1989

# Software cost estimation of the Advanced Training System project using three computer-based models

Douglas Poteat
*The University of Montana*

# Software Cost Estimation of the Advanced Training System Project Using Three Computer-Based Models

By

Douglas Poteat

B.S., Wofford College, 1974

Presented in Partial Fullfillment

of the Requirements for the Degree of

Master of Business Administration

University of Montana

1989

Approved by:

James E Novitzki

Chairman, Board of Examiners

E C Murray

Dean, Graduate School

June 12, 1989

Date




TABLE OF CONTENTS

# LIST OF ILLUSTRATIONS

LIST OF TABLES

# CHAPTER I

# INTRODUCTION

## Background

One of the critical challenges confronting the Department of Defense, from the present to the next century, is the proper management of the rapid growth area of software development. In the past the software acquisition process has been characterized by cost and schedule overruns. Original projections of cost are often based on traditional engineering projections of required manpower. Within the last few years, automated parametric models have been officially accepted as providing more consistent estimates.[1] The basis of these models is statistical research into previous development efforts. Defense contractor proposals are now including, for government evaluations, results from these parametric projections.

This paper provides an analysis using three automated costing models. The paper determines the range of cost for a new Advanced Training System (ATS) software development

[1]Elizabeth K. Bailey, Thomas P. Frazier, and John W. Bailey, A Descriptive Evaluation of Automated Software Cost-Estimation Models IDA Paper P-1979 (Washington, DC: Institute for Defense Analyses, October 1986), 37.

project. The new project will be managed at the Air Force Systems Command (AFSC) Human Systems Division located at Brooks Air Force Base, Texas.

## Purpose of the Research

The purpose of the study is to estimate range of costs of a software project, using three separate computer-based models. Previous efforts to estimate the cost of the Advanced Training System used only man-hour evaluations. The concern from the Air Force Cost Center is that these earlier assessments have not used recognized automated cost models, which are based on detailed studies of historical research and development (R&D) information. The results from the analysis should produce more consistent cost figures. The immediate usefulness of the paper will be valuable since it will serve as a basis for the fair market cost/price of the contract between the potential software developer and the Federal Government.

## Methodology

The method used consisted of obtaining from the assigned engineers and systems analysts, an explanation of the project in terms of "descriptors" used by each of the three computer-based models. Descriptors are the method of characterizing the software to the estimating models. The

descriptors are presented and defined in the sequence of input into the model. The other critical information needed for the parametric models is the estimated lines of software code. The procedure was to ask specific questions and to use survey type forms for a unanimous group decision. The result of the procedure was to obtain (1) the best judgment and, (2) the worst case input values.

## Sources of Data

Data gathering was limited to the above cited technical team assigned to the project. Expert advice on the computer models was obtained from the source of the models and the Air Force Cost Center.

## Limitations of the Investigation

The limits of the study may be grouped into three brief categories. These are the boundaries of the study, the uniqueness of Ada projects, and proprietary information.

As the title states, the required hardware for the Advanced Training System will not be estimated in this study. The focus is only on the development of the software. The ATS will use off-the-shelf equipment and these prices are relatively easy to confirm.

The ATS requires the use of Ada. The unique aspects of Ada software projects, as compared to other efforts using

different computer languages, are not fully known. Therefore, this area will not be addressed. All three models have been recommended by the Air Force Cost Center.

The equations used in PRICE-S and Softcost-Ada are proprietary. This is considered an impairment in reviewing the mathematical principles. Further, it is not possible to verify that the REVIC program correctly uses the Constructive Cost Model (COCOMO) estimating equations, since the source coding is not available.[2]

Another limitation was the large number of variables for a sensitivity analysis. A sensitivity analysis modifies one computer input value for the model, while holding other elements constant. The result is to observe the impact on the total cost. The process is repeated for all inputs. Due to the number of possible variables to be altered, especially with three models, another method was selected. The low/high alternative is perhaps a more useful way to obtain the cost range when multiple cost models are used.[3] The procedure is to specify the best judgment value, and the worst case. The worst case will display the higher cost.

---

[2]Common literature usage dictates "COCOMO model" even though the "MO" is referencing the word "model."

[3]Mike Helton, Software Cost Analyst, Air Force Cost Center, interviewed by author, 16 January 1989, Washington, D. C., telephone conversation, Air Force Cost Center, Washington, D. C.

The outcome is normally effective. Limited sensitivity analysis will be performed on the schedule of the project.

## Assumptions

The basic assumption was made that the low/high approach is adequate for the purpose of this paper, given that three estimating methods are employed.

An important assumption was also made regarding the lines of code (LOC). The assumption is the estimate by the engineers for the lines of code is correct. Almost all parametric models use projections for the amount of software code, to establish the magnitude of the project. A descriptive assessment of the software modules is provided in Chapter II.

## Organization of Paper

Chapter two describes the software project in concept and characteristics. Chapter three provides the first of three different software estimates, starting with the PRICE-S. Chapter four continues the study with the COCOMO/REVIC model. Chapter five provides an analysis using the Softcost-Ada Model. Chapter six compares the results, including the original engineering estimate. The final chapter is the summary of findings, conclusions, and recommendations.

# CHAPTER II

# DESCRIPTION OF THE SOFTWARE PROJECT

## General Concept

Air Training Command (ATC), the ultimate user of the new system, is one of the largest training organizations in the world. It is responsible for all Air Force technical training. Currently, the command teaches over 6,000 courses and trains 176,000 technical students per year. The ATC units affected by the Advanced Training System are at six geographically separated bases. The prime reason for the existence of these bases is the technical training.[4]

The current major problem of conducting training is the immense classroom instruction time required. As the technology level of aerospace equipment has increased, the complexity of the courses has also dramatically risen. In practical terms, the effect has been increasing job demands for maintaining accuracy in course content, developing newer courses, and actively instructing more material. For example, it has been estimated that between the years 1985

[4]Department of the Air Force, Headquarters Air Training Command, Operational Concept Document for the Advanced Training Systems ([San Antonio, Texas]: U.S. Department of the Air Force, Headquarters Air Training Command, 26 October 1987), 11.

and 2000 the training emphasis will shift to the applied mechanical and electrical engineering fields. This will account for a net increase of 30 percent in the number of required instructional hours.[5] Another consideration is the lead time for developing qualified instructors.

To respond to the challenge, the Air Force desires advanced automation incorporating the use of Ada to answer the future needs. Therefore, the objective of the Advanced Training System is to increase the effectiveness and efficiency of training provided by the Air Training Command.[6]

The programming of ATS will be capable of utilizing off-the-shelf computers, thus no hardware development is contemplated. The training system will serve as the comprehensive, state-of-the-art training system for the Air Force. It must also be entirely written in a new computer language, Ada. The Defense Department is mandating the Ada usage in all new programs. The hope is to reduce the resources required to support multiple computer languages, in the latter years of software projects.[7] The future

---

[5] Ibid., 11.

[6] Ibid., 9.

[7] S. D. Conte, H. E. Dunsmore, and V. Y. Shen, Software Engineering Metrics and Models (Reading, Massachusetts: The Benjamin/Cummings Publishing Company, Inc., 1986), 14-15.

maintenance expense of software represents a considerable percentage of the total software life cycle cost.[8]

One desired goal in the development of ATS is to build highly portable software. The concept of portability refers to software capable of being written once and moved to many different types of computers. To rewrite code or maintain multiple versions is a more costly endeavor. Minor interfacing software will be used and may require modification. Further, the system must employ a modular concept to allow easy tailoring and flexibility at the various bases.

The Advanced Training System must perform a variety of functions. First, it must provide an authoring system. The authoring system may be considered a word processing system with advanced capabilities. These features would include graphics, and other aids to the educational writer. The authoring system would be used to build basic instructional material. Second, it must enhance the management of trainees. This would include monitoring trends for individual students, as well as class performance. The generation, evaluation, and scoring of tests would also be required. Third, it must assist the management of training resources. The scheduling of instructors, classes, and

[8]C. R. Vick and C. V. Ramamoorthy, ed., Handbook of Software Engineering (New York: Van Nostrand Reinhold Company Inc., 1984), 197.

classrooms has always been a demanding task. The system should perform these examples of routine administrative tasks.[9]

## Specific Characteristics

The software must have a "System Level." This would almost be analogous to a Disk Operating System for a microcomputer. However, under the existing concept, this system will actually be operating in concert with the real disk operating system, Unix. Unix will be used at three levels of computers. The lowest level, used by a student or typical instructor, would be at a microcomputer. The middle layer would use a minicomputer to supervise base operations. A comparable concept would be the administration of a college. The overall monitoring device for the Air Training Command will be a mainframe computer. The System Level must provide a strong interfacing ability for the distributed data base. Interfacing would be software of one computer interacting with the software of another. A distributed data base is a network of computers in which each terminal can support, or at least update, information on the data base software.

---

[9] Operational Concept Document for the Advanced Training Systems, pp. 1-50.

The next functional delineation would be the "System Controls." This area of the software is really part of the System Level software. It was extracted for the analysis, because it represents a significant percentage of the lines of code with dissimilar characteristics. One major function of the System Controls is to provide security to insure decreased access at each of the lower levels. The security will require more data storage and retrieval than the System Level portion of ATS.

The third type of category is the "Management" software. Essentially, this portion of the software will direct almost all operations. One of the prime functions of the management software will be to develop a training plan for each discipline. The purpose of the plan is to insure complete coverage of school objectives on a scheduled basis. One may consider the analogy of a business school, tracking not only the subject of the course, but also the micro-detailed information. A specific example of the degree of detailed information would be investment finance theories. Last, it will update instructor records to document courses they have taught and annotate their educational background upon completion of additional training.

Fourth, "Student Management" will assign the student to various courses and track attendance. More importantly, it

will monitor the performance of each student, their permanent records (transcripts, etc.), and the performance of a particular class. The significant concept is that the software is student oriented.

"Authoring" is designed to be a desktop publishing system that is more comprehensive than word-processing. The current goal is to build a system for persons with a minimum knowledge of computers. It must be remembered the instructor will be an expert in his/her field, but not necessarily computer-oriented. Additionally, with the expected high turnover of military members due to relocation and retirement, the authoring system needs to be easy to use. In this module, the art of presentation will be optimized to achieve the greatest benefit to the student. A link to an "expert system" may be a future capability of the software. This approach would allow expertise of former instructors to be utilized.

"Delivery" will, simply stated, transmit the courseware and interact with the student. Courseware is the lesson plan presented from software in either text format, information pictorials, or self-paced tutorial tests. However, the conceived delivery software will perform more than communication services. With the application of artificial intelligence, the presentations will be modified based on the individual's previous performance. The

software will thus simulate the method an instructor would use. It should also restore the student to the ending location from a previous session, even on different computers at different bases. It must verify the students identification number against the data base to determine what access to courseware they may have. Finally, the software should update the data base to reflect the current progress of the student, including test scores and the position in training schedule.

"Testing" will present material to the student and it will be interactive. The software will orient the test question in various styles using artificial intelligence to increase the benefit to the student. For further test generation, the module will choose randomly from the pool of questions meeting the learning objective. Later, the Testing software can administer the tutorial lessons using computer-aided instruction, be "on-line" for actual tests, or produce copies of tests for the instructor. The variety of output devices would include simple laser printers and computer produced slides. Since the Testing module is test oriented, the history of the test question and the entire test will be produced along with the statistical information. The history would be maintained to increase the effectiveness of the lecture and to suggest possible changes.

"Evaluation" will be a unit of the software devoted to the analysis. Two types of analysis are needed: internal and external. Internal evaluation, as described by Air Training Command, is a review of how the students are performing on a selected training objective. External evaluation is examining the method of instruction. It is a procedure of appraising the training material and the presentation technique.

The final unit of software required is the "Data Base Management System." Since existing software is available to meet the distributed data base requirements, this item may be purchased. For purposes of the lines of code estimation, a commercial data base product, Oracle, was used as an analogous system.[10]

Table 1 displays the lines of software code that must be written for the final program, according to the above specific description of requirements. Additional information is presented by lower levels. The LOC is also estimated by units of 500 or 3000. However, the data does reflect the engineers' detailed reasoning in the lines of code estimate.

---

[10] Dan Conners, to Human Systems Division/YAT, 13 April 1987. Memorandum for the Record. Brooks Air Force Base, Texas.

Table 1
## ATS Module Breakout

| | LOC 500 | LOC 3000 | SUBTOTAL | TOTALS |
|---|---|---|---|---|
| **System Level** | | | | |
| Operating System Interface | 0 | 1 | 3,000 | |
| Ada Language Interface | 0 | 6 | 18,000 | |
| Ada Development Tools | 0 | 2 | 6,000 | |
| Network Interfacing | 0 | 0 | 0 | |
| Workstation Interfacing | 30 | 0 | 15,000 | |
| Database Report/Query Language | 3 | 2 | 7,500 | |
| Courseware Conversion | 0 | 2 | 6,000 | |
| | | | | 55,500 |
| **System Controls** | 67 | 5 | 48,500 | 48,500 |
| **Management** | | | | |
| Training Plan Development | 55 | 0 | 27,500 | |
| Resource Management | 12 | 0 | 6,000 | |
| Instructor Management | 6 | 0 | 3,000 | |
| Manage Course Material | 12 | 0 | 6,000 | |
| | | | | 42,500 |
| **Student Management** | 74 | 8 | 61,000 | 61,000 |
| **Authoring** | | | | |
| Support Inputs | 15 | 10 | 37,500 | |
| Intergrate Inputs | 3 | 1 | 4,500 | |
| Support Outputs | 21 | 5 | 25,500 | |
| | | | | 67,500 |
| **Delivery** | | | | |
| Computer Aided Instruction | 0 | 2 | 6,000 | |
| Database Interaction | 3 | 1 | 4,500 | |
| | | | | 10,500 |
| **Testing** | | | | |
| Test Item Pool | 5 | 2 | 8,500 | |
| Test Instrument Records | 6 | 1 | 6,000 | |
| Perform Tests On-Line | 2 | 0 | 1,000 | |
| Score Off-Line Tests | 1 | 0 | 500 | |
| | | | | 16,000 |
| **Evaluation** | | | | |
| Internal | 9 | 4 | 16,500 | |
| External | 1 | 4 | 12,500 | |
| | | | | 29,000 |
| **TOTAL** | | | | **330,500** |

* Note: LOC denotes Lines of Code

# CHAPTER III

# DEVELOPMENT OF THE PRICE-S ANALYSIS

## Background of the Model

The basic PRICE model for hardware was developed in the 1960's by the RCA Government Systems Division (GSD) as an internal management method to cross-check the conventional engineering assessment of costs. It rapidly became accepted as the standard throughout the GSD business units such as the Government Communications Systems, Missile and Surface Radar, Astro-Electronics and Automated Systems divisions of the company. With the apparently successful application of the PRICE model to several highly visible U.S. defense programs, including the Space Shuttle and the B-1 bomber electronics, its use spread to industry as well as the Federal Government. In 1975, RCA established PRICE Systems as a self contained business unit within GSD. Later, the scope of PRICE was expanded to include software (PRICE S), total life cycle support costs (hardware/PRICE HL, software/PRICE SL), and custom microelectronic chips (PRICE

M).[11] In 1987, PRICE Systems was acquired by General Electric.

Since the PRICE models are marketed only as proprietary items, detailed mathematical knowledge is closely guarded and unknown.[12] Other software costing experts also advise that few equations are provided for an open, debatable forum.[13] Using a parametric approach to estimating, the PRICE model cost estimates are not produced from labor tables, but rather equations at the historical perspective of projects. In essence, the modeling is based on Cost Estimating Relationships (CERs) that describe variations in observations. According to PRICE Systems, their model is more of a process approach rather than a data base where one "fits data." In this regard, PRICE S is a dynamic system.

The latest PRICE S model (released in January 1988) is a product resulting from several years of continued research and refinement. It contains, according to PRICE Systems, the known effects of the new Department of Defense Military Standard 2167A. The document, formally known as Defense System Software Development, is increasing costs for

---

[11] PRICE Parametric Cost Models: An Executive Guide (Cherry Hill, New Jersey: PRICE Systems, [undated]), pp. 1-4.

[12] Conte, 329.

[13] Martin L. Shooman, Software Engineering: Design, Reliability, and Management (New York: McGraw-Hill, Inc., 1983), 447.

software projects because of stringent documentation requirements. More importantly, for the immediate subject, the unique characteristics of Ada are also contained in the PRICE S model.

## Build-Up of the Input Values

### General Discussion

Appendix 1 contains questions and answers pertaining to the inputs of the PRICE S model. The number of PRICE inputs, or descriptors, are fewer for simple projects, but greater if the effort is complex. For example, a new project may involve multiple computer languages, commercial software, and existing programming which needs modification. One common requirement in all the parametric models is an estimate of the lines of code. It is by far the most important parameter in software cost projections.[14] One negative aspect of PRICE S is that some descriptors appear to overlap. This is particularly true in the skill of the assigned contractor personnel, both individually and as a software team.

The needs of the future software system are almost constant. The variables are the personnel, experience, and productivity of the winning contractor. The PRICE model concentrates these functions in only two inputs. For the

[14]Vick, pp. 470-482.

best judgment and worst case scenarios the Complexity One input and the Productivity Factor were changed. The impact of these adjustments approximate the variations made in the inputs of the other models. All remaining values for the PRICE S inputs remain the same.

Prior to specific inputs for the ATS project, a financial file is created to stipulate a general wage rate. The previous engineering estimate used the same amount, as did the other models in this paper. Exact labor categories and wages, such as management, engineering, and others, cannot be specified in the automated models. It is possible to alter the average number of hours worked per month, but this was not necessary since the PRICE S model defaults to 152 work hours per month. To establish a comparison, REVIC/COCOMO and the Softcost-Ada parametric model also used the 152 work hours per month.

For the actual input, PRICE S requires the software project to be entered as units of code known as boxes. The boxes correspond in name to the modules of software listed in Table 1 and Appendix 2.

## Specific Inputs

### Language

The first input to the PRICE S model is the type of computer language to be used for the software development.

Although the correct selection for our purpose is Ada, twenty choices are possible.

Source Lines of Code

Source Lines of Code (SLOC) describes the size of the project. The PRICE parametric model does not require the extensive information as presented Table 1, regarding units of 500 or 3000 lines of code.

Fraction

Fraction, or FRAC, is related to the lines of code. It is a percentage of the lines that are not instructions to the computers. These are either statements of data or informational notes for programmers, and are usually a small percentage. The principal concept is that these statements in the software do not require work effort. In the original process of estimating SLOC for all cost models and Appendix 2, the Fraction was assumed to be zero.

Complexity One

Complexity One (known as CPLX1) measures the product familiarity, personnel skills, software tool availability and changing requirements of the job. This input greatly affects the calculation of an optimal schedule. Product familiarity for ATS would mean the extent of corporate

experience with computer-based training and data bases. Some companies excel in writing certain types of software, but do poorly in developing other applications. With respect to the personnel perspective, the descriptor is quantifying the talent of the developers of the software. The software engineering team may be outstanding, average, or relatively inexperienced. Software tools are time saving devices. An analogous example would be the use of macros and other pre-existing templates in a personal computer spreadsheet package, such as Lotus 1-2-3. The user does not have to rebuild them each time. In a similar way, the programmer can use existing software programs. Examples of software tools would include the following:

(1) An editor to view and alter the contents of the software
(2) The debugger to locate and correct errors
(3) A compiler which translates the programming language into machine language
(4) An interpreter which executes the programming language as it reads them
(5) A linker which puts together programs from several sources

The availability of the software tools has a strong impact on the amount of labor needed for a project. The input for the Complexity One value is 1.1 for the best judgment. This describes a nominal value for an Ada project. For worst case value an adjustment was made to 1.3. The additional 20 percent represents a mix of experience in personnel talent, including hiring new employees for the project. Further,

the element of changing requirements is introduced. The parameter of 1.3 reflected the experience of the TRW corporation in recently completed Ada projects.[15]

Complexity Two

Complexity Two (CPLX2) provides the model with the complicating effects of the software and hardware interactions. The major reason for potential problems would be a concurrent software and hardware development effort. A nominal value of "1" is for software projects without complications. Increases are made for new hardware. Hardware developed in parallel demands even greater value increments. The ATS projects will use existing commercial computers, therefore the input is 1.

Productivity Factor

The efficiency, productivity, and skill levels are captured by the Productivity Factor or PROFAC. It describes the ability of the individuals or team assigned to the project. According to PRICE Systems, overlap does exist between Complexity One and the Productivity Factor. Both are important in determining the final cost of the development. The best judgment value of PROFAC was judged

[15] Jim Otte, Cost Analyst, PRICE Systems, interviewed by author, 24 February 1989, Dayton, Ohio, telephone conversation, Dayton, Ohio.

to be 5.2, while the worst case used 4.0. An inverse relationship is present in the grading scale. The range for commercial software projects is from 8.0 to 4.0. The PROFAC for aerospace applications begins at 5.0 and continues to 3.0. The inputs used in this study were based on current a verbal assessment from PRICE Systems.[16]

Mix, New Design, New Code

The three above descriptors are very related. All of them require seven numbers to input into the PRICE S model. Each of the seven positions represents a type of application of the software. The numbers correspond to a decimal fraction, with the entire line equal to one. The application ranges from simple data storage and retrieval to intricate mathematical operations. The amount of work to be accomplished for the design, and coding or programming varies for each type of application. The Mix input explains how the software is to be used. Complex software is more expensive than unsophisticated programs. The New Design conveys how much new architecture and engineering is needed. New Code indicates the amount of programming and typing that is necessary. It may be possible to use existing design and code from previous software projects. However, with the Advanced Training System, all new design and code is needed.

[16]Otte, interviewed by author, 24 February 1989.

Appendix 2 displays the modules of the software with the Mix, New Design, and New Code values converted into percentages of required application.

## Platform

Platform plays a major role in the computations of costs and schedules in the PRICE models. As one increases the importance of the operating environment, the software must be better. Therefore, platform is a function of the testing and ultimately, the contract. It is approximately equal to Required Software Reliability in the REVIC/COCOMO model. The eight possible values identify operating locations of the software. The Platform for the ATS project is identified as a Military-Specification (MIL-SPEC) ground site. The input value is 1.2.

## Management Complexity

The effects of developing software at multiple corporate locations and even multinational projects are noted by Management Complexity. Also known as CPLXM, this input is rarely changed from the normal value of 1. The ATS program will be constructed at one location.

Integration (Internal and External)

Integration, in creating software, is to combine smaller units of code into larger ones. The critical aspect of integration is insuring the unified program works correctly. The PRICE S parametric model has two entries for Integration. The difference is based on the idea of the Computer Software Configuration Item or CSCI. This is the level where software is managed and tracked, from the formal changes to the programming and documentation. It is a relatively new concept in software writing. Integration within a CSCI is internal. External Integration is the combining of two or more CSCIs. All integration for the ATS software was judged to be simple. The Integration value range is 0 to 1. Values less than .5 describe a non-difficult effort of uniting the software. Based on suggestions from a member of the General Electric PRICE staff, the internal value is .1 and the external number is .3.[17]

Utilization

Utilization (UTIL) describes the extra effort needed to overcome limited computer processor capabilities. It is

[17]Earl King, Senior Analyst, Operations Staff, PRICE Systems, interview by author, 16 May 1988, Moorestown, New Jersey, telephone conversation, PRICE Systems, Moorestown, New Jersey.

expressed as a decimal fraction of the total memory capacity used. The range of values is from greater than zero to less than 1. At amounts greater than .5, increases in cost can be expected. For ATS, the nominal value of .5 was assumed. Processor limitations for computer-based training will not exist.

### Schedule

The schedule for the baseline scenario (or base case) for all parametric models was the time calculated by the particular model. The calculated schedule is the optimal time for the type and amount of work to be performed. Theoretically, for software development, a reduction from the optimal schedule of time would increase costs. A sensitivity analysis of the schedule was later accomplished. The starting date for the PRICE S model was March 1989.

## Results of Analysis

### Dollar Amounts

### Best Judgment

The total for the software development effort, with the calculated schedule, is $42.5 million (constant 1989 dollars). The PRICE S model produces a variety of other information for the estimator. By using the various output options, the model can provide the financial requirements by

functional category. These categories would be roughly equivalent to departments in the corporation. Examples would include the senior engineering known as Systems Engineering and Program Management, and the design section. For the purpose of comparison, the dollars were converted to percentages. Table 2 contains these amounts. Definitions of categories are contained in Appendix 3.

TABLE 2

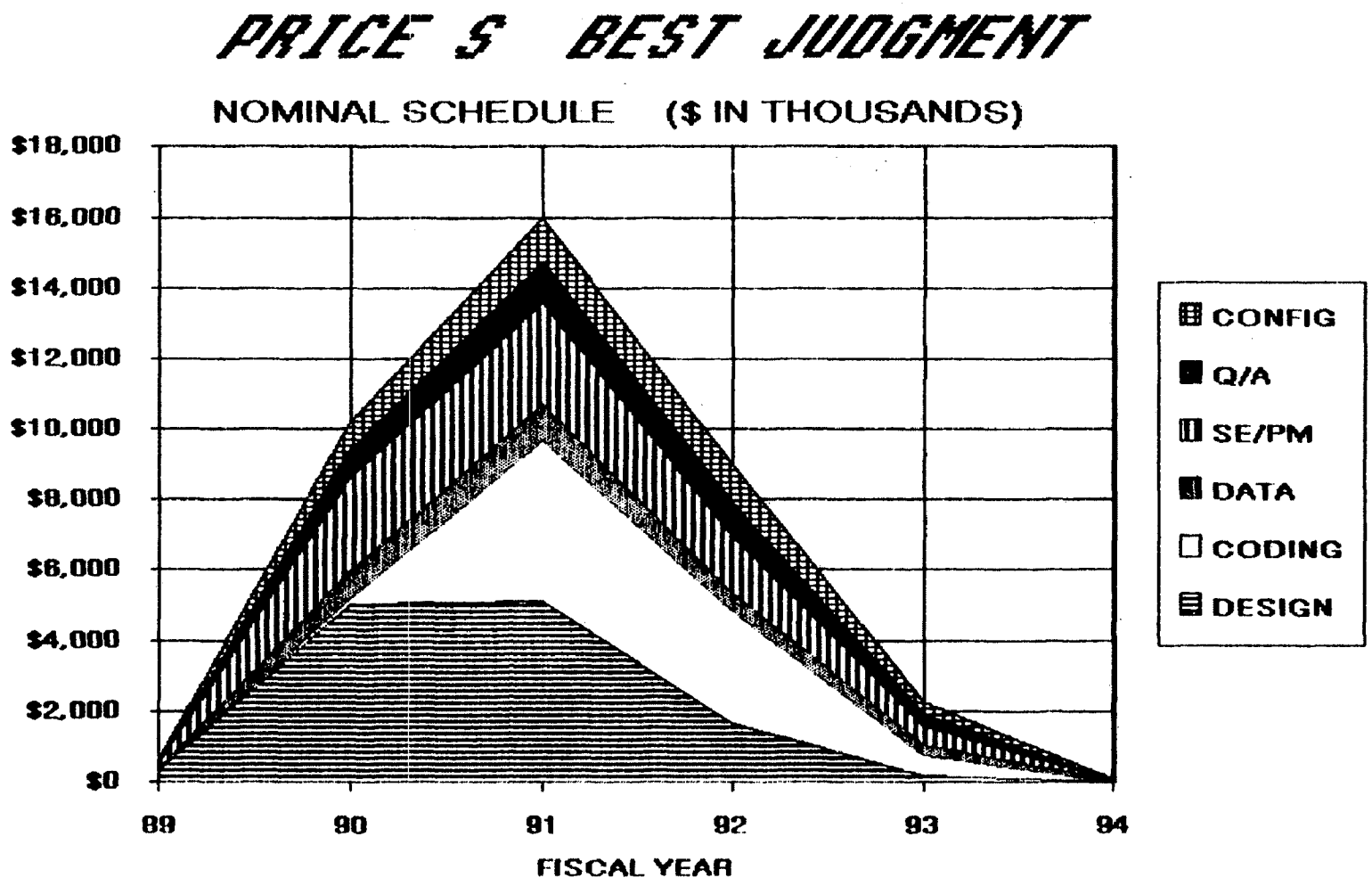
PRICE S BEST JUDGMENT AS A
PERCENT OF TOTAL DOLLARS BY FUNCTIONAL CATEGORY

| Functional Category | Percent of Total Dollars |
|---|---|
| Design | 33.8 |
| Coding/Programming | 21.8 |
| Data | 7.9 |
| Systems Engineering and Program Management | 18.8 |
| Quality Assurance | 8.7 |
| Configuration | 9.1 |

Figure 1 provides the profile of anticipated spending by functional category over the government fiscal years. It exhibits the estimated amounts of money spent for six functional areas from 1989 through 1994.

Other monetary data are also available in the PRICE S results for the phases of a software project. The phases of the effort normally proceed in a timely sequence. However, the exact duration of the phases will vary depending on the uniqueness of the software and the requirements. The later

Fig. 1


PRICE S BEST JUDGMENT
NOMINAL SCHEDULE ($ IN THOUSANDS)
$18,000
$16,000
$14,000
$12,000
$10,000
$8,000
$6,000
$4,000
$2,000
$0
89
90
91
92
93
94
FISCAL YEAR
CONFIG
Q/A
SE/PM
DATA
CODING
DESIGN

phases may have some concurrent actions. Table 3 provides the amounts required by phases for the best judgment parameters. Appendix 3 also lists the phases of a software project.

TABLE 3

PRICE S BEST JUDGMENT AS A
PERCENT OF TOTAL DOLLARS BY SOFTWARE PROJECT PHASE*

| Phase of the Project | Percent of Total Dollars |
|---|---|
| System Concept | 2.0 |
| System and Software Requirements | 3.0 |
| Software Requirements | 14.9 |
| Preliminary Design | 11.3 |
| Detailed Design | 18.0 |
| Code and Testing | 13.5 |
| CSCI Testing | 19.1 |
| System Testing | 6.0 |
| Operational Test and Evaluation | 2.7 |
| System Integration | 9.4 |

*Phases arranged in time sequence

Worst Case

With the worst case inputs, the cost model is forced to the higher amount of $61.6 million, or a 45 percent increase. The percentages of funding required for the functional categories, and by phases, changed only by insignificant amounts. These are depicted in tables 4 and 5, respectively. Figure 2 depicts the prospective worst case cost of functional categories. The areas of Design and Coding have particularly changed compared to Figure 1.

Fig. 2

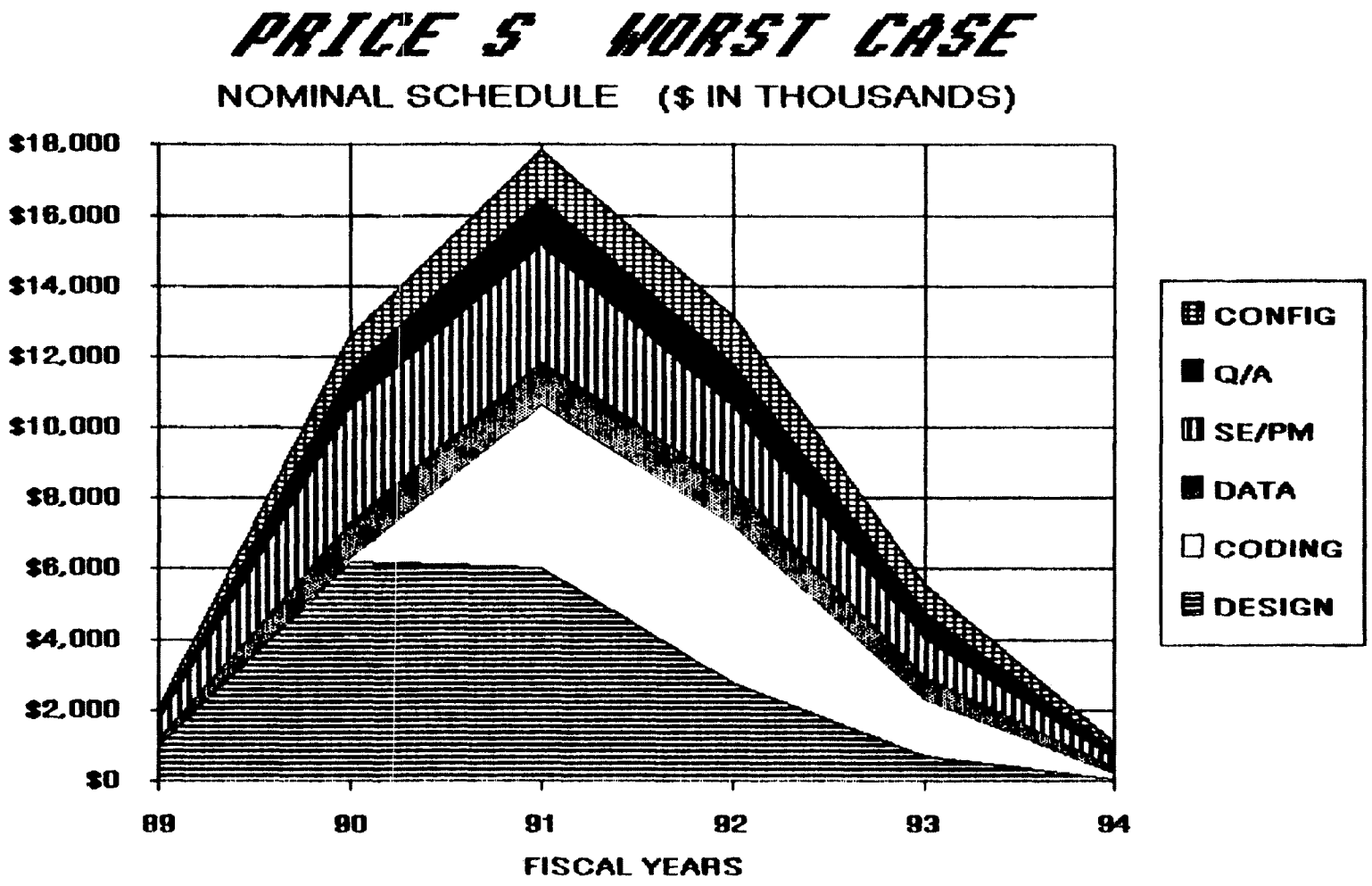

PRICE S WORST CASE
NOMINAL SCHEDULE ($ IN THOUSANDS)
$18,000
$16,000
$14,000
$12,000
$10,000
$8,000
$6,000
$4,000
$2,000
$0
89
90
91
92
93
94
FISCAL YEARS
CONFIG
Q/A
SE/PM
DATA
CODING
DESIGN

TABLE 4

PRICE S WORST CASE AS A
PERCENT OF TOTAL DOLLARS BY FUNCTIONAL CATEGORY

| Functional Category | Percent of Total Dollars |
|---|---|
| Design | 33.5 |
| Coding/Programming | 21.3 |
| Data | 8.1 |
| Systems Engineering and Program Management | 19.0 |
| Quality Assurance | 8.8 |
| Configuration | 9.2 |

TABLE 5

PRICE S WORST CASE AS A
PERCENT OF TOTAL DOLLARS BY SOFTWARE PROJECT PHASE

| Phase of the Project | Percent of Total Dollars |
|---|---|
| System Concept | 2.1 |
| System and Software Requirements | 3.2 |
| Software Requirements | 15.6 |
| Preliminary Design | 11.7 |
| Detailed Design | 18.5 |
| Code and Testing | 13.0 |
| CSCI Testing | 18.4 |
| System Testing | 5.8 |
| Operational Test and Evaluation | 2.6 |
| System Integration | 9.1 |

## Schedule

The parametric model, using the best judgment inputs, calculated an optimum schedule of 38 months for the project. The worst case descriptors caused the duration of the effort to increase by seven months, to a total of 45 months. This favorably compares to the program office schedule of 42

months. The desired schedule of the program did specify fixed periods for the phases.

## Effort

Effort is the total quantity of person-months in a project. The effort divided by the months of the schedule would equal the average number of individuals needed for a project. PRICE S estimates the best judgment would require 3,644 person-months. The worst case increases the requirement to 5,295.

## Productivity

Productivity is expressed in the PRICE model as an obscure factor. It ranges from 4.0 for complex military software, to 7.0 for commercial programming products. Most cost models for software estimate the lines of code produced per person-month. PRICE Systems indicates no mathematical relationship exists to successfully convert to the apparent standard measurement criteria.[18] The best judgment inputs for the ATS program indicated a Productivity Factor of 5.2. The worst case was calculated at 4.0. These lower values,

[18]Claude Wilton, Senior Analyst, West Coast Operations Staff, PRICE Systems, interview by author, 27 February 1989, Los Angeles, California, telephone conversation, PRICE Systems, Los Angeles, California.

suggesting intricate software, are within a valid range for a military Ada language program.[19]

## Sensitivity of Schedule

### Dollar Amounts and Schedule

To determine the sensitivity and impact of decreasing the schedule, varying percentages of the nominal schedule were imposed on all the parametric models. These reductions are to 95, 85 and 75 percent of the nominal amount. Further, a 42 month period desired by the Program Office was included in the sensitivity analysis. The effect of reducing the calculated schedule in the best and worst cases is shown in table 6, and graphically in Figure 3. As the schedule decreases in the PRICE S inputs, the costs for the project will generally increase. Minor changes in the schedule will also display a modest financial rise. However, the effect is lost due to rounding. An additional increase was encountered due to the Program Office schedule possessing fixed periods for the various software phases. The cost growth of roughly 39 percent, from the 95 percent of nominal schedule (43 months), to the duration requested by the Program office (42 months), can be attributed to the inefficiency of having a fixed schedule.

---

[19]Wilton, interviewed by author, 27 February 1989.

TABLE 6

PRICE S SCHEDULE SENSITIVITY RESULTS
(CONSTANT 1989 $ IN MILLIONS)

| | Best Judgment | | Worst Case | |
|---|---|---|---|---|
| | Dollars | Months | Dollars | Months |
| Nominal | $42.5 | 38 | $61.6 | 45 |
| 95% of Nominal | 42.5 | 36 | 61.6 | 43 |
| 85% of Nominal | 45.0 | 32 | 65.0 | 38 |
| 75% of Nominal | 50.1 | 29 | 72.9 | 34 |
| Program Office Schedule | 58.0 | 42 | 85.9 | 42* |

*Schedule with fixed time periods for phases of the project.

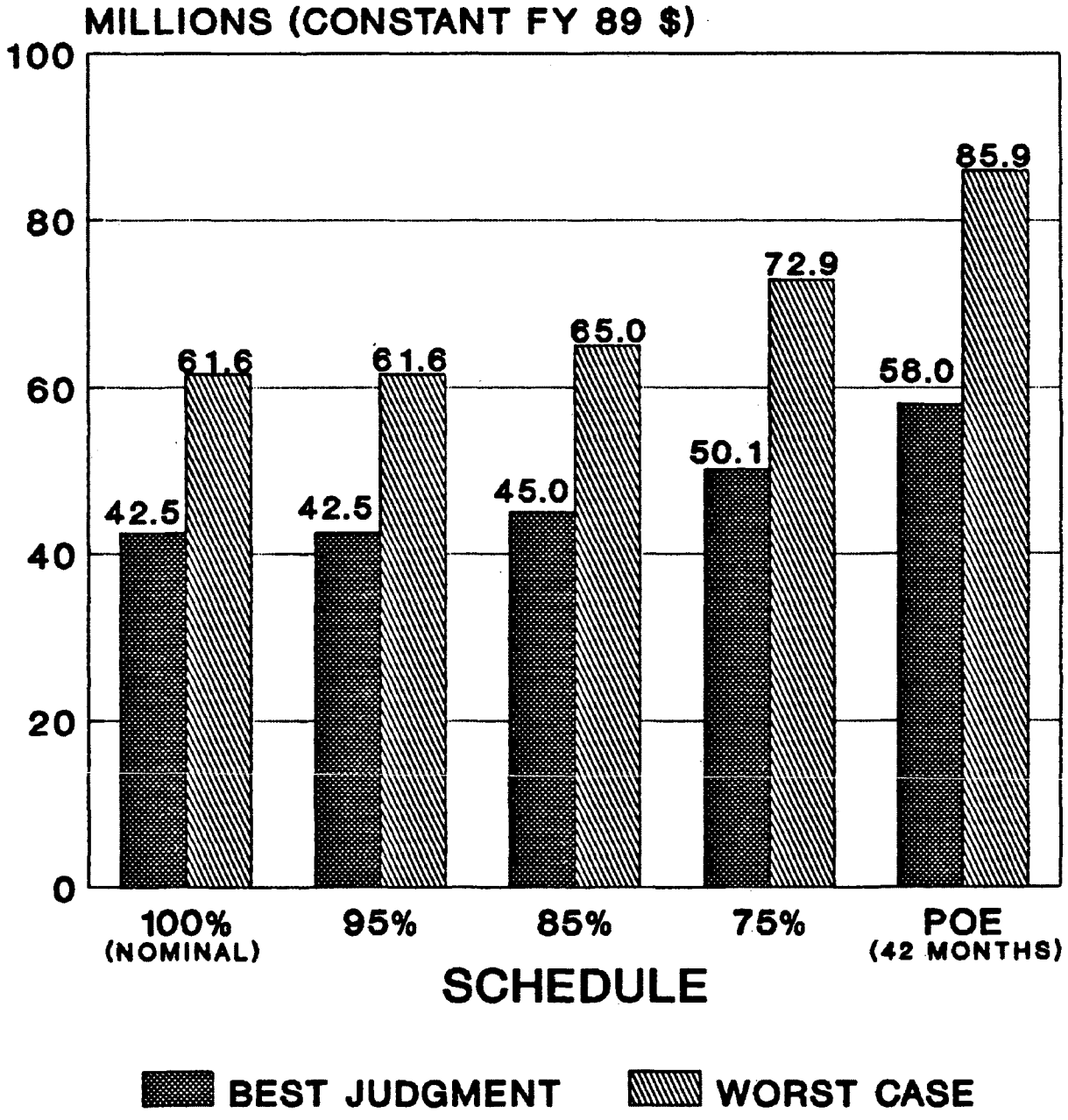
The trend with respect to the functional categories displays a consistent pattern in both the best judgment and worst cases. As the schedule is reduced, expenditures for Design, Data, and Systems Engineering and Program Management decline. Steady increases of two to three-tenths of a percent are noted in Quality Assurance costs. Advances of seven to eight-tenths of a percent occur regularly in funding needs for Configuration. The situation of short schedules apparently demands more rework in the area of information management and formal tracking of the project.

## Productivity

The Productivity Factor remains constant in the best judgment situations at 5.2. The worst case factor was

Fig. 3

# PRICE S
# SCHEDULE SENSITIVITY



stable at 4.0. Thus, no productivity loss or increase can be measured in a sensitivity analysis with PRICE S.

# CHAPTER IV

# DEVELOPMENT OF THE REVIC/COCOMO ANALYSIS

## Background of the Model

The REVIC computer program is an automated version of the COCOMO software cost estimating model. COCOMO was developed by Dr. Barry Boehm and published in his book Software Cost Economics in 1981. It is the most complete and well-documented of all the models. By strict interpretation, Dr. Boehm's model is not truly automated. It has three levels: basic, intermediate, and detailed. The REVIC program is based on the intermediate level. The COCOMO model is not formed by regression methods--but by experience, cost estimating relationships, and trial and error. The data base used to derive the model consisted of 63 programs written in different languages such as Fortran, Cobol, PL/1 and Joval from 1964 to 1979, primarily at TRW Systems, Inc. The program sizes varied from 2000 to 1,000,000 lines of code excluding comments.[20] The sample projects were divided into three separate complexities defined by product type, certain attributes of the project,

[20] Conte, 303.

and by the team's talent.[21] The software programs varied from the scientific and business areas to the supervisory and control type of software. The research into the cost drivers was based on the Delphi-type technique.[22] Although the model is heavily used, there has been an absence of published verification of the model using completed software projects.[23]

## Build-Up of the Input Values

### General Discussion

The survey for the REVIC/COCOMO cost model is attached as Appendix 4. With this parametric program, twenty-two inputs and the lines of code projections are necessary to estimate. Unlike PRICE S, the REVIC version of COCOMO has default multipliers. The default multipliers, or descriptors, are the nominal value. The projected lines of code by the module name are also needed. One sophisticated feature missing from the model is automated warnings. In PRICE S, the warnings indicate the values used are either too low or too high for the other given parameters. Additionally, the cost analyst cannot adjust or calibrate

---

[21] Tom DeMarco, Controlling Software Projects: Management, Measurement & Estimation. Foreword by Barry W. Boehm (New York: Yourdon, Inc., 1982), 163.

[22] Conte, 303.

[23] Ibid., 304.

the program based on historical information. This may be accomplished in PRICE but only with extensive data and parametric expertise.[24] All acronyms for the descriptors in REVIC are in capitalized letters.

## Specific Inputs

### Analyst Capability

The first REVIC input was the Analyst Capability or ACAP. It is an attempt to quantify team skills of the software engineers. The model stresses that the assessment is not simply a measure of the analysts' years of experience but effectiveness, as well. The analysts to which it is referring will plan the software architecture and produce the overall initial design requirements for the project. The possible value permits a total range of five selections from two lower categories--the nominal amount, and two stronger team skill appraisals. The chosen value for the best judgment, without the knowledge of who may win the contract, is the nominal amount. The worst case selection was determined to be low.

### Programming Team Capability

The next value to be judged was the capability of the programming team or PCAP. It is intended to be an

[24] Helton, interviewed by author, 16 January 1989.

evaluation of the programming team. The programming department will be the individuals who accomplish the detailed design after the preliminary design of the engineers. They also will write the actual code, and later, merge the various components of the code. The merging is the integration and testing phase. The five values are categorized by mathematical percentiles. The best judgment was assumed to be nominal. The higher cost producing worst scenario was critiqued as very low.

Project Application Experience

The Project Application Experience or AEXP is an assessment of the development and design team's familiarity with projects of this type. Specifically, this would reference computer-based training. The estimate is the average amount in years. The least input is less than four months, the nominal input is rated at three years, and the highest at twelve years. Due to the lack of broad computer aided training knowledge of the potential bidders, the nominal experience level was chosen from the possible five options for all cases.

Language Experience

The LEXP factor is used to record the programming crew's exposure to the computer language. Ada has been in

existence for several years, but until recently, waivers have been granted in the Defense Department for using other software languages. The effect is that the experience level in industry is not extremely high. The five ratings range from no experience to a maximum of more than two years. The nominal input was again selected for the best judgment and the worst case.

Execution Time Constraints

The computer, or more correctly, the central processing unit (CPU), alternates between the fetch cycle for locating the instructions, and the execution cycle where it performs the instructions. The execution is defined merely as the processing of the instruction. This input (TIME) measures as a percentage, the available time of the CPU that will be used by the software. Four selections are available from no restriction on execution time to a 60 to 95 percent utilization. With the higher percentage, the design of the code is more complex, requiring increased manpower in the project. For example, software for a fighter plane's radar electronics would need to be rapidly processed, thus the input value would be a high percentage. Computer-based training does not have timing constraints. The best judgment position, as well as the worst case, is the very low parameter.

Main Storage Constraints

The storage capacity of the computers that will ultimately use the newly designed software is also an important consideration in the estimate. When memory storage is limited, the software must be designed to operate with greater efficiency with respect to the amount of code. The reduction requires more effort in design and coding for either the initial design or a redesign, because the code is too large. Extra effort may be required in the quality control aspect of the project. Four possibilities are presented for the Main Storage Constraints input, STOR. The envisioned training system has no restrictions on memory. Both the expected and worst case input values are identical at nominal.

Virtual Machine Volatility

During the design of a software project an almost simultaneous hardware development effort can be underway. The Virtual Machine Volatility or VIRT is intended to ascertain how much change will be present in the design and development of the hardware. The frequency of these changes causes fluctuations in the software design. As more hardware modifications occur, increased manpower is consumed to maintain compatibility. The Advanced Training System will use existing commercial computers. Thus, the input of

machine volatility is very low for the best judgment and worst case.

Computer Turnaround Time

The factor of computer turnaround time (TURN) measures the intended computer response interval where the final software will operate. A lower rating would indicate a faster time for processing, either to print or perform other action. The range is from six minutes to a very slow time of twelve hours. The best judgment input for Advanced Training System is low. If one considers the student waiting for computer interaction, the worst case scenario is nominal. At the nominal selection, 60 percent of the computer processor time would be available for the software.

Requirement Volatility

The aspect of rework in software design is entered into the cost model by the requirements volatility (RVOL) factor. This input attempts to estimate the customer specified changes during the life of the software development project. Formally these changes are known as Engineering Change Proposals (ECPs). The impact can be significant in total cost increases. ECPs require changes in engineering manpower, time for management oversight, and the preparation of the ECP itself, including legal review. The customer in

the ATS program is the Air Training Command. Five choices are again present in the REVIC model from redirection to major redirections. The best judgment preference is high. The higher cost option for the worst case is very high.

Required Software Reliability

An increasing concern in the development of computer programs is required software reliability. The input of RELY will indicate to the model the appropriate level demanded. As the need for reliability increases, so does the human effort to test the software for problems, potential design flaws, and rework. The lower reliability would equate to a slight inconvenience if the software fails, as in internally produced testing programs. The stricter reliability would cover the spectrum from severe financial loss to potential life threatening situations, as in the software designed for nuclear power plant operations. The optimum reliability for computer-based training in both situations is nominal.

Data Base Size

The input for DATA to determine the design effects of large data bases that must be maintained and manipulated. The four choices are present from low to very high. The Advanced Training System will have extremely large and

multiple data bases. The best judgment and the worst case values must be very high.

Software Product Complexity

To quantify the degree of difficulty, the software product complexity (CPLX) value is employed. Six selections are possible. The low range presents merely simple computer routines. The midrange choices increase the mathematical requirements. The upper limit of complexity emphasizes scientific applications. The difficulty of intelligent computer-based training indicates high for best judgment and for the worst case.

Required Reusability

One of the basic reasons to shift to the use of Ada in the Department of Defense is the hope of producing reusable software. To construct computer code for reuse requires more labor in design and perhaps programming. The input for required reusability, or RUSE, has four levels. The range starts with no reuse, increasing ultimately to utilization of the software in any other project. The opinion regarding reuse of the ATS software dictates a best judgment position of high and a worst case position of very high.

Modern Programming Practices

Modern Programming Practices, entered as MODP in the computer model, signifies a management style to programming. The applied use of standardized planning techniques in the code, such as data flow diagrams and structured architecture, increases efficiency of the project. It also decreases concerns regarding compatibility of modules. Modularity is the programming technique of constructing software as several discrete parts.[25] MODP has five categories ranging from no use to routine application of the methods. The reason the best judgment selection is very high can be rationalized. The approach to writing Ada is modular. The worst case is high, since the winner of the ATS contract may not have fully implemented the modern management style to programming.

Use of Software Tools

As previously explained under the Complexity One input of the PRICE S model, software tools are labor saving utility computer programs. They are written for the language in development efforts. The cost estimator using the REVIC model has seven graduated choices for TOOL. The selection for both best judgment and worst case is nominal.

[25] Ibid., 197.

Classified Security Application

The model permits two options for classified security application (SECU), either unclassified or classified. The classified environment involves the "need to know" principle. Individuals are given information directly related to their work. Since compatibility is a function of information, a classified project is inherently more expensive due to waste or possible rework. A computer-based training project does not require a security classification. The best judgment and worst case opinions are identical.

Management Reserve for Risk

The input for management reserve (RISK) allows the operator of the model to insert a percentage for uncertainty. The parametric models of this paper are being calculated either without management reserve, or at very low if it is a mandatory input, for comparison purposes. In the final determined price of a project, management reserve may be added for contingencies.

Required Development Schedule

The REVIC model defaults to a nominal or an expected normal project development schedule. The input SCED may be used to force a compression of the nominal schedule, but not lower than 75 percent. In software development, a reduction

in the normal schedule will increase the cost. The best judgment will use the calculated schedule. The worst case will also employ the calculated schedule for those particular inputs. Lesser schedules for the effort were assumed in the sensitivity analysis.

Software Development Mode

The COCOMO model categorizes software development mode into three types. These are organic, embedded, and semi-detached. The organic is a smaller size software effort with relaxed schedule requirements. For example, software built for internal use by a corporation. Embedded software is at the other extreme. It is normally a large development process. The schedules are very demanding in this environment. The level or degree of innovation is very high. Semi-detached is a compromise position on the spectrum.[26] The REVIC adaptation of the COCOMO theories includes this feature. The best judgment and worst case is for the semi-detached mode.

Hours per Person-month

The REVIC model also permits flexibility in altering the person-month hours. The assumed hours per month are 152 unless the cost estimator changes the value. No change was

[26] Ibid., 301.

implemented for calculation regarding ATS. The yearly amount is equal to 1,824 hours. The upper limit of the model could be 248 hour per month, or a yearly total of 2,976.

### The Cost per Person-hour

An important constant in the comparison of the parametric models is the average cost per worker. The composite amount used in all the computer-based estimates and the previous person-hour approach is $71.41. All results, including the original program estimate, were later inflated using the approved Department of Defense inflation rates. For inflating constant year 1987 dollars to constant year 1989, the multiplicative factor is 1.072.[27] The model will otherwise assume an average total cost rate of about $135 thousand per year (a minor difference).

## Results of Analysis

### Dollar Amounts

### Best Judgment

The REVIC model estimates the Advanced Training System project will cost $29.3 million (constant 1989 dollars), at the nominal (calculated) schedule. Functional categories

[27] Letter from Mr. Joseph T. Wagner, HQ Air Force Systems Command (AFSC) /Cost Analysis (ACC) to all AFSC Product Divisions, 19 December 1988.

are not addressed in the results. However, the amount for phases as a percent of the total effort are calculated and presented in table 7. The names of the phases are different in REVIC than the Military Standard 2167A, and therefore, PRICE S. They are also reduced in number. For clarity, the military standard labels have been substituted in the table.

TABLE 7

REVIC/COCOMO BEST JUDGMENT AS A PERCENT OF TOTAL DOLLARS BY SOFTWARE PROJECT PHASE*

| Phase of the Project | Percent of Total Dollars |
|---|---|
| System Concept | N/A |
| System and Software Requirements | |
| Software Requirements | 9.0 |
| Preliminary Design | 17.2 |
| Detailed Design | 21.6 |
| Code and Testing | 16.4 |
| CSCI Testing | |
| System Testing | 16.4 |
| Operational Test and Evaluation | N/A |
| System Integration | 19.4 |

*Phases arranged in time sequence

Worst Case

The cost results for the Advanced Training System under the worst conditions, according to the REVIC/COCOMO model is

$109.7 million (constant 1989 dollars). This represents a substantial increase of 275 percent. The pattern of percentages in the software development phases remained constant.

## Schedule

The calculated schedule for the best judgment base case is 82.9 months, or almost twice the desired schedule of the ATS Program Office. The duration outcome for the worst case is 131.7 months. The variation is a four-year increase.

## Effort

Optimistically, the best judgment position indicates 2,513 person-months of effort. The staffing level under this environment would vary from 14 to 51 individuals. The worst case effort climbs to 9,428 person-months, with the minimum number of personnel more than doubling to 33. The growth rate of the maximum staff follows the identical ratio, producing the number of 122 persons.

## Productivity

The REVIC/COCOMO model measures productivity in Source Lines of Code per person-month. The number is for the average size of staff in the software producing firm directly working for the completion of the project. The

value for the Advanced Training System is 160 for the base case under the best judgment conditions. The worst case decreased the amount by 73 percent, to 42.8 lines of code per month.

## Sensitivity of the Schedule

### Dollar Amounts and Schedule

The effect of reducing the optimal calculated schedule is displayed in Table 8 (also see Figure 4). The table indicates the relative increases in cost in the best judgment and worst case are almost equal as a percentage. The model was unable to accept the Program Office estimate of 42 months for the project. The amount is less than 75 percent of the calculated schedule. The table also indicates the increases in the best judgment and worst case are almost equal as a percentage.

TABLE 8

REVIC/COCOMO SCHEDULE SENSITIVITY RESULTS
(CONSTANT 1989 $ IN MILLIONS)

| | Best Judgment | | Worst Case | |
|---|---|---|---|---|
| | Dollars | Months | Dollars | Months |
| Nominal | $29.3 | 82.9 | $109.7 | 131.7 |
| 95% of Nominal | 29.3 | 78.8 | 109.7 | 125.1 |
| 85% of Nominal | 31.6 | 70.4 | 135.0 | 111.9 |
| 75% of Nominal | 35.9 | 62.2 | 135.0 | 98.8 |
| Program Office Schedule | N/A* | 42** | N/A | 42** |

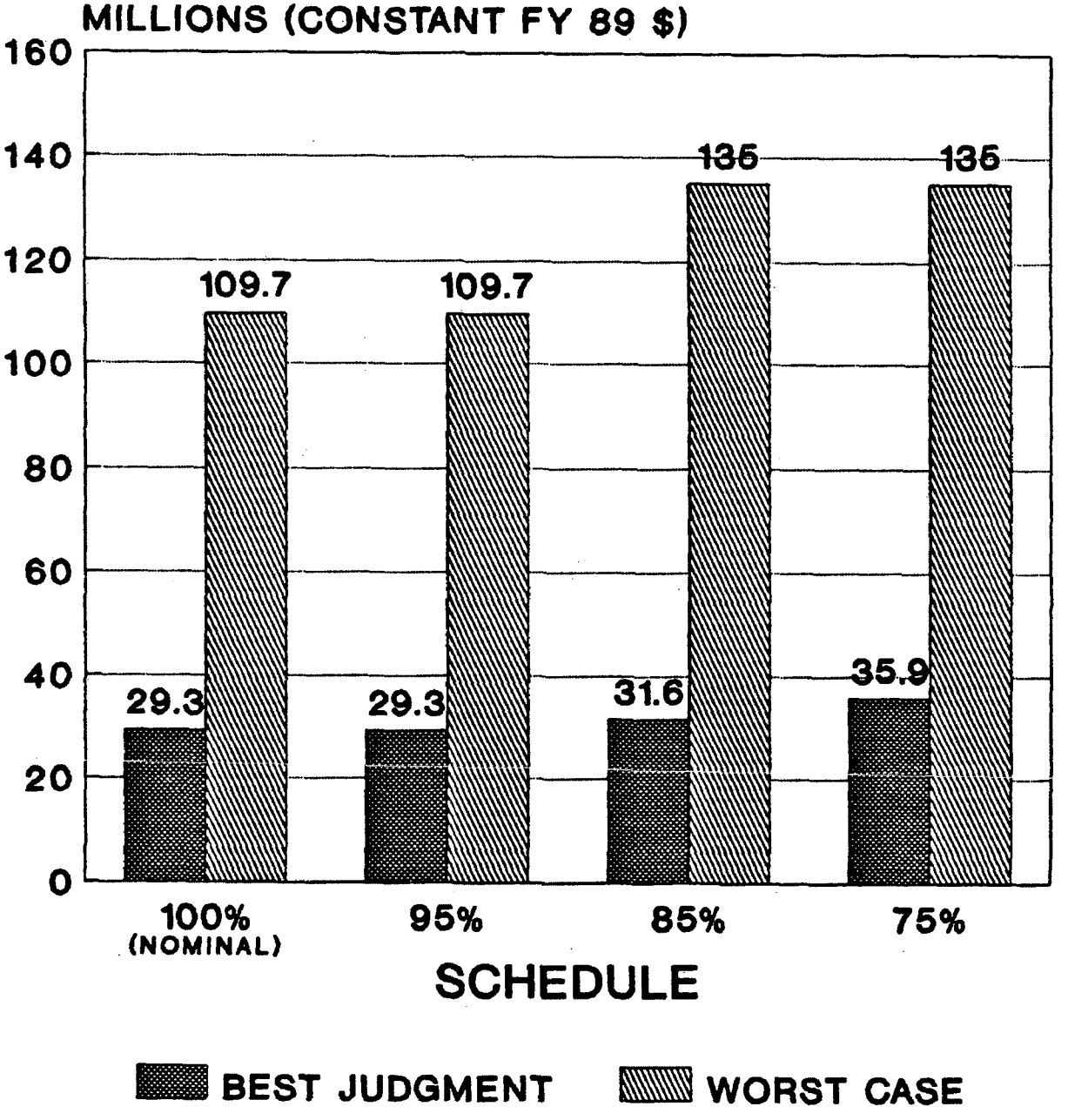
*The REVIC model is not capable of reducing the schedule below 75% of the nominal.
**Schedule with fixed time periods for phases of the project.

## Productivity

Under the best judgment conditions, the highest productivity was obtained at 160.5 lines of code per person-month. Compressing the schedule to 85 percent of the nominal decreased the productivity by 8 percent. At 75 percent of the nominal schedule, the productivity decreased by an additional 12 percent. The highest productivity rate for the worst case was 42.8 lines of code per month. Only one decline, at the 85 percent of the nominal schedule, was detected. The total decrease of the worst case was identical to the previous best judgment amount of 19 percent.

Fig. 4

# REVIC/COCOMO SCHEDULE SENSITIVITY



# CHAPTER V

# DEVELOPMENT OF THE SOFTCOST-ADA MODEL ANALYSIS

## Background of the Model

The Softcost-Ada cost estimating model is a product from Reifer Consultants, Inc. in Torrance, California. It was introduced in 1987 and is being used by 17 organizations in three nations. The program is compatible with International Business Machines' personal computers, using a Microsoft Disk Operating System (MS-DOS). Originally, the program was developed based upon a detailed statistical analysis of 75 completed Ada projects in five aerospace firms. Later, it was tested against a data base representing 12 million lines of delivered code.[28] According to Donald Reifer, the primary developer of the model, a basic assumption of the program is that cost decreases with the size of the software being developed due to productivity gains (in contrast with the COCOMO model).[29] Other models, including COCOMO, assume the costs will

---

[28]Personal letter from Douglas Willens, Marketing Director, Reifer Consultants, Inc., to author, 25 January 1989.

[29]Donald J. Reifer, "Ada's Impact: A Quantitative Assessment" (Torrance, California: Reifer Consultants, Inc., 10 September 1987), 8-9.

increase because of the added complexity of inter-group communication. Further, unlike COCOMO, Softcost-Ada assumes that the major factors which affect cost (cost drivers) do not act independently.[30] In Dr. Boehm's COCOMO model, and therefore the COCOMO-based REVIC automated version, the cost drivers are only multipliers.

## Build-Up of the Input Values

### General Discussion

The Softcost-Ada cost model survey is listed as Appendix 5. Twenty-eight inputs are presented to the estimator in four basic areas. The estimated amount of the lines of code are also required to reflect the magnitude of the project. The descriptors are very similar to the REVIC/COCOMO model. Unlike the REVIC, this model is oriented to Ada. Consequently, the survey was more meaningful to the government software personnel. Unlike the two previous parametric models, the Softcost-Ada descriptors or inputs do not have acronyms.

---

[30] Donald J. Reifer, "Softcost-Ada: User Experiences and Lessons Learned at the Age of One" (Torrance, California: Reifer Consultants, Inc., 15 May 1988), 7.

## Specific Inputs

### Type of Software System

The first input for the Softcost-Ada model is for the type of software. The types may be broadly grouped into five categories. The first includes military applications such as command and control, telecommunication, and avionics. This obviously represents the bulk of defense related work. However, the second type of scientific and testing software also share a large percent of the military budget. The third is utility software labeled as tools. This category consists of small programs. General automation and data processing systems are very large and are analogous to systems used by banks or similar institutions. The final category of "other" was suggested for computer-based training by the source of the model.[31]

### System Architecture

The second requirement for the Softcost-Ada model is to identify how the operating system will perform in relationship to the hardware and the type of data base. The input describes the anticipated hardware environment. The seven options of choices range from a centralized and single computer to a distributed data base with numerous computers. The Advanced Training System will have multiple layers of

[31]Helton, interviewed by author, 16 January 1989.

computers. Each layer, and almost each computer, will have a data base of files.

Number of Software Organizations

The next input for the Softcost-Ada model is the number of organizations that must interact for a successful project. As the number increases, the communication and time to communicate rises dramatically. Organizations include consultants, government test agencies, and customers. The numbers selected are three for best judgment, and four for worst case.

Organizational Interface Complexity

The degree of difficulty in dealing with the various organizational computer connections is the topic of this input. The complexity of communication grows as the number of interfaces increase. Further, the geography of the customer has an effect on the difficulty and cost of the project. The Advanced Training System will use multiple geographically distributed locations. The best judgment and the worst case impression indicate the use of the high and very high, respectively. Very high would be the most costly of the five choices.

Required Development Schedule

Softcost-Ada also allows flexibility in the approach to the anticipated schedule. The middle position of the five values is the nominal schedule. No documentation is provided to determine the definition of the average time. However, one may alter the model to compress the amount to 85 or even 75 percent of the nominal schedule. The converse is also available at 120 and the upper limit of 130 percent of the average. Compressing the schedule to less than the normal time, requires tasks to be performed before all information is available. Effort must be later expended to insure compatibility in the software project. If the schedule is lengthened, work may require more time than usual without any improvement in software features quality. The best judgment position and the worst case used the calculated schedule produced by the model. The compressed schedules are reserved for the sensitivity analysis.

Resource Availability

The resource availability input recognizes the limitation of assets committed to one project. The range of five possibilities covers the variety from austere facilities, equipment, and staff, to a technology-enriched office and a very capable software department. The increased communication of local area networks is identified

in the upper two choices. Improved facilities and especially increased staff capabilities result in a more efficient operation and a lower ultimate cost.

Security Requirements

Reifer Consultants have interpreted security requirements in their model to be a description of the final software and physical security. This is contrast with the simple COCOMO/REVIC viewpoint of classified or unclassified. With six selections, one may choose none, from a range of increasing security levels. Computer-based training should have secure data bases. No other security requirement exists. The nominal value is the preference for both the best judgment decision and the worst case.

Degree of Standardization

Since this particular model is tailored for the Ada language, the level of programming standardization is required. A current problem in the industry, regarding Ada programming, is standardization. The upper levels of choice in the cost model have very detailed military guidelines. Commercial standards are considered the normal level of use. The required amount of standardization is high for both the best judgment condition and the worst case.

Scope of Support

The cost aspects of customer relations and product support are conveyed to the model with the scope of support. With four degrees of support, only the maximum level of support is considered for both the high and low estimates. The Federal Government requires extensive cost and schedule reporting on contracts other than Firm Fixed Price (FFP) agreements. The ATS contract will not be a FFP situation; therefore, the highest rating identifies these mandatory requirements.

Use of Modern Software Methods

Regarding use of Modern Software Methods, the Softcost-Ada descriptor is equivalent to Modern Programming Practices in the REVIC model. The possible number of selections are the same. The best judgment and worst case positions for both models remain constant at high and very high, respectively.

Use of Peer Reviews

Quality control is the subject of peer reviews. The government has formalized the review process to include multiple management and technical evaluations in the area of software inspections. The management oversight would inquire into established company procedures. For example,

questions might be asked such as, "How is work actually authorized?" Further, "Are expenses, which are pooled into overhead, allowed by the Federal Government for defense related work?" The technical inspection would be in-depth to the extent of comparing the existing military standards to statements of work in the contract. The only preference suitable for both best judgment and the worst situation is the highest review criteria.

Use of Software Tools/Environments

The Use of Software Tools is present (as in the COCOMO/REVIC program) with six possible options. The model, being specifically for Ada projects, highlights more technical terms. The word "environment" simply implies an automated surrounding of software tools. The selections are nominal for both.

Software Tool/Environment Stability

The Ada programming environment is currently evolving with better software tools. The problem in the past has been a capable compiler (a software tool) for the computer language. The primary reason is the relatively young age of Ada. The input of stability measures two aspects in the possible six selections. First, it measures the rework that is required because of defective software tools. Secondly,

the model quantifies the added cost from the inefficiency perspective, i.e., the extra work that is needed. The appraisal for the tool/environment stability is very high for best judgment. The conjecture for the worst position is high. Greater stability indicates reduced costs for the project.

Ada Usage Factor

Ada is anticipated to have a higher initial cost than other computer languages. The usage factor addresses the possibility that multiple languages may be used in the development of the software programming. Although the five choices range from 50 to 100 percent, the Advanced Training System is required to be written totally in Ada. The very high factor is the only option that applies.

Product Complexity

The Product Complexity parallels the Software Product Complexity of the previous model. Potential software is divided into six increasingly difficult mathematics and logic routines, according to Softcost-Ada. As compared to COCOMO/REVIC, this model clarifies the explanations of the logic and library descriptors. Further, the Reifer model amplifies awareness to time sensitive and concurrent tasks

performed by the software. The ATS programming will be in the extra high range of complexity.

Requirements Volatility

Requirements Volatility possesses the same meaning as the phrase in the REVIC parametric model. The current model also describes the five selections, but in percentages of known requirements versus more general expressions of change. The selected inputs are equivalent. The best judgment and worst cases are high with the possibility of great change in requirements.

Degree of Optimization

The amount of optimization is similar to the REVIC parameter of "Time." It measures the efficient use of the processor. The effectiveness of the central processing unit (CPU) is rated as a fraction of the potential full use. Increased work in the software design and coding should produce a more efficient use of the processor. The best judgment recommendation is nominal. The worst case was also viewed as being nominal. If more processor capacity is needed, off-the-shelf computers are viewed as the less expensive alternative to more software design effort.

Degree of Real-Time

Real Time can have slight variations in definition. As applied by Softcost-Ada, it refers to the programming reacting so rapidly that it depends on other instantaneous software operations to perform well. Five alternatives are given to describe the type of desired software. The relationship of work effort to the Degree of Real Time in software is simple. Real time requires more labor and more talent. The best judgment selection is nominal, and the worst case is identical.

Degree of Reuse

Inherently, the language of Ada is structured for ease of "reuse." It is possible, however, to design and write the coding for easier reuse. The additional cost for the planned modular coding could be viewed as an investment. For computer-based training the situation may not immediately demand the anticipated reuse. However, designing software for reuse in later training programs may be more economical for the future. Of the five alternatives, the selection for best judgment and worst case is extra high.

Data base Size

Data base size is also an input for the Ada specific model. The method for calculating the size is the same, as well as the possible range in the selection choices. The proper decision is to use only the largest size for the ATS program in the low and high predictions.

Ada Experience Profile

The input for experience in the computer language is similar to the REVIC Language Experience factor. The intent is to gauge the effectiveness of the software team. However, the REVIC/COCOMO measured only time. The Softcost-Ada method to determine experience is to use the number of completed projects in the Ada language. A stipulation in the ATS contract requires successful completion of three projects.

Analyst Capability

This is identical to the previous parametric model. With the classification divided by mathematical percentile, the government technical team chose the nominal value for the best judgment. For the worst case, the very low value was selected. Both favorably compare to the REVIC model's inputs.

Applications Experience

Application Experience parallels the REVIC/COCOMO model. The inputs are duplicated at nominal for the best case and low for the worst scenario.

Ada Environment Experience

No equivalent exists in the previous cost model, or specifically in PRICE S, for familiarity with the software tools. These utility programs are collectively known as an environment. The best judgment position was estimated at nominal. The worst situation would only merit a low rating.

Ada Language Experience

Softcost-Ada measures the duration the average software worker has been writing with the Ada language, similar to the method used in REVIC/COCOMO cost model. The experience input, to some degree, appears redundant to the following parameter of methodology. The government software specialists recommended nominal and low for the best/worst positions, respectively.

Ada Methodology Experience

Software engineering studies suggest the development process of Ada efforts is unlike other software languages. Exposure to the design methods significantly reduces total

program costs. The effect has been compared to the learning curve phenomenon taught in business courses. Of the three automated cost models, the input for Ada Methodology Experience is the most distinctive. Curiously, the choice for experience is not clear in months and years of experience. However, a technical paper authored by Donald Reifer, the ultimate source of the Softcost-Ada model, recognizes the number of completed projects--a different criterion. Only the nominal option was selected for both low and high cost conditions.

### Team Capability

Although Team Capability in the COCOMO/REVIC parametric model identifies only programming teams, the Reifer Consultants cost model recognizes more. The higher values are assigned for participatory and interdisciplinary team approaches to problem solving. It identifies a highly involved style of software engineering in the development process. The nominal value was selected for both cases.

## Results of Analysis

### Dollar Amounts

### Best Judgment

The estimate for the best judgment, with the calculated schedule, is $32.3 million (constant 1989 dollars). The

model does not precisely estimate functional categories, such as design or programming. It allocates percentages to four areas. These are development, management, configuration, and quality. Management is 10 percent of development, with the others being approximately 5 and 6 percent, respectively. The phases of the project are also identified differently from the Military Standard 2167A, PRICE S, or the REVIC/COCOMO model. For clearness, the Softcost-Ada phases in Table 9 are approximately listed in the format of the military standard.

TABLE 9

SOFTCOST-ADA BEST JUDGMENT AS A
PERCENT OF TOTAL DOLLARS BY SOFTWARE PROJECT PHASE*+

| Phase of the Project | Percent of Total Dollars |
|---|---|
| System Concept | N/A |
| System and Software Requirements<br>Software Requirements<br>Preliminary Design<br>Detailed Design | 50.0 |
| Code and Testing | 15.0 |
| CSCI Testing<br>System Testing | 35.0 |
| Operational Test and Evaluation<br>System Integration | N/A |

*Phases arranged in time sequence.
+According to Reifer Consultants, these percentages, as calculated by the model, do not vary under any circumstances.[32]

Worst Case

The worst case descriptors produced an amount of $71.9 million (constant 1989 dollars). The total is 2.24 times the best judgment scenario. The functional areas and phases did not change in the allocation of required financial resources.

[32]Pat Kane, Software Engineer, Reifer Consultants, Inc., interviewed by author, 7 March 1989, Torrance, California, telephone conversation, Torrance, California.

## Schedule

Softcost-Ada forecasts a schedule of 60.1 months for the best judgment values. The inputs for the worst case increase the total software development time to 80.9 months.

## Effort

The best judgment result for the person-months of effort is 2,774.9. The average staffing level is over 46 persons. The worst case values altered the amount to 6,182 months of work. The higher average staffing level is 76 individuals. A major feature of Softcost-Ada is a tabular matrix for the relationship of effort and confidence levels. The model theorizes as the number of person-months increase, a greater confidence is expected to complete the work on the proper schedule. Surprisingly, the calculated schedules for both the low and high scenarios have a confidence of less than 50 percent. The maximum level of confidence is approximately 85 percent.

## Productivity

The values of productivity are extremely similar to the REVIC/COCOMO amounts. The lines of code per person-month are 106 and 47.6 for the best and worst cases, respectively. However, Softcost-Ada actually provides a range. For each schedule adjustment the range scale will vary. Further,

each value in the range has a probability. The probability appears to be a function of the effort. As the number of person-months of effort increases, the confidence level grows, but the effectiveness of the average employee decreases.

## Sensitivity of the Schedule

### Dollar Amounts and Schedule

The calculated schedule in the Softcost-Ada model was reduced to analyze the changes in cost. The outcome is presented in table 10, as well as Figure 5. The increase in cost, as the schedule decreases to 75 percent of the nominal amount, is greater as a percent in the best judgment scenario. The situation reverses when the Program Office estimate of 42 months is used.

TABLE 10

SOFTCOST-ADA SCHEDULE SENSITIVITY RESULTS
(CONSTANT 1989 $ IN MILLIONS)

| | Best Judgment | | Worst Case | |
|---|---|---|---|---|
| | Dollars | Months | Dollars | Months |
| Nominal | $32.3 | 60.1 | $71.9 | 80.9 |
| 95% of Nominal | 37.0 | 57.1 | 81.3 | 76.9 |
| 85% of Nominal | 39.6 | 51.1 | 86.8 | 68.8 |
| 75% of Nominal | 42.5 | 45.1 | 93.2 | 60.7 |
| Program Office Schedule | 44.4 | 42.0 | 115.7 | 42* |

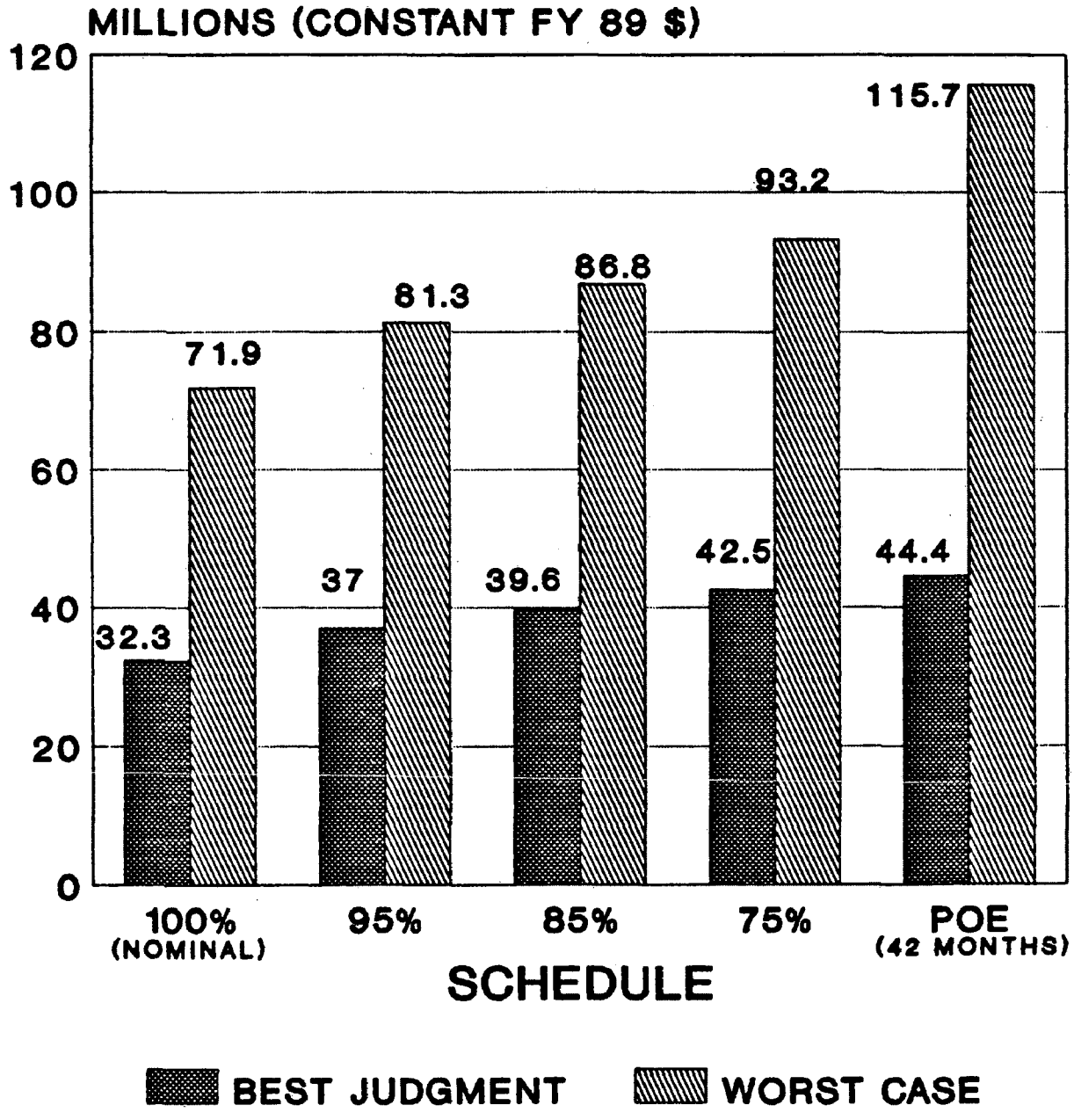
*Schedule with fixed time periods for phases of the project.

## Productivity

The highest productivity with the best judgment was the previously mentioned 106 lines of code per person-month. The amount decreased by roughly 6 percent with each reduction in the calculated schedule. The final productivity loss was only 3 percent from the 75 percent of nominal schedule to the Program Office desired duration for the program. The worst case productivity was only 47 lines of code per month for the nominal, and 95 percent of nominal schedule. The rate of decrease in productivity parallels the best judgment outcome with one exception. The exception is from the 75 percent of nominal to the 42 months requested by the Program Office. The change represents a 20 percent drop.

Fig. 5

# SOFTCOST-ADA
## SCHEDULE SENSITIVITY



# CHAPTER VI

# COMPARISON OF THE RESULTS

## Summarization of Inputs

The last three chapters have shown some similarities and many differences with respect to the inputs for the three cost models examined. Table 11 highlights some of the major differences in input between the models. The software descriptors for PRICE S are unique, unlike Softcost-Ada which is similar in approach to the COCOMO-based REVIC model. The latter parametric models have numerous descriptors which serve as simple multipliers. All models use the major cost driver of lines of code for the determination of project magnitude. Reliability is also a common characteristic with the models merely having a different name for their input. Another principle in the cost consideration is schedule. The PRICE model requires the start date only or the start date plus the details for any of the phases. The REVIC and Softcost-Ada models require only an expert opinion as to the degree of schedule compression. One must compare the schedule calculated by the model to the desired time in months. The inputs for personnel in the PRICE model are consolidated into two

Table 11

## OVERALL SOFTWARE COST MODEL INPUT COMPARISON

### Similiar Characteristics

| Categories | PRICE S | REVIC | Softcost-Ada |
|---|---|---|---|
| **Major Cost Driver** | Lines of Code | Lines of Code | Lines of Code |
| **Reliability** | Platform | Required Reliability | Product Complexity |
| **Hardware Limitation** | Utilization | Main Storage and Execution Time Time Constraints | Degree of Optimization |

Table 11-Continued.

OVERALL SOFTWARE COST MODEL INPUT COMPARISON

Dissimiliar Characteristics

| Categories | PRICE S | REVIC | Softcost-Ada |
|---|---|---|---|
| **Schedule** | Actual Dates: Start date only, or with more information | Expert Opinion | Expert Opinion |
| **Personnel** | PROFAC and Complexity One | Analyst Capability, Team Capability, Application and Language Experience | Multiple experience factors (Language, Methodology, etc.) |
| **Mix of the Type of Code** | Described by Percentage | N/A | N/A |
| **General Hardware Description** | N/A | N/A | System Architecture |
| **Management Reserve** | Optional (Actual Percentage) | Required (not Percentage) | Required (not Percentage) |

parameters. The other models need more parameters to describe the capability of the programmers and engineers. Various types of software are inherently more costly than others. PRICE S addresses the issue; however, the REVIC and Softcost-Ada models indicate the impact is minimal or cannot be calculated.

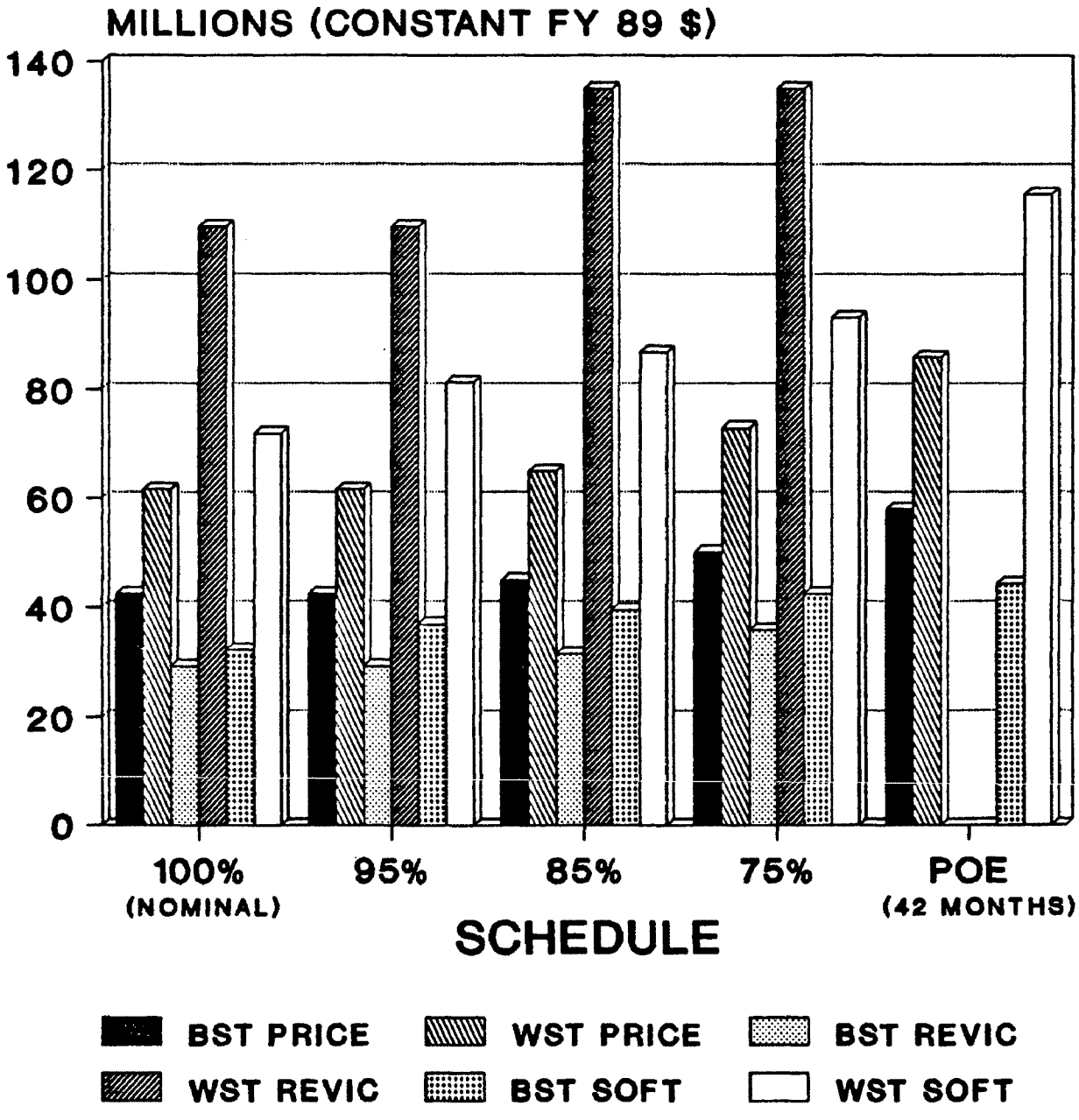
## Dollar Amounts

The variation in final totals of the software models is surprising, as a reasonable cost range does not emerge. The best judgment inputs produced the smallest size of variation at $16.2 million. The highest amount of $42.5 million, calculated by the PRICE S program, is almost twice the amount estimated by the REVIC version of COCOMO. In contrast, Softcost-Ada is only 10 percent more than the REVIC. The worst case distribution is a significantly larger variation of $47.8 million. However, the PRICE and REVIC/COCOMO models reverse the previous order. PRICE S assumes the lower estimated position at $61.6 million, while REVIC is 78 percent more ($109.7 million). Softcost-Ada retains the middle position at 16.7 percent above the PRICE S generated amount.

Figure 6 depicts a global view of the cost analysis of all models for both scenarios. Under any circumstances costs rise as the schedule is decreased. PRICE S is

Fig. 6

# PRICE S/REVIC/SOFTCOST
## BEST JUDGMENT/WORST CASE



consistently the highest for the best judgement and constantly the lowest for the worse case. The REVIC/COCOMO model results indicate the lower cost projections for the best judgement set of assumptions. It also displays extreme sensitivity to the parameters for the worse case by producing exceptionally high forecasts. The REVIC model was not accomplished for the Program Office Estimate of 42 months due to limitations of the computer program.

Another element in the comparison of dollar amounts is the allocation of financial resources in the various phases of a software project. The models are relatively consistent in the dispersion of money as a percentage (reference Table 12). However, the models do not consistently address all stages of software development identified in the Military Standard 2167A. Therefore, the phases must be grouped to establish any relationship. Further, only PRICE S estimates the System Concept phase and the Operational Test and Evaluation period of the project. The other models assume these costs do not require forecasting.

TABLE 12

COMPARISON OF GROUPED PHASES OF A SOFTWARE PROJECT AS A PERCENT OF TOTAL DOLLARS ESTIMATED

| Phases* | Parametric Models | | |
|---|---|---|---|
| | PRICE S** Best/Worst | REVIC | Softcost |
| System and Software Requirements<br>Software Requirements<br>Preliminary Design<br>Detailed Design | 47.2/49 | 47.8 | 50 |
| Code and Testing | 13.5/13 | 16.4 | 15 |

*Phases defined in Military Standard 2167A accounting for the remaining percentages would include:
- Systems Concept
- CSCI Testing
- System Testing
- Operational Testing and Evaluation
- Systems Integration

**Only PRICE S has different percentages for the best judgment and worst case inputs.

Insufficient information is available from all the models to compare the allocation of money to the various functional categories of labor.

## Schedule

PRICE S consistently calculates the shortest schedule of the models. It also has the smallest difference between

the best judgment and worst case situations at 18 percent (reference Figures 7 and 8). The REVIC parametric model possesses the longest schedule in both scenarios. Further, the variation in the magnitude of the results, at 59 percent, is also the greatest. Softcost-Ada estimates are between the two extremes for the schedule length. Additionally, its low and high differences are in the moderate position.

The Program Office schedule of 42 months is supported by PRICE S. Radical results opposing this shorter project duration exist with the other parametric models. With respect to the schedule analysis, a recognizable relationship between the financial amount estimated and the calculated schedules does not exist.

## Effort

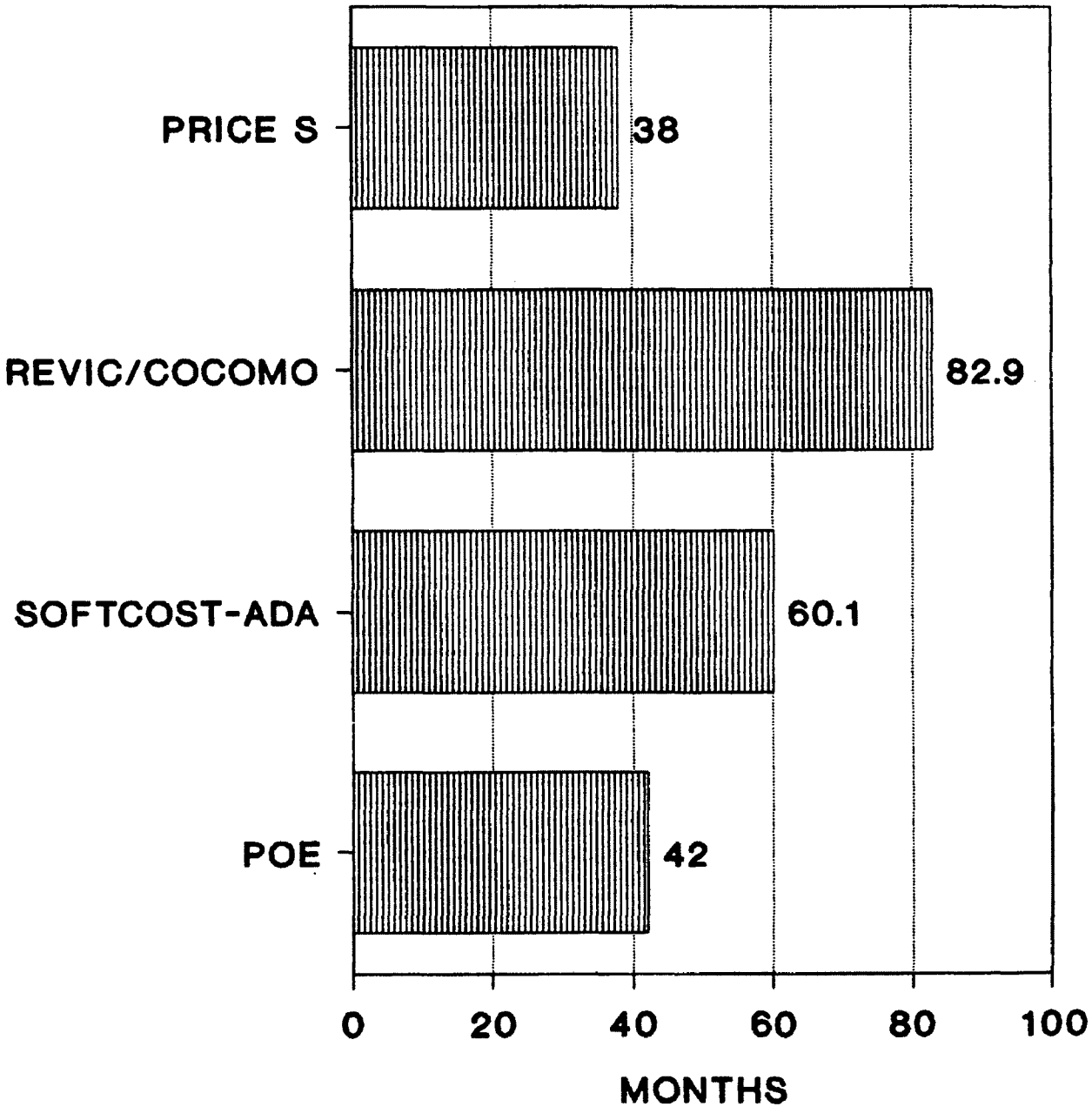
Effort is another expression of cost. It is simply a measurement in person-hours. The relationship of effort in the three models is identical in comparison to the dollar amounts.

## Productivity

The PRICE S Productivity Factor cannot be used in a direct comparison. The concept of the PRICE S factor is useful, but only in relation to other PRICE estimates. The

Fig. 7

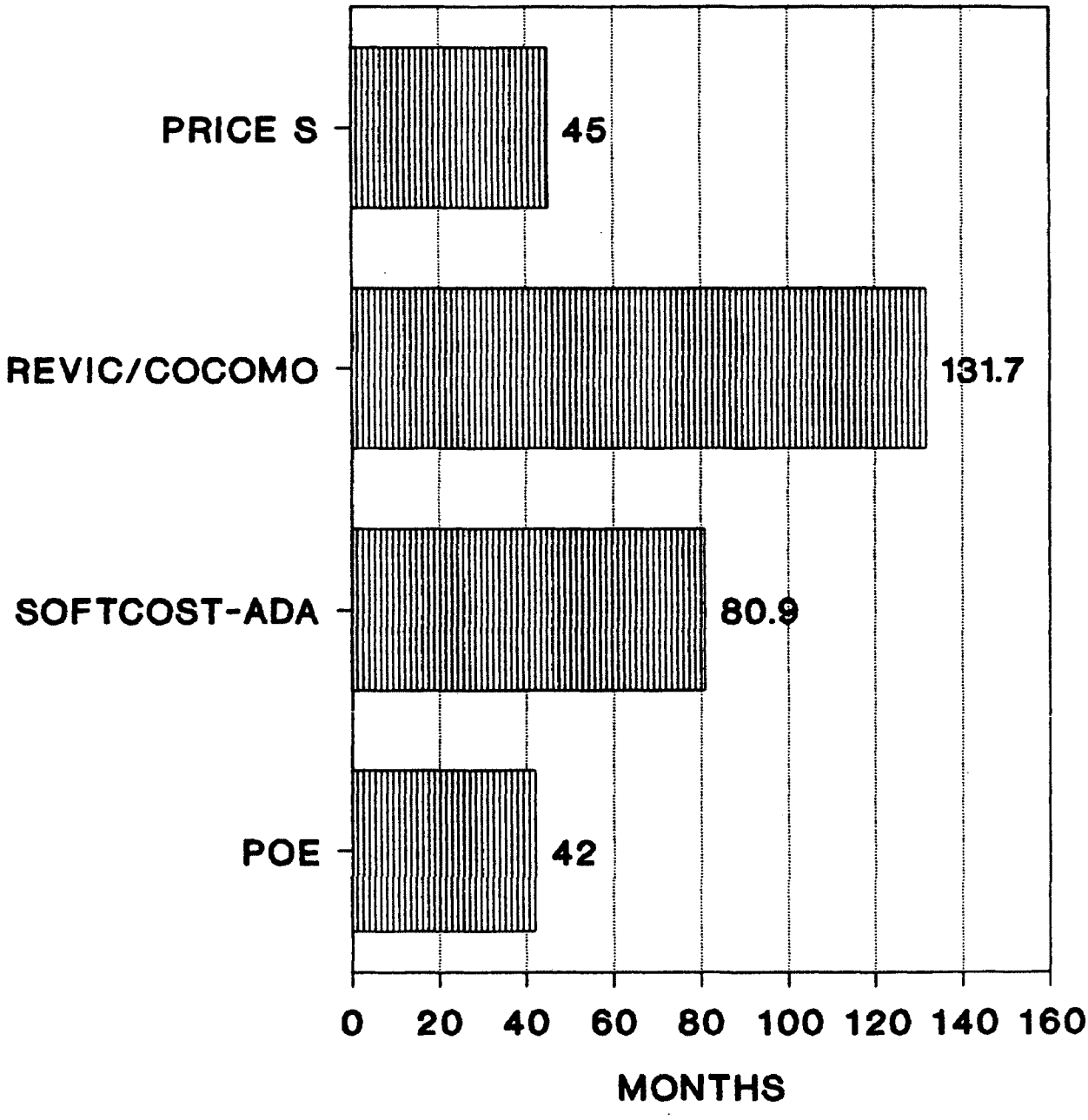
# CALCULATED SCHEDULE
# BEST JUDGMENT




PROGRAM OFFICE ESTIMATE (POE) SHOWN FOR COMPARISON

Fig. 8

# CALCULATED SCHEDULE WORST CASE




PROGRAM OFFICE ESTIMATE (POE) SHOWN FOR COMPARISON

factor is merely on an arbitrary scale. With the other models productivity is calculated in source lines of code produced per month by the average software worker. The REVIC model projected 160.5 lines for optimal circumstances. The worst case achieved only 42.8 lines per month. Softcost-Ada forecasted 106 and 47.6, respectively in the low/high appraisals. A 51.4 percent difference between the two latter models, in the best judgment scenario, changes to an 11 percent difference for the severe conditions. No clear conclusion may be inferred from the productivity data.

## Comparison to the Program Office Estimate

The Program Office estimate (POE) approached the software development project using a completely different method. Accordingly, only the final totals may be compared with the results of the parametric models. The innovative procedure used program management software for a personal computer. Work was entered as tasks, and arranged in the sequence in which they must be performed. The amount of needed hours to accomplish the tasks was also recorded. The initial result of person-hours by fiscal year was exported to an electronic spreadsheet program. Within the spreadsheet computer program, it was possible to add the hours and multiply by a composite wage rate ($71.41). The original software estimate was written in 1987, therefore,

the total was inflated for two years to reflect constant year 1989 dollars.

The adjusted program office estimate, excluding general and administrative expenses and profit, is $24 million. If one uses this estimate as the baseline, the following variations arise as shown in Table 13. All parametric estimates for software development are higher than the program office estimate by at least 22.1 to 77.1 percent.

TABLE 13

DIFFERENCES USING THE 1987 PROGRAM OFFICE SOFTWARE ESTIMATE AS THE BASELINE ($ ARE CONSTANT 1989, IN MILLIONS)

| | Best Judgment | | Worst Case | |
|---|---|---|---|---|
| | Differences in Dollars | Percent Increase | Differences in Dollars | Percent Increase |
| PRICE S | $18.5 | 77.1 | $37.6 | 157.0 |
| REVIC/COCOMO | 5.3 | 22.1 | 85.7 | 357.1 |
| SOFTCOST-ADA | 8.3 | 34.6 | 47.9 | 199.6 |

*The nominal (calculated) schedule results are used for the comparison.

Clearly, the focus of the review of Table 13 should be on the best judgment situation. A government estimate is constructed to determine funding requirements and cannot be based on the worst case assessments.

# CHAPTER VII

## SUMMARY OF FINDINGS AND CONCLUSIONS

The software costs obtained from these models are consistently higher than the Program Office software development cost estimate. Due to the POE methodology of using estimated man-hours, the only comparison possible is with the final financial total. The models produce a best judgment range from 22 to 77 percent greater than the POE using their calculated schedule. This would suggest probable cost growth above the previously anticipated amount.

Projections by REVIC and Softcost-Ada forecast a much longer development schedule than the 42 months desired by the program office. The differences in estimating the time to complete the software project may account for the variation in cost forecasts of the models. With respect to the sensitivity of time, the models also indicate an increase in costs for reductions below the schedule calculated by the model to complete the ATS project. However, a decrease of 5 percent or less in the schedule may not have a significant effect. The PRICE S outcome has suggested the planned times for phases should be considered

forecasts and not precise deadlines. The artificially scheduled times could increase costs. A conclusion may be inferred from the sensitivity analysis, that a review of the schedule requirements should be performed.

Regarding the issue of timing expenditures by software development phase, the parametric models are almost identical. Approximately 50 percent of funding will be spent from the system requirements phase to the detailed design of the project. However, the remaining 50 percent is spread basically among the latter phases of testing, evaluation and integration. The models also differ in approaches to these phases and the allocation of percentages of required funding (refer to Table 12).

No firm conclusion was established in regard to the cost of the Advanced Training System. The range of cost estimates for the best judgement scenario and the calculated schedule was from $29.3 to 42.5 million. Without a reasonably consistent number, the determination of the exact amount above the from the Program Office estimate cannot be made. The software cost models clearly indicate an increase of resources is needed when compared to the Program Office cost estimate.

Automated models can also estimate the approximate magnitude of cost for timely managerial decisions.[33] This

[33]Helton, interviewed by author, 16 January 1989.

paper has concentrated on a best judgement and worst case approach in estimating software development costs. A recommendation for further study should include a sensitivity analysis for the software descriptors of the various models. The purpose would be to isolate the major determinants of cost and any potential inconsistencies. An Air Force Cost Center review is currently underway to determine the accuracy of the parametric models in Ada language projects. The research involves numerous completed software programs. It will also analyze the limitations of models based on personal computers (REVIC and Softcost-Ada) versus the more complex ones hosted on mainframes (PRICE S).[34] Additional studies into ATS should also incorporate these findings.

[34]Helton, interviewed by author, 16 January 1989.

## PRICE S Questions and Answers

1. **Question:** What is the magnitude of the project with respect the lines of software code?

**Answer:** In 1987, the lines of code was estimated at approximately 330,000 (the estimate summary is provided in Table 1 of the paper). The forecast, based on expert opinion, introduces the software by function although the final result may differ to some degree.

2. **Question:** What is the character of the project (Program Application)?

**Answer:** The project will be an advanced computer-based training application. The final result will be highly interactive with the student. Another viewpoint may state the software will be a massive state-of-the-art data base.

3. **Question:** What is the degree of New Design and Code (How much new work is needed?)

**Answer:** No current software architecture or programming code is available to use in the Advanced Training System. Therefore, everything must be built.

4. **Question:** Who will do the work (Productivity)?

**Answer:** As always in a government program, this is the greatest concern. One simply does not know.

5. **Question:** What hardware constraints (Utilization)?

**Answer:** No hardware constraints for ATS. A cheaper alternative to more design and coding is to purchase "off-the-shelf" hardware. In an airplane, one may not be able to use this alternative due to weight or space limitations.

6. **Question:** Are customer specifications and reliability requirements very high?

**Answer:** The military has extensive reviews for software development projects. The issue of reliability would be less of a cost driver.

7. **Question:** What complicating factors exist (Development Environment)?

**Answer:** Due the intense demand for the system the development time is only 42 months. Further, the ATS is also a distributed data base which is a rather new concept.

## MIX, NEW DESIGN, NEW CODE

**System Level Subsystem**

| APPLICATION | MIX | New Development DESIGN | New Development CODE |
|---|---|---|---|
| Data Storage and Retrieval | 30 % | 100 % | 100 % |
| Real Time Command and Control | 10 % | 100 % | 100 % |
| String Manipulation | 30 % | 100 % | 100 % |
| Operating Systems | 30 % | 100 % | 100 % |
| | 100 % | | |

**System Controls Subsystem**

| APPLICATION | MIX | New Development DESIGN | New Development CODE |
|---|---|---|---|
| Data Storage and Retrieval | 40 % | 100 % | 100 % |
| On-line Communications | 10 % | 100 % | 100 % |
| Mathematical Operations | 20 % | 100 % | 100 % |
| String Manipulation | 30 % | 100 % | 100 % |
| | 100 % | | |

**Management Subsystem**

| APPLICATION | MIX | New Development DESIGN | New Development CODE |
|---|---|---|---|
| On-line Communications | 90 % | 100 % | 100 % |
| Operating Systems | 10 % | 100 % | 100 % |
| | 100 % | | |

**Student Management Subsystem**

| APPLICATION | MIX | New Development DESIGN | New Development CODE |
|---|---|---|---|
| Data Storage and Retrieval | 50 % | 100 % | 100 % |
| String Manipulation | 50 % | 100 % | 100 % |
| | 100 % | | |

(Continued)
**MIX, NEW DESIGN, NEW CODE**

**Authoring Subsystem**

| | | New Development | |
|---|---|---|---|
| APPLICATION | MIX | DESIGN | CODE |
| Data Storage and Retrieval | 40 % | 100 % | 100 % |
| Interactive Operations | 20 % | 100 % | 100 % |
| String Manipulation | 40 % | 100 % | 100 % |
| | 100 % | | |

**Delivery Subsystem**

| | | New Development | |
|---|---|---|---|
| APPLICATION | MIX | DESIGN | CODE |
| Data Storage and Retrieval | 50 % | 100 % | 100 % |
| Interactive Operations | 10 % | 100 % | 100 % |
| String Manipulation | 40 % | 100 % | 100 % |
| | 100 % | | |

**Testing Subsystem**

| | | New Development | |
|---|---|---|---|
| APPLICATION | MIX | DESIGN | CODE |
| Data Storage and Retrieval | 50 % | 100 % | 100 % |
| Interactive Operations | 10 % | 100 % | 100 % |
| Mathematical Operations | 10 % | 100 % | 100 % |
| String Manipulation | 30 % | 100 % | 100 % |
| | 100 % | | |

**Evaluation Subsystem**

| | | New Development | |
|---|---|---|---|
| APPLICATION | MIX | DESIGN | CODE |
| Data Storage and Retrieval | 40 % | 100 % | 100 % |
| Mathematical Operations | 20 % | 100 % | 100 % |
| String Manipulation | 40 % | 100 % | 100 % |
| | 100 % | | |

# PHASE AND COST ELEMENT DEFINITIONS*

## PHASE DEFINITIONS

### System Concept

The system concept phase is the initial planning and concept exploration phase for the system. During this initial phase, customer operational requirements are evaluated and refined through the use of feasibility and trade-off studies and analyses. These studies may be conducted on several different solutions, from which one will be chosen. The requirements for the selected solution will be documented in the Preliminary System Segment Specification. The Preliminary System Segment Specification is the prime deliverable product from this phase and will be evaluated by the customer at the System Requirements Review. Successful completion of this review completes the System Concept phase. Although the Preliminary System Segment Specification contains both hardware and software initial requirements, the PRICE S model only estimates the cost associated with the software requirements.

### System Software Requirements Analysis

This phase is a continuation of the previous phase in defining the system software requirements. The Preliminary System Segment Specification is completed, and Computer Software Configuration Items (CSCI) are identified. Trade-off analysis, and architecture and feasibility studies are accomplished to assist in completing the Preliminary Software Requirements Specification and Preliminary Interface Requirements Specifications. The Operational Concept Document is completed, depicting the mission requirements of the system, and how it will be supported within its operational environment. The Software Development Plan, which states how the software effort will be managed and controlled is also completed. The System Design Review is conducted at the end of this phase, with the purpose of reviewing the above documents for consistency, understandability and traceability. The system's Functional Baseline is a product of this phase.

### Software Requirements Analysis

During this phase, all Computer Software Configuration Item (CSCI) requirements stated in the System Segment

Specification are evaluated and defined. Functional, performance and database requirements for each CSCI are identified and defined, along with formal testing requirements. Engineering requirements for each CSCI are documented in the Software Requirements Specification. Internal and external CSCI interface requirements are defined, along with formal testing requirements, and are documented in the Interface Requirements Specification. At the completion of this phase, the Software Specification Review is conducted by the customer. The purpose of this review is to ensure that all requirements are allocated to the appropriate CSCI. This is accomplished by the customer reviewing the Software Requirements Specification and the Interface Requirements Specification for consistency, understandability and traceability of all system requirements defined, and allocated to each CSCI. The System Allocated Baseline results from this phase.

**Preliminary Design**

The Preliminary Design phase consists of producing a top level software design for each Computer Software Configuration Item (CSCI). This CSCI top level design depicts the functions that will be performed along with identifying mathematical models, data and function flows of the software. This is accomplished by allocating the requirements from the Software Requirements Specification and the Interface Requirements Specification to the Top Level Computer Software Components (TLCSC) and Lower Level Computer Software Components (LLCSC) within each CSCI. The Software Top Level Design Document contains all top level design information for each CSCI, including processing and TLCSC interface relationships. This document should demonstrate that all top level requirements for each CSCI have been evaluated and allocated as they relate to the Software Requirements Specification, and Interface Requirements Specification. Effort is also expended during this phase in completing the System Integration and Test Plans, and CSCI Software Test Descriptions. Preliminary versions of the Computer Resources Integrated Support Document, Computer Systems Diagnostic Manual, Computer Systems Operators Manual and The Software Users Manuals are completed and submitted to the customer for review during the Preliminary Design Review. At the completion of this phase, a Preliminary Design Review is accomplished, ensuring that the requirements allocation from each CSCI to TLCSCs and LLCSCs are adequate and meet the system requirements. The Software Test Descriptions will also be reviewed,

ensuring that the means of data recording, reduction, and analysis methods are acceptable.

### Detailed Design

This phase consists of decomposing each CSCI Software Top Level Computer Software Component requirement allocated during the preliminary design phase, down to specific components and units, where each unit performs a single function. Database and internal and external interface designs are completed. This phase should result in a "code to" design document. This process is documented in the Software Detailed Design Document. Unit test requirements, test responsibilities, test cases in terms of expected results and evaluation criteria, and schedules for unit testing of all CSCs will be documented in the software development files. Test cases for each formal CSCI test will also be developed and documented in the Software Test Description. Preliminary versions of the Firmware Support Manual, and the Software Programmers Manual are also completed and submitted to the customer for review at the Critical Design Review. The Critical Design Review is conducted at the end of this phase for each CSCI. The purpose of this review is to ensure that all requirements have been allocated to the respective CSCI units, and to review unit testing and CSCI level testing procedures and documentation.

### Code and Test

The objective of this phase is to produce a deliverable source and object code for each unit that comprises the CSCI. All units are coded in accordance with the code standards described in the Software Development Plan. Unit test is conducted in accordance with procedures developed during the detailed design and the results are documented in the software development files. The purpose of the tests is to ensure that all logic and algorithms employed by each CSC are correct and that the CSC satisfies its specified requirements. Computer Software Component integration and test is accomplished during this phase, prior to the Test Readiness Review. All test results will be reviewed during the Test Readiness Review conducted to ensure that the system is ready for CSCS integration and Functional Configuration Audit.

### CSCI Testing

During this phase, formal tests are conducted for each CSCI to demonstrate to the customer that the CSCI satisfies its allocated requirements. All testing will be conducted by independent personnel and results are documented in the Software Test Report. After the testing is complete, the code is prepared for delivery as specified in the Software Requirements Specification and the Functional Configuration Audit and Physical Configuration Audit are performed. After authenticated completion of these two audits, the Software Specification for the CSCI is entered into the Product Baseline, and CSCI developmental Configuration is complete.

### System Test

Previous to this phase, the software was tested against the CSCI Software Requirements Specification, only. During this phase, hardware/software integration takes place and system test, ensuring that the system meets the requirements stated in the System Segment Specification. All tests of the systems will be conducted by personnel who are independent from the individuals responsible for the development. The objective of this phase is to verify that the system performance of the hardware and software configuration items comply with hardware and software development specification, operational requirements, and interface requirement specifications. At the successful completion of this phase, the development article is provided to the customer.

### Operational Test and Evaluation

During this phase, the system is tested by the operational users or a designated customer test group, with the purpose of testing the operational effectiveness and suitability of the deployed system. Flight tests and other special tests occur during this phase. Deficiencies identified during this phase may result in additional development activity or modifications to the system.

## *COST ELEMENT DEFINITIONS*

Throughout the software development process, six major functions or elements of costs are performed, all of which are essential to the success of the project.

**Design**

The design cost element contains all costs attributed to the software engineering design department. These costs include Engineering supervision, technical and administrative support and vendor liaison required for software development effort.

**Programming**

The programming cost element contains all costs attributed to writing the source code and subsequently testing it.

**Data**

This cost element contains all costs associated with software deliverable documentation. This includes responding to the "Contractor Data Requirements List" (CDRL) which contains requirements for delivery of all requirements, design, maintenance, and user manuals, i.e., Systems Segment Specification, Top Level Design and Detailed Design Specifications, Programmer and user manuals, etc.

**System Engineering/Project Management**

This element includes the System Engineering to define the software, and the Project Management effort required to manage the software development project. The system engineering activity encompasses the effort to define the system requirements, and the integrated planning and control of the technical program efforts of the design engineering, specialty engineering development of test procedures, and system oriented testing and problem resolution. Project Management efforts include managing the software development program in accordance with all procedures identified in the Software Development Plan, design review activities, and administrative duties.

**Quality Assurance**

This cost element includes the effort required to conduct internal reviews and walkthroughs to evaluate the quality of the software and associated documentation. Activities included in this element are evaluation of the Software Development Plan, software development library, and the Software Configuration Management Plan.

**Configuration Control or Configuration Management**

This activity involves the determination, at all times, of precisely what is and is not an approved part of the system. To accomplish this, it is necessary to perform three tasks. The first involves incorporating requirements and specifications into the Functional and Allocated Baselines. Once a document has been incorporated into the baseline, changes may only be made through the configuration control task. This task involves the evaluation of changes, and corrections to the baseline. Finally, it is necessary to provide for the dissemination and control of approved baseline material. Configuration Control also reviews the

test procedures and ensures compliance with test plans and specification.

*Extracted from the PRICE S Reference Manual 1987, pp A-1 to A-5.

# SURVEY RESULTS

## COCOMO/REVIC PROJECT FACTORS

### ANALYSTS CAPABILITY - ACAP

ACAP measures the system engineers capability as a team average. The analysts define the software architecture and produce the preliminary design specifications. This includes requirements identification and decomposition as well as preliminary design of the Computer System Configuration Item (CSCI) and its component Computer Subsystem Components.

Enter the two letter code from the table below which best describes the capabilities of the entire team. Note that the rating does not necessarily correspond to years of experience, but rather attempts to quantify skills.

| RATING | EXAMPLES | |
|---|---|---|
| _____ | VL | New personnel with no experience (15%) |
| Worst | LO | Functional team with low effectivity (35%) |
| Best | NM | Average team with nominal effectivity (55%) |
| _____ | HI | Strong team with good effectivity (75%) |
| _____ | VH | Strong team with many top people (90%) |

### PROGRAMMING TEAM CAPABILITY - PCAP

PCAP measures the capability of the programmers who will be the ones that actually perform the detailed CSCI/CSC design during the critical design phase of the contract and write/test the physical code during the coding and integration testing phases. Enter the two letter code that best describes the capabilities of the programming team.

| RATING | EXAMPLES | |
|---|---|---|
| _____ | VL | 15TH PERCENTILE TEAM |
| Worst | LO | 35TH PERCENTILE TEAM |
| Best | NM | 55TH PERCENTILE TEAM |
| _____ | HI | 75th PERCENTILE TEAM |
| _____ | VH | 90TH PERCENTILE TEAM |

PROJECT APPLICATION EXPERIENCE - AEXP

AEXP attempts to measure the familiarity of the design and development team with this specific program. Enter the two letter code which best describes the experience of the overall team with projects of this type.

| RATINGS | EXAMPLES | |
|---|---|---|
| _____ | VL | NO EXPERIENCE (less than 4 months) |
| _____ | LO | LIMITED EXPERIENCE (1 year) |
| Both | NM | NOMINAL EXPERIENCE (3 years) |
| _____ | HI | BETTER THAN AVERAGE (6 years) |
| _____ | VH | EXPERTS (more than 12 years) |

LANGUAGE EXPERIENCE - LEXP

LEXP measures the design and programming team's experience with the language that will be used to implement the design in the software. Enter the two letter code which best describes the team's experience.

| RATINGS | EXAMPLES | |
|---|---|---|
| _____ | VL | NEVER USED BEFORE |
| _____ | LO | LESS THAN 1 YEAR EXPERIENCE |
| Both | NM | AT LEAST 1 YEAR EXPERIENCE |
| _____ | HI | 2 YEARS EXPERIENCE |
| _____ | VH | MORE THAN 2 YEARS EXPERIENCE |

EXECUTION TIME CONSTRAINTS - TIME

TIME measures the approximate percentage of available CPU execution time that will be used by the software. Enter the two letter code that best describes the approximate amount of utilization.

| RATINGS | EXAMPLES | |
|---|---|---|
| Both | VL | NO CONSTRAINTS ON EXECUTION TIME |
| _____ | LO | NO CONSTRAINTS ON EXECUTION TIME |
| _____ | NM | FOR 60% UTILIZATION |
| _____ | HI | FOR 70% UTILIZATION |
| _____ | VH | FOR 85% UTILIZATION |
| _____ | XH | for 95% or more |

MAIN STORAGE CONSTRAINTS - STOR

STOR measures the amount of constraint imposed on the software due to main memory limitations in the target computer. If memory is a problem, more time must be spent on design and coding. Enter the two letter code that best describes this system's main storage constraints.

| RATINGS | EXAMPLES | |
|---|---|---|
| Both | NM | NO MEMORY CONSTRAINTS |
| | HI | FOR 70% UTILIZATION |
| | VH | FOR 85% UTILIZATION |
| | XH | FOR 95% OR HIGHER UTILIZATION |

VIRTUAL MACHINE VOLATILITY - VIRT

VIRT measures the amount of changes the host and target computers are experiencing during the design and development phases. The more the system changes during these phases, the more work is required to keep the software design and code compatible with the system's hardware and software. Enter the two letter code which best describes the frequency of changes to both the development and target computer's hardware and software.

| RATINGS | EXAMPLES | |
|---|---|---|
| Both | VL | NO CHANGES EXPECTED |
| | LO | ONE CHANGE EVERY 6 MONTHS |
| | NM | ONE CHANGE EVERY 3 MONTHS |
| | HI | ONE CHANGE EVERY MONTH |
| | VH | SEVERAL CHANGES EVERY MONTH |
| | XH | CONCURRENT DEVELOPMENT - CONSTANTLY CHANGING |

COMPUTER TURNAROUND TIME - TURN

TURN measures the time spent waiting for the host, or developmental computers, to complete an action such as a compile or printing a listing. Enter the two letter code which best describes the development environment.

| RATINGS | EXAMPLES | |
|---|---|---|
| _____ | VL | LESS THAN 6 MINUTES |
| Best | LO | LESS THAN 30 MINUTES |
| Worst | NM | LESS THAN 4 HOURS |
| _____ | HI | MORE THAN 4 HOURS |
| _____ | VH | MORE THAN 12 HOURS |

REQUIREMENTS VOLATILITY - RVOL

RVOL measures the amount of project design and development rework that is the result of changes in customer specified requirements. This factor can have a very large effect on the total development time and effort, but should be used very carefully. Most projects will be put on contract after negotiations based on the known requirements. The expectation is that any changes in requirements will result in an Engineering Change Proposal (ECP) which will adjust the contract price accordingly. This factor compensates for the extra system engineering and management effort to evaluate the changes in requirements, estimate the design impacts, prepare the ECPs, and change the software. This factor should not be used to arbitrarily built in a management reserve, instead use the Mgmt Reserve for Risk factor.

| RATINGS | EXAMPLES | |
|---|---|---|
| _____ | LO | ESSENTIALLY NONE |
| _____ | NM | SMALL, NONCRITICAL REDIRECTIONS |
| Best | HI | OCCASIONAL MODERATE REDIRECTIONS |
| Worst | VH | FREQUENT MODERATE OR OCCASIONAL MAJOR REDIRECTIONS |
| _____ | XH | FREQUENT MAJOR REDIRECTIONS |

REQUIRED SOFTWARE RELIABILITY - RELY

RELY quantifies the required reliability of the finished software. As the required reliability increases, more time must be spent in the critical design and testing phases. Enter the two letter code which best describes the required reliability of the finished system in terms of what effects the software failure would have on the user.

| RATINGS | EXAMPLES | |
|---|---|---|
| _____ | VL | SLIGHT INCONVENIENCE |
| _____ | LO | EASILY RECOVERABLE LOSS |
| Both | NM | MODERATE RECOVERABLE LOSS |
| _____ | HI | FOR MIL-STD OR HIGH FINANCIAL LOSS |
| _____ | VH | FOR POSSIBLE LOSS OF LIFE |

DATA BASE SIZE - DATA

DATA attempts to determine the effects on the software development due to the size of the data base which must be maintained and manipulated. Enter the two letter code which best describes the size and complexity of the program's data base effort.

| RATINGS | EXAMPLES | |
|---|---|---|
| _____ | LO | VERY SMALL EFFORT (DB BYTES/PROGRAM DSI <10) |
| _____ | NM | NOMINAL SIZE EFFORT (10<= D/P <100) |
| _____ | HI | LARGE AND COMPLEX EFFORT (100<= D/P <1000) |
| Both | VH | THE LARGEST (D/P =>1000) |

SOFTWARE PRODUCT COMPLEXITY - CPLX

CPLX attempts to quantify the complexity of the software product to be developed. Use the examples below to select the two letter code that best describes the required software's complexity.

| RATINGS | EXAMPLES | |
|---|---|---|
| _____ | VL | OFFLINE SIMPLE PRINT ROUTINES |
| _____ | LO | OFFLINE DATA PROCESSING |
| _____ | NM | DATA PROCESSING AND MATH ROUTINES |
| Both | HI | SOME H/W I/O AND ADVANCED DATA STRUCTURES |
| _____ | VH | REAL TIME APPLICATIONS AND ADVANCED MATH |
| _____ | XH | EXTREMELY COMPLEX SCIENTIFIC PROCESSING SUCH AS SIGNAL PROCESSING AND EPHEMERIC CALCULATIONS |

REQUIRED REUSABILITY - RUSE

RUSE measures the extra effort needed to generalize software modules when it must be developed specifically for reuse in other software packages. Enter the two letter code that best describes the required reusability of this software.

| RATINGS | EXAMPLES | |
|---|---|---|
| _____ | NM | NOT FOR REUSE ELSEWHERE |
| Best | HI | REUSE WITHIN SINGLE-MISSION PRODUCTS |
| Worst | VH | REUSE ACROSS SINGLE PRODUCT LINE |
| _____ | HX | REUSE IN ANY APPLICATION |

MODERN PROGRAMMING PRACTICES - MODP

MODP quantifies the use of modern programming practices such as structured design, data flow diagrams, data dictionaries, etc. Enter the two letter code which best describes this program's use of such practices.

| RATINGS | EXAMPLES | |
|---|---|---|
| _____ | VL | NO USE OF MPPs |
| _____ | LO | BEGINNING USE OF MPPs |
| _____ | NM | SOME USE OF MPPs BY MORE EXPERIENCED TEAM MEMBERS |
| Worst | HI | GENERAL USE OF MPPs BY ALL TEAM MEMBERS |
| Best | VH | ROUTINE USE OF MPPs WITH STRONG COMPANY TRAINING |

USE OF SOFTWARE TOOLS - TOOL

TOOL measures the use of automated software tools such as CASE (computer aided system engineering) tools, Ada Programming Support Environments, etc. Enter the two letter code which best describes the use of tools in this program.

| RATINGS | EXAMPLES | |
|---|---|---|
| _____ | VL | VERY FEW, PRIMITIVE TOOLS |
| _____ | LO | BASIC MICRO TOOLS |
| Both | NM | BASIC MINI TOOLS |
| _____ | HI | BASIC MAXI TOOLS |
| _____ | VH | EXTENSIVE TOOLS, LITTLE INTEGRATION |
| _____ | XH | MODERATELY INTEGRATED ENVIRONMENT (UNIX OR MAPSE) |
| _____ | XX | FULLY INTEGRATED ENVIRONMENT (APSE) |

CLASSIFIED SECURITY APPLICATION - SECU

SECU measures the extra work required to develop software either in a classified security area, or for a classified security application.
Note that this factor does not relate to the certification of security processing levels, as specified by the NSA. Use the complexity factor to account for that type of application. Enter the two letter code which describes this program's environment.

| RATINGS | EXAMPLES | |
|---|---|---|
| _____ | VL | |
| _____ | LO | |
| Both | NM | UNCLASSIFIED |
| _____ | HI | CLASSIFIED (SECRET, TOP SECRET) |
| _____ | VH | |
| _____ | XH | |

MANAGEMENT RESERVE FOR RISK - RISK

RISK allows you to add in a percentage factor to account for varying levels of program risk. This factor should only be used very carefully to assess the upper limits of program costs. Enter the two letter code as desired. Normally, this value should be left at VL.

| RATINGS | EXAMPLES | |
|---|---|---|
| Both | VL | VERY LOW PROGRAM RISK (GROUND SYSTEMS) |
| | LO | LOW PROGRAM RISK (MILITARY SPEC GROUND SYSTEMS) |
| | NM | MEDIUM PROGRAM RISK (UNMANNED AIRBORNE SYSTEMS) |
| | HI | HIGH PROGRAM RISK (MANNED AIRBORNE SYSTEMS) |
| | VH | VERY HIGH PROGRAM RISK (UNMANNED SPACE APPLICATIONS) |
| | XH | EXTRA HIGH PROGRAM RISK (MANNED SPACE APPLICATIONS) |

REQUIRED DEVELOPMENT SCHEDULE - SCED

SCED measures the effects that schedule compression or stretchout have on the total effort. Forcing the schedule below the nominal schedule predicted by the model will always increase the total effort. The model will not let you force the schedule below approximately 75% of the nominal schedule it calculates. You cannot change this factor in this screen. It is automatically calculated as a percentage of the nominal schedule whenever you use any of the options from the constraint menu, such as forcing the total schedule or specifying the staffing/schedule per development phase.

Note: The Best Judgment and the Worst Case used the calculated schedule. The Program Office schedule is 42 months.

SURVEY RESULTS

SOFTCOST - ADA PROJECT FACTORS

TYPE OF SOFTWARE SYSTEM

| | | | |
|---|---|---|---|
| ____1) | Automation System | ____6) | Scientific System |
| ____2) | Avionics System | ____7) | Simulation System |
| ____3) | Command & Control System | ____8) | Telecommunica-tions System |
| ____4) | Data Processing System | | |
| ____5) | Environment/Tool System | ____9) | Test System |
| | | Both 10) | Other |

Note: Reifer Consultants suggested "Other" for the ATS program.

SYSTEM ARCHITECTURE

| | |
|---|---|
| _____1) | Centralized (single processor) |
| _____2) | Tightly-coupled (multiple processor) |
| _____3) | Loosely-coupled (multiple processor) |
| _____4) | Federated (functional processors communicating via a bus) |
| _____5) | Distributed (centralized database) |
| Both 6) | Distributed (distributed database) |
| _____7) | Other |

NUMBER OF SOFTWARE ORGANIZATIONS

Enter the number of organizations that will be involved in the software effort (e.g., development, test, Q.A., etc).

Best Judgment: 3 organizations
Worst Case : 4 "

ORGANIZATIONAL INTERFACE COMPLEXITY

| RATING | | EXAMPLES |
|---|---|---|
| _____ | VL | Single interface with collocated customer |
| _____ | LO | Single interface with single customer |
| _____ | NM | Multiple internal and single external interfaces |
| Best | HI | Multiple internal and external interfaces |
| Worst | VH | Multiple geographically distributed internal and external interfaces |

REQUIRED DEVELOPMENT SCHEDULE

| RATING | | EXAMPLES |
|---|---|---|
| _____ | VL | 75% of nominal schedule |
| _____ | LO | 85% of nominal schedule |
| Both | NM | Nominal schedule |
| _____ | HI | 120% of nominal schedule |
| _____ | VH | 130% of nominal schedule |

Note: The base case used the nominal schedule.

RESOURCE AVAILABILITY

| RATING | | EXAMPLES |
|---|---|---|
| _____ | VL | Extreme equipment, facility and staff limitations |
| _____ | LO | Computer shared and remotely accessible. Less than 30% of staff available when needed. |
| _____ | NM | Interactive access to dedicated computer resources. Less than 50% of staff available when needed. |
| _____ | HI | Dedicated facilities with multiple LAN-servers/worker. Less than 70% of staff available when needed. |
| Both | VH | Software Factory with multiple LAN-servers/worker. Skilled staff available when needed. |

SECURITY REQUIREMENTS

| RATING | | EXAMPLES |
|---|---|---|
| _____ | LO | None |
| Both | NM | Database integrity |
| _____ | HI | Physical security |
| _____ | VH | Demonstrably correct trusted system Physical security |
| _____ | EH | Verifiably correct trusted system Physical security |

DEGREE OF STANDARDIZATION

| RATING | | EXAMPLES |
|---|---|---|
| _____ | VL | None |
| _____ | LO | Use Ada programming standards |
| _____ | NM | Use commercial life cycle standards |
| Both | HI | Use tailored military standards |
| _____ | VH | Use untailored military standards |

SCOPE OF SUPPORT

| RATING | EXAMPLES | |
|---|---|---|
| _____ | LO | No support to nonsoftware organizations |
| _____ | NM | Liaison support to nonsoftware organizations |
| _____ | HI | Extensive support to system test organizations |
| Both | VH | Extensive support to system engineering and test organizations. CSSR/CSCSC reporting requirements. |

USE OF MODERN SOFTWARE METHODS

| RATING | EXAMPLES | |
|---|---|---|
| _____ | VL | No use |
| _____ | LO | Structured programming |
| _____ | NM | Object-oriented design plus structured programming |
| Best | HI | Ada packaging methods used as part of an overall structured methodology |
| Worst | VH | Integrated life cycle methodology which exploits Ada reusability concepts |

USE OF PEER REVIEWS

| RATING | EXAMPLES | |
|---|---|---|
| _____ | VL | No use |
| _____ | LO | Quality inspections/audits |
| _____ | NM | Design and code walkthroughs |
| _____ | HI | Design and code inspections |
| Both | VH | Peer management reviews, design and code inspections |

USE OF SOFTWARE TOOLS/ENVIRONMENTS

| RATING | EXAMPLES | |
|---|---|---|
| _____ | VL | Basic Ada language tools |
| _____ | LO | MAPSE, plus access to host tools |
| Both | NM | MAPSE, plus access to host/target tools |
| _____ | HI | APSE as part of the environment |
| _____ | VH | APSE as the development environment |
| _____ | EH | Full, integrated, life cycle APSE |

SOFTWARE TOOL/ENVIRONMENT STABILITY

| RATING | EXAMPLES | |
|---|---|---|
| _____ | VL | Buggy compiler. APSE change every 2 weeks |
| _____ | LO | Stable but incapable compiler. APSE change every month. New tool rate 1 per week. |
| _____ | NM | Stable compiler. APSE change every 3 months. New tool rate 1 per month. |
| Worst | HI | Stable compiler. APSE change every 4 months. New tool rate 1 per quarter. |
| Best | VH | Stable compiler capable of tasking. APSE change every 6 months. New tool rate 1 per quarter. |
| _____ | VH | Stable and fully capable compiler. APSE change every 6 months. New tool rate 1 per 6 months. |

ADA USAGE FACTOR

| RATING | EXAMPLES | |
|---|---|---|
| _____ | VL | Less than 50% of the software written in Ada |
| _____ | LO | Less than 75% of the software written in Ada |
| _____ | NM | Less than 85% of the software written in Ada |
| _____ | HI | Less than 95% of the software written in Ada |
| Both | VH | 100% of the software written in Ada |

PRODUCT COMPLEXITY

| RATING | EXAMPLES | |
|---|---|---|
| _____ | VL | Straight line code. Standard types. General Structures. Simple math. No tasking. |
| _____ | LO | Simple Operators. Standard types. General Structures. Simple math. Simple data manipulation. No tasking |
| _____ | NM | Straight forward logic. Generics and standard structures. Standard I/O. Simple Tasking. |
| _____ | HI | Highly nested logic. Numeric Types. Libraries of packages and generics. Complicated I/O. Concurrent tasking. |
| _____ | VH | Stochastic logic. Unique types. Libraries of packages, tasks and generics. Sophisticated math and I/O. Rendezvous. |
| Both | EH | Dynamic resource allocation. Unique types. Special libraries. Time dependent task scheduling. Multiple exception handlers. Optimization and efficiency concerns. |

REQUIREMENTS VOLATILITY

| RATING | | EXAMPLES |
|---|---|---|
| _____ | LO | Essentially no changes |
| _____ | NM | Over 60% of requirements are well established |
| Both | HI | Over 50% of requirements are well established |
| _____ | VH | Over 30% of requirements are well established |
| _____ | VH | Less than 30% of requirements are well established |

DEGREE OF OPTIMIZATION

| RATING | | EXAMPLES |
|---|---|---|
| _____ | LO | Uses maximum of 50% of available processor resources |
| Both | NM | Uses maximum of 75% of available processor resources |
| _____ | HI | Uses maximum of 85% of available processor resources |
| _____ | VH | Uses maximum of 95% of available processor resources |
| _____ | EH | Uses more than 100% of available processor resources |

DEGREE OF REAL-TIME

| RATING | | EXAMPLES |
|---|---|---|
| _____ | LO | Essentially batch response |
| Both | NM | Interactive with limited Ada tasking |
| _____ | HI | Interrupt driven with tasking in milliseconds |
| _____ | VH | Concurrent tasking with rendezvous in milliseconds |
| _____ | EH | Concurrent tasking with rendezvous in nanoseconds |

DEGREE OF REUSE

| RATING | | EXAMPLES |
|---|---|---|
| _____ | LO | Essentially no packaging for reuse. |
| _____ | NM | Less than 10% of software packaged for reuse. |
| _____ | HI | Less than 20% of software packaged for reuse. |
| _____ | VH | Very High - Less than 30% of software packaged for reuse. Effective use of package, generic and task library units. |
| Both | EH | Over 30% of software packaged for reuse. Extensive use of package, generic and other library units. |

DATA BASE SIZE

| RATING | | EXAMPLES |
|---|---|---|
| _____ | LO | Database (bytes) as a % of total program size < 19 |
| _____ | NM | Database (bytes) as a % of total program size < 100 |
| _____ | HI | Database (bytes) as a % of total program size < 1000 |
| Both | VH | Database (bytes) as a % of total program size > 1000 |

ADA EXPERIENCE PROFILE

Enter the average number of Ada projects completed by the team that will be assigned to this project. A project is defined as a delivery of a product packaged and prepared using Ada concepts (Ada PDL, a software build, etc.).

Average Number: 3

ANALYST CAPABILITY

| RATING | | EXAMPLES | |
|---|---|---|---|
| Worst | VL | Bottom | (15th percentile) |
| _____ | LO | Below average | (35th percentile) |
| Best | NM | Average | (60th percentile) |
| _____ | HI | Top performer | (80th percentile) |
| _____ | VH | Superstar | (90th percentile) |

APPLICATIONS EXPERIENCE

| RATING | | EXAMPLES |
|---|---|---|
| _____ | VL | Less than 4 months experience |
| Worst | LO | Less than 1 year of experience |
| Best | NM | Between 1 - 3 years experience |
| _____ | HI | Between 3 - 6 years experience |
| _____ | VH | Over 6 years experience |

ADA ENVIRONMENT EXPERIENCE

| RATING | | EXAMPLES |
|---|---|---|
| _____ | VL | Less than 3 months experience |
| Worst | LO | Between 3 - 6 months experience |
| Best | NM | Between 6 - 12 months experience |
| _____ | HI | Over 1 year of experience |

ADA LANGUAGE EXPERIENCE

| RATING | EXAMPLES | |
|---|---|---|
| _____ | VL | Less than 3 months experience |
| Worst | LO | Between 3 - 6 months experience |
| Best | NM | Between 6 - 12 months experience |
| _____ | HI | Between 1 -2 years experience |
| _____ | VH | Over 2 years experience |

ADA METHODOLOGY EXPERIENCE

| RATING | EXAMPLES | |
|---|---|---|
| _____ | VL | Less than 3 months experience |
| _____ | LO | Between 3 - 6 months experience |
| Both | NM | Between 6 - 12 months experience |
| _____ | HI | Between 1 - 2 years experience |
| _____ | VH | Very High - Over 2 years experience |

TEAM CAPABILITY

| RATING | EXAMPLES | |
|---|---|---|
| _____ | VL | Not used |
| _____ | LO | Design teams |
| Both | NM | Programming teams |
| _____ | HI | Participatory teams |
| _____ | VH | Interdisciplinary teams |

## Sources Consulted


Bailey, Elizabeth K., Thomas P. Frazier, and John W. Bailey. A Descriptive Evaluation of Automated Software Cost-Estimation Models IDA Paper P-1979. Washington, DC: Institute for Defense Analyses, October 1986.

Conners, Dan. Memorandum for the Record. Presented to Human Systems Division/YAT, 13 April 1987. Brooks Air Force Base, Texas.

Conte, S. D., H. E. Dunsmore, and V. Y. Shen. Software Engineering Metrics and Models. Reading, Massachusetts: The Benjamin/Cummings Publishing Company, Inc., 1986.

DeMarco, Tom. Controlling Software Projects: Management, Measurement & Estimation. Foreword by Barry W. Boehm. New York: Yourdon, Inc., 1982.

Department of the Air Force, Headquarters Air Training Command. Operational Concept Document for the Advanced Training System. [San Antonio, TX]: U.S. Department of the Air Force, Headquarters Air Training Command, 26 October 1987.

Helton, Mike, Software Cost Analyst, Air Force Cost Center, interviewed by author, 16 January 1989, Washington, D. C., telephone conversation, Air Force Cost Center, Washington, D. C.

Kane, Pat. Software Engineer, Reifer Consultants, Inc., interviewed by author, 7 March 1989, Torrance, California, telephone conversation, Torrance, California.

King, Earl, Senior Analyst, Operations Staff, PRICE Systems. Interview by author, 16 May 1988, Moorestown, New Jersey. Telephone conversation. PRICE Systems, Moorestown, New Jersey.

Otte, Jim. Cost Analyst, PRICE Systems, interviewed by author, 24 February 1989, Dayton, Ohio, telephone conversation, Dayton, Ohio.

PRICE Parametric Cost Models: An Executive Guide. Cherry Hill, New Jersey: PRICE Systems [undated].

PRICE S Reference Manual. Moorestown, New Jersey: RCA Corporation, 1987.

Reifer, Donald J. "Ada's Impact: A Quantitative Assessment" Torrance, California: Reifer Consultants, Inc. 10 September 1987.

_____ "Softcost-Ada: User Experiences and Lessons Learned at the Age of One" Torrance, California: Reifer Consultants, Inc., 15 May 1988.

Shooman, Martin L. Software Engineering: Design, Reliability, and Management. New York: McGraw-Hill, Inc., 1983.

Vick, C. R., and C. V. Ramamoorthy., ed. Handbook of Software Engineering. New York: Van Nostrand Reinhold Company Inc., 1984.

Wagner, Joseph T. Headquarters Air Force Systems Command (AFSC) /Cost Analysis (ACC) Letter to all AFSC Product Divisions, 19 December 1988.

Willens, Douglas. Marketing Director, Reifer Consultants, Inc., Personal letter to author, 25 January 1989.

Wilton, Claude. Senior Analyst, West Coast Operations Staff, PRICE Systems, interview by author, 27 February 1989, Los Angeles, California, telephone conversation, PRICE Systems, Los Angeles, California.