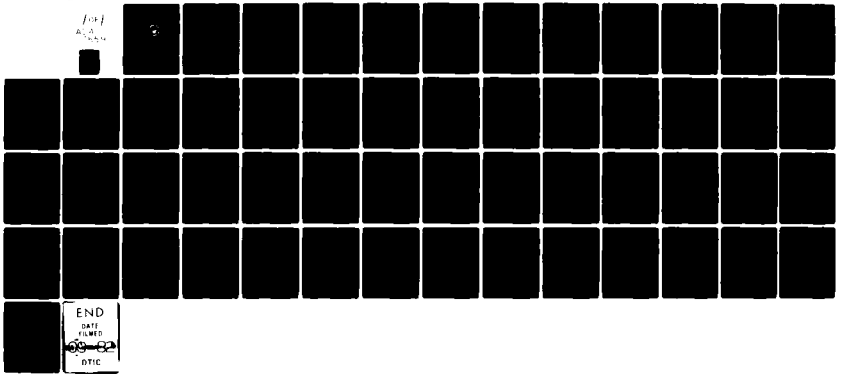
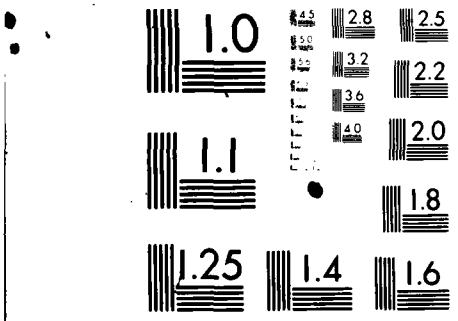
AD-A117 859 NAVAL POSTGRADUATE SCHOOL MONTEREY CA F/G 9/2
APPLICATION OF SELECTED SOFTWARE COST ESTIMATING MODELS TO A TA--ETC(U)
MAR 82 W B COLLINS

UNCLASSIFIED NL



END
DATE FILMED
DTIC



(2)

NAVAL POSTGRADUATE SCHOOL
Monterey, California

AD A117559

THESIS

APPLICATION OF SELECTED SOFTWARE COST ESTIMATING MODELS TO A TACTICAL COMMUNICATIONS SWITCHING SYSTEM: TENTATIVE ANALYSIS OF MODEL APPLICABILITY TO AN ONGOING DEVELOPMENT PROGRAM

by

William B. Collins

March 1982

Thesis Advisor: D. C. Boger

Approved for public release; distribution unlimited

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A117559 / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Application of Selected Software Cost Estimating Models to a Tactical Communications Switching System: Tentative Analysis of Model Applicability to an Ongoing Development Program	5. TYPE OF REPORT & PERIOD COVERED Master's Thesis; March 1982 / 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) William B. Collins	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940	12. REPORT DATE March 1982 / 13. NUMBER OF PAGES 54
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report) / 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Software Cost Models	Software Cost Analysis
Software Cost Estimation	TRI-TAC
Software Cost Factors	AN/TTC-42
Software Costing Techniques	Unit Level Circuit Switch

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This study analyzes the applicability of software cost estimating models to a tactical information processing system. In particular, the Boehm and Putnam models are used to obtain predictions which are compared to contractor estimates for the TRI-TAC AN/TTC-42 Unit Level Circuit Switch. Factors affecting model performance are also examined. Conclusions address the requirement for more accurate estimation of

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

software project scope and clarification of model input parameter definitions.

A

Approved for public release; distribution unlimited

Application of Selected Software Cost Estimating Models to a Tactical Communications Switching System: Tentative Analysis of Model Applicability to an Ongoing Development Program

by

William B. Collins
Major, United States Marine Corps
B.S., United States Naval Academy, 1971

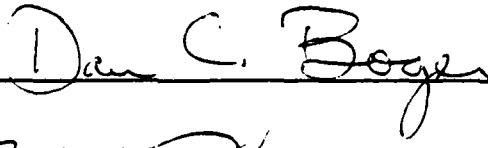
Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN TELECOMMUNICATIONS SYSTEMS MANAGEMENT

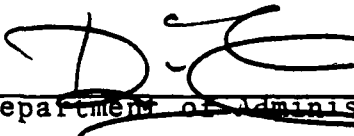
from the

NAVAL POSTGRADUATE SCHOOL
March 1982

Author: 

Approved by: 
Thesis Advisor


Second Reader


Chairman, Department of Administrative Sciences


Dean of Information and Policy Sciences

ABSTRACT

This study analyzes the applicability of software cost estimating models to a tactical information processing system. In particular, the Boehm and Putnam models are used to obtain predictions which are compared to contractor estimates for the TRI-TAC AN/TTC-42 Unit Level Circuit Switch. Factors affecting model performance are also examined. Conclusions address the requirement for more accurate estimation of software project scope and clarification of model input parameter definitions.

TABLE OF CONTENTS

I. INTRODUCTION

Since the introduction of the stored program computer, a reversal in the proportion of development cost attributed to hardware and to software has occurred (Boehm: pp. 17 & 18, 1981). Unfortunately this increase in the contribution of software to overall development cost has not been accompanied by a similar increase in the accuracy of software cost and schedule estimating procedures (Putnam: pp. 13 & 14, 1980; Thibodeau: pp. 5-1 & 5-2, 1981; ITT: Sec 400.22, 1981). Two of the models which have been proposed to remedy this situation will be the subject of this study. The Constructive Cost Model (COCOMO) (Boehm, 1981) and the Putnam Model (Putnam, 1980) will be applied to an ongoing automated system development program in an effort to assess their applicability in the study environment.

The vehicle for the study will be the AN/TTC-42 tactical telephone switching central, which together with a smaller switchboard, the SB-3865, comprises the Unit Level Circuit Switch (ULCS) family. These two equipments are designed to provide "reliable, ... and highly mobile telephone switching equipments for 'unit level' organizations such as division and brigade..." (Huebner-Wright, 1980). In many ways they resemble the private automatic branch exchanges (PABX) that have become common in fixed plant telephone installations

during the last decade, but they possess additional capabilities needed on the modern battlefield, for example, high mobility and cryptographic protection. Both switchboards, or circuit switches as they are officially known, are being developed by the U.S. Marine Corps as part of a Department of Defense mandated move away from the analog, non-secure, and often incompatible telephone systems currently in use by U.S. military forces to a digital cryptographically secure and interoperable environment. If this change occurred too rapidly a large portion of the existing tactical telephone plant would have to be discarded well before the end of its useful service life. To avoid this a multi-service effort, the Joint Tactical Communications (TRI-TAC) Program, is developing an integrated system of hybrid analog and digital tactical circuit switches, including the AN/TTC-42, which will be able to operate in today's analog environment, in the mixed analog and digital transition period and finally in the all digital scenario.

Placing the burden of compatibility on the TRI-TAC systems may appear to be a simple solution to the problem of integrating a wide variety of equipment into a common system, but implementing that simple concept requires complicated solutions. When this requirement is tied in with others such as faster and more reliable service, and cryptographic protection, a purely hardware-oriented solution is infeasible.

In the AN/TTC-42, the hardware is complex, but the complexity has been held to tenable levels by transferring much of it to software. Two Intel 8080A microprocessors, one each for the switching (SCPU) and monitor (MCPU) control fucntions, are used to operate the switch as a real-time, event-driven, table-controlled system. The programs required to accomplish this are quite complex, and would be much larger than they are but for the use of even more complex control tables. When an event occurs that requires processing by one of the microprocessors, an interrupt is generated to the unit concerned. The unit identifies the element of its environment that initiated the interrupt and the point in the unit's program where processing is required to begin by means of the control tables. This technique saves processing time and memory space, but does so at the cost of utilizing some very sophisticated and difficult-to-develop software.

The Naval Electronic Systems Command (NAVELEX) acts as the ULCS program manager for the Marine Corps. NAVELEX in turn is aided in the performance of its ULCS responsibilities by a support contractor, the Federal Systems Group of Calculon Corporation. Actual development of the circuit switches is being conducted by the International Telephone and Telegraph Corporation's Defense Communications Division (ITT) under a contract awarded in August 1977. The contract price was $19.5M, and ITT agreed to contribute another $5.4M in cost

sharing. This cost sharing offer made ITT's excellent technical proposal even more appealing, and was reputedly made by the company in a bid to expand its technological base. Completion was scheduled for thirty-six months from the contract award date.

ITT prepared a highly sophisticated environment for the development of the ULCS software. ITT software design philosophy (ITT, 1981) places heavy emphasis on the complete identification of software performance requirements before the software design phase commences. Top-down design is stressed and modular integrity is maintained to the maximum degree permitted by performance requirements. Structured programming is used to implement the top-down design and program design language (PDL) is employed to enhance modularity and to ease the transition from the program design to code. The code itself is written by chief programmer teams in XAS-8 assembly language, a Pascal derivative that lends itself to structured programming techniques.

Extensive software tools were developed and employed to implement this design and programming philosophy. A UNIX interactive timesharing system is used to edit code, maintain program files and to generate program documentation. The XAS-8 source code is translated to Intel 8080A object code by a macro assembler, and a link editor is employed to assemble object routines for testing. The SCPU, MCPU, their

interfaces and hardware environments are emulated on a DEC PDP 11/70 computer. This has permitted extensive testing of the software before hardware completion. In addition, an environmental simulator is used to construct precisely controlled and completely reproducible test scenarios. Configuration control and resource (memory and processor) utilization are continually scrutinized, and an independent quality assurance agent is employed.

Notwithstanding the sophistication of the development environment, as the ULCS project progressed it became apparent that cost overruns and schedule slippages were occurring. These resulted in an ITT proposal for rephasing the project (known in the program jargon as rebaselining) in April 1979, and another two years later in July 1981. The 1979 rebaselining was considered to be the result of project growth and increased project scope in the ratio of two to one (Crandell: p. 1, 1981), with software among the critical path items. The contractor seems to have seriously underestimated both the size and the difficulty of the software development for the ULCS in its contract proposal; so much so that even though the design was frozen after the 1979 rebaselining, another rephasing was required by 1981. This time software was considered responsible for one-half of the overrun problem (Crandell: p. 33, 1981). The project's history through the 1981 rebaselining is summarized in Table 1 where

TABLE 1

ULCS PROGRAM HISTORY (NAVELEX: 1977-1982; CRANDELL, 1981)

Year	* Est Prgm Size (Kbytes)	# Est Dev Cost ($M)	# Est Dev Sched (Mos)
1977	91	4.1	24
1979	117	6.8	48
1981	235	14.0	66

	# Remaining Software Budget ($M)	# Remaining Software Schedule (Mos)
1977	4.1	24
1979	4.6	29
1981	4.8	19

* AN/TTC-42

\# AN/TTC-42 and SB-3865

ULCS software development cost and schedule at contract award and at each of the rebaselinings is presented along with the estimate of AN/TTC-42 program size (SCPU plus MCPU) for those points. In addition, the estimated remaining development cost and schedule at each point are also presented. Although the cost figures for SB-3865 software development could not be separated from those for the AN/TTC-42, some idea of the error introduced may be gleaned from the AN/TTC-42's position as "software parent" to the smaller circuit switch. Of an

estimated 2408 ULCS software routines only 210, or 8.7 percent, are SB-3865 unique. The remainder were either adopted whole or modified from AN/TTC-42 routines.

The primary intent of this study is to discover the extent to which software estimating models such as COCOMO and Putnam are applicable in development scenarios similar to that of the ULCS. The predictions of all three models (ITT, COCOMO, and Putnam) will be compared in the hope of gaining some insight into the software estimating process. Although the models possess features in addition to cost and schedule estimation, only these will be examined in detail. In particular, techniques for estimating program size will not be considered. The estimates of program size produced by ITT will be accepted as having formed the basis for its cost and schedule estimates, and are therefore considered valid for generating similar estimates with the COCOMO and Putnam models. Finally, development cost will be measured only in units of labor, for example, man-hours. No attempt is made to monetarize these values.

II. MODEL DESCRIPTIONS

A. CATEGORIZATION OF MODELS

The two models for estimating software cost and development time that are the subject of this study and the model used by the development contractor will be described in this chapter. The description of each model will consider its basic estimating methodology; origin and source data base, if known; required inputs; the evaluation process, and model outputs. In addition, model features that are beyond the scope of this study will be briefly summarized for the sake of completeness.

B. THE ITT MODEL

The ITT cost estimating model is described in (ITT: Sec 400.22.2.6, 1981). Although the cover sheet of (ITT, 1981) identifies the publication date as August 1981, the opening sections are dated 1980, and the software cost estimating section is dated October 6, 1981. This researcher was unable to determine the exact extent to which (ITT, 1981) reflects the cost estimating methodology used for the ULCS contract proposal and the two rebaselinings, but conversations with contractor software personnel persuaded this researcher that even if particulars have changed, the underlying philosophy and general methodology remain the same. As will be described below, this philosophy stresses disassembly of the

software product into the smallest possible components. This procedure is followed in the belief that the estimating error associated with these smaller and presumably better understood elements is proportionately smaller than that of larger elements. The estimating philosophy also stresses the requirement for an extensive data base of carefully collected information from previous projects. It notes that estimates of software development efforts are meaningless unless they can be calibrated against previous performance. (Unfortunately the researcher was not able to discern the composition of the current ITT data base.) Finally, ITT rejects the existence of an algorithmic solution to the cost estimating problem, and stresses the role of individual judgment in software cost estimating to a much greater degree than do COCOMO and Putnam. Interestingly, although (ITT, 1981) seems quite concerned with estimating the amount of effort required to develop software products, the model assumes that scheduled development time is a given. This corresponds with industry practices described in (Boehm, 1981) and (Putnam, 1980).

The ITT procedure requires as an input only the overall number of lines of code to be produced by the project. The term "lines of code" is undefined in (ITT, 1981). No differentiation is made between executable lines of code, data declarations or comments. Further, there is no reference to the language level used for project estimating

purposes. The procedure does identify a quantity referred to as "Total Software Effort" measured in lines of code and divisible into support, operational and hardware support categories. The support category encompasses the software required to produce deliverable programs and data bases. It includes such items as "compilers, linkers, editors, debuggers," and "test environment." The operational category consists of the deliverable products themselves, examples of which are "operating systems, communications interfaces, interrupt handlers, on-line diagnostics," and "application data base processing." The hardware support category supports the development of new equipment by assisting the efforts of departments other than software engineering (ITT: Sec 400. 22.2.6, p. 4, 1981).

Given an input in terms of lines of code separated into the categories described above, the ITT estimating process (ITT: Sec 400.22.2.6, pp. 4-6, 1981) requires that each category of software be disassembled until no sub-item is either more than 5 percent of its category or 1000 lines of code if the total software effort is less than 20,000 lines of code. The next step is to compute a "Project Difficulty Factor" (PDF) using the following algorithm:

1. An "Instruction Mix Weight" is assigned to each sub-item. This value is obtained from a table which forms part of the RCA Price S software estimating model. It represents the relative difficulty of producing software for such

applications as operating systems and mathematical operations.

2. The percent of total software effort represented by each sub-item is determined.

3. The "Project Difficulty Factor" (PDF) is computed according to the formula:

$$PDF = \sum_{i=1}^{n} (w_i)(\%_i) \ , \qquad (1)$$

where: w_i = the Instruction Mix Weight of the ith sub-item, and

$\%_i$ = the percent of the total code represented by the ith sub-item.

Following calculation of the "Project Difficulty Factor," the relative complexity of the project under consideration is determined by computing a "Project Complexity Factor" (PCF) for each sub-item using another facet of the RCA Price S model, "Complexity Adjustment" values. These values are provided for personnel, product familiarity and complicating factors. For personnel, the ratings range from "Outstanding crew" to "Relatively inexperienced." Product familiarity similarly ranges from "Old hat, redo of previous work" to "New line of business," and complicating factors includes such items as "New hardware" or "Hardware developed in parallel." When these values have been determined they are entered into equation (2) to obtain the complexity factor.

$$PCF = 1.0 + PV + PFV + CFV, \quad (2)$$

where: PV = Personel value,

PFV = Product familiarity value, and

CFV = Complexity factor value.

When the preceding steps have been completed, an estimate of the effort (man-hours) required to develop each sub-item of code is generated based on a comparison of the Project Difficulty and Complexity Factors with the historical data base. Although no specific directions are presented for making this comparison, it seems to be highly subjective as the estimator is cautioned to include notes "that describe the thought process for determining the estimate" in the project documentation. To arrive at an overall effort estimate, the sub-item estimates are summed with independently determined estimates for non-coding activities such as management, and the development of test plans and documentation. The non-coding estimates are supposed to be based on the amount of coding effort required, but no specific directions for determining the non-coding effort values are given.

(ITT, 1981) does not specify the periods within the development process that are included by the effort estimating methodology, but the data obtained for this study include all effort estimated for the software portion of the project work breakdown structure (WBS) from the drafting of

the Program Design Specification (PDS) to completion of the software integration and test phase. The Program Design Specification is a key project document which is described as follows in (ITT, 1981):

> The Program Design Specification (PDS) describes the structure and design of the software necessary to implement the operational, performance and functional requirements defined in the Program Performance Specification. The program architecture is defined in the PDS, and the programming approach for developing the software is specified.

Software integration and test involves the assembly of the entire deliverable software product, and verification that it meets all of the specified requirements. These points were selected as seeming to best meet the input criteria of the COCOMO and Putnam models.

C. COCOMO

The COnstructure COst MOdel or COCOMO (Boehm, 1981) is a comprehensive software life-cycle cost and schedule estimating model. The primary emphasis of the model is on the development portion of the life-cycle which COCOMO defines as beginning "at the beginning of the product design phase... and end(ing) at the end of the (software) integration and test phase." (Boehm: p. 59, 1981) This definition seems consistent with that used to gather the ITT data on which the study will be based. COCOMO can be employed at three levels of sophistication which in increasing order are Basic, Intermediate and Detailed. The Basic and Intermediate

models are utilized in this study, but insufficient project data was available to allow the Detailed model to be employed.

COCOMO was derived from a study of sixty-three widely varied software development projects which are summarized in (Boehm: pp. 496 & 497, 1981). Boehm describes the method by which he arrived at his estimating relationships in the following quotation:

> The calibration and evaluation of COCOMO has not relied heavily on advanced statistical techniques. After trying to apply advanced statistical techniques to software cost estimation, and after observing similar efforts by others, I have become convinced that the software field is too primitive, and software cost driver interactions too complex, for standard statistical techniques to make much headway; and that more initial progress could be made by trying to formulate empirically the nature of the interactions between cost drivers, using functional forms which reflected the best available perspectives and data on software life-cycle phenomenology. (Boehm: p. 493, 1981)

In contrast with the ITT model, COCOMO is an algorithmic model which posits definite relationships between model inputs and outputs, although depending on the version of COCOMO used, individual judgment may still have a substantial impact on model performance. The Basic model is almost completely deterministic, and only one input requires the exercise of judgment on the part of the user. The Intermediate and Detailed models provide increasing amounts of scope for the application of user judgment.

1. Basic COCOMO

Basic COCOMO (Boehm: pp. 57-113, 1981) is both the least sophisticated and the least accurate of the three

COCOMO models. For inputs it requires only the number of thousands of delivered source instructions in the software product and the mode in which the product is being developed. Delivered source instructions are defined (Boehm: pp. 58 & 59, 1981) as follows:

> Delivered. This term is generally meant to exclude nondelivered support software such as test drivers. However, if these are developed with the same care as delivered software, with their own reviews, test plans, documentation, etc., then they should be counted.
>
> Source Instructions. This term includes all program instructions created by project personnel and processed into machine code by some combination of preprocessors, compilers, and assemblers. It excludes comment cards and unmodified utility software. It includes job control language, format statements, and data declarations. Instructions are defined as lines of code or card images. Thus, a line containing two or more source statements counts as one instruction; a five-line data declaration counts as five instructions.

By software development mode Boehm means a general description of the software product and its development environment. Three modes are defined -- organic, embedded and semi-detached. Organic projects are characterized by relatively small development teams having extensive experience with systems similar to the one under development, and operating in a stable, familiar, in-house environment. No particular importance is attached to early completion of the project. An embedded mode project is developing a product that "must operate within (is embedded in) a strongly coupled complex of hardware, software, regulations and operational

procedures...." A great deal of emphasis is placed on early or at least timely completion of the project as schedule delays may result in large cost overruns. The semi-detached mode is characterized either by an intermediate level of project characteristics or by a mixture of organic and embedded mode characteristics. The development mode determines which of three sets of cost and schedule estimating equations (Boehm: p. 75, 1981) is used in the Basic model. In the case of the AN/TTC-42 the rigid interface specifications between the circuit switch software and TRI-TAC software, as well as the equally inflexible interfaces between the circuit switch hardware and software place the development effort in the embedded mode.

Once the number of thousands of delivered source instructions (KDSI) and the development mode have been determined, an estimate of project software development effort (MM) in man-months (1 man-month = 152 hours of working time) can be made using equation (3) in the case of the embedded mode.

$$MM = 3.6\ (KDSI)^{1.20} \qquad (3)$$

In addition to providing estimates of software development effort and schedule, Basic COCOMO has two other features of note. It provides a breakdown of the development effort and schedule estimates by phase and it provides a means for

estimating the amount of effort required to maintain the software once it has been developed. Schedule and effort distributions over the three phases covered by the model (Product Design, Programming, and Integration and Test) and the Plans and Requirements phase are provided for several project sizes. The distributions for intermediate size projects are obtained by linear interpolation. Estimated Annual Maintenance Effort is determined based on the amount of effort required to produce the software and a parameter known as Annual Change Traffic, which is defined as, "The fraction of the software product's source instructions which undergo change during a (typical) year, either through addition or modification." (Boehm: p. 71, 1981)

2. Intermediate COCOMO

Intermediate COCOMO (Boehm: pp. 114-163, 1981) provides increased estimating accuracy over that available from Basic COCOMO by using two additional inputs, Equivalent Delivered Source Instructions (EDSI) and an Effort Adjustment Factor (EAF). The primary input to the model is still thousands of delivered source instructions (KDSI), but provision is made in the Intermediate model for differentiating between newly developed software and existing software adapted for use in the project. The model provides an algorithm that converts the quantity of adapted code into an equivalent number of new delivered source instructions (EDSI)

based on such factors as the percent of the adapted software's design that required modification in order to function in the new environment. When adapted software is used on a project, the Equivalent Delivered Source Instructions are divided by 1000 and summed with KDSI to provide a new effort estimating equation input -- KEDSI. The AN/TTC-42 uses no adapted software; so in this case KDSI equals KEDSI.

The other new input, the Effort Adjustment Factor (Boehm: pp. 117-120, 1981) provides a means of modifying the amount of development effort estimated to be required for a project based on fifteen cost driver attributes which are listed in Table 2. Cost drivers are generally rated on a scale of very low to extra high although some do not have values at one or both extremes of the scale. Once the software cost driver ratings have been determined, the associated Software Development Effort Multipliers (SDEMs) are extracted from (Boehm: p. 118, 1981). The Effort Adjustment Factor can then be calculated with equation (5).

$$EAF = \prod_{i=1}^{15} SDEM_i \qquad (5)$$

In developing Intermediate COCOMO development effort estimates the Basic COCOMO relationship, equation (3), is retained but with a different coefficient which was empirically determined by Boehm to provide better results in the

TABLE 2

COCOMO COST DRIVER ATTRIBUTES (BOEHM: PP. 115 & 116, 1981)

Product Attributes	
RELY	Required Software Reliability
DATA	Data Base Size
CPLX	Product Complexity
Computer Attributes	
TIME	Execution Time Constraint
STOR	Main Storage Constraint
VIRT	Virtual Machine Volatility
TURN	Computer Turnaround Time
Personnel Attributes	
ACAP	Analyst Capability
AEXP	Applications Experience
PCAP	Programmer Capability
VEXP	Virtual Machine Experience
LEXP	Programming Language Experience
Project Attributes	
MODP	Use of Modern Programming Practices
TOOL	Use of Software Tools
SCED	Required Development Schedule

Intermediate model. The general case of the Intermediate model is one in which there is no adapted software and all of the cost driver ratings are nominal. The Software Development Effort Multiplier for a nominal cost driver rating is one, which by equation (5) means that the Effort Adjustment Factor for the general case is also one. The resulting Intermediate COCOMO nominal effort, $(MM)_{NOM}$,

estimating equation is equation (6).

$$(MM)_{NOM} = 2.8\ (KDSI)^{1.20} \qquad (6)$$

For cases in which adapted software is utilized or for which non-nominal software development cost driver ratings exist, adjusted software development effort, $(MM)_{DEV}$, is computed by equation (7).

$$(MM)_{DEV} = 2.8\ (KEDSI)^{1.20}\ (EAF) \qquad (7)$$

The Intermediate COCOMO software development schedule (TDEV) estimating equation is equation (8). It is the Basic COCOMO relationship modified by the substitution of adjusted development effort $(MM)_{DEV}$ for development effort (MM).

$$TDEV = 2.5\ (MM)_{DEV}^{.32} \qquad (8)$$

In addition to estimates of software development effort and schedule based on overall program size, Intermediate COCOMO provides procedures (Boehm: pp. 146-152, 1981) for computing these same estimates from smaller units of code called components. Boehm does not define the term, but directions for employing components in the estimating process and examples in

(Boehm, 1981) indicate that it refers to a major software subdivision one step below the overall product. Examples provided in (Boehm: p. 155, 1981) for a communications front end processor include task control, communications, and status monitoring. As it is unlikely that the cost driver ratings for all of the software components will be identical to that for the overall product, the effect of the use of this two-level estimating procedure should be to increase model accuracy.

In an extension of the Effort Adjustment Factor concept, Intermediate COCOMO enables the user to compute an adjusted estimate of Annual Maintenance Effort (Boehm: pp. 129-132, 1981) utilizing the cost driver attributes in Table 2 with the exception of Required Development Schedule. The Intermediate COCOMO distribution of effort by phase and schedule is the same as in the Basic model.

3. Detailed COCOMO

Detailed COCOMO contains two extensions to the Intermediate model, phase-sensitive effort multipliers and a three-level product hierarchy. In Intermediate COCOMO, effort multipliers are assumed to have one value applicable over the life of the project. The Detailed model provides discrete effort multiplier values for the cost drivers during each phase of the project. The Detailed model also utilizes a three-level product hierarchy of module, subsystem,

and system in place of the two-level hierarchy used in the Intermediate model. At the lowest level, the module, the cost drivers most likely to affect the output of individual programmers are applied, while the remaining cost drivers are used at the subsystem level. These lower level estimates are aggregated and used to produce project estimates of effort and schedule at the system level. The estimated Annual Maintenance Effort is identical to that of the Intermediate model, and the phase and schedule distribution of effort is the same as that in the Basic model.

D. THE PUTNAM MODEL

The Putnam model comprises those portions of a proprietary software development system, the Software Life Cycle Model (SLIM), that have been placed in the public domain (Putnam; Thibodeau: pp. A-63 - A-80, 1981). The general principles of the proprietary system are the same as those in the Putnam model, but SLIM contains additional features which improve estimating accuracy, particularly for projects of less than 70,000 source statements (Putnam: pp. 319 & 320, 1980; Thibodeau: p. A-68, 1981). The model is based on the premise that the rate at which effort is expended in the solution of development problems follows a so-called Rayleigh-Norden distribution (Putnam: pp. 17 & 18, 1980). Putnam's development of this concept while at the U.S. Army Computer Systems Command and also with data from the U.S. Air Force's Rome

Air Development Center (RADC) led to the formulation of the model (Putnam: pp. 32 & 37, 1980).

The model draws its utility from the proposition that within defined limits, software development efforts (E) and development time (t_d) can be substituted for each other. The relationship describing the rate at which one may be traded for the other is described by the Software Equation, equation (9).

$$S_s = C_k \left(\frac{E}{.4}\right)^{1/3} t_d^{4/3} \qquad (9)$$

The Software Equation requires as inputs the amount of effort to be expended in developing the software (E), the length of time allotted for software development (t_d), and the technology constant (C_k). The nominal output of the model is program size in source statements (S_s), although as noted above the more likely case uses inputs of program size, technology constant, and one of the remaining parameters to determine either required development effort given the development time or development time given effort.

The example of the Software Equation given in equation (9) uses the effort expended during the development period as the input parameter. This version of the equation was used as development effort in one of the parameters of interest in this study. The more common form of the Software

Equation uses life cycle effort (K). The conversion from life cycle to development effort is accomplished using equation (10).

$$E = .4(K) \tag{10}$$

This relationship is based on the premise that the maximum rate of expenditure of effort on a project occurs just prior to the completion of the development period (Putnam: pp. 7, 17 & 314, 1980). By analyzing the project data available to him, Putnam determined that at this point forty percent of the life cycle effort has been expended. His concept of what constitutes development time (t_d) does not extend much beyond the explanation offered above for development effort. The figure contained in (Boehm: p. 314, 1981) shows two phases, design and coding and test and validation, in the development period, but provides no additional information concerning their composition. In addition, a portion of the system installation effort is also contained in this period. Insufficient information concerning the model is available to make a judgment concerning the degree to which the available ITT data meet the model criteria.

The technology constant (C_k) is a dimensionless value "which is somehow a measure of the state-of-technology being applied to the project." (Putnam: p. 168, 1980). It "is

obtained by calibrating the model using historical data that are representative of the project to be estimated" (Thibodeau: p. A-77, 1981), and "measures any throughput constraints that impede the progress of programmer/analysts.... C_k is quantized...(and) varies in a set sequence of allowable values (Fibonacci sequence.)" (Putnam: p. 7, 1980)

The Software Equation is not applied in a vacuum. As noted above the degree to which effort and development time may be substituted for each other is not unlimited. For each type of development undertaken by an organization, for example, "stand alones" and "rebuilds," there is a relationship of the form given in equation (11) that defines the "Difficulty Gradient" (∇D) for that particular organization and type project.

$$\left(\frac{E/.4}{t_d^3}\right) = \nabla D \qquad (11)$$

The difficulty gradient defines the minimum feasible combinations of development effort and time required to produce a software product of a given size. It is derived from a study of the software development organization's previous performance.

In addition to the Software Equation, the Putnam model demonstrates an expected value technique for estimating project size from independent estimates of the largest possible, least possible and most likely project sizes. The

model also demonstrates the use of the Monte Carlo method for determining the sensitivity of the model to uncertainties in the model inputs. Linear programming solutions to development effort and schedule optimization problems are discussed, as are techniques for determining optimal project man-loading. An example of dynamic project control using differential equations is presented, and the elements for a software cost estimating data base are recommended.

III. DATA AND EVALUATION PROCEDURE

The data evaluated in this study are listed in Appendix A. They comprise three data points which were extracted from the 1979 and 1981 rebaselining documentation (NAVELEX, 1977-1982) by the Naval Electronic Systems Command's support contractor, the Federal Systems Group of Calculon Corporation. Two of the data points (1977 and 1979) provide estimates of program size, development effort and schedule. The third data point (1981) includes estimates of program size and development schedule only. In addition to these data points a stand along estimate of program size (Crandell, 1981) for 1981 was obtained. The Crandell estimate was obtained from a report to the Director of the Joint Tactical Communications (TRI-TAC) Office concerning the December 1981 status of the ULCS program. The report contains a recapitulation of the project's history, discusses the probable results of continuing the existing program management approach, and makes recommendations for improving program performance.

ULCS program size estimates are maintained by ITT in lines of program design language (PDL) and bytes. Program design language is a pseudo-higher order language which aids programming personnel in applying the principles of top-down design and structured programming. In the case of this project it is closely related to the formal programming

language (XAS-8) and supposedly provides a smoother transition from program design documentation to the actual programs than is provided by such techniques as flow charting. Program size in lines of PDL was determined by count. Program size in bytes was also determined by count and is used to generate two additional estimates in terms of 8080A instructions and lines of XAS-8 assembly code. Program estimates in bytes are convertible to 8080A instructions at the rate of 1.8 bytes per instruction, and to lines of XAS-8 at the rate of 3.1 bytes per line (NAVELEX, 1977-1982). The 8080A conversion rate is based on an "average" 8080A instruction size. Actual 8080A instructions occupy one, two or three bytes. The XAS-8 figure was obtained from a NAVELEX study of a sample of the source code and its corresponding object code. The Crandell estimate, which was made in an unspecified quantity, "lines of code" (LOC), was derived from a computer sort of 5 percent of the source code.

Estimated development effort was submitted by the contractor in man-hours which are considered convertible for the purposes of this study to man-months (MM) at the Boehm rate of 152 manhours per man-month. A man-year (MY) is defined as twelve man-months. The estimates encompass all work charged to software from the start of the program design specifications to the completion of software integration and test. Unfortunately the estimates for the AN/TTC-42 and the

SB-3865 could not be separated and the figures listed in Appendix A are for the entire ULCS project, as noted in the introduction. An estimate of the effort expended up to the 1979 rebaselining was also obtained. Estimated development schedule was provided in terms of start and stop dates for the period covered by the estimates of development effort.

Additional model inputs, unique to COCOMO or Putman, were required before the evaluation process could begin. In the case of COCOMO, information was required to compute the effort adjustment factor (EAF). Estimates of the cost driver ratings were solicited from both contractor and NAVELEX software personnel, all of which corresponded closely. The researcher's synthesis of these estimates is contained in Appendix B, and yields an EAF value of 1.11. In the case of the Putnam model, a value for the technology constant (C_k) was required. After review, a value of 10040 was adopted as representing a conservative estimate of the advanced state of software engineering technology employed by ITT. This figure was extracted from a 1979 article (Putnam: p. 317, 1980) and "represents an average state-of-the-art development environment using on-line interactive development." Given the unusual sophistication of the ITT software development environment the researcher feels that this figure is somewhat conservative.

The evaluation of the data proceeded as follows. Since none of the available estimates of program size met the criteria for delivered source instructions (Boehm: pp. 58 & 59, 1981), all available estimates of program size were used as inputs to the COCOMO model. The estimates of development effort and schedule which resulted were then compared to the ITT estimates and a percent difference figure of the form given by equation (12) was generated.

$$\frac{\text{Boehm} - \text{ITT}}{\text{ITT}} = \Delta\% \qquad (12)$$

In addition, contractor estimates for development effort and schedule were entered into the equations in order to obtain the implied values of the other software parameters. The results of the Basic and Intermediate COCOMO models are contained in Appendices C and D respectively.

The Putnam model was evaluated by using ITT estimates of development effort and schedule with the assumed technology constant of 10040 as inputs to the model. The predicted program size was then compared to the estimated program size in lines of XAS-8, which was considered the metric most closely resembling that defined for the model by Putnam. The estimates were also used to generate a percent difference figure ($\Delta\%$) in the manner described above. The results of the Putnam evaluation are contained in Appendix E.

IV. MODEL APPLICABILITY AND PERFORMANCE

The most significant factor in accounting for model performance is the amount and type of information available for evaluation. Thibodeau quotes another researcher on this subject:

> All of the data used...were data of opportunity, i.e., we took what we were able to get in the time available. Hard data on the costs of computer programming...are scarce commodities both in computer programming organizations and in the published literature. Few numerical data are recorded; fewer yet are recorded under "controlled" conditions, and still fewer are suitable for generalization to other situations.... (Thibodeau: pp. 6-11 & 6-12, 1981)

The next most important factor is the scope of the work to be done as measured by program size. The data which the models were developed to explain consist of completed projects and, as such, the relationships to be observed among the data are static. In a dynamic situation like the AN/TTC-42 program, relationships among project estimating parameters similar to those in the model data bases may not exist until the end of the development period. This inability to forecast project scope is particularly damaging in the case of staffing. Both models (Boehm: pp. 89-94, 1981; Putnam: pp. 25-27, 1980) develop desired staffing levels along a time line that represents development schedule. Putnam provides examples of what happens when the required staffing levels are not maintained. One of these examples

(Putnam: p. 27, 1980) illustrates the situation that occurs when management adds personnel in step-like increments, but makes the additions only after perceived slips in development schedule. Instead of following the Rayleigh curve, staffing either exceeds requirements and labor is wasted, or there is too little labor available and the program slips. The researcher believes that, in effect, this situation is similar to the AN/TTC-42 program in which the amount of work required is constantly increasing. Staffing is rarely at the optimal value, because the correct value of program size on which to base estimates of development effort and staffing is constantly changing and never known with certainty until after the fact. The Putnam model provides a trade-off mechanism for time and development effort, but this is not an open-ended relationship. It is bounded below by the difficulty gradient, so that, at some point the schedule stretchout caused by increases in project size cannot be overcome by adding more people.

In addition to uncertainty concerning the size of the program, the units used to measure size are also a problem. ITT's method of accounting for program size in bytes does not meet the requirements of the models. Unfortunately, neither do any of the other measures of program size that are readily available. The metric most similar to the delivered source instructions and source statements required would seem to be lines of XAS-8 assembly code. However,

this is deceptive. While it is true that XAS-8 is what the programmers write, a prime requisite of the models, it is not the language in which they test and debug. XAS-8 input to the macro assembler produces an output of 8080A machine instructions which actually execute on the processors. Unlike the case in which a higher order language (HOL) is used, debugging for the AN/TTC-42 is not done in the HOL, but in 8080A instructions. No mechanism is mentioned in the models for handling such an anomaly.

An important aspect of program size not easily assessed is the effect of the control tables used in this real-time, event-driven system. Putnam does not address the handling of data elements, and Boehm approaches them from the point of view that if a programmer writes a data declaration it counts as a delivered source instruction. (Actually, as noted in the discussion of his model, he considers a delivered source instruction to be a "card image" which could contain multiple data declarations.) This manner of handling data does not seem applicable to the control tables. All concerned with the project agree that these control tables are extremely difficult to construct, and, as program size has expanded, so has the size of the tables. From a 1977 estimate of 22 kbytes, the tables have expanded to a 1981 estimate of 53 kbytes. This 138 percent increase in the most difficult part of the programming effort may have

had a much greater effect on the project than would the same increase in the number of lines of XAS-8, particularly if the tables became increasingly complex in an effort to prevent the program from exceeding the main storage constraint.

The COCOMO cost driver rating for the main storage constraint is "Very High" but still understates the effect that it has had on the program. To deliver a functional product the contractor clearly must provide sufficient memory to accommodate the program and data. The AN/TTC-42 main memory storage capacity and estimates of combined program and data storage requirements for each data point are contained in Appendix A. (Note: These figures accord with the Boehm definition of main storage [Boehm: p. 410, 1981], and do not include data stored in a bulk storage unit which was part of the 1977 and 1979 estimates.) While the main memory constraints indicated in Appendix A would not seem particularly significant, they must be viewed in light of the continued growth of the program and data base over the development period. Although at the time the project started in 1977 the programmers had an 18 percent reserve of memory, this hand dwindled away to 2 percent by the time of the 1979 rebaselining.

If the programmers had been uninhibited by the approaching exhaustion of memory resources during the 1977-1979 period this would have had little effect on the project, but the researcher considers this to have been extremely unlikely.

Instead, it is more likely that, as the memory reserve eroded, the contractor would have placed increasing emphasis on utilizing the most memory-efficient methods possible to implement program tasks. Presumably, this constraint increased the difficulty of programming system routines and may have led to increased complexity and size of the control tables as a memory conservation measure. Effort may also have been expended reworking already completed routines to compress their memory requirements. The point here is not so much the value of the main storage constraint at the start of the baseline, but rather the increasing pressure placed on the project personnel to fit the expanding program and data into available memory as the hardware redesign limit was approached. More than likely this process was repeated during the 1979-1981 time frame.

Another factor adversely affecting model performance is the manner in which the Intermediate COCOMO effort adjustment factor was estimated. The cost driver ratings used for the evaluation represent average ITT values over the life of the project. This situation is complicated by the presence of more than one software development organization. The ITT portion of the programming effort, with which the personnel interviewed were most familiar, was confined to the switching control processor. ITT subcontracted the development of the MCPU software, the software tools, and the environmental simulator. No estimate of the competence of the subcontractor

personnel was obtained, and, had one been available, an attempted synthesis of the subcontractor and ITT personnel values would have produced a figure of little significance. However, if this data could have been coupled with sufficient program structure information in order to use the Detailed COCOMO model, the results might have been quite different than those produced by the Intermediate model.

In addition to lack of knowledge of the ITT subcontractor personnel qualifications, the researcher was not able to obtain estimates of the programming effort required to produce the software tools and the environmental simulator in use on the project. This effort must have been substantial given the sophistication of the tools described in the introduction, and it is clearly included in Boehm's model by his definition of deliverable source instructions. The inclusion of this data in the Boehm model inputs might have substantially reduced the variance between the ITT and COCOMO effort predictions.

Other difficulties encountered in applying the models came from determining the technology constant and the appropriate time units to use with the Putnam model. No explicit guidance is provided by Putnam for determining the value of the technology constant. If, as Putnam states, the technology constant is to be determined from the past performance of the organization, then it encompasses considerably more than "the use of modern programming practices, the language

used, the development environment (on-line, interactive development versus batch), and the availability of the development machine," that he mentions. In some way it is measuring the overall software development capability of the organization, and this use of the technology constant makes it what Thibodeau (Thibodeau: p. A-69, 1981) terms a self-calibrating model. This is particularly significant for model performance as Thibodeau found in a study of eight software cost estimating models that the factor most significant in increasing model accuracy was calibration to the development enviornment (Thibodeau: p. 1-8, 1981). In view of this, the lack of a technology constant actually based on the ITT development environment must be considered a serious shortcoming of the modeling process.

The difficulty with the time scale arose from statements by Putnam (Putnam: pp. 5 & 53, 1980) that the input to his model may be measured in a variety of units. The examples in (Putnam, 1980) are calculated in years and man-years, and no mention is made of any changes to the software equation that are required if other units are used. For comparison the researcher conducted calculations with the Putnam model using two different time scales for the same inputs. The results in Appendix E indicate that time units are not interchangeable without the use of a different technology constant.

No correlation between the results of the ITT procedure and the models was posited, and none appears in the results of the evaluation. A significant trend noted was that the models consistently predicted that substantially less time and effort should have been required to complete the project than was predicted by ITT for all model inputs except bytes. Basic COCOMO produced percent difference figures for development effort and schedule estimates ranging from a best case of minus thirteen and minus twenty-nine percent respectively for the 1977 Intel 8080A instruction input, to minus eighty-seven and minus seventy-eight percent for the 1979 PDL input. Intermediate COCOMO estimates closely followed those of the Basic model, but substantial variations were noted for the Putnam model. When inputs of effort in man-months and development time in months were used with this model, estimates of program size exceeding those made by the contractor by several thousand percent resulted. In addition, when the effort expended up to the 1979 rebaselining (190 man-months) is input to the Basic COCOMO model, a value of 27251 delivered source instructions results. This correlates quite well with the 29498 lines of XAS-8 estimated from the rebaselining proposal, but ITT estimated a development schedule of forty-eight months while the model predicted that the 27 KDSI should have been developed in thirteen months.

V. CONCLUSIONS

Little is likely to be accomplished using the COCOMO and Putnam models on software development projects like the AN/TTC-42 until the size of the program that has to be developed can be estimated with reasonable accuracy or at least fixed between useful limits. (But see [Thibodeau: p. 5-29, 1981] for the performance of a model that does not use program size as an input.) Even were accurate estimates of this and other estimating parameters available, however, the utility of the models would still be severely limited by the differences between the inputs as they are defined in the models and as they are available in the development environment. Where the parameters are clearly defined (COCOMO) this problem is amenable to solution by software development personnel. However in the case of the Putnam model the parameters are not sufficiently well defined, with the possible exception of the program size parameter, to provide a useful guide for a data collection effort. In particular, the means of determining the technology constant used with this model are completely inadequate.

APPENDIX A

DATA

	1977	1979	1981
Program Size			
Bytes	91,443	116,640	234,750
8080A Instructions	50,802	64,800	130,417
XAS-8	29,498	37,626	75,726
Lines of Code (Crandell)			100,000
Software Development:			
Effort (ULCS)			
Man-hours	70,195	100,682	
Man-months	462	662	
Man-years	38.5	55.2	
Schedule (AN/TTC-42)			
Months	24	48	66
Years	2	4	5.5
Data (Bytes)			
Total	105,494	127,413	154,166
Control Tables	22,285	22,780	53,000
Bulk Storage	36,000	54,700	-0-

APPENDIX B

EFFORT ADJUSTMENT FACTOR

	Rating	Effort Multiplier
Product Attributes		
RELY	Very High	1.40
DATA	Low	.94
CPLX	Very High	1.30
Computer Attributes		
TIME	Nominal	1.00
STOR	Very High	1.21
VIRT	Low	.87
TURN	Low	.87
Personnel Attributes		
ACAP	High	.86
AEXP	High	.91
PCAP	Nominal	1.00
VEXP	Very Low	1.21
LEXP	Nominal	1.00
Project Attributes		
MODP	Very High	.82
TOOL	Very High	.83
SCED	Very High	1.10

Effort Adjustment Factor = 1.11

APPENDIX C

BASIC COCOMO

In this appendix, the software development parameters have been generated using the data in the left hand column as inputs to the Basic COCOMO estimating equations, equations (3) and (4). The inputs are either ITT estimates (Bytes, PDL, development effort and development schedule) or are generated from ITT estimates (8080A instructions and XAS-8) as described in the text. To simplify comparison of the estimates of software development effort and schedule produced by the BASIC COCOMO model and those estimated by ITT, a percent difference figure is calculated for all cases except 1981 development effort for which no ITT estimate was available. In addition, where ITT estimates of software development effort or schedule are used as inputs, an implied value for program size is generated.

1. 1977

	KDSI	MM	Δ%	TDEV	Δ%
Bytes	91.443	812	76	21	-11
8080A	50.802	401	-13	17	-29
XAS-8	29.498	209	-55	14	-42
PDL	11.429	67	-86	10	-60
MM = 462	57	462	-0-	18	-26
TDEV = 24	124	1174	154	24	-0-

2. 1979

	KDSI	MM	Δ%	TDEV	Δ%
Bytes	116.640	1088	64	23	-51
8080A	64.800	537	-19	19	-61
XAS-8	37.626	280	-58	15	-68
PDL	13.954	85	-87	10	-78
MM = 662	77	662	-0-	20	-58
TDEV = 48	756	10240	1447	48	-0-

3. 1981

	KDSI	MM	Δ%	TDEV	Δ%
Bytes	234.750	2518		31	-54
8080A	130.417	1244		24	-63
XAS-8	75.726	648		20	-70
PDL	29.343	208		14	-79
LOC	100	904		22	-67
TDEV = 66	1732	27702		66	-0-

APPENDIX D

INTERMEDIATE COCOMO

The input parameters used with the Basic COCOMO model are here employed with the Intermediate COCOMO estimating equations, equations (7) and (8). As no adapted software is used in the AN/TTC-42 programs, the values for KEDSI and KDSI are equal. The Effort Adjustment Factor is calculated in Appendix B, and was found to be 1.11. Percent difference figures for the Intermediate COCOMO and ITT estimates are calculated in the same manner as in Appendix C, and again, no comparison of estimated software development effort at the 1981 rebaseline point was possible for lack of an ITT value.

1. 1977

	KEDSI	MM	Δ%	TDEV	Δ%
Bytes	91.443	701	52	20	-15
8080A	50.802	346	-25	16	-32
XAS-8	29.498	180	-61	13	-45
PDL	11.429	58	-87	9	-62
MM = 462	65	462	-0-	18	-26
TDEV = 24	140	1176	154	24	-0-

2. 1979

	KEDSI	MM	Δ%	TDEV	Δ%
Bytes	116.640	939	42	22	-53
8080A	64.800	464	-30	18	-63
XAS-8	37.626	242	-63	14	-70
PDL	13.954	73	-89	10	-79
MM = 662	87	662	-0-	20	-58
TDEV = 48	854	10240	1447	48	-0-

3. 1981

	KEDSI	MM	Δ%	TDEV	Δ%
Bytes	234.750	2174		29	-56
8080A	130.417	1074		23	-65
XAS-8	75.726	559		19	-71
PDL	29.343	179		13	-80
LOC	100	781		21	-68
TDEV = 66	1957	27702		66	-0-

APPENDIX E

THE PUTNAM MODEL

In this appendix the Software Equation, equation (9), is used to generate estimates of program size in source statements (S_s) from an assumed technology constant (C_k) and ITT estimates of development effort (E) and schedule (t_d) in two time scales. The estimated number of source statements is compared to the number of lines of XAS-8 source code estimated to have been required and a percent difference figure is calculated for these values. Lack of a 1981 ITT effort estimate prevented a program size estimate from being made at that point.

1. 1977

t_d = 24 mos	t_d = 2 yrs
E = 462 MM	E = 38.5 MY
C_k = 10040	C_k = 10040
S_s = 7,292,493	S_s = 115,942
XAS-8 = 29498	XAS-8 = 29498
Δ% = 24622	Δ% = 293

2. 1979

t_d = 48 mos	t_d = 4 yrs
D = 662 MM	E = 55.2 MY
C_k = 10040	C_k = 10040
S_s = 20,716,737	S_s = 329,438
XAS-8 = 37626	XAS-8 = 37626
Δ% = 54960	Δ% = 776

LIST OF REFERENCES

(Boehm, 1981). B. W. BOEHM, Software Engineering Economics, Prentice-Hall, Englewood Cliffs, New Jersey, 1981.

(Crandell, 1981). R. M. CRANDELL, "ULCS Program Evaluation Final Report," December 1981. Unpublished report to the Director, Joint Tactical Communications Office.

(Huebner-Wright, 1980). R. E. HUEBNER and B. T. WRIGHT, "Bridging the Gap -- Analog to Digital Tactical Switching," Signal, November 1980, pp. 41-46.

(ITT, 1981). International Telephone and Telegraph Corporation - Defense Communications Division, "Engineering Operations Manual 400.22 -- Software Engineering Minimum Standards and Procedures," ITTDCD, Nutley, New Jersey, August 1981.

(NAVELEX, 1977-1982). A Collection of unpublished extracts from Naval Electronic Systems Command reports and data summaries in the possession of the author, 1977-1982.

(Proposal). International Telephone and Telegraph Corporation - Defense Communications Division, Appendix I, "Software Management Plan," to the ITTDCD Unit Level Circuit Switch contract proposal, no date.

(Putnam, 1980). L. H. PUTNAM, Tutorial on Software Cost Estimating and Life-Cycle Control: Getting the Software Numbers, IEEE, New York, New York, 1980.

(Thibodeau, 1981). R. THIBODEAU, An Evaluation of Software Cost Estimating Models, Final Technical Report, RADC-TR-81-144, General REsearch Corporation, June 1981. DTIC No. AD-A104226.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3.	Department Chairman, Code 54 Department of Administrative Sciences Naval Postgraduate School Monterey, California 93940	1
4.	Assistant Professor Dan C. Boger, Code 54Bk Department of Administrative Sciences Naval Postgraduate School Monterey, California 93940	1
5.	Commander Marshall L. Sneiderman, SC USN Code 54Zz Naval Postgraduate School Monterey, California 93940	1
6.	Litutenant Colonel J. F. Mullane, USMC Code 0309 Naval Postgraduate School Monterey, California 93940	1
7.	Project Manager PME-154 ULS Naval Electronic Systems Command Washington, D.C. 20360	10
8.	Major William B. Collins, USMC PME-154 ULS Naval Electronic Systems Command Washington, D.C. 20360	2


FILMED
9-8