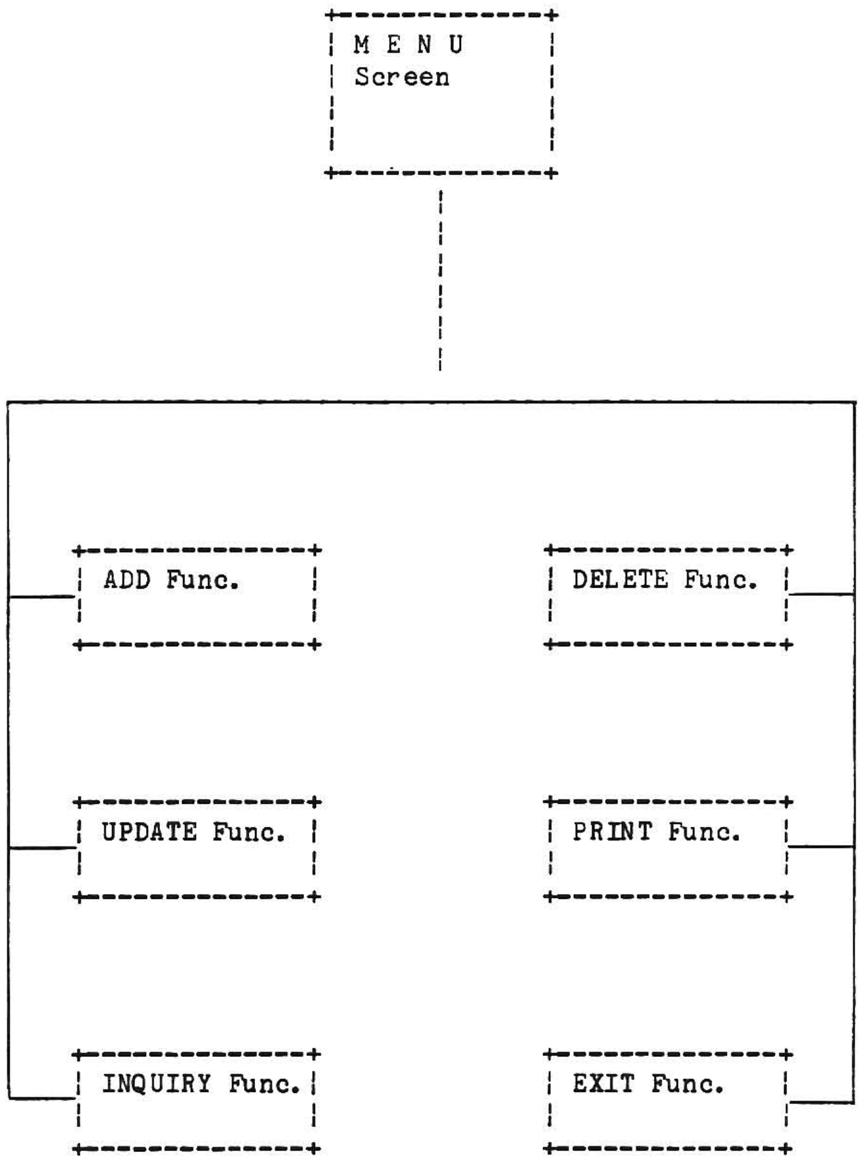


Figure 3-4. Hierarchy Block Diagram of Interactive Processing Software Component

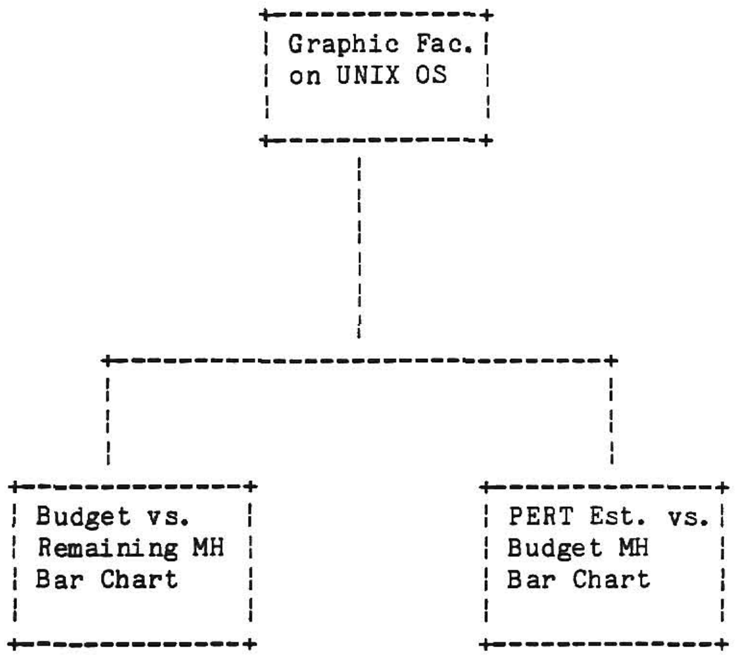

Graphic Fac. on UNIX OS
Budget vs. Remaining MH Bar Chart
PERT Est. vs. Budget MH Bar Chart

Figure 3-5. Hierarchy Block Diagram of Graphic Generation Software Component

The module specifications were then designed from the diagrams. For the interactive processing component of the system, one module (interact.p) was coded to perform all data management and interactive tasks. A program flow chart of interact.p is given in Figure 3-6. A program/module design specification was developed that includes all input, output, and functional specifications. See Appendix A - Program/Module Design: interact.p.

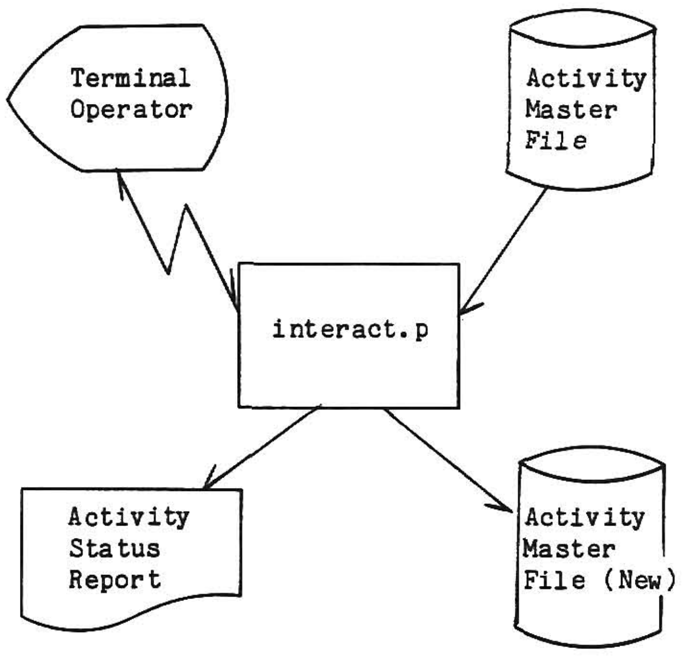


Figure 3-6. Program Flow Chart of interact.p

The graphic generation component defined in the requirements required the development of a module (grafgen.p) to output formated data for the graphic facilities. See Figure 3-7 for program flow chart. The program/module design specification can be referenced in Appendix B - Program/Module Design: grafgen.p.

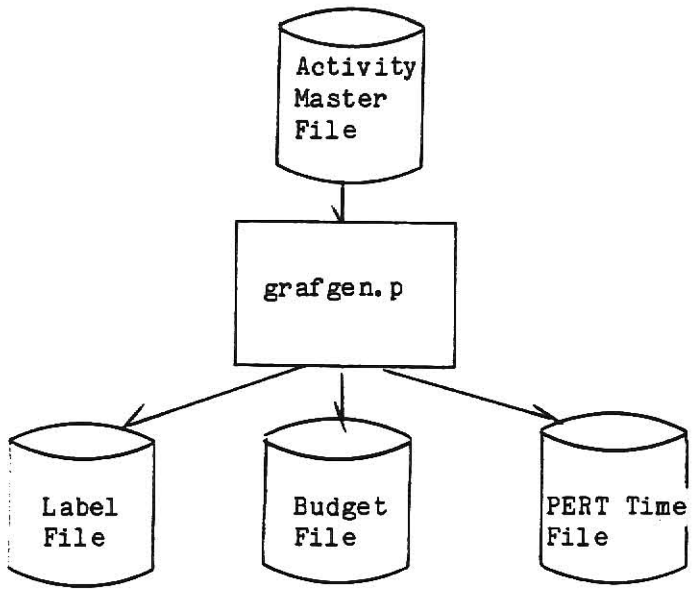


Figure 3-7. Program Flow Chart of grafgen.p

3.4 Data and File Organization

The next step in the design phase was the design and organization of the data. The system requirements and output requirements previously developed in the requirements were the basic source of information to this step. Files were set up which include an Activity Master File and three subfiles which include a Label File, Budgeted Manhours File, and Pert Time File.

The Activity Master File is a sequential file. It contains all the activities of the project. File creation and maintenance is via on-line data entry. When an activity is created, a unique code will be manually assigned. The activity record will contain that unique code number, a description of the activity, and other required information. Once established on file the code will be the access key. Refer to Appendix C for data structure.

The remaining three files--Label, Budgeted Manhours, and PERT Time, are sequential transient files. They are system generated containing extracted data from the Activity Master File formated for the graphic facilities. Extracted data includes x axis labeling information, budgeted manhours, remaining manhours, and PERT time estimates. Refer to Appendices D, E, and F for data structures.

3.5 Input and Output

The design also included a menu screen listing the six required functions. The terminal operator may select any one of the six functions. From there the design prompts the terminal operator through the session returning him to the menu screen for another session. In this way, no necessary information can be missed. Refer to Figures 2-1 and 2-2 for a layout of the menu screen, and an activity data display.

CHAPTER FOUR - Summary

In this Master's Report, we presented the development and implementation of the Project Management System. This implementation was part of a series of Master's implementation projects to provide tools and features for a prototype of a software development environment that would be suitable for either the Japan FGCS project or the U.S. DoD STARS program.

In CHAPTER ONE, we presented the results of a literature survey of FGCS and STARS, two major research and development efforts, pointing out relationships between the implementation-Project Management System and the literature.

The implementation requirements and design were addressed in CHAPTER TWO and CHAPTER THREE, respectively. We discussed what the implementation had to do to be helpful, what the validation criteria was, and also how the requirements influenced the design.

Up until now, computer industry's emphasis has been on the technical aspects of software development, with very little research in the area of software engineering project management. As a result, development problems have persisted. Hopefully, the research and development efforts of the U.S. STARS program and the Japan FGCS project will produce and promote the use of effective software engineering project management techniques.

Future work on the Project Management System could address deficiencies that time did not permit to be considered in this first implementation. Some of these areas are discussed below.

From a run time standpoint, the graphics package executes satisfactorily except for the mapping of the plot files to the plotter. Currently, the plot files have to be copied from the 8/32 to the Plexus to access the graphic facilities. This could be seconds or an hour depending on network traffic. Since the plotter is connected directly to the 8/32, another network copy form the Plexus to the 8/32 is required to finally plot the bar charts. Better hardware and software system configuration would eliminate the network copies and reduce total elapse time.

Access security is not explicitly provided by the Project Management System, but is implicit in the UNIX Operating System. An authorized user is one that has a valid user name and password. This entitles a user complete access to the Project Management System. The user could add, update, delete, or inquire at will. Changes to the software to password protect each function would be appropriate.

In contrast to full screen display, this implementation supports line by line display. With line by line there is the problem of scrolling off the screen. Software changes to implement full screen would be substantial, but conceivable.

In closing, the author was very pleased to have had the opportunity to work on the project. The literature survey provided an increased awareness of research and development occurring in software engineering project management and computer systems. Also the project provided the opportunity to learn and use Berkeley Pascal and the Graphic Facilities on the UNIX Operating System.

REFERENCES

[Beck-Perkins, 1983]

Berk, L. L. and T. E. Perkins, "A Survey of Software Engineering Practice: Tools, Methods, and Results," IEEE Transaction on Software Engineering, Vol. SE-9, No. 5, September 1983, pp. 541-561.

[Carlson-Druffel-Fisher-Whitaker, 1980]

Carlson, W., L. Druffel, D. Fisher, and D. Whitaker, "Introducing Ada," Proc. ACM, Oct. 1980, pp 263-271.

[Cochran, 1983]

Cochran, H.T., "Fourth-Generation Languages," Computerworld, Vol. 17, No. 24A, June 15,1983, pp. 47-50.

[Cooper, 1978]

Cooper, J. D., "Corporate Level Software Management," IEEE Transactions on Software Engineering, SE-4, No. 4, July 1978, pp. 319-326.

[De Roze-Nyman, 1978]

De Roze, B. C. and T. H. Nyman, "The Software Life Cycle - A Management and Technological Challenge in the Department of Defense," IEEE Transactions on Software Engineering, Vol. SE-4, No. 4, July 1978, pp. 309-317.

[DoD, 1982]

"Report of the DoD Task Force on Software Problems," Office of the Deputy Under Secretary of Defense (Research and Advanced Technology), Washington, DC, July 1982.

[DoD, 1983]

"Software Technology for Adaptable, Reliable Systems (STARS) Program Strategy," Department of Defense, National Technical Information Service, Springfield, Va., Stock No. AD A128981, Mar. 1983.

[Druffel-Redwine-Riddle (1), 1983]

Druffel, L. E., S. T. Redwine, Jr., and W. E. Riddle, "The STARS Program: Overview and Rationale," Computer, Vol. 16, No. 11, Nov. 1983, pp. 21-29.

[Druffel-Redwine-Riddle (2), 1983]

Druffel, L. E., S. T. Redwine, Jr., and W. E. Riddle, "The DoD Stars Program," Computer, Vol., 16, No. 11, November 1983, pp. 9-11.

[Duda-Gaschnig, 1981]

Duda, R. O. and J. G. Gaschnig, "Knowledge-Based Expert Systems Come to Age," Byte, Vol. 6, No. 9, Sept. 1981, pp. 238-281.

[Evans-Piazza-Dolkas, 1983]

Evans, M. W., P. Piazza, and J. B. Dolkas, Principles of Productive Software Management, Wiley-Interscience, New York, N. Y., 1983.

[Frank, 1983]

Frank, W. L., "PIPS: A New Revolution in the Making?" Computerworld, Vol. 17, No. 15, April 11, 1983, pp. 49.

[Freeman-Wasserman, 1983]

Freeman, P. and A. I. Wasserman, "Ada Methodologies Concepts and Requirements," ACM Software Engineering Notes, Vol. 8, No. 1, Jan. 1983, pp. 4-12.

[Hessinger, 1984]

Hessinger, P. R., "Strategies For Implementing Fourth-Generation Software," Computerworld, Vol. 18, No. 8, Feb. 2, 1984, pp ID/1-ID/11.

[JIPDC, 1981]

Japan Information Processing Development Center, "Proceedings, International Conference on Fifth-Generation Computer System," Oct., 1981.

[Joy-Graham-Haley, 1980]

Joy, W. N., S. L. Graham, and C. B. Haley, Berkeley Pascal User's Manual, Version 2.0, October, 1980. Computer Science Division, University of California, Berkeley 94720.

[Keider, 1974]

Keider, S. P., "Why Projects Failed," Datamation, Vol. 20, No. 12, Dec. 1974, pp. 53-55.

[Kernighan-Plauger, 1976]

Kernighan, B. W. and P. J. Plauger, Software Tools, Addison-Wesley, Reading, Mass., 1976.

[Kim, 1983]

Kim, K. H., "A Look at Japan's Development of Software Engineering Technology," Computer, Vol. 16, No. 5, May 1984, pp. 26-37.

[Lord, 1982]

Lord, K. W., CDP**** Review Manual A Data Processing Handbook, Van Norstand Reinhold, New York, N. Y., 1983.

[Marbach, 1983]

Marbach, W. D., "The Race to Build A Supercomputer," Newsweek, July 4, 1983, pp. 58-64.

[Martin, 1983]

Martin, E. W., "Strategy for a DoD Software Initiative," Computer, Vol. 16, No. 3, Mar. 1983, pp. 52-59.

[Matsumoto, 1980]

Matsmoto, Y. et al., "SWB System: A Software

**** "CDP" and "Certificate in Data Processing" are registered trademarks of the Institute for Certification of Computer Professionals (ICPP).

Factory," Software Engineering Environments, H. Hunke, ed., North-Holland, New York, 1980, pp. 305-318.

[McClure, 1981]

McClure, C. L., Managing Software Development and Maintenance, Van Norstand Reinhold, New York, N. Y., 1981.

[Moto-oka, 1982]

Moto-oka, T., ed., Fifth-Generation Computer Systems, North-Holland, New York, 1982.

[Norrie, 1984]

Norrie, C., "Supercomputers for Superproblems: A Architectural Introduction," Computer, Vol. 17, No. 3, March 1984, pp. 62-74.

[Osborne, 1979]

Osborne, A., Running Wild-The Next Industrial Revolution, Osborne/McGraw-Hill, New York, N. Y., 1979.

[Pressman, 1982]

Pressman, R. S., Software Engineering A Practitioner's Approach, McGraw-Hill, New York, N. Y., 1982.

[Sumney, 1982]

Sumney, L. W., "VHSIC: A Promise of Leverage," IEEE Spectrum, Vol. 10, No. 10, Oct. 1982, pp. 93-94.

[Tajima-Matsubara, 1984]

Tajima, D. and T. Matsubara, "Inside the Japanese Software Industry," Computer, Vol. 17, No. 3, March 1984, pp. 34-43.

[Thayer-Lahman, 1977]

Thayer, R. H. and J. H. Lehman, "Software Engineering Project Management: A State-of-the-Art Report," A Collection of Technical Papers, in AIAA Computers in Aerospace Conference, (Oct. 31 - Nov.

2, 1977), pp. 153-167.

[Thayer-Pyster-Wood, 1980]

Thayer, R.H., A. Pyster, and R.C. Wood, "The Challenge of Software Engineering Project Management," Computer, Vol. 13, No. 8, Aug. 1980, pp. 51-59.

[Thayer-Pyster-Wood, 1982]

Thayer, R. H., A. Pyster, and R.C. Wood, "Validation Solutions to Major Problems in Software Engineering Project Management," Computer, Vol. 15, No. 8, Aug. 1982, pp. 65-77.

[Toffler, 1980]

Toffler, A., The Third Wave, Morrow Publishers, New York, N. Y., 1980.

[Togai, 1984]

Togai, M., "Japan's Next Generation of Robots," Computer, Vol. 17, No. 3, March 1984, pp. 19-25.

[Treleaven-Lime, 1982]

Treleaven, P. C. and I. G. Lime, "Japan's Fifth Generation Computer Systems," IEEE Computer, Vol. 15, No. 8, Aug. 1982, pp. 79-88.

[Weil, 1982]

Weil, U., Information Systems In The 80's Products, Markets, and Venders, Prentice-Hall, Englewood Cliffs, N. J., 1982.

APPENDICES

Appendix A - Program/Module Design: interact.p

PROGRAM/MODULE DESIGN

PROGRAM NAME: interact.p

PURPOSE: Interactive processing and PERT Analysis

NARRATIVE: This is a Berkeley Pascal program. It provides the interactive user with a menu screen of the following six functions:

ADD, UPDATE, DELETE, INQUIRY, PRINT, and EXIT

These functions will provide for entering, maintaining, inquiring, and reporting of activities. Reference Figure for screen layout.

In addition, all subsequent actions between terminal operator and system is by prompts. At the conclusion of each function (with the exception of function EXIT) the terminal operator is returned to the menu screen for another selection.

Calculations are performed on the Activity Master File for a PERT analysis and remaining manhours.

The PERT analysis is perform by cumulating each activities budgeted manhours with all that activity's predecessor activities for a total cumulative manhours. The second calculation, remaining manhours, is the difference between budgeted manhours and expended manhours.

There are two inputs defined by the following variables:

standard file input
oldfile

Three outputs files are defined by the following variables:

standard file output
newfile
printout

The standard file input is used for input from the terminal operator during an interactive session. The oldfile defines the current Activity Master File for processing.

Output form the programs to the terminal operator is through standard file output. The newfile variable defines the updated Activity Master File. while printout defines the Activity Report generated by the print function.

Appendix B - Program/Module Design: grafgen.p

PROGRAM/MODULE DESIGN

PROGRAM NAME: grafgen.p

PURPOSE: Extract data for graphics generation.

NARRATIVE: This is a Berkeley Pascal program. It inputs the current Activity Master File and outputs three subfiles of extracted information in the proper formats for graphics generation.

The input file current, Activity Master File, is defined through the standard file input. Three output files are defined by the following variable:

lablfile
budfile
pertfile

Variable lablfile contains the activity code and descriptions for axis labeling. Variable budfile contains budgeted manhours and remaining manhours. Variable pertfile contains budgeted manhours and Pert time estimates.

Appendix C - Data Structure: Activity Data

The Activity Master File is a sequential text file. It contains all activity data for a project on the Project Management System. The following represents a file layout.

Record type	Activity Record
Activity Code:	A two digit user defined code assigned to an activity.
Activity Description:	Description of activity up to up to thirty characters in length.
Start Date:	Starting date of activity. Format of YYMMDD.
Completion Date:	Completion date of activity ; Format of YYMMDD.
Budgeted Manhours:	Allocated manhours for the activity.
Expended Manhours:	Latest expenditure of manhours against the activity.
Remaining Manhours:	System calculation of the difference between budgeted manhours and expended manhours.
Optimistic Time:	PERT time assigned to the activity, what it would be take if everything went right.
Pessimistic Time:	PERT time assigned to the activity, what is would take if everything went wrong.
Most Likely Time:	estimate.
Programmer Last Name:	Personnel assigned to activity.
Predecessors:	Predecessor activities; up to five.
Successors:	Successor activities; up to five.

Appendix D - Data Structure: Label File

The Label File is a sequential text file. It contains information for labeling the x axis of bar charts produced by the Project Management System. The following represents a file layout.

Record type

Label Record

Activity Code:	Numerical code assigned to the activity.
Activity Description:	Description of activity

Appendix E - Data Structure: Budgeted Manhours File

The Budgeted Manhours File is a sequential text file. It is used as a plotting file for bar chart construction. The following represents a file layout.

Record type

Budgeted Manhours Record

Bar Number:	Positional value of bar for bar chart generation
Budgeted Manhours:	Allocated manhours for the activity
Bar Number:	Positional value of bar for bar chart generation
Remaining Manhours:	Calculated value of budgeted manhours less expended manhours

Appendix F - Data Structure: PERT Time File

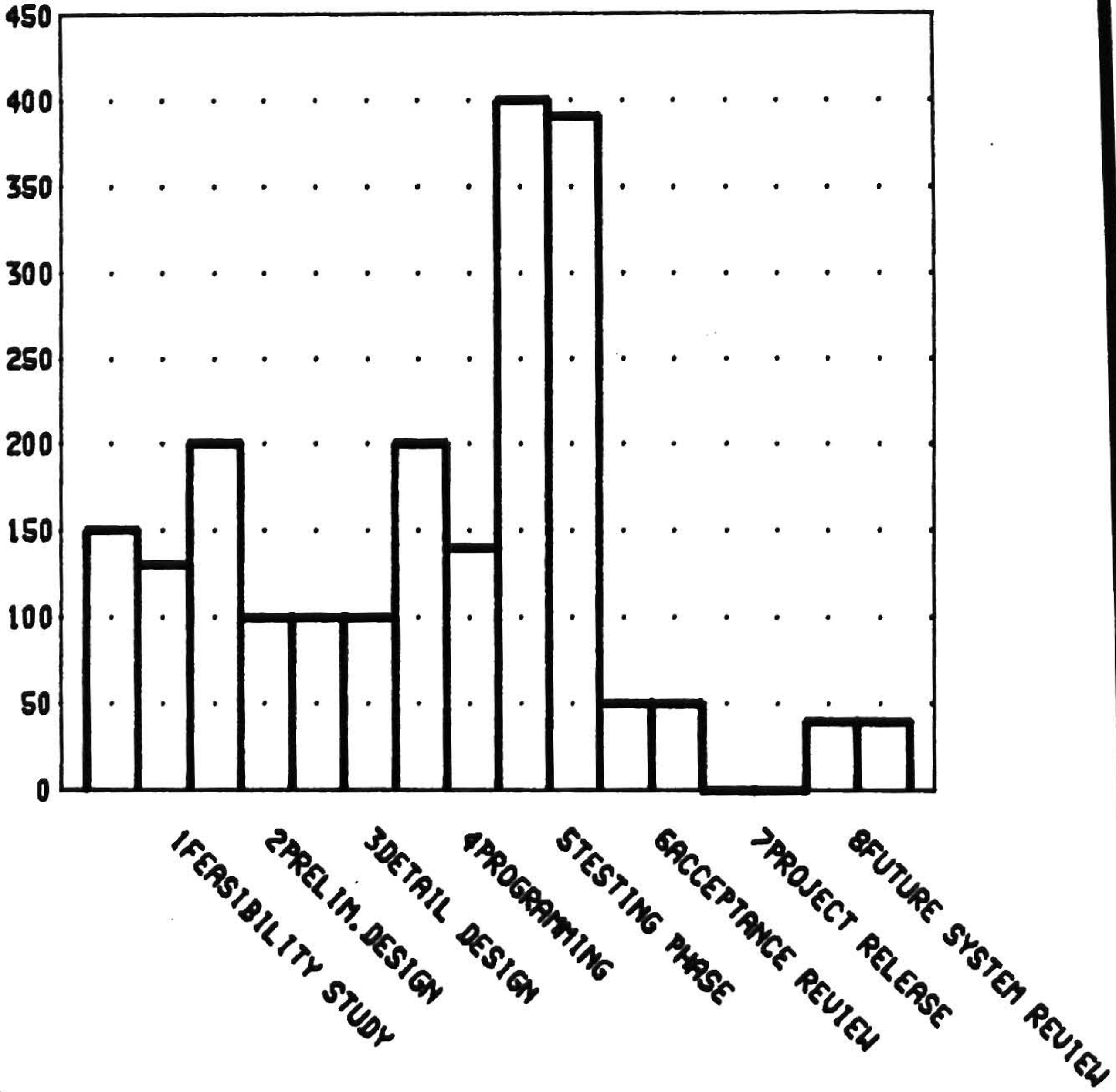
The PERT Time File is a sequential text file. It is used as a plot file for bar chart construction. The following represents a file layout.

Record type

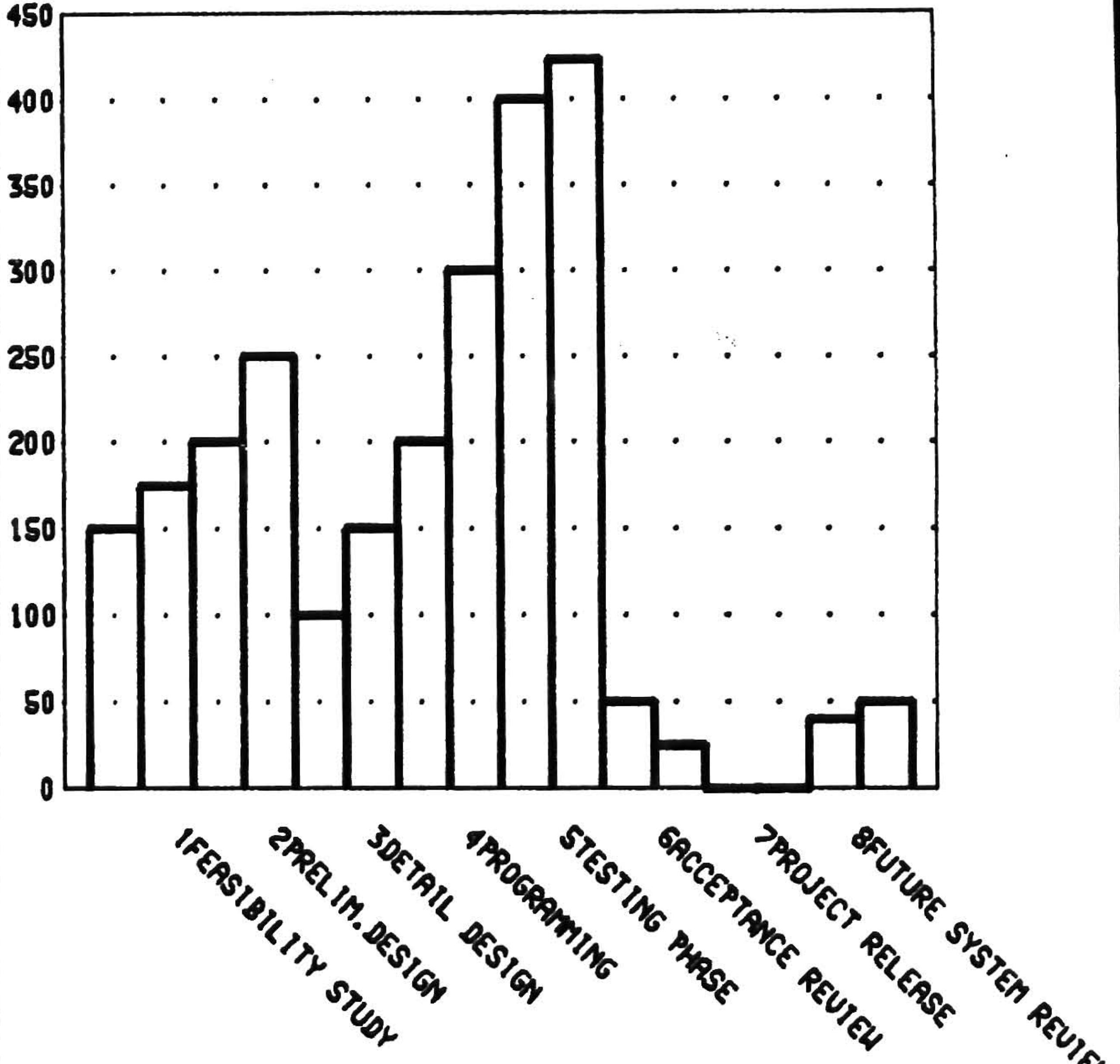
Pert Time Record

Bar Number:	Positional value for bar chart generation
Budgeted Manhours:	Allocated manhours for the activity
Bar Number:	Positional value for bar chart generation
Pert Time Estimate:	(optimistic + pessimistic + 4*most likely) / 6)

Appendix G - Bar Charts



BUDGET VS REMAINING MANHOURS BY ACTIVITY



BUDGET VS PERT ESTIMATED MANHOURS BY ACTIVITY

Appendix H - Source Code: interact.p

```
program interact(input, output, oldfile, newfile, printout);
label
    100;

const
    blank = ' ';
    maxtab = 99;

    maxpre = 5;
    maxsuc = 5;


type
    activityrecords =
        record
            code: integer;
            desc: array [1..30] of char;
            sdte: integer;
            cdte: integer;
            budhrs: integer;
            exphrs: integer;
            remhrs: integer;
            cumhrs: integer;
            opttme: integer;
            pettme: integer;
            mlktme: integer;
            pgmr: array [1..10] of char;
            pre: array [1..5] of
                record
                    predec: integer
                end;
            suc: array [1..5] of
                record
                    succes: integer
                end
        end;

var
    printout, oldfile, newfile: text;
    onerecord: activityrecords;
    userdata: activityrecords;

    table: array [1..maxtab] of activityrecords;
    w, x, y, z: integer;
    recordsread, recordswritten: integer;
    field, selection, numberofrecords: integer;
    answer: char;
    procedure skiplines;
    begin
        for x := 1 to 3 do
            writeln
    end; { skiplines }

    procedure sort(numberofrecords: integer);
(* sort records by code *)
    var
        workspace: activityrecords;
        i, j: integer;
    begin
        for i := 1 to numberofrecords - 1 do
            for j := 1 to numberofrecords - i do
                if table[j].code > table[j + 1].code
                then begin
                    (* swap [j] and [j+1] *)
                    workspace := table[j];
                    table[j] := table[j + 1];
                    table[j + 1] := workspace
                end
    end; { sort }

    procedure enterdesc;
    begin
        with userdata do begin
            writeln('enter description: ');
            for x := 1 to 30 do
                desc[x] := blank;
            x := 0;
            while not eoln and (x < 30) do begin
                x := x + 1;
                read(desc[x])
            end;
            readln;
            table[w].desc := desc
        end

    end; { enterdesc }

    procedure entersdte;
    begin
        with userdata do begin
            writeln('enter start date as yymmdd:');
            readln(sdte);
            table[w].sdte := sdte
        end

    end; { entersdte }
```

```
procedure entercdte;
begin
    with userdata do begin
        writeln('enter completion date as yymmdd:');
        readln(cdte);
        table[w].cdte := cdte
    end

end; { entercdte }

procedure enterhrs;
begin
    with userdata do begin
        writeln('            enter manhours');
        writeln('budgeted  expended  remaining
                cumulative');
        readln(budhrs, exphrs, remhrs, cumhrs);
        table[w].budhrs := budhrs;
        table[w].exphrs := exphrs;
        table[w].remhrs := remhrs;
        table[w].cumhrs := cumhrs
    end

end; { enterhrs }

procedure entertme;
begin
    with userdata do begin
        writeln('        enter pert times');
        writeln('optimistic pestimistic most likely');
        readln(opttme, pettme, mlktme);
        table[w].opttme := opttme;
        table[w].pettme := pettme;
        table[w].mlktme := mlktme
    end

end; { entertme }

procedure enterpgmr;
begin
    with userdata do begin
        writeln('enter programmer last name');
        for x := 1 to 10 do
            pgmr[x] := blank;
        x := 0;
        while not eoln and (x < 10) do begin
            x := x + 1;
            read(pgmr[x])
        end;
        readln;
        table[w].pgmr := pgmr
    end

end; { enterpgmr }

procedure enterpre;
begin
    with userdata do begin
        for x := 1 to 5 do
            pre[x].predec := 0;
        for x := 1 to 5 do
            table[w].pre[x].predec := 0;
        writeln
        ('enter predecessor activities for code',
               table[w].code);
        writeln('are there any? enter y or n');
        read(answer);
        readln;
        x := 0;
        while (answer = 'y') and (x < maxpre) do begin
            x := x + 1;
            writeln('enter predecessor:');
            read(pre[x].predec);
            readln;
            table[w].pre[x].predec := pre[x].predec;
            writeln
            ('more predecessors for', table[w].code,
                    '?', '  enter y or n');
            readln(answer)
        end
    end

end; { enterpre }

procedure entersuc;
begin
    with userdata do begin
        for x := 1 to 5 do
            suc[x].succes := 0;
        for x := 1 to 5 do
            table[w].suc[x].succes := 0;
        writeln
        ('enter successor activities for code',
                table[w].code);
        writeln('are there any? enter y or n');
        read(answer);
        readln;
        x := 0;
        while
         (answer = 'y') and (x < maxsuc) do begin
            x := x + 1;
            writeln('enter successor:');
```

```
              read(suc[x].succes);
              readln;
              table[w].suc[x].succes := suc[x].succes;
              writeln
              ('more successors for', table[w].code,
                   '?', '  enter y or n');
              readln(answer)
          end
      end
  end; { entersuc }

  procedure display;
  begin
      writeln(' ');
      writeln
      ('code:', table[w].code, '                    ',
            'programmer: ', table[w].pgmr);
      writeln
      ('activity description :', table[w].desc);
      writeln(' ');
      writeln('start date:', table[w].sdte: 6,
            '      completion date:',
            table[w].cdte: 6);
      writeln(' ');
      writeln('manhours...');
      writeln('budgeted hrs.   :', table[w].budhrs,
              ' expended hrs.  :',
              table[w].exphrs);
      writeln('remaining hrs.  :', table[w].remhrs,
              ' cumulative hrs.:',
              table[w].cumhrs;
      writeln(' ');
      writeln('pert times...');
      writeln('optimistic:', table[w].opttme: 3,
              ' pessimistic:', table[w].pettme: 3,
              ' most likely:', table[w].mlktme: 3);
      writeln(' ');
      writeln
      ('---------------------------------------');
      writeln
      ('   predecessors: ', '   ', 'successors: ');

      for x := 1 to maxsuc do begin
        writeln('    ', table[w].pre[x].predec, '    ',
                  table[w].suc[x].succes)
      end;
      skiplines
  end; { display }

  procedure print;
  begin
```

```
        writeln(printout, ' ');
        writeln(printout, 'code:', table[w].code,
                '                    '
                , 'programmer: ', table[w].pgmr);
        writeln(printout, 'activity description :',
        table[w].desc);
        writeln(printout, ' ');
        writeln
        (printout, 'start date:', table[w].sdte: 6,
                 '       completion date:',
                 table[w].cdte: 6);
        writeln(printout, ' ');
        writeln(printout, 'manhours...');
        writeln(printout, 'budgeted hrs.   :',
                table[w].budhrs,
                expended hrs.:', table[w].exphrs);
        writeln(printout, 'remaining hrs.  :',
                table[w].remhrs,
                ' cumulative hrs.:', tanble[w].cumhrs);
        writeln(printout, ' ');
        writeln(printout, 'pert times...');
        writeln
        (printout, 'optimistic:', table[w].opttime: 3,
  ' pesstimistic:', table[w].pettme: 3,
  ' most likely:', table[w].mlktme: 3);
        writeln(printout, ' ');
        writeln(printout,'------------------------------
                 ----------');
        writeln(printout, ' predecessors: ', ' ',
                 'successors: ');

        for x := 1 to maxsuc do begin
           writeln
            (printout, '     ', table[w].pre[x].predec,
            '    ', table[w].suc[x].succes)
        end;
        skiplines
    end; { print }

    procedure calculate;
(* c a l c u l a t e    h o u r s *)
    var
        x: integer;
    begin
        for x := 1 to maxtab do begin
            table[x].remhrs := table[x].budhrs - table[x].
             exphrs;
            table[x].cumhrs := 0;
            table[x].cumhrs := table[x].cumhrs + table[x].
             budhrs;
            for y := 1 to maxpre do
```

```
                if table[x].pre[y].predec <> 0 then begin
                    z := table[x].pre[y].predec;
                table[x].cumhrs:=table[x].cumhrs +
                table[z].cumhrs
                end
        end
    end; { calculate }

(* m a i n l i n e   c o d e *)
begin
    writeln(' program interact started ');
    reset(oldfile);
    rewrite(newfile);
    rewrite(printout);
    numberofrecords := maxtab;
    recordsread := 0;
    recordswritten := 0;
(* read oldfile into table *)
    while not eof(oldfile) do begin
        read(oldfile, onerecord.code);
        with onerecord do begin

            for x := 1 to 30 do
                desc[x] := blank;
            x := 0;
            while
             not eoln(oldfile) and (x < 30) do begin
                x := x + 1;
                read(oldfile, desc[x])
            end;
            readln(oldfile);

            readln(oldfile, sdte);
            readln(oldfile, cdte);
            readln
            (oldfile, budhrs, exphrs, remhrs, cumhrs);
            readln(oldfile, opttme, pettme, mlktme);

            for x := 1 to 10 do
                pgmr[x] := blank;
            x := 0;
            while
             not eoln(oldfile) and (x < 10) do begin
                x := x + 1;
                read(oldfile, pgmr[x])
            end;
            readln(oldfile);

            for x := 1 to 5 do
                pre[x].predec := 0;
            for x := 1 to 5 do
```

```
                suc[x].succes := 0;
            for x := 1 to 5 do
                read(oldfile, pre[x].predec);
            readln(oldfile);
            for x := 1 to 5 do
                read(oldfile, suc[x].succes);
            readln(oldfile)
        end;
        recordsread := recordsread + 1;

        (* insert in table by code *)
        w := onerecord.code;
        if (w > 0) and (w <= maxtab) then
            table[w] := onerecord
    end;
(* calculate cumulative hours *)
    calculate;
(*  interactive computing  *)
100:
    skiplines;
    writeln('-----------------------------------');
    writeln(' p r o j e c t  m a n a g e m e n t ');
    writeln('               m e n u ');
    writeln(' ');
    writeln('[1] add     [2] inquiry   [3] delete');
    writeln('[4] print   [5] update    [6] exit');
    writeln(' ');
    writeln('     enter your selection:');
    writeln('-----------------------------------');
    read(selection);
    readln;
    while (selection > 6) or (selection < 1) do begin
        writeln('invalid selection. try again');
        writeln('enter your selection');
        read(selection);
        readln
    end;
(* a d d *)
    if selection = 1 then begin
        skiplines;
        writeln('mode: add');
        writeln('enter activity code:');
        y := 1;
        while y <> 0 do begin
            with userdata do begin
      (* following loops if code is out of range 1-99 *)
                w := 0;
                while (w < 1) or (w > maxtab) do begin
                    read(code);
                    readln;
                    w := userdata.code;
                    if (w > 0) and (w <= maxtab) then
                                (* range ok? *)
                        if table[w].code = 0 then begin
                            table[w].code := w;
                            writeln('code:', w, '...is new')
                        end else begin
                            writeln('code', w, '...is
                            already on file');
                            writeln('enter activity code');
                            w := 999
                        end     (* set out of range *)
                    else begin
                        writeln('code', w, ' is invalid');
                        writeln('enter activity code')
                    end
                end;

                table[w].code := userdata.code;
                enterdesc;
                entersdte;
                entercdte;
                enterhrs;
                entertme;
                enterpgmr;
                enterpre;
                entersuc;
                writeln('are you finish?  enter y or n');
                readln(answer);
                if answer = 'y' then
                    y := 0
                else
                    writeln('enter activity code:')
            end
        end;
        calculate;
        writeln('add mode ended')
    end;
(* u p d a t e *)
    if selection = 5 then begin
        skiplines;
        writeln('mode: update');
        writeln('enter activity code:');
        y := 1;
        z := 1;
        while y <> 0 do begin
            with userdata do begin
      (* following loops if code is out of range 1-99 *)
                w := 0;
                while (w < 1) or (w > maxtab) do begin
                    read(code);
                    readln;
    w := userdata.code;
    if (w > 0) and (w <= maxtab) then
            (* range ok? *)
        if table[w].code = 0 then begin
            writeln('code:', w, '...is does
             not exist');
            writeln('enter activity code:');
            w := 999
        end else begin
            (* set out of range *)
            writeln('code', w, '...is
              on file');
            display
        end
    else begin
        writeln('code', w, ' is invalid');
        writeln('enter activity code:')
    end
end;

writeln('         field selection');
writeln(' ');
writeln('[1] description');
writeln('[2] start date');
writeln('[3] completion date');
writeln('[4] manhours');
writeln('[5] pert times');
writeln('[6] programmer last name');
writeln('[7] predecessors');
writeln('[8] successors');
writeln(' ');
while z <> 0 do begin
    writeln('enter field number:');
    read(field);
    readln;
    while
     (field > 8) or (field < 1) do begin
        writeln('invalid field number.
          try again');
        read(field);
        readln
    end;
    if field = 1 then
        enterdesc;
    if field = 2 then
        entersdte;
    if field = 3 then
        entercdte;
    if field = 4 then
        enterhrs;
    if field = 5 then
```

```
                    entertme;
                if field = 6 then
                    enterpgmr;
                if field = 7 then
                    enterpre;
                if field = 8 then
                    entersuc;
                writeln
                ('are you finish with this code?
                  enter y or n');
                readln(answer);
                if answer = 'y' then
                    z := 0
            end;

            z := 1;          (* reset interloop *)
            writeln('update another activity code?
                enter y or n');
            readln(answer);
            if answer = 'n' then
                y := 0
            else             (* continue *)
                writeln('enter activity code:')
        end
    end;
    calculate;
    writeln('update mode ended')
  end;
(* print report *)
  if selection = 4 then begin
    writeln(printout, '      ',
     'p r o j e c t   m a n a g e m e n t');
    writeln(printout, '     ', '
      activity report');
    for w := 1 to maxtab do
        if table[w].code <> 0 then begin
            print
        end;
    writeln('print mode ended')
  end;
(* activity i n q u i r y *)
  if selection = 2 then begin
    skiplines;
    writeln('mode: inquiry');
    writeln('enter activity code: ');
    writeln('to terminate...enter code 0');
    y := 1;
    while y <> 0 do begin
        read(userdata.code);
        readln;
        y := userdata.code;
```

```
          w := userdata.code;
          (* range check *)
          if (w > 0) and (w <= maxtab) then
              (* find code *)
              if table[w].code = userdata.code
              then begin
                                (* output inquiry *)
                  writeln('     ',
               'p r o j e c t  m a n a g e m e n t');
                  writeln
                ('     ', '            status inquiry');
                  display
              end else
                  writeln('code not found')
          else
              writeln('code is invalid');

          if y = 0 then
              writeln('inquiry mode ended')
          else
              writeln('enter activity code:')
      end
  end;

(* activity d e l e t e *)
  if selection = 3 then begin
      skiplines;
      writeln('mode: delete');
      writeln('enter activity code: ');
      writeln('to terminate...enter code 0');
      y := 1;
      while y <> 0 do begin  .
          read(userdata.code);
          readln;
          y := userdata.code;
          w := userdata.code;
          (* range check *)
          if (w > 0) and (w <= maxtab) then
              (* find code *)
              if table[w].code = userdata.code
              then begin
                                (* output inquiry *)
                  writeln('     ',
                 'p r o j e c t  m a n a g e m e n t');
                  writeln
                  ('     ', '            delete report');
                  display;
                  (* delete record *)
                  table[w].code := 0;
                  writeln('record sucessfully delete')
              end else
```

```
                    writeln('code not found')
                else
                    writeln('code is invalid');

                if y = 0 then
                    writeln('delete mode ended')
                else
                    writeln('enter activity code:')
            end;
            calculate
        end;

    (* sort activity in table by code and write to newfile *)
        if selection <> 6 then
            goto 100;
        begin
            calculate;
            sort(numberofrecords);

            for w := 1 to maxtab do
                if table[w].code <> 0 then begin
                    with table[w] do begin
                        write(newfile, code);
                        for x := 1 to 30 do
                            write(newfile, desc[x]);
                        writeln(newfile);
                        writeln(newfile, sdte);
                        writeln(newfile, cdte);
                        writeln(newfile,
                        budhrs, exphrs, remhrs, cumhrs);
                        writeln
                        (newfile, opttme, pettme, mlktme);
                        for x := 1 to 10 do
                            write(newfile, pgmr[x]);
                        writeln(newfile);
                        for x := 1 to 5 do
                            write(newfile, pre[x].predec);
                        writeln(newfile);
                        for x := 1 to 5 do
                            write(newfile, suc[x].succes);
                        writeln(newfile)
                    end;
                    recordswritten := recordswritten + 1
                end;
            writeln('newfile successfully written');
            writeln
            ('   old record count was ', recordsread);
            writeln
            ('   new record count is  ', recordswritten);
            writeln('program interact ended')
        end
```

```
end.
```

Appendix I - Source Code: grafgen.p

```
program grafgen(input, output, lablfile, budfile, pertfile);

const
    blank = ' ';
    maxtab = 99;

    maxpre = 5;
    maxsuc = 5;


type
    activityrecords =
        record
            code: integer;
            desc: array [1..30] of char;
            sdte: integer;
            cdte: integer;
            budhrs: integer;
            exphrs: integer;
            remhrs: integer;
            cumhrs: integer;
            opttme: integer;
            pettme: integer;
            mlktme: integer;
            pgmr: array [1..10] of char;
            pre: array [1..5] of
                record
                    predec: integer
                end;
            suc: array [1..5] of
                record
                    succes: integer
                end
        end;

var
    lablfile, budfile, pertfile: text;
    onerecord: activityrecords;
    x: integer;
    pertest: integer;
    recordsread, recordswritten: integer;

    procedure writelabel;
(* write labels *)
    begin
        with onerecord do begin
            write(lablfile, code: 3);
            for x := 1 to 30 do
                write(lablfile, desc[x]);
            writeln(lablfile)
        end
    end; { writelabel }

    procedure writeremhrs;
    begin
(* write PERT File *)
        with onerecord do begin
            write(pertfile, recordsread);
            writeln(pertfile, budhrs);
            write(pertfile, recordsread);
            pertest
             := (opttme + pettme + 4 * mlktme) div 6;
            writeln(pertfile, pertest)
        end
    end; { writeremhrs }

    procedure writebudhrs;
(* write budget hours *)
    begin
        with onerecord do begin
            write(budfile, recordsread);
            writeln(budfile, budhrs);
            write(budfile, recordsread);
            writeln(budfile, remhrs)
        end
    end; { writebudhrs }

begin
    writeln('program grafgen started ');
    rewrite(lablfile);
    rewrite(budfile);
    rewrite(pertfile);
    recordsread := 0;
    recordswritten := 0;
(* read from the standard input *)
    while not eof do begin
        read(onerecord.code);
        with onerecord do begin

            for x := 1 to 30 do
                desc[x] := blank;
            x := 0;
            while not eoln and (x < 30) do begin
                x := x + 1;
                read(desc[x])
            end;
            readln;
            readln(sdte);
            readln(cdte);
            readln(budhrs, exphrs, remhrs, cumhrs);
            readln(opttme, pettme, mlktme);

            for x := 1 to 10 do
                pgmr[x] := blank;
            x := 0;
            while not eoln and (x < 10) do begin
                x := x + 1;
                read(pgmr[x])
            end;
            readln;

            for x := 1 to 5 do
                pre[x].predec := 0;
            for x := 1 to 5 do
                suc[x].succes := 0;
            for x := 1 to 5 do
                read(pre[x].predec);
            readln;
            for x := 1 to 5 do
                read(suc[x].succes);
            readln
        end;
        recordsread := recordsread + 1;

        writeremhrs;
        writelabel;
        writebudhrs;
        recordswritten := recordswritten + 1
    end;
    writeln('all files successfully written');
    writeln('    in record count was ', recordsread);
    writeln('   out record count is  ', recordswritten);
    writeln('program grafgen ended')
end.
```

Appendix J - Graphics Commands

budcmmd

```
hist -xa,b $1 | label -b,x,r-45,F$2 | title -v"budget vs
remaining manhours by activity" | td
```

budplot

```
hist -xa,b $1 | label -b,x,r-45,F$2 | title -v"budget vs
remaining manhours by activity" | sd | splot
```

pertcmmd

```
hist -xa,b $1 | label -b,x,r-45,F$2 | title -v"budget vs
pert estimated manhours by activity" | td
```

perplot

```
hist -xa,b $1 | label -b,x,r-45,F$2 | title -v"budget vs
pert estimated manhours by activity" | sd | splot
```

Appendix K - Shell Commands

execpms

```
echo
echo interactive processing
echo
px pms
echo
echo graphic generation?
echo enter: execgen
echo
echo hardcopy report?
echo enter: execprint
```

execgen

```
echo
echo graphic generation
echo
px gen < newfile
echo
echo network copy to ksuplx1 started
echo when prompted enter: login name and password
echo
echo lablfile:
netcp lablfile ksuplx1:lablfile
echo
echo budfile:
netcp budfile ksuplx1:budfile
echo
echo pertfile
netcp pertfile ksuplx1:pertfile
echo
echo network copy to ksuplx1 ended
echo
echo now wait for messages from ksuplx1
```

execprint

```
cat printout | lpr
```

budgetbar

```
budcmmd budfile lablfile
```

budgetplot

```
budplot budfile lablfile
```

<u>pertbar</u>

```
pertcmmd pertfile lablfile
```

<u>pertplot</u>

```
perplot pertfile lablfile
```

Appendix L - User's Guide

PROJECT MANAGEMENT SYSTEM
User's Guide

Introduction

This discussion provides the basic information you need to get started on the Project Management System.

Logging In

Type command: execpms

When you have logged in successfully, the system will display the menu screen.

Adding Activity

To add new activity to the system, select function [1] ADD. The system will then prompt for the activity data. After successfully adding the activity record, the system will prompt for the addition of another activity record. A negative response will return you to the menu.

Updating Activity

To update existing activity select function [5] UPDATE. The system will then display a numbered list of fields for possible update and a prompt for an activity code. Enter your activity code. The system will then prompt for a field to be updated. Enter the appropriate field number from the numbered list of fields. The system will then prompt for the update. After successfully updating the field, the system prompts for more updating. A negative response will return you to the menu.

Deleting Activity

To delete activity, select function [3] DELETE. The system will then prompt for an activity code. After successfully deleting the activity record, the system will prompt for another activity code for deletion. A negative response will return you to the menu.

Activity Inquiry

To inquire, select function [2] INQUIRY. The system will then prompt for an activity code. After successfully displaying the activity record, the system will prompt for another activity code for display. A negative response will return you to the menu.

Hardcopy Report

To obtain a hardcopy of all activity in the system, select function [4] PRINT. The system will output all activity records on file, then return you to the menu.

To Exit

To exit from the system, select function [6] EXIT. The system will terminate the session.

Generating Bar Charts

To generate bar charts, type command: execgen

The system will prompt for user name and password to copy plot files from the 8/32 to the Plexus computer. From the Plexus execute the following instructions.

Constructing bar chart "Budget versus Remaining Manhours by Activity"

Type command: budgetbar, for CRT image.

Type command: budgetplot, for plotter hardcopy.

Constructing bar chart "Budget versus PERT Estimated Manhours by Activity"

Type command: pertbar, for CRT image.

Type command: pertplot, for plotter hardcopy.

Appendix M - Manual Page Descriptions

NAME

execpms - Project Management System

SYNOPSIS

```
execpms
```

DESCRIPTION

The Project Management System is an interactive system that maintains a data base of manhours by project activities. The system calculates cumulative manhours to completion by activity and tracks remaining manhours by activity. In addition to generating an output activity status report, graphic displays are constructed on either a graphic display terminal or a plotter. Graphic displays include a bar chart of manhours allocation against remaining manhours by activity and a bar chart showing PERT time estimates to allocated manhours for each activity.

FILES

standard file input
standard file output
oldfile
newfile
printout

SEE ALSO

execgen, execprint, budget[bar, plot], pert[bar, plot]

BUGS

none

NAME

execprint - Project Management System "Activity Status Report"

SYNOPSIS

```
execprint
```

DESCRIPTION

This part of the Project Management System prints a hardcopy of report titled "Activity Status Report" from a print image file.

FILES

printout

SEE ALSO

execpms, execgen, budget[bar, plot], pert[bar, plot]

BUGS

none

NAME

execgen - Project Management System Graphics Generation

SYNOPSIS

```
execgen
```

DESCRIPTION

This portion of the Project Management System extracts from the Activity Master File and outputs three subfiles in the proper format for graphics generation. These subfiles are then copied (netcp) from the 8/32 to the Plexus for actual bar chart construction.

FILES

```
newfile
lablfile
budfile
pertfile
```

SEE ALSO

execpms, execprint, budget[bar, plot], pert[bar, plot]

BUGS

none

NAME

budget - Project Management System
Graphics Generation -
"Budget versus Remaining Manhours by Activity"

SYNOPSIS

```
budget[ options ]
```

DESCRIPTION

This part of the Project Management System constructs a bar chart of budget versus remaining manhours by activity. Options and their meanings are as follows:

bar - CRT image

plot - plotter hardcopy

FILES

lablfile
budfile

SEE ALSO

execpms, execgen, execprint, pert[bar, plot]

BUGS

none

NAME

```
 pert       - Project Management System
              Graphics Generation -
              "Budget versus PERT Time Estimates by Activity"
```

SYNOPSIS

```
                 pert[ options ]
```

DESCRIPTION

This part of the Project Management System constructs a bar chart of budget versus PERT time estimates by activity. Options and their meanings are as follows:

bar	- CRT image
plot	- plotter hardcopy

FILES

```
      lablfile
      pertfile
```

SEE ALSO

execpms, execgen, execprint, budget[bar, plot]

BUGS

none

AN IMPLEMENTATION OF A SOFTWARE ENGINEERING PROJECT MANAGEMENT SYSTEM: A TOOL FOR A PROTOTYPE SOFTWARE ENGINEERING ENVIRONMENT

by

OLIVER BERT CASTLE

B. S., Miami University, Oxford, Ohio, 1973

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1984

ABSTRACT

This Master's Report documents the rational, development, and implementation of a software engineering project management system that would provide features for a prototype of a software development environment suitable for the Japanese Fifth-Generation Computer Systems (FGCS) project, the United States Department of Defense (DoD) STARS (Software Technology supporting the development of Adaptable, Reliable Systems) program, or other future development environments.

The Project Management System is an interactive system that maintains a data base of manhours by project activities. The system calculates cumulative manhours to completion by activity and tracks remaining manhours by activity. In addition to generating an output activity status report, graphic displays are constructed on either a graphic display terminal or a plotter. Graphic displays include a bar chart of manhours allocation against remaining manhours by activity and a bar chart showing PERT time estimates to allocated manhours for each activity. The implementation tool is Berkeley Pascal under the UNIX* Operating System on

* A trademark of AT&T Bell Laboratories.

the 8/32 Computer at Kansas State University Computer Science Department.