JPL PUBLICATION 81-7

Deep Space Network Software Cost Estimation Model

Robert C. Tausworthe

April 15, 1981

National Aeronautics and
Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

JPL PUBLICATION 81-7

Deep Space Network Software Cost Estimation Model

Robert C. Tausworthe

April 15, 1981

National Aeronautics and
Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

The research described in this publication was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

Contents

Figures

Tables

Abstract

This report presents a parametric software cost estimation model prepared for JPL Deep Space Network (DSN) Data Systems implementation tasks. The resource estimation model modifies and combines a number of existing models, such as those of the General Research Corp., Doty Associates, IBM (Walston-Felix), Rome Air Force Development Center, University of Maryland, and Rayleigh-Norden-Putnam. The model calibrates the task magnitude and difficulty, development environment, and software technology effects through prompted responses to a set of approximately 50 questions. Parameters in the model are adjusted to fit JPL software life-cycle statistics. The estimation model output scales a standard DSN Work Breakdown Structure, which is then input to a PERT/CPM system, producing a detailed schedule and resource budget for the project being planned.

Deep Space Network Software Cost Estimation Model

I. Introduction

The early-on estimation of required resources and schedule for the development and maintenance of software has probably been the least precise aspect of the software life cycle, and yet, an orderly and rational attempt must be made in order to plan and organize an implementation effort. The existence of an orderly and rational approach implies the existence of a resource and schedule model that accepts as input the technical requirements to be achieved — the magnitude of the task; the physical, environmental, human, and management constraints assumed or known to be in effect; the history base of similar and dissimilar experience; the means, alternatives, and technology available to the task; and a theory which is capable of correlating these parameters with measured results.

The least precise of such models is one which relies entirely on experience, intuition, and luck. It is sometimes referred to as a "WAG", or "Wildly Aspiring Guess." (More often, the acronym is somewhat differently derived, but with the same general connotation.) When a more formalized, mathematical model with some statistical verification can be formulated, the model appellation is upgraded to "Scientific WAG", or "SWAG."

The prediction of human behavior is the problem of estimating events in a stochastic process governed by an unknown probability function. The goal of a SWAG model is therefore to predict the events in such a way as to produce minimum variance (or risk). The optimum SWAG model can predict only to the limit imposed by the statistical distribution actually characterizing the human activity.

The optimal SWAG model would require the precise quantification of all technical, environmental, and human behavioristic parameters, and would combine these into a mathematical formula producing maximum likelihood or least mean square error results. Lacking this precise quantification, the best that one may hope for in a SWAG model is that it accommodate the principal factors affecting the estimate variance (or risk) in a way that reduces the variance (or risk) from what it would be, had that factor not been included.

There are a number of SWAGs in existence. Fourteen software cost estimation models are summarized in Ref. 1, and nine are evaluated for JPL use in Ref. 2. None of these, by itself, seemed to the author to contain sufficient range of application and adaptability to the diverse kinds of software being produced at JPL to quantify the relevant cost factors and risks with sufficient accuracy to be useful. Taken all together, however, these models seemed to possess, in their union, the potential for as good a SWAG as could be obtained at the current state of the art.

An IBM study (Walston-Felix, Ref. 3) reported the analysis of 60 software projects with respect to 68 variables believed to influence productivity. Of these, 29 showed a significant, high correlation with productivity and were included in their estimation model.

A number of models reported in Ref. 1 (General Research Corp., Doty Associates, TRW, Air Force Electronic Systems Division, Tecolote, Aerospace Corp., et al.), as well as statistics from the University of Maryland (Ref. 4), provide productivity data with best-fit curves using many fewer parameters.

Still other models, notably the Rayleigh-Norden Putnam model (Refs. 5, 6) presuppose a few-parameter, specific mathematical model, which is then calibrated using available industry data to provide tradeoffs between effort, duration, and risk.

Several models (e.g., Wolverton, Ref. 7) proceed to detail resource expenditures into the various phases of activity. Some of the models are fully automated, such as PRICE-S (Ref. 8), SLIM (Ref. 9), and SLICE (Ref. 10). The others appear to be calculative, or perhaps small programs.

The software cost model described in this report is fully automated; it borrows and extends features from many of the models above. It utilizes 7 factors from the GRC model, 29 factors from, or similar to, the Walston-Felix model, and incorporates an inherited (or existing) code model due to the author, exposed herein. It utilizes the "PERT" technique to estimate the expected size and variance of the software to be produced. It utilizes a modification of the Rayleigh-Norden-Putnam model to check on the feasibility of resource estimates. It applies the estimated effort, staff, and duration to a standard Work Breakdown Structure (WBS) developed for DSN software tasks, and automatically produces a task plan and schedule to be used at the initial system/subsystem planning, software implementation planning, or software maintenance planning stages of a project.

There are 68 parameters within the model which relate to productivity, duration, staffing level, documentation, and computer resources. Another 69 parameters divide the total estimated effort among each WBS subtask; an additional 69 relate total duration to subtask duration. Subtask precedences are adjustable, and drive a PERT/Critical-Path-Method scheduling algorithm.

The outputs of the model include estimates and variance values for program size, staff productivity, effort, duration, staffing, documentation, and computer resources, together with complete scheduling early/late start/finish and float-time data, plus a Gantt chart (schedule bar chart) of the planned activities.

All parameters in the automated model are easily altered by a simple text editing process, without recompilation of the programs. For all its seeming complexity, the model itself is simple to use. Only a series of questions relating to size and environment need to be answered.

The ensuing sections of this report describe the model and its parameters, and discuss the source or deviation of parameter values and other elements of the model. The values of parameters cited in this report are subject to modification and refinement as further calibration and usage of the model proceeds. The reader interested in a current set of values at any time subsequent to the publication of this report may consult the Software Specification Document (Ref. 11).

Concerning accuracy: at this writing, insufficient data has been collected from DSN projects to optimize the parameters of the model to fit DSN productivity, duration, etc. Therefore, the model accuracy is unknown, as pertaining to DSN predictions. However, the model does fit industry statistics (or can be made to fit any of the cited source models) by proper parameter selection. For this reason, it is felt that the few JPL data points that have been factored in will yield accuracy figures as good as, or perhaps better than, the other models in their stated environments.

II. Model Description

The Software Cost and Resource Estimation Model (the acronym SCAREM is *not* used!) overview appears in Fig. 1 in data-flow-diagram format. Program size and staff productivity factors are separately estimated and then combined to estimate effort, staffing, duration, documentation, and computer resources. The model produces uninflated dollar costs for documentation and computer resources. Both estimated mean and variance values for all resources are output.

Estimated values are presented in the automated model to the user as advice. The user is admonished to use these figures with sufficient risk biases to ensure project completion within a desired confidence value. The model then accepts any two of the three parameters: effort, average staff, and duration.

These entries are checked against a model akin to the Rayleigh-Norden-Putnam model, but altered to conform with power-law fits to measured data. Warnings are issued if the entries are unreasonable. The user may alter the input estimates, if desired, for another check.

Once acceptable resources and duration have been decided, the model proceeds to produce a standard DSN software implementation work breakdown structure and schedule without further input from the user.

A. Estimation of Program Size

The size of the software task is measured in "equivalent" Kilo-Source-Lines of Executable Code (KSLEC). A source line of executable code is defined basically as a source language

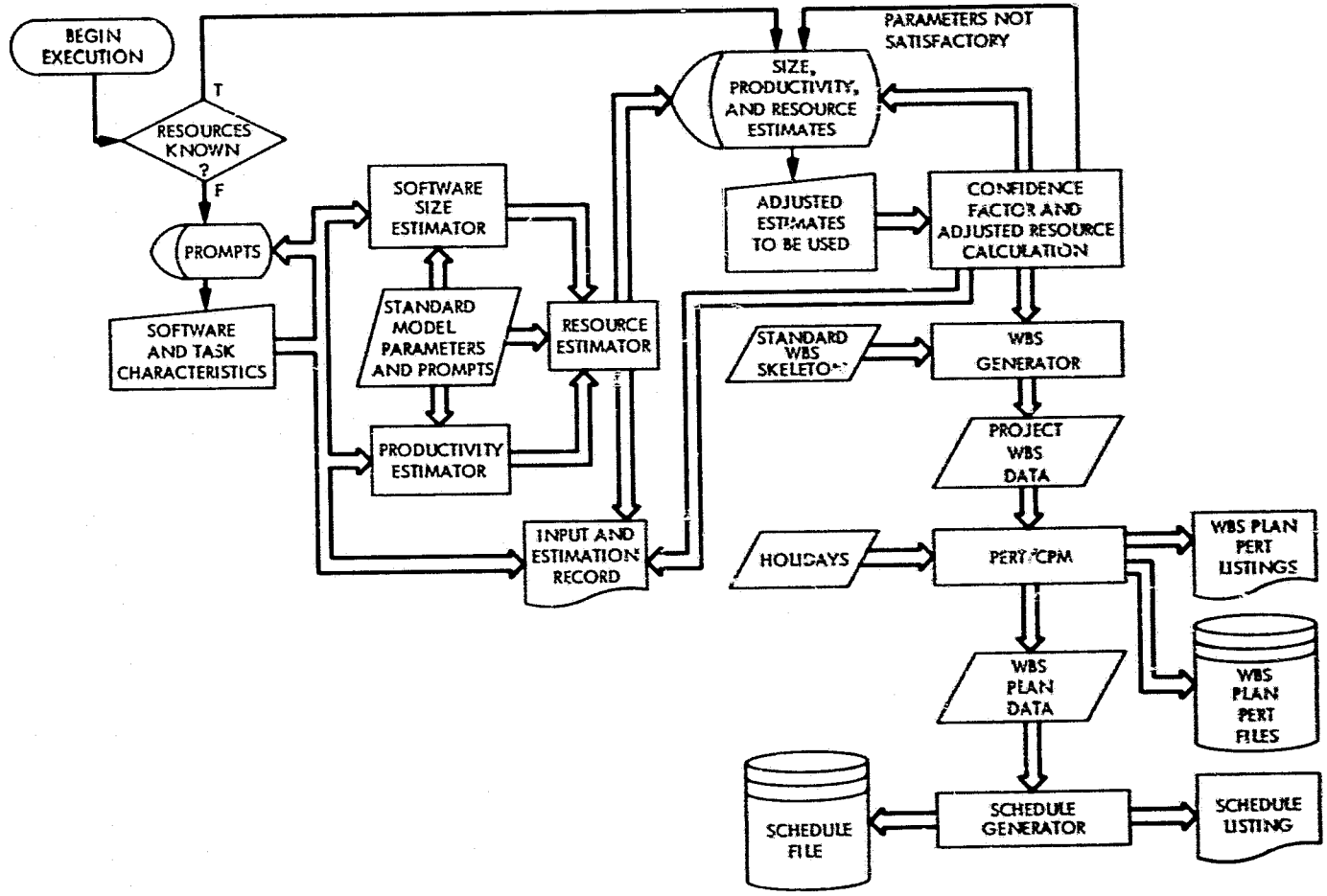

Fig. 1. The software cost and resource estimation model data flow

statement occupying one physical line in the source (display) medium that results in generation of object code, reservation of storage, or definition of data type. Comments are excluded, as are statements merely defining labels and equivalences of identifiers. If several basic statements may appear on one physical source line, each such statement should be added separately into the KSLEC count.

Source lines of new code are weighted differently than lines of reused code, in proportion to the relative amount of effort required to adapt the inherited code to the current task. Even deleted lines of code contribute to the programming effort, and therefore increase the "equivalent" KSLEC count.

The programming tasks involved with the generation of new code and reuse of existing code are depicted in Fig. 2. The effort to specify, produce, document, and test a new line of code is normalized to unity; the lines of code added, changed, deleted, and retested only contribute to the equivalent line count according to relative effort. The extent of existing-module modification is measured by the number of lines added in, the number changed, and the number deleted. The number of equivalent lines of code produced is then defined to be

$$L_{eq} = L_{new} + bL_{mod} + cL_{add} + dL_{chg} + eL_{del} + gL_{removed} + hL_{retest} \quad (1)$$

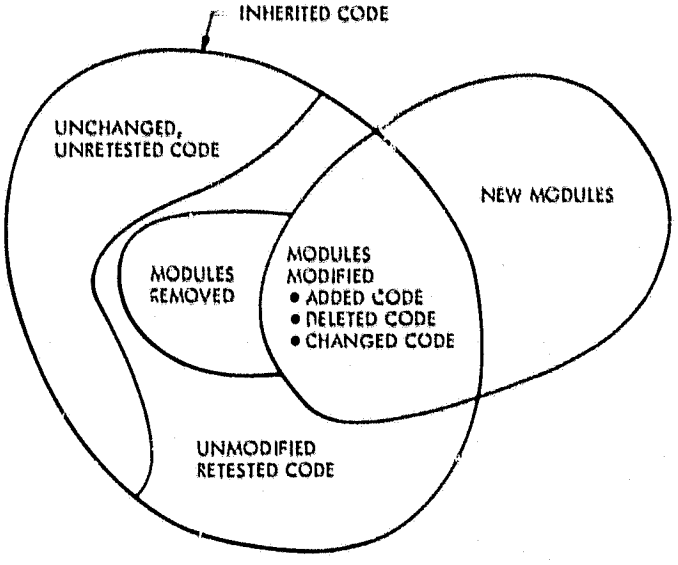

Fig. 2. Software implementation tasks related to new and inherited code activities

in which parameters b, c, d, e, g, and h are chosen to account for the expected effort required for each corresponding component.

The assumptions with respect to each component are the following:

(1) New code is subjected to the entire standard implementation process.

(2) The recognition of the reuse of existing code is made in the architectural phase, so code added, changed or deleted from modules goes only through subsequent phases.

(3) Added code takes the same effort as new code in corresponding phases where activity takes place.

(4) Changed code requires the same design and testing effort as new code, but less documentation and coding effort.

(5) Deleted lines from existing modules require reduced design, coding, and documentation effort, and no testing of the deleted lines.

(6) Any module changed is completely retested and requalified.

(7) Deleted modules require less architectural, interface, and detailed design considerations than new code; only that coding effort required to remove the unwanted code contributes to the coding time; no testing of the deleted code is possible; and documentation effort involves removal of entire sections of existing material and minor cleanup. Retesting is covered in modules which interfaced with the deleted module.

(8) Retested unmodified code requires revalidation of the interface design and retesting efforts only.

The analysis in Section IIID produces the following estimated typical values for the DSN environment:

$$L_{eq} = L_{new} + 0.27\,L_{mod} + 0.53\,L_{add} + 0.24\,L_{chg} - 0.15\,L_{del} + 0.11\,L_{remove} + 0.27\,L_{retest} \quad (2)$$

Each of the code-size parameters in Eq. (2) is estimated by the "PERT" technique (Ref. 12). This technique presumes that guesses for L are governed by a beta distribution (Ref. 13),

$$p(x) = B(s+1, t+1)\,x^s(1-x)^t \text{ for } 0 \leq x \leq 1 \quad (3)$$

where $B(\ ,\ )$ is the beta function (Ref. 14),

$$x = \frac{I - I_{min}}{I_{max}}$$

and $r = s + t$. This distribution has mean value

$$I_{avg} = \frac{I_{min} + I_{max} + r I_0}{r + 2} \tag{4}$$

in which I_0 is the value of I yielding the maximum value of $p(x)$.

The variance of I about I_{avg} is

$$var(I) = \frac{[(r+1)(I_{max} - I_{min})^2 + r^2 (I_0 - I_{min})(I_{max} - I_0)]}{(r+2)^2 (r+3)} \tag{5}$$

For a given value of r, the variance of I is maximum when

$$s = t = \frac{r}{2}$$

giving

$$Max\ [var(I)] = \frac{(I_{max} - I_{min})^2}{4 (r+3)} \tag{6}$$

The value $r = 4$ is quite often used (Ref. 6) and is adopted here. The estimators adopted in the model are calculated from guesses for I_{max}, I_{min}, and I_0:

$$I_{est} = \frac{I_{max} + 4 I_0 + I_{min}}{6} \tag{7}$$

$$I_{var} = \frac{5 (I_{max} - I_{min})^2 + 16 (I_{max} - I_0)(I_0 - I_{min})}{252} \tag{8}$$

The variance value with $s = t = 2$ gives a standard deviation of

$$I_{sigma} = \frac{I_{max} - I_{min}}{5.29} \tag{9}$$

The denominator value of Eq. (9) usually published in the literature (Ref. 6) is 6, which underestimates the variance by about 13%.

The estimated value for I_{est} is composed of the usual weighted sum of the I_{est} for each parameter, and its standard deviation is the square-weighted root-sum-square of the individual deviations. The weights in Eq. (1) are used.

B. Estimation of Productivity

In this model, the productivity P is defined as total equivalent KSLFC (here denoted I) produced, divided by the total staff effort (here denoted W),

$$P = \frac{I}{W}, \text{ KSLFC/staff-month} \tag{10}$$

A number of data bases (e.g., Refs. 3, 15, 16) have shown that I and W are correlated through a power-law relationship

$$W = \frac{I^a}{P_1} \tag{11}$$

where P_1 is the average 1-KSLFC productivity rate. The productivity at other values of I is given approximately by

$$P = P_1 I^{1-a} \tag{12}$$

The value of P_1 is set primarily by technology and environment. In fact, industry studies show that P_1 may vary by as much as 50:1 as a function of such factors. The value of a, however, in each environment where data is available, shows a relative constancy, at a value near unity.

It seems intuitive, all other things being equal, that P should not increase as a program's size rises; however, the least-square power-law fits to data bases yield a-values of 0.91 (IBM, Ref. 3), 0.991 (Doty, Ref. 1), 0.986 (University of Maryland, Ref. 4), and 0.975 (RADC, Ref. 16). One reason for the indicated increase in productivity for larger programs may be the increased usage of higher technology to develop these larger programs. Whatever the reason, the consistency of these figures seems to indicate that a linear power-law relationship, if anything, is slightly conservative.

The model presented here has compensations for the use of higher technology. The value for a assumed by this model, therefore, is unity. However, the model implementation keeps this value parametric, and a can be changed, if desired.

Several models have contributed to the formula by which P_1 is calculated, principally those of GRC (Ref. 17) and IBM (Ref. 3). The form of P_1 is

$$P_1 = P_0 A_1 A_2 \tag{13}$$

where P_0 is a constant factor, and A_1 and A_2 are difficulty, technology, and environmental adjustment factors. The value of A_1 is computed on the basis of parameters judged by GRC to be significant,

$$A_1 = [(1 + A_{lang} + A_{t\text{-}crit} + A_{c\text{-}crit}) A_{diff} A_{stable} A_{exp}]^W \quad (14)$$

The component adjustments are as follows:

(a) Language Adjustment, A_{lang}:

$$A_{lang} = f_{assy} \frac{L_{assy}}{L} \quad (15)$$

where f_{assy} is an assembly language factor, and L_{assy} is the amount of assembly language used.

The DSN value, $f_{assy} = 0.9$, is derived in Section IIIE. GRC uses a value of $f_{assy} = 3.5$.

(b) Time-Critical-Code Adjustment, $A_{t\text{-}crit}$:

$$A_{t\text{-}crit} = f_{t\text{-}crit} \frac{L_{t\text{-}crit}}{L} \quad (16)$$

where $f_{t\text{-}crit}$ is a factor which compensates for design/code/test time for code in which timing is critical, and $L_{t\text{-}crit}$ is the amount of such code. The DSN value, $f_{t\text{-}crit} = 0.7$, is derived in Section IIIF. The GRC value is 1.6.

(c) Capacity-Criticality Admustment, $A_{c\text{-}crit}$:

$$A_{c\text{-}crit} = f_{c\text{-}crit} \frac{L_{c\text{-}crit}}{L} \quad (17)$$

where $f_{c\text{-}crit}$ is a factor which compensates for capacity-constrained portions of the program, and $L_{c\text{-}crit}$ is the amount of code which contributes to the need for extra effort in solving the capacity problem. The DSN value, $f_{c\text{-}crit} = 1.0$, is derived in Section IIIF. The GRC value is 7.

(d) Difficulty Adjustment, A_{diff}:

$$A_{diff} = 1 + (f_{hard} - 1) \frac{L_{hard}}{L} + (f_{easy} - 1) \frac{L_{easy}}{L} \quad (18)$$

where f_{hard} and f_{easy} are factors which relate to the relative effort required for "hard" and "easy" parts of the program, and L_{hard} and L_{easy} are the amounts of code adjudged to be "hard" and "easy," respectively. The DSN values, $f_{hard} = 1.2$ and $f_{easy} = 0.8$, are taken from data published in TRW reports (Ref. 6).

(e) Requirements and Design Stability Adjustment, A_{stable}:

$$A_{stable} = L_{none} + f_{base} L_{base} + f_{many} L_{many} + f_{free} \frac{L_{free}}{L} \quad (19)$$

Here, L_{none} is the amount of code which is expected to derive from well-understood, stable requirements, not expected to change. Factors f_{base} and L_{base} relate to effort and amount of code resulting from requirements expected to change moderately, but which are kept under baseline control. Factors f_{many} and L_{many} derive from the effort and amount of code for which requirements are expected to produce many changes, but, again, will do so under baseline control. The parameters f_{free} and L_{free} compensate for the accommodation of requirements allowed to change freely, not baseline-controlled. The values used in the DSN model, $f_{base} = 1.35$, $f_{many} = 1.9$, and $f_{free} = 2.3$, are taken directly from the GRC model.

(f) Experience Adjustment, A_{exp}:

$$A_{exp} = 1 - f_{exp} T_{exp} \text{ if } T_{exp} \leq \frac{1 - f_{full\text{-}up}}{f_{exp}}$$

$$= f_{full\text{-}up} \text{ if } T_{exp} > \frac{1 - f_{full\text{-}up}}{f_{exp}} \quad (20)$$

Here T_{exp} is the average staff training in years, f_{exp} is the training rate, and $f_{full\ up}$ is the fully trained productivity factor. The values $f_{exp} = 0.06$ and $f_{full\text{-}up} = 0.4$ derive from the GRC model.

The second productivity adjustment factor, A_2, derives from the Walston-Felix data (Ref. 3) published by IBM:

$$A_2 = \prod_{i=1}^{N} \frac{P_{hi(i)}}{P_{lo(i)}}^{wx_i} = \exp\left[w \sum_{i=1}^{N} x_i \log \frac{P_{hi(i)}}{P_{lo(i)}}\right] \quad (21)$$

in which N factors that correlate with productivity and their effects on productivity were considered. $P_{hi(i)}$ values were measured when factor i was applied and contributed to productivity, and $P_{lo(i)}$ resulted when factor i was not applied and thus lowered productivity. The use of x_i relates to the

extent factor f is present in the application being estimated. The values used for x_f are either +1, 0, or −1. The parameter w was chosen to best fit the data collected.

The various factors contributing to the adjustment, the acceptable responses, and the log-ratio values used in the DSN model were taken, for the most part, directly from Walston-Felix data. A few others have been added, due to their variability in the DSN environment. The exponent value w was chosen to give a 50:1 variation between the extreme values.

The factors, responses, and log-ratio values used in the DSN model are listed in Appendix A. The value for w is computed by

$$w = \frac{f_{wf}}{\sum_{i=1}^{M} \log \frac{P_{hi(i)}}{P_{lo(i)}}} \tag{22}$$

in which $f_{wf} = 1.95$ produces the 50:1 spread in productivity adjustment. The value M extends the sum over both the IBM and GRC model factors.

IBM and other data show that the total effort adjustment may be expected to deviate, at a given L, from the estimated value by a standard deviation factor of about $A_{sigma} = 1.73$.

C. Estimation of Implementation Task Duration

The IBM, University of Maryland, and RADC statistics suggest that the average duration T required for L KSLEC and W staff-months effort is approximated by

$$T = T_1 W^{f_t} \tag{23}$$

where T_1 is the 1-KSLEC average duration, and f_t is a time-factor exponent found from industry statistics. The value used in the DSN model, $T_1 = 4.8$, was adjusted to fit limited available DSN data, and $f_t = 0.356$ was the average power-law for the more extensive IBM, RADC, and GRC data.

D. Average Staff

The average staff, in persons, results from manipulation of the duration equation,

$$S = \frac{W}{T} = \frac{1}{T_1} W^{1-f_t} \tag{24}$$

The staffing exponent, $1 - f_t = 0.644$, implied in the DSN model compares with measured values averaging 0.629 across industry.

E. Documentation Sizing and Cost

IBM statistics showed a nearly linear relationship between pages of documentation and lines of code, whereas the University of Maryland measured almost a square-root relationship. DSN experience over six Mark-III Data System programs revealed an exponent about midway in between (0.83). A study of Software Specification Document (SSD) user needs (Ref. 18) recommended that this document be about 40-50 pages per KSLEC for programs in the 30 KSLEC vicinity. The formula used for the model for the number of pages of documentation is

$$D_{pp} = D_1 L^{c_{doc}} \tag{25}$$

The model uses $D_1 = 90$ and $c_{doc} = 0.83$ to match the DSN experience and SSD guidelines.

The documentation cost is found by a straight dollar-per-page rate; a figure of $30/page is used in the current model.

F. Computer Resources

IBM and TRW give statistical figures for computer time costs as functions of lines of code and total effort, and also as a fraction of total cost. The DSN, however, mostly has dedicated minicomputers for which operational costs are not assessed to the implementation task on a usage basis. TRW does, however, also estimate a linear relationship between CPU time required per machine code instruction of about $C_1 = 25.2$ CPU hours per thousand instructions.

The DSN model computes CPU resources as

$$T_{cpu} = C_1 \left[L_{assy} + (L - L_{assy}) f_{HOL} \right]^{c_{cpu}} \tag{26}$$

The higher-order-language expansion factor, $f_{HOL} = 2.4$, used in the DSN model is the GRC figure, based on JOVIAL. The exponent value $c_{cpu} = 0.96$, given by Walston and Felix (who give dollar costs, rather than CPU time), is adopted to account for the general trend of CPU time with program size.

If CPU dollar cost is relevant, the model computes this at a straight dollar-per-CPU-hour figure (zero in the DSN model, but a parameter is available for other applications).

G. Variance Computation

We assume in this model that measurements of values of a quantity y for a given x satisfy a power-law formula of the form

$$y = kx^v \tag{27}$$

in which p is a known, fixed value, but k exhibits some statistical distribution about a best-power-law fit,

$$y_0 = k_0 x^p \tag{28}$$

The values of k_0 and p are chosen to minimize the logarithmic error over the observed data.

$$err\,[\ln(y)\,|\,x)] = \sum_{i=1}^{N} (\ln y_i - \ln y_0^2) \tag{29}$$

The variation in y is expressed as a $(X/)$ -factor to be applied to y_0 at a given x.

If, in a particular instance, there is uncertainty about the value of x, then the y variation is described by

$$var\,(\ln y) = E\left(\ln \frac{k}{k_0} + p \ln \frac{x}{x_0}\right)^2 \tag{30}$$

where $\ln(x_0)$ is the mean of $\ln(x)$.

If k_2 and x_1 are defined as those values such that

$$\begin{aligned} \ln^2 \frac{k_1}{k_0} &= E\left(\ln^2 \frac{k}{k_0}\right) \\ \ln^2 \frac{x_1}{x_0} &= E\left(\ln^2 \frac{x}{x_0}\right) \end{aligned} \tag{31}$$

then we can write

$$y = k_0 x^p\,(X/k_2) \tag{32}$$

where k_2 is a standard deviation factor

$$k_2 = exp\left[\left(\ln^2 \frac{k_1}{k_0} + p^2 \ln^2 \frac{x_1}{x_0}\right)^{1/2}\right] \tag{33}$$

Factors such as this are applied in the model to each power law, using k_1/k_0 values measured from industry statistics, and x_1/x_0 values assumed to be one-sigma estimations of the x-parameter. The x-parameter in the model ultimately derives from the program length, L, whose variance was addressed in Section IIA.

H. Modified Rayleigh-Norden-Putnam Model

Guided by the SWAG-estimated values, the user of the model enters the effort, duration, and staff parameters intended for use. As a check on reasonableness, comparisons of these parameters with the Rayleigh-Norden-Putnam model are made. However, some adjustments in this model have been made in order to make it fit the average power-law variations described above.

The basic differential equation describing the average work effort, w, is taken to be of the form

$$w' = At^r (K - w) \tag{34}$$

in which $w = w(t)$ is the cumulative effort expended up to time t (months), K is the total life-cycle staff-months effort, r is an exponent that accounts for learning curve and sets the "pace" of the work, and A is a constant scaling time.

Norden and Putnam assume $r = 1$ (a linear learning curve); however, this value of r is not consistent with the observed power-law relationships between L, T, and W.

The solution to the modified equation is

$$w = K\left\{1 - \exp\left[-\left(\frac{r}{r+1}\right)\left(\frac{t}{t_0}\right)^{r+1}\right]\right\} \tag{35}$$

where the value A has been replaced by r/t_0^{r+1}, in which t_0 is the value at which w' is maximum (time of maximum staffing).

We use T to denote the time of acceptance test completion and transfer to operations (in months), whence $w(T) = W$. Then

$$T/t_0 = \left[\frac{r+1}{r} \ln\left(\frac{1}{1 - \frac{W}{K}}\right)\right]^{1/(r+1)} \tag{36}$$

As in the Putnam work, a "difficulty index," D may be defined as

$$D = \frac{Kr}{t_0^{r+1}} \tag{37}$$

and a "difficulty gradient factor" can be similarly defined as

$$\nabla D = (r+1)\,\frac{D}{2t_0} \tag{38}$$

These reduce to the Putnam expressions when $r = 1$. Note: Putnam expresses these values using time units of years, so comparisons require a scale change,

$$D_{Put} = 12^r D$$
$$\nabla D_{Put} = 12^{r+1} \nabla D \tag{39}$$

The number of lines of code is taken to be of the same general form as Putnam's "software equation,"

$$L = c_k K^p t_0^q \tag{40}$$

(Putnam takes $p = 1/3$, $q = 4/3$, a 1:4 relationship.) The values for p and q used in the DSN software cost model are chosen to satisfy both power-law productivity constraints and time-effort tradeoffs.

The ratio W/K is the implementation-effort/total-life-cycle-effort ratio, which Putnam estimated to be constant, at about 0.95, and NASA measurements (Ref. 19) found to be about 0.88. Such ratios set the proportionality between T and t_0 to produce the equations

$$L = c_p W^p T^q = c_p W^{p+q} S^{-q} = c_p T^{p+q} S^p \tag{41}$$

for an appropriately defined technology constant c_p related to the productivity coefficient P_1 evaluated by the model.

We may use the power-law formula for L and the definition of D to eliminate T. When difficulty is held constant at a value D_0 corresponding to the power-law, or "average difficulty" of projects, we find that the following relationship must hold:

$$\frac{1}{a} = p + \frac{q}{r+1} \tag{42}$$

Use of the power-law expression for average T similarly produces the relationships

$$f_t = \frac{1}{r+1}$$
$$c_p = (P_0 A_1 A_2)^{-1/a} T_1^{-q}$$
$$D_0 = r K/W \left(\frac{T}{t_0}\right)^{r+1} T_1^{-(r+1)} \tag{43}$$
$$\nabla D_0 = (r+1) \frac{T}{t_0} \frac{D_0}{2T}$$

For a given L, the ratio q/p sets the time vs effort tradeoff relationship. Use of the factor $f = q/p = -\log (W_1/W)/\log (T_1/T)$ permits p and q to be expressed as

$$p = \frac{1}{a(1 + f_t f)}$$
$$q = \frac{f}{a(1 + f_t f)} \tag{44}$$

The model contains an input parameter $W_1/W = W_{1/2}$, defined to be the effort ratio for which $T_1/T = 1/2$. This parameter is used to specify the value of f as $f = \log_2 (W_{1/2})$.

The equation for L is thereby

$$L = (P_0 A_1 A_2)^{-1/a} \left(\frac{D_0}{D}\right)^{q/(r+1)} W^{1/a} \tag{45}$$

and the expression for productivity P as

$$P = (P_0 A_1 A_2)^{-1/a} \left(\frac{D_0}{D}\right)^{q/(r+1)} W^{(1/a)-1} \tag{46}$$

If $f = q/p$ were set to the Putnam value (viz., 4) then a value $W_{1/2} = 16$ would result. While the best value of f for DSN tasks has not been determined, the consensus of managers and programmers is that a 16:1 effort factor to reduce duration by a factor of 2 is too extreme, and not within their experience. Rather, a $W_{1/2} = 1.5$ value is probably closer to actuality in the DSN environment. This assumption produces the values

$$f = 0.585$$
$$p = 0.828$$
$$r = 0.484 \tag{47}$$
$$c_p = 0.0621/(A_1 A_2)$$

The value $W/K = 0.88$ mentioned earlier produces

$$\frac{T}{T_1} = 1.53$$
$$D_0 = 0.788, \text{ or } D_{Put} = 7.06 \tag{48}$$
$$\nabla D_0 = 0.169/T, \text{ or } \nabla D_{Put} = 182/T$$

Difficulty and gradient calculations are not currently used in the model, however.

When L and W are given, T and S are then given by

$$T = \left(\frac{L}{c_p W^p}\right)^{1/q}$$
$$S = \frac{W}{T} \tag{49}$$

When L and T are given, W and S are

$$W = \left(\frac{L}{c_p T^q}\right)^{1/p}$$
$$S = \frac{W}{T} \tag{50}$$

and when L and S are given, W and T are

$$W = \left(\frac{L S^q}{c_p}\right)^{1/(p+q)}$$
$$T = \frac{W}{S} \tag{51}$$

That is, only one of the average W, T, S parameters need be specified for a given L. The others can then be computed (and are in the model).

If any pair of the W, T, S parameters is proposed, the lines-of-code value determined from the "software equation" represents the average number of lines of code that can be supported by the proposed resources. The difference between the size-model estimate and the software equation estimate is a margin on which confidence levels may be calculated.

One should note, however, that the software equation does not remain valid for tasks so small that average staff drops below unity. In this case, we revert back to the power-law equation to yield the needed effort and solve for T and S from the relation $W = TS$.

Confidence in completing the software task within resources and budget when staff is variable is

$$C(W,T) = P\{t \leq T, w \leq W\} \tag{52}$$

where $P\{\cdot\}$ is the probability function. That is, the confidence level is the probability that both effort W *and* duration T are not exceeded.

Under the presumptions that the conditional densities $p(T \mid W, L)$ and $p(W \mid L)$ are log-normal (which are fairly well borne out by RADC data, see Ref. 16), and that $p(L)$ can be estimated by a log-normal density as well, then the confidence factor is readily calculated. The log of the software equation provides the conditional mean for log (T) in the first; the log of the power-law formula provides the conditional mean for log W in the second; and the code size estimator provides the mean of log (L) in the third. The result is the integral

$$C(W,T) = \frac{1}{2(\pi)^{1/2}} \int_{-\infty}^{X} [1 + \operatorname{erf}(u + vx)] \exp(-x^2)\, dx \tag{53}$$

in which

$$X = \frac{\ln\left(\frac{T}{t}\right)\left(\frac{Y}{2Z}\right)^{1/2}}{s_T}$$
$$u = \frac{\left(\frac{Z}{2QY}\right)^{1/2} \ln\frac{W}{w}}{s_W}$$
$$v = \frac{p s_W / q s_T}{(Q)^{1/2}} \tag{54}$$
$$Y = 1 + \frac{s_L^2}{p^2 s_W^2}$$
$$Q = Y + \frac{s_L^2}{q^2 s_T^2}$$
$$Z = Q + \frac{p^2 s_W^2}{q^2 s_T^2}$$

The values t and w are duration and effort values on the software equation curve producing the expected number of lines of code; s_L, s_W, s_T, represent the standard deviation in log (L), log (W), and log (T), respectively.

The $C(W,T)$ appears to be a function of arbitrary parameters t,w on the curve $L_{est} = c_p w^p t^q$. The triple, $(L_{est}\ w,t)$, represents the point from which margins are measured to (L,W,T), the resource values to be used in the project. The model chooses w,t to maximize $C(W,T)$. The optimum values of w and t satisfy the equation

$$[1 + \mathrm{erf}(Fx + Gy)] \exp\left(-\frac{(x^2+y^2)}{2}\right)$$

$$= H[1 + \mathrm{erf}(Jx + Ky)] \exp\left(-\frac{A^2x^2}{2}\right) \qquad (55)$$

where

$$x = \frac{\ln\left(\frac{T}{t}\right)}{s_T}$$

$$y = \frac{\ln\left(\frac{W}{w}\right)}{s_W}$$

$$A = \left(\frac{1}{Z}\right)^{1/2}$$

$$F = \left(\frac{Y}{2Q}\right)^{1/2}$$

$$G = \frac{ps_w}{qs_T(2QY)^{1/2}} \qquad (56)$$

$$H = \frac{Yps_w}{qs_T Z^{1/2}}$$

$$J = \frac{ps_W}{qs_T}\left(\frac{Y}{2ZQ}\right)^{1/2}$$

$$K = \left(\frac{Z}{2QY}\right)^{1/2}$$

Equation (55) is solved by Newton's method, subject to the constraint $\log(L/L_{est}) = ps_W y + qs_T x$ and the confidence integral is evaluated by numeric integration.

I. Standard Work Breakdown Structure Model

The DSN Data Systems Section has adopted a standard WBS format for its software implementation tasks. The format of subtasks is shown in Table 1. The table also shows the average reported effort in 31 recent upgrade tasks for each line item, adjusted for future tasks based on an analysis of the problems encountered. Requirements and subsystem design resources were not reported with the subsystem implementation activities; however, estimates for these have been factored into the WBS model.

The model permits selection of the starting point for WBS generation: at the System/Subsystem Functional Requirements (FRD) commencement; at the beginning of the Software Requirements Definition (SRD); or at the beginning of the Software Design Definition (SDD), at which time the software architectural considerations are developed.

Once a starting point has been selected by the user, the model apportions staff effort over the remaining subtasks (using percentage parameters in the standard WBS data base), and calculates subtask durations (also via percentage figures in the data base) to conform with the selected overall effort duration. These percentages are currently chosen to produce a relatively constant effort profile over the implementation project. This profile is a trend in the small DSN efforts currently under way. The percentages are alterable, however, to fit any assumed profile for other tasks.

The standard WBS data base also contains precedence information among subtasks, which is used by the PERT/CPM scheduling algorithm. Such precedences were set by logical and management criteria to form a rational progression of activities and milestones commensurate with DSN standard software implementation practices. Precedences for each task are specified in the form of the decimalized number of the task(s) which must complete before that particular task may begin.

All items in the standard WBS data base can be altered to fit another project by a mere text editing process.

J. Operational Modes

The DSN Software Cost Model software has a size/productivity estimation mode, which is optional; if estimates of staff effort, average staff size, and task duration are known a priori, the estimation mode may be skipped.

Regardless of the means by which amount of code staff effort, average staffing, and project duration are estimated, these figures are used to calculate a confidence level. The user may adjust estimates, as desired, until a suitable set of resource parameters is obtained before proceeding to the schedule generation process.

Table 1. Standard work breakdown structure

Task	Effort, %	Duration, %
0 Start	0.0	0.0
1. System plans, requirements, and design	0.0	0.0
1.1 Define subsystem requirements	2.5	3.2
1.2 FRD	0.0	0.0
1.2.1 Write all sections	0.7	0.9
1.2.2 Edit and release FRD	0.3	0.4
1.3 Level B review	0.5	0.6
1.4 Define system architecture	3.0	3.9
1.5 FRD	0.0	0.0
1.5.1 Write all sections	0.7	0.7
1.5.2 Edit and release	0.3	0.4
1.6 Level C review	0.5	0.6
2. Software planning and requirements	0.0	0.0
2.1 Define software requirements	4.0	5.6
2.2 SRD	0.0	0.0
2.2.1 Write all sections	0.7	0.7
2.2.2 Edit and release	0.3	0.4
2.3 Level D review	0.4	0.6
3. Software architecture and design definition	0.0	0.0
3.1 Define software architecture	5.0	7.2
3.2 SSD	0.0	0.0
3.2.1 Write all sections	0.6	0.6
3.2.2 Edit and release	0.3	0.4
3.3 System interface design	3.6	3.6
3.4 Level E review	0.4	0.6
4. Software detailed design and production	0.0	0.0
4.1 SSD	0.0	0.0
4.1.1 Write Sections 1, 2, 3	1.0	1.0
4.1.2 Write Section 4	3.0	4.4
4.1.3 Write Section 5	7.0	7.0
4.1.4 Write Section 6	0.7	0.7
4.1.5 Write Section 7	1.5	1.5
4.1.6 Edit and release	0.9	0.9
4.2 SOM	0.0	0.0
4.2.1 Write preliminary draft	1.3	1.9
4.2.2 Complete all sections	1.5	1.5
4.2.3 Edit and release	0.6	0.6
4.3 High-level design review	0.4	0.7
4.4 Module production and integration	0.0	0.0
4.4.1 Executive and control	2.9	5.6
4.4.2 I/O modules	2.9	5.6
4.4.3 Interface handlers	2.9	5.6
4.4.4 Function A	2.9	5.4
4.4.5 Function B	2.9	5.4
4.4.6 Function C	2.8	5.2
4.4.7 Function D	2.8	5.2
4.4.8 Function E	2.8	5.2
4.4.9 Function F	2.8	5.2
4.5 Special tasks	0.0	0.0
4.5.1 Support software	2.0	2.0
4.5.2 Other	1.0	1.0
4.6 Acceptance readiness review	0.4	0.7
5. Software test and transfer	0.0	0.0
5.1 Verification tests	4.5	8.1
5.2 Contingency	4.0	4.0
5.3 SIT	0.0	0.0
5.3.1 Write all sections	2.3	2.3
5.3.2 Edit and release	0.4	0.4
5.4 Acceptance tests	3.3	4.1
5.5 Demonstration tests	3.5	4.8
5.6 Transfer, CDF to COF	1.2	1.6
6. Management tasks and milestones	0.0	0.0
6.1 CDF activities	6.0	6.0
6.2 Develop preliminary budget	0.5	0.6
6.3 Develop system implementation plan	0.5	0.6
6.4 Draft software implementation plan	1.0	1.4
6.5 Revise implementation plan	2.0	2.9

Output is offered on hard copy or computer files, at user selection. Schedules are output beginning with FRD, SRD, or SDD activity.

K. Risk-Bias Computation

The size and resource estimates produced by the model are mean-estimate figures; however, variance figures are also given for risk-bias computations, as appropriate. If a schedule is generated using the mean estimates given, then the confidence factor for performing the implementation on time and within budget is only about 25%. That is, only about one-quarter of the projects using the mean estimates should expect to deliver within schedule and budget! If a project desires a confidence factor higher than 25%, then resource estimates must be increased by some amount to produce the higher confidence level.

III. Selection and Calibration of Parameters

This section contains a data base of statistics and computations used in the formulation of the DSN Software Cost Model.

A. Productivity Statistics

The following formulas represent best-fit parametric models of observed data on KSLEC vs effort. In most cases, the unit of measure was not actually KSLEC, but something probably linearly related, such as delivered source lines (including comments) or object instructions.

$W = 5.2\ L^{0.91}$ (IBM)

$W = 1.27\ L^{0.986}$ (University of Maryland)

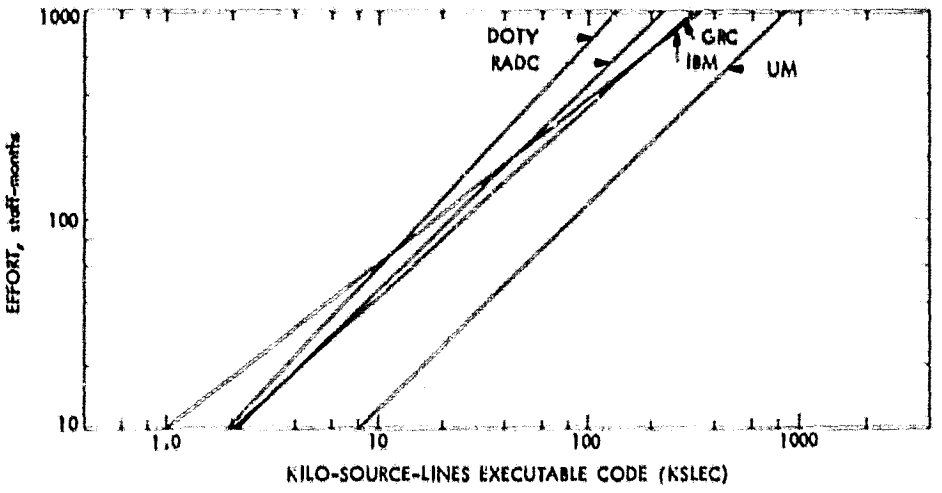

Fig. 3. Industry estimates of effort vs source lines of code delivered

$W = 10.1\ L^{0.79}$ (GRC, high-order languages)

$W = 4.86\ L^{0.976}$ (RADC)

$W = 5.25\ L^{1.057}$ (Doty)

These are shown in Fig. 3.

The exponent for L in the DSN model was chosen as unity for the following reasons: (1) the proximity of all the measured exponent values to unity, (2) the wide scatter of points in each case, and (3) the intuitive feeling that productivity cannot rise, all other things being the same, when the program size increases.

B. Staff Size Statistics

Staffing level industry statistics yielded least variation among best-fit models. Some of the results are

$S = 0.54\ W^{0.6}$ (IBM)

$= 0.409\ W^{0.65}$ (*)

$S = 0.198\ W^{0.786}$ (University of Maryland)

$= 0.196\ W^{0.79}$ (*)

$S = 0.989\ L^{0.609}$ (RADC)

$= 0.369\ W^{0.624}$ (*)

$= 0.388\ W^{0.641}$ (*)

The figures asterisked (*) are computed from other measured statistics by the same company listed. For example, if project duration was given as a function of effort W, then the staffing could be computed by $S = W/T$ for comparison to the best-fit data. These staffing best-fit data are plotted in Fig. 4.

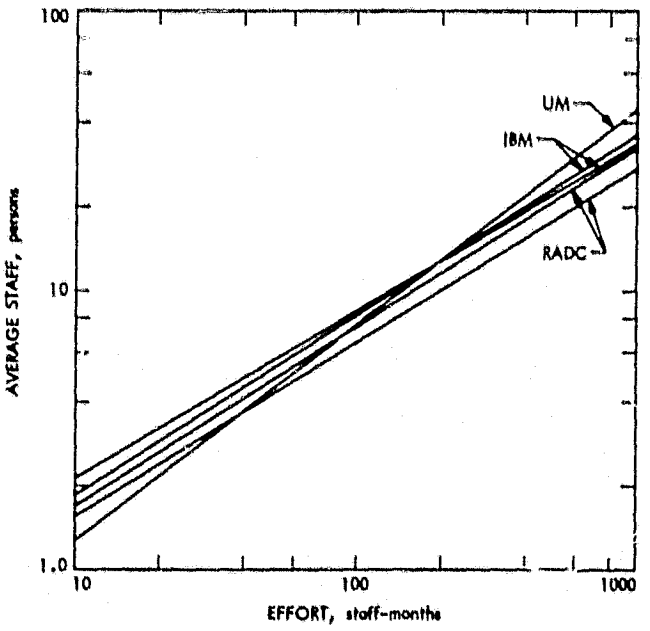

Fig. 4. Average staffing required vs project effort

The DSN model computes S from effort W and duration T directly,

$$S = \frac{W}{T}$$

C. Project Duration Statistics

Duration statistics seem to produce slightly different effort exponents across the industry estimates, but seem to converge in the 100 staff-months effort region. The fits are

$T = 2.47\ W^{0.35}$ (IBM)

$= 2.13\ W^{0.4}$ (*)

$T = 5.10\ W^{0.2}$ (University of Maryland)

$= 5.19\ W^{0.206}$ (*)

$T = 4.55\ L^{0.349}$ (RADC)

$= 2.59\ W^{0.358}$ (*)

As in the previous section, the figures marked by asterisk (*) are not direct measurements, but computed from direct-measurement power-law fits. These duration data are presented in Fig. 5.

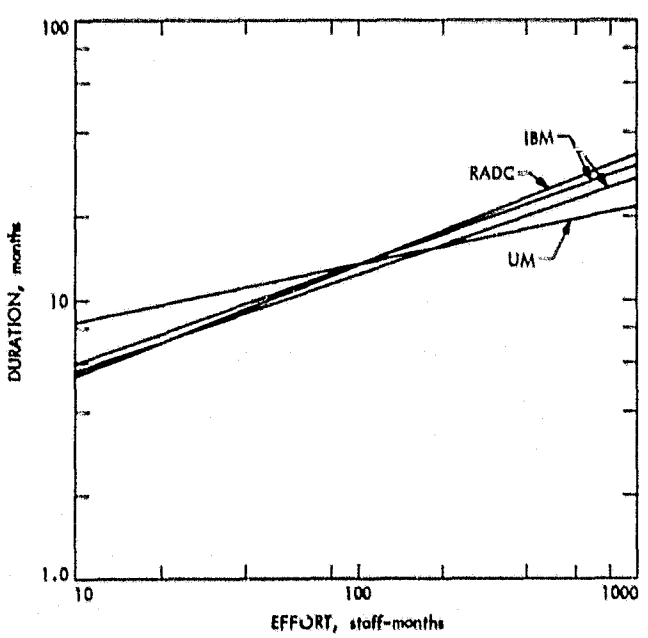

Fig. 5. Average duration of software projects vs the effort required

The DSN effort-exponent was computed from a least-squares fit to an equal number of points chosen from each fit above, as well as the implied durations taken from staffing statistics in Section IIIB. The multiplier coefficient was taken to fit observed DSN data.

D. Effort Factors of Inherited Code Utilization

The assumed effort factors for the utilization of existing code are shown in Table 2. The standard WBS entries are shown, along with the estimated effort required for each in a one-staff-year project (242 staff days). The entries for "New Module" are the numbers allocated to each activity, adding up to 242; that is, if the project were to develop only entirely new code, all 242 staff-days would be used.

For each other column, the entries are the estimated resources spent in each activity, as if the entire project were totally involved with only that activity heading the column, on the same amount of code as addressed by the effort in column 1. The required effort to be applied is scaled from the relative amounts of all code to be utilized. The "equivalent" number of new lines of code for any of the columns is found by multiplying the work factor times the number of lines hypothetically being processed.

E. High-Level Language Adjustment

The statistics for usage of high-level languages vs assembly language are widely diverse in their effects on project effort. GRC reports (Ref. 15) best-fit curves to measured statistics of

$W = 9.38\ L^{0.79}$ (assembly language sources)

$W = 10.12\ L^{0.79}$ (high-order language source*)

The (*) statistic is derived from the measured ratio of object instructions to high-level source statements. These two equations suggest nearly a 1:1 ratio of effort on a source line basis. However, GRC also proposes a 4.5:1 cost factor in their Military Sales Software Cost Model (Ref. 17). Therefore, an analysis similar to that of Section 3.4 was undertaken to estimate the high-level vs assembly-language tradeoff. The assumptions were:

(a) Design, coding and unit testing activities would increase proportionately to the code-expansion ratio of the language. A value of 2.4:1 (GRC) was used.

(b) Documentation of the code would also increase in the same ratio, to keep the same relative degree of detail (pages/KSLEC).

Table 2. Inherited code effort requirement estimates

Activity	Changed module						
	New module	Added code	Deleted code	Changed code	Same code	Deleted module	Retested code
Requirements and design							
FRD phase	6						
FDD phase	7						
SRD phase	10						
SDD phase	13	13	4	13	4	4	
SIS activity	10	10	3	10	3	3	
SSD phase	22	22	5	6	5	5	
Special	6	6		3			
Other	3	3		1			
Coding							
SSD phase	23	23	3	7		3	
Special	6	6		3			
Other	4	4		2			
Testing							
Integration	12	12		12	12		
Verification	12	12		12	12		
Contingency	2	2		2	2		2
Acceptance test	10	10		10	10		10
Demonstration	10	10		10	10		10
Documentation							
FRD	3						
FDD	3						
SRD	3						
SDD	3	3	1	2		1	
SSD	31	31	4	10		4	
SOM	3	3	1	2		1	
STT	6	6	1	3		1	2
Management							
FRD review	2						
FDL review	2						
SRD review	5						
Management plan	7						
SDD review	3	3	1	1	1	1	
High level design review	1	1	1	1	1	1	
Acceptance review	1	1	1	1	1		1
Transfer	1	1	1	1	1		1
CDF SSD phase	9	9	2	9	2	2	
CDL test phase	3	3	1	2	1		3
Total	242	194	29	123	65	26	41
Factor	1.0	0.80	0.12	0.51	0.27	0.11	0.17

(c) Acceptance tests, being functional tests, would be the same for the two. The reduced number of functions to be tested per KSLEC of assembly code as compared to the higher-order language was felt to be negated by the larger number of errors likely to be found.

(d) Requirements and architectural design tend to be top-level activities depending on numbers of functions. The number of functions which can be provided per KSLEC is less in assembly language by the language-level factor. Hence these activities should take the same effort on a per-function basis.

Table 3 shows the estimated resource budget based on a one-staff-year effort, high-order language effort. The factors above have been applied to the individual subtasks to arrive at

Table 3. Assembly language effort requirement estimates

Activity		FRD start	SRD start	SDD start
Requirements and design				
FRD phase		6		
FDD phase		7		
SRD phase		10	10	
SDD phase		13	13	13
SIS activity		10	10	10
SSD phase	× 2.4	53	53	53
Special	× 2.4	14	14	14
Other	× 2.4	7	7	7
Coding				
SSD phase	× 2.4	55	55	55
Special	× 2.4	14	14	14
Other	× 2.4	10	10	10
Testing				
Integration	× 2.4	29	29	29
Verification	× 2.4	29	29	29
Contingency	× 2.4	5	5	5
Acceptance test		10	10	10
Demonstration		10	10	10
Documentation				
FRD		3		
FDD		3		
SRD		3	3	3
SDD		3	3	3
SSD	× 2.4	74	74	74
SOM		3	3	3
STT		12	12	12
Management				
FRD review		2		
FDD review		2		
SRD review		5	5	
Management plan		7	7	7
SDD review		3	3	3
High level design review		1	1	1
Acceptance review		1	1	1
Transfer		1	1	1
CDE SSD phase	× 1.3	12	12	12
CDE test phase	× 1.3	4	4	4
Total		421	398	380
Factor		1.74	1.82	1.89

the estimated assembly language overhead. The effects of the three permitted starting points are shown.

F. Capacity and Timing Constraint Adjustment

Capacity-constrained effects on required software effort are also diverse in industry, and did not seem to fit with DSN experience. IBM published (Ref. 3) a 3.8:1 ratio for a severely constrained task. GRC reported (Ref. 15) a 5.15:1 ratio and recommended an 8:1 ratio of effort be used in its Military Sales Model (Ref. 17).

Capacity constraints can often be relieved by program optimization or segmentation, when mass storage resources are available, but timing considerations then also sometimes become a factor, and TRW estimates a 2.5:1 factor in effort for such timing constraints. GRC concurs in this figure. TRW also estimates a 3:1 increase in effort for real-time programs over sequential programs.

Inasmuch as capacity and timing constraints often require coding in assembly language, it was felt that the three were correlated in the same data and that the effort factor of all three taken simultaneously was overcompensation. The DSN productivity on the Mark III Data System (10 source lines/day in assembly language, with real-time and severe capacity constraints) tended to favor this hypothesis.

Tables 4 and 5 are estimated resource budgets for capacity- and timing-contrained developments. Since these factors directly affect code design, coding, and testing, the emphasis of the extra effort is applied in these subtasks.

G. DSN Documentation Statistics

The documentation page count of six Mark III Data System tasks representing over 100 KSLEC produced data presented in Table 6. These data gave best-fit power-law statistics of

$$D_{SRD} = 6.29\, L^{0.499} \quad (X/1.50)$$

$$D_{SDD} = 21.3\, L^{0.314} \quad (X/1.63)$$

$$D_{SOM} = 15.3\, L^{0.727} \quad (X/1.21)$$

$$D_{SSD} = 136\, L^{0.832} \quad (X/1.21)$$

$$D_{SST} = 7.71\, L^{1.09} \quad (X/1.61)$$

$$D_{tot} = 175\, L^{0.829} \quad (X/1.21)$$

It was generally felt, however, that the SRD and SDD documentation represented by these figures was not detailed enough, whereas the SSD was too detailed in some areas, and yet not detailed enough in others. A study (Ref. 18) recommended about 46 SSD pages/KSLEC (without listings), down about 43% from the average of 80 pages/KSLEC actually produced for the SSDs, or down about 33% from the total.

The power law curves above were therefore adjusted up 10% for the SRD and SDD, and down by about 43% for the SSD to arrive at the recommended figure

$$D_{tot} = 120\, L^{0.823} \quad (X/1.2)$$

Table 4. Capacity constraint effort requirement estimates

Activity		FRD start	SRD start	SDD start
Requirements and design				
FRD phase		6		
FDD phase		7		
SRD phase		10	10	
SDD phase	× 2	26	26	26
SIS activity	× 2	20	20	20
SSD phase	× 5	110	110	110
Special	× 4	24	24	24
Other	× 4	12	12	12
Coding				
SSD phase	× 2	46	46	46
Special	× 2	12	12	12
Other	× 2	8	8	8
Testing				
Integration	× 2	24	24	24
Verification	× 2	24	24	24
Contingency	× 2	4	4	4
Acceptance test	× 2	20	20	20
Demonstration	× 2	20	20	20
Documentation				
FRD		3		
FDD		3		
SRD		3	3	
SDD		3	3	3
SSD		31	31	31
SOM		3	3	3
STT		6	6	6
Management				
FRD review		2		
FDD review		2		
SRD review		5	5	
Management plan		7	7	7
SDD review		3	3	3
High level design review		1	1	1
Acceptance review		1	1	1
Transfer		1	1	1
CDF SSD phase	× 1.3	12	12	12
CDF test phase	× 1.3	4	4	4
Total		463	440	422
Factor		1.91	2.01	2.10

Table 5. Time critical module effort estimates

Activity		FRD start	SRD start	SDD start
Requirements and design				
FRD phase		6		
FDD phase		7		
SRD phase		10	10	
SDD phase	× 1.5	20	20	20
SIS activity	× 1.5	15	15	15
SSD phase	× 3	66	66	66
Special	× 3	18	18	18
Other	× 4	9	9	9
Coding				
SSD phase	× 2	46	46	46
Special	× 2	12	12	12
Other	× 2	8	8	8
Testing				
Integration	× 2	24	24	24
Verification	× 2	24	24	24
Contingency	× 2	4	4	4
Acceptance test	× 2	20	20	20
Demonstration	× 2	20	20	20
Documentation				
FRD		3		
FDD		3		
SRD		3	3	
SDD		3	3	3
SSD		31	31	31
SOM		3	3	3
STT		6	6	6
Management				
FRD review		2		
FDD review		2		
SRD review		5	5	
Management plan		7	7	7
SDD review		3	3	3
High level design review		1	1	1
Acceptance review		1	1	2
Transfer		1	1	1
CDF SSD phase	× 1.3	12	12	12
CDF test phase	× 1.3	4	4	4
Total		399	376	358
Factor		1.649	1.72	1.78

Table 6. DSN documentation statistics

Subsystem	KSLOC	SRD	SDD	SSD	SOM	STT	Total
				(pages)			
CPA OP-F/A	22.4	63	123	1974	193	431	2784
TPA OP-F	49.3	43	58	3678	233	736	4748
MDA OP-D	15.1	22	33	919	85	53	1112
DST OP-C	3.7	18	38	380	36	52	524
MON OP-C	12.2	11	73	1318	128	162	1692
CMF OP-D	11.8	17	23	1340	95	66	1541

The SRD, SDD, and SSD recommended best-fit sizes are:

$$D_{SRD} = 7.1\ L^{0.492} \quad (\times/1.5)$$

$$D_{SDD} = 23.4\ L^{0.313} \quad (\times/1.6)$$

$$D_{SSD} = 78.4\ L^{0.831} \quad (\times/1.2)$$

$$D_{SOM} = 15.3\ L^{0.727} \quad (\times/1.2)$$

$$D_{STT} = 7.71\ L^{1.09} \quad (\times/1.8)$$

IV. Typical Example of the Model Operation

The software prototype implementation of the DSN Software Cost Model was programmed for the JPL Automated Office Data Center/Terminal Work Station (AODC/TWS, Ref. 20), which utilizes a Z-80 CPU, 64K-bytes RAM, diskette storage, and a CP/M-compatible operating system.

The user merely enters the program name (SOFTCOST) and answers prompted questions. A blank form is available (see Fig. 6) for data entry by an operator, should the user so desire. The example on the completed form shown in Fig. 7 produces the outputs shown in Figs. 8 through 12. Outputs are selectable; all or none can be generated.

V. Summary and Conclusion

The Software Cost Model reported here is the first of a series of refinements. As the model is used and as performance data are collected, no doubt changes will be made: adjustments of parameters, alterations of formulas, modifications of formats, added and deleted input requirements, additional kinds of outputs, and so forth. One extension currently envisioned is the automated transfer of the WBS data base generated by the model into the WBS-based project control system currently used in the DSN Data Systems implementation tasks, by means of networking the software model host computer with the project control system host.

If the model presented seems complex, it is justly so, for the factors which affect human performance are generally complex and unpredictable, except in statistical terms. One sample function chosen from a stochastic ensemble is hardly ever "average" or "typical." One must expect variations between actual behavior and predictions by the model.

The directions for the future are to refine the model for greater accuracy (within the human performance estimation capacity limits), to extend the utility of the model throughout the entire software life cycle, and to provide the basis for indicating needed new computer technology, software methodology, and tools.

TITLE: _____ CPCI: _____
ECR/ECO: _____ PROG. ID.: _____
SUBSYS: _____ Date Estimated: _____

Answer the following items to the best of your estimation.

1. How much new code is to be produced (completely new modules)?
   Maximum value, kilo-lines executable source(99% confidence level)? _____
   Expected value, kilo-lines executable source? _____
   Minimum value, kilo-lines executable source(99% confidence level)? _____

2. How much code exists in modules requiring modification?
   Maximum value, kilo-lines executable source(99% confidence level)? _____
   Expected value, kilo-lines executable source? _____
   Minimum value, kilo-lines executable source(99% confidence level)? _____

3. How much code will be deleted from these existing modules?
   Maximum value, kilo-lines executable source(99% confidence level)? _____
   Expected value, kilo-lines executable source? _____
   Minimum value, kilo-lines executable source(99% confidence level)? _____

4. How much code will be added to these existing modules?
   Maximum value, kilo-lines executable source(99% confidence level)? _____
   Expected value, kilo-lines executable source? _____
   Minimum value, kilo-lines executable source(99% confidence level)? _____

5. How much code will be changed in other ways in these modules?
   Maximum value, kilo-lines executable source(99% confidence level)? _____
   Expected value, kilo-lines executable source? _____
   Minimum value, kilo-lines executable source(99% confidence level)? _____

6. How much code will be deleted as entire modules from existing code?
   Maximum value, kilo-lines executable source(99% confidence level)? _____
   Expected value, kilo-lines executable source? _____
   Minimum value, kilo-lines executable source(99% confidence level)? _____

7. How much of the remaining existing code must be retested?
   Maximum value, kilo-lines executable source(99% confidence level)? _____
   Expected value, kilo-lines executable source? _____
   Minimum value, kilo-lines executable source(99% confidence level)? _____

8. Expected percentage of code to be developed actually delivered
   (0-90, 91-99, 100)? _____

9. How many different kinds of input/output data items per 1000 lines of
   new or modified code(>80, 16-80, 0-15)? _____

10. Overall complexity of program and data base architecture
    (high, medium, low)? _____

11. Complexity of code logical design(high, medium, low)? _____

12. What percent of the programming task is in Assembly language? _____

13. What percent of the new or modified code must be storage-optimized? _____

Fig. 6. Blank form for software cost model inputs

14. What percent of the new or modified code must be timing-optimized? _____

15. What percent of the total programming task is 'easy'? _____

16. What percent of the total programming task is 'hard'? _____

17. When is work to start, on the(FRD/FDD, SRD, SDD)? _____

18. What percent of the total program requirements are well established, stable, and will not be altered before delivery? _____

19. What percent of the requirements are likely to change slightly before delivery, but will do so under baseline change control? _____

20. What percent of the requirements are likely to change more drastically before delivery, but will do so under baseline control? _____

21. Complexity of program functional requirements(high, medium, low)? _____

22. Expected user involvement in requirements definition (much, some, none)? _____

23. Customer experience in application area(much, none, some)? _____

24. Customer/implementor organizational interface complexity (high, normal, low)? _____

25. Interfaces with other SW development projects or organizations (many, few, none)? _____

26. Efficiency of implementing organization(poor, ok, good)? _____

27. Overall implementation personnel qualifications and motivation (low, average, high)? _____

28. Percentage of programmers doing functional design who will also be doing development(<25, 25-50, >50)? _____

29. Previous programmer experience with application of similar or greater size and complexity(minimal, average, extensive)? _____

30. What is the average staff experience, in years, obtained from work similar to that required in the task being estimated? _____

31. Previous experience with operational computer to be used (minimal, average, extensive)? _____

32. Previous experience with programming language(s) to be used (minimal, average, extensive)? _____

33. Use of top-down methodology(low, medium, high)? _____

34. Use of structured programmer team concepts(low, medium, high)? _____

35. Use of Structured Programming(low, medium, high)? _____

Fig. 6 (contd)

36. Use of design and code inspections(low, QA, peer)? _____

37. Classified security environment for computer(yes, , no)? _____

38. Hardware under concurrent development(much, some, none)? _____

39. Percent of work done at primary development site (<70, 70-90, >90)? _____

40. Development computer access mode(remote, scheduled, demand)? _____

41. Percent of development computer access availability(<30, 30-60, >60)? _____

42. Quality of SW development tools and environment(poor, ok, good)? _____

43. Maturity of system and support software(buggy, ok, good)? _____

44. Overall adverse constraints on program design (severe, average, minimal)? _____

45. Is the program real-time, multi-task(chiefly, some, no)? _____

46. SW to be adaptable to multiple computer configurations or environments (yes, , no)? _____

47. Adaptation required to change from development to operational environment(much, some, minimal)? _____

Values to be used for planning are:
- Kilo-lines of code= _____
- Effort (person-months): _____
- Duration (months): _____
- Average staff (persons): _____

Is output to be saved in a file? _____

Name of output file to be created: _____

Schedule start date: _____

Select desired outputs and output media, or enter RETURN only for defaults. Defaults are 1A, 2A, and 3A. Choices are:

1=Gantt Chart	A=file
2=PERT data, 132 width	B=line printer
3=PERT data, 80 width	

Choice(s): _____

Fig. 6. (contd)

TITLE: VERSION CONTROL EDITOR CDE: Angus Day
ECR/ECO: E 80.126 PROG. ID.: HVP-H2-OP-0.2
SUBSYS: XA1.6 Date Estimated: 10 Nov 80

Answer the following items to the best of your estimation.

1. How much new code is to be produced (completely new modules)?
   Maximum value, kilo-lines executable source(99% confidence level)? 5.5
   Expected value, kilo-lines executable source? 3.5
   Minimum value, kilo-lines executable source(99% confidence level)? 2.1

2. How much code exists in modules requiring modification?
   Maximum value, kilo-lines executable source(99% confidence level)? 6.9
   Expected value, kilo-lines executable source? 6.6
   Minimum value, kilo-lines executable source(99% confidence level)? 6.3

3. How much code will be deleted from these existing modules?
   Maximum value, kilo-lines executable source(99% confidence level)? .4
   Expected value, kilo-lines executable source? .3
   Minimum value, kilo-lines executable source(99% confidence level)? .2

4. How much code will be added to these existing modules?
   Maximum value, kilo-lines executable source(99% confidence level)? .7
   Expected value, kilo-lines executable source? .6
   Minimum value, kilo-lines executable source(99% confidence level)? .4

5. How much code will be changed in other ways in these modules?
   Maximum value, kilo-lines executable source(99% confidence level)? 1.2
   Expected value, kilo-lines executable source? .9
   Minimum value, kilo-lines executable source(99% confidence level)? .7

6. How much code will be deleted as entire modules from existing code?
   Maximum value, kilo-lines executable source(99% confidence level)? 1.4
   Expected value, kilo-lines executable source? 1.3
   Minimum value, kilo-lines executable source(99% confidence level)? 1.1

7. How much of the remaining existing code must be retested?
   Maximum value, kilo-lines executable source(99% confidence level)? 2.1
   Expected value, kilo-lines executable source? 1.9
   Minimum value, kilo-lines executable source(99% confidence level)? 1.7

8. Expected percentage of code to be developed actually delivered (0-90, 91-99, 100)? 91-99

9. How many different kinds of input/output data items per 1000 lines of new or modified code(>80, 16-80, 0-15)? 16-80

10. Overall complexity of program and data base architecture (high, medium, low)? med

11. Complexity of code logical design(high, medium, low)? low

12. What percent of the programming task is in Assembly language? 9

13. What percent of the new or modified code must be storage-optimized? 9

Fig. 7. Example of input form usage

14. What percent of the new or modified code must be timing-optimized? 9

15. What percent of the total programming task is 'easy'? 20

16. What percent of the total programming task is 'hard'? 80

17. When is work to start, on the(FRD/FDD, SRD, SDD)? FRD/FDD

18. What percent of the total program requirments are well established, stable, and will not be altered before delivery? 80

19. What percent of the requirements are likely to change slightly before delivery, but will do so under baseline change control? 10

20. What percent of the requirements are likely to change more drastically before delivery, but will do so under baseline control? 5

21. Complexity of program functional requirements(high, medium, low)? low

22. Expected user involvement in requirements definition (much, some, none)? much

23. Customer experience in application area(much, none, some)? some

24. Customer/implementor organizational interface complexity (high, normal, low)? normal

25. Interfaces with other SW development projects or organizations (many, few, none)? few

26. Efficiency of implementing organization(poor, ok, good)? good

27. Overall implementation personnel qualifications and motivation (low, average, high)? high

28. Percentage of programmers doing functional design who will also be doing development(<25, 25-50, >50)? 25-50

29. Previous programmer experience with application of similar or greater size and complexity(minimal, average, extensive)? average

30. What is the average staff experience, in years, obtained from work similar to that required in the task being estimated? 6

31. Previous experience with operational computer to be used (minimal, average, extensive)? min

32. Previous experience with programming language(s) to be used (minimal, average, extensive)? min

33. Use of top-down methodology(low, medium, high)? high

34. Use of structured programmer team concepts(low, medium, high)? high

35. Use of Structured Programming(low, medium, high)? high

Fig. 7 (contd)

```
36. Use of design and code inspections(low, QA, peer)?                    QA
37. Classified security environment for computer(yes, , no)?             no
38. Hardware under concurrent development(much, some, none)?             none
39. Percent of work done at primary development site
    (<70, 70-90, >90)?                                                   70-90
40. Development computer access mode(remote, scheduled, demand)?         Demand
41. Percent of development computer access availability(<30, 30-60, >60)? 30-60
42. Quality of SW development tools and environment(poor, ok, good)?     ok
43. Maturity of system and support software(buggy, ok, good)?            ok
44. Overall adverse constraints on program design
    (severe, average, minimal)?                                          min
45. Is the program real-time, multi-task(chiefly, some, no)?             no
46. SW to be adaptable to multiple computer configurations or environments
    (yes, , no)?                                                         no
47. Adaptation required to change from development to operational
    environment(much, some, minimal)?                                    min

Values to be used for planning are:
   Kilo-lines of code=                                                   —
   Effort (person-months):                                               —
   Duration (months):                                                    16 mos
   Average staff (persons):                                              2

Is output to be saved in a file?                                         yes

Name of output file to be created:                                       VCEDIT

Schedule start date:                                                     17 NOV 80

Select desired outputs and output media, or enter RETURN only for
defaults. Defaults are 1A, 2A, and 3A. Choices are:

                  1=Gantt Chart                  A=file
                  2=PERT data, 132 width         B=line printer
                  3=PERT data, 80 width
Choice(s):                                                               1B, 2B, 3A
```

Fig. 7 (contd)

TITLE: VERSION CONTROL EDITOR
ECR/ECO: e80.176
SUBSYS: X21.6

CDE: Angus Day
PROG. ID.: HUP-D2-OP-D.2
Date Estimated: 14NOV80
Model Data Version 1.3 31OCT80

Answer the following items to the best of your estimation.

1.	How much new code is to be produced (completely new modules)?	
	Maximum value, kilo-lines executable source(99% confidence level)?	3.5
	Expected value, kilo-lines executable source?	3.3
	Minimum value, kilo-lines executable source(99% confidence level)?	3.1
2.	How much code exists in modules requiring modification?	
	Maximum value, kilo-lines executable source(99% confidence level)?	6.9
	Expected value, kilo-lines executable source?	6.6
	Minimum value, kilo-lines executable source(99% confidence level)?	6.3
3.	How much code will be deleted from these existing modules?	
	Maximum value, kilo-lines executable source(99% confidence level)?	.4
	Expected value, kilo-lines executable source?	.3
	Minimum value, kilo-lines executable source(99% confidence level)?	.2
4.	How much code will be added to these existing modules?	
	Maximum value, kilo-lines executable source(99% confidence level)?	.7
	Expected value, kilo-lines executable source?	.6
	Minimum value, kilo-lines executable source(99% confidence level)?	.4
5.	How much code will be changed in other ways in these modules?	
	Maximum value, kilo-lines executable source(99% confidence level)?	1.2
	Expected value, kilo-lines executable source?	.9
	Minimum value, kilo-lines executable source(99% confidence level)?	.7
6.	How much code will be deleted as entire modules from existing code?	
	Maximum value, kilo-lines executable source(99% confidence level)?	1.4
	Expected value, kilo-lines executable source?	1.3
	Minimum value, kilo-lines executable source(99% confidence level)?	1.1
7.	How much of the remaining existing code must be retested?	
	Maximum value, kilo-lines executable source(99% confidence level)?	2.1
	Expected value, kilo-lines executable source?	1.9
	Minimum value, kilo-lines executable source(99% confidence level)?	1.5
8.	Expected percentage of code to be developed actually delivered (0-90, 91-99, 100)?	91-99
9.	How many different kinds of input/output data items per 1000 lines of new or modified code(>80, 16-80, 0-15)?	16-80
10.	Overall complexity of program and data base architecture (high, medium, low)?	MEDIUM
11.	Complexity of code logical design(high, medium, low)?	LOW
12.	What percent of the programming task is in Assembly language?	9
13.	What percent of the new or modified code must be storage-optimized?	9

Fig. 8. Hard copy format input parameter record

14. What percent of the new or modified code must be timing-optimized?	9
15. What percent of the total programming task is 'easy'?	20
16. What percent of the total programming task is 'hard'?	30
17. When is work to start, on the(FRD/FDD, SRD, SDD)?	FRD/FDD
18. What percent of the total program requirments will be established and stable before design, and will not be altered before delivery?	80
19. What percent of the requirements are likely to change slightly before delivery, but will do so under baseline change control?	10
20. What percent of the requirements are likely to change more drastically before delivery, but will do so under baseline control?	5
21. Complexity of program functional requirements(high, medium, low)?	LOW
22. Expected user involvement in requirements definition (much, some, none)?	MUCH
23. Customer experience in application area(much, none, some)?	SOME
24. Customer/implementor organizational interface complexity (high, normal, low)?	NORMAL
25. Interfaces with other SW development projects or organizations (many, few, none)?	FEW
26. Efficiency of implementing organization(poor, ok, good)?	GOOD
27. Overall implementation personnel qualifications and motivation (low, average, high)?	HIGH
28. Percentage of programmers doing functional design who will also be doing development(<25, 25-50, >50)?	25-50
29. Previous programmer experience with application of similar or greater size and complexity(minimal, average, extensive)?	AVERAGE
30. What is the average staff experience, in years, obtained from work similar to that required in the task being estimated?	6
31. Previous experience with operational computer to be used (minimal, average, extensive)?	MINIMAL
32. Previous experience with programming language(s) to be used (minimal, average, extensive)?	MINIMAL
33. Use of top-down methodology(low, medium, high)?	HIGH
34. Use of structured programmer team concepts(low, medium, high)?	HIGH
35. Use of Structured Programming(low, medium, high)?	HIGH

Fig. 8 (cont'd)

36. Use of design and code inspections(low, QA, peer)?	QA
37. Classified security environment for computer(yes, , no)?	NO
38. Hardware under concurrent development(much, some, none)?	NONE
39. Percent of work done at primary development site (<70, 70-90, >90)?	70-90
40. Development computer access mode(remote, scheduled, demand)?	DEMAND
41. Percent of development computer access availability(<30, 30-60, >60)?	30-60
42. Quality of SW development tools and environment(poor, ok, good)?	OK
43. Maturity of system and support software(buggy, ok, good)?	OK
44. Overall adverse constraints on program design (severe, average, minimal)?	MINIMAL
45. Is the program real-time, multi-task(chiefly, some, no)?	NO
46. SW to be adaptable to multiple computer configurations or environments (yes, , no)?	NO
47. Adaptation required to change from development to operational environment(much, some, minimal)?	MINIMAL

Fig. 8 (contd)

```
Estimated Overall Parameters:
                                                  *average value
                                        +1-sigma                    -1-sigma
Adjusted Lines of code= 6182 SLEC
 6280            6085
Effort= 26.5 person-months
 45.8             15.3
Staff productivity= 233 SLEC/staff-month
 404             135
Duration= 15.7 months
 18.0             12.9
Avg. Staff= 1.7
  2.9             1.0
Documentation= 537 pages     $16.1K
  645             448            $19.4K              $13.4K
Computer CPU time= 319 hours      $0.0K
  478             212             $0.0K               $0.0K

Use these figures to arrive at Effort, Duration, and Staffing
requirements. Include factors to provide acceptible risk
and confidence levels.


Values specified are:
   Kilo-lines of code=                                                  6.18
   Effort (person-months):                                              32.0
   Duration (months):                                                   16.0
   Average staff (persons):                                              2.0

    For the numbers you have entered, a reasonableness check indicates that
the average project would produce 7303 lines of code, using 32 staff-months
of resources and 16 months of duration, with an average staff of 2 persons,
for a productivity of 228 SLEC/staff-month.

The level of confidence in delivering 6182 lines of code,
on-time and within resources= 33 %.

Is output to be saved in a file?                                          Y

Name of output file to be created:                                    VCEDIT

Schedule start date:                                                17NOV80


Select desired outputs and output media, or enter RETURN only for
defaults. Defaults are 1A, 2A, and 3A. Choices are:

                  1=Gantt Chart                 A=file
                  2=PERT data, 132 width        B=line printer
                  3=PERT data, 80 width

Choice(s):                                                        1B,2B,3A
```

Fig. 9. Output of SWAG estimator using the "typical software project" parameters

PAGE 1

TITLE: VERSION CONTROL EDITOR
ECR/ECO: e80.176
SUBSYS: X21.6

CDE: Angus Day
PROG. ID.: HUP-D2-GP-D.2
STATUS AS OF: 14NOV80

CODE	TASK	WHO	EFF ORT	START EARLY DAY	START EARLY DATE	START LATE DAY	START LATE DATE	FINISH EARLY DAY	FINISH EARLY DATE	FINISH LATE DAY	FINISH LATE DATE	FLOAT TIME DAYS
*0.	START		0	0	17NOV80	0	17NOV80	0	17NOV80	0	17NOV80	0
*1.	Sys Plans, Reqts, & Design		0	32	9JAN81	32	9JAN81	32	9JAN81	32	9JAN81	0
* 1.1	Define Subsys Reqts		15	0	17NOV80	0	17NOV80	10	3DEC80	10	3DEC80	0
* 1.2	FRD		0	13	8DEC80	13	8DEC80	13	8DEC80	13	8DEC80	0
1.2.1	Write all sections		4	0	17NOV80	9	2DEC80	3	26NOV80	12	5DEC80	9
* 1.2.2	Edit and release FRD		2	12	5DEC80	12	5DEC80	13	8DEC80	13	8DEC80	0
* 1.3	Level B Review		3	13	8DEC80	13	8DEC80	15	10DEC80	15	10DEC80	0
* 1.4	Define Sys Architecture		18	15	10DEC80	15	10DEC80	27	31DEC80	27	31DEC80	0
* 1.5	FDD		0	30	7JAN81	30	7JAN81	30	7JAN81	30	7JAN81	0
1.5.1	Write all sections		4	15	10DEC80	27	31DEC80	17	12DEC80	29	6JAN81	12
* 1.5.2	Edit and release		2	29	6JAN81	29	6JAN81	30	7JAN81	30	7JAN81	0
* 1.6	Level C Review		3	30	7JAN81	30	7JAN81	32	9JAN81	32	9JAN81	0
*2.	SW Planning and Reqts		0	58	16FEB81	58	16FEB81	58	16FEB81	58	16FEB81	0
* 2.1	Define Software Reqts		25	32	9JAN81	32	9JAN81	50	4FEB81	50	4FEB81	0
* 2.2	SRD		0	56	12FEB81	56	12FEB81	56	12FEB81	56	12FEB81	0
2.2.1	Write all sections		4	32	9JAN81	53	9FEB81	34	13JAN81	55	11FEB81	21
* 2.2.2	Edit and release		2	55	11FEB81	55	11FEB81	56	12FEB81	56	12FEB81	0
* 2.3	Level D Review		2	56	12FEB81	56	12FEB81	58	16FEB81	58	16FEB81	0
*3.	SW Architecture and Design		0	93	6APR81	93	6APR81	93	6APR81	93	6APR81	0
* 3.1	Define SW architecture		31	58	16FEB81	58	16FEB81	81	19MAR81	81	19MAR81	0
* 3.2	SDD		0	91	2APR81	91	2APR81	91	2APR81	91	2APR81	0
3.2.1	Write all sections		4	58	16FEB81	88	30MAR81	60	18FEB81	90	1APR81	30
* 3.2.2	Edit and release		2	90	1APR81	90	1APR81	91	2APR81	91	2APR81	0
3.3	System Interface Design		22	58	16FEB81	78	16MAR81	70	4MAR81	90	1APR81	20
* 3.4	Level E Review		2	91	2APR81	91	2APR81	93	6APR81	93	6APR81	0
*4.	SW Detailed Design & Prod		0	273	22DEC81	273	22DEC81	273	22DEC81	273	22DEC81	0
4.1	SSD		0	315	23FEB82	327	11MAR82	315	23FEB82	327	11MAR82	12
4.1.1	Write Sections 1,2,3		6	93	6APR81	268	15DEC81	96	9APR81	271	18DEC81	175
* 4.1.2	Write Section 4		18	93	6APR81	93	6APR81	107	24APR81	107	24APR81	0
4.1.3	Write Section 5		43	107	24APR81	249	16NOV81	129	27MAY81	271	18DEC81	142
4.1.4	Write Section 6		4	93	6APR81	269	16DEC81	95	8APR81	271	18DEC81	176
4.1.5	Write Section 7		9	93	6APR81	266	11DEC81	98	13APR81	271	18DEC81	173
4.1.6	Edit and release		6	312	18FEB82	324	8MAR82	315	23FEB82	327	11MAR82	12
4.2	SQM		0	314	22FEB82	327	11MAR82	314	22FEB82	327	11MAR82	13
* 4.2.1	Write preliminary draft		8	107	24APR81	107	24APR81	113	4MAY81	113	4MAY81	0
4.2.2	Complete all sections		9	133	2JUN81	266	11DEC81	138	9JUN81	271	18DEC81	133
4.2.3	Edit and release		4	312	18FEB82	325	9MAR82	314	22FEB82	327	11MAR82	13
* 4.3	High-level Design Review		2	131	29MAY81	131	29MAY81	133	2JUN81	133	2JUN81	0
* 4.4	Module Production & Integ		0	271	18DEC81	271	18DEC81	271	18DEC81	271	18DEC81	0
* 4.4.1	Executive and control		18	113	4MAY81	113	4MAY81	131	29MAY81	131	29MAY81	0
* 4.4.2	I/O Modules		18	133	2JUN81	133	2JUN81	151	26JUN81	151	26JUN81	0
* 4.4.3	Interface handlers		18	151	26JUN81	151	26JUN81	169	23JUL81	169	23JUL81	0

Fig. 10. Full PERT/CPM work breakdown structure and schedule table output of the software cost model

TITLE: VERSION CONTROL EDITOR
ECR/ECO: e80.176
SUBSYS: X21.6

CDE: Angus Day
PROG. ID.: HUP-D2-OP-D.2
STATUS AS OF: 14NOV80

CODE	TASK	WHO	EFFORT	START EARLY DAY	START EARLY DATE	START LATE DAY	START LATE DATE	FINISH EARLY DAY	FINISH EARLY DATE	FINISH LATE DAY	FINISH LATE DATE	FLOAT TIME DAYS
* 4.4.4	Function A		18	169	23JUL81	169	23JUL81	186	17AUG81	186	17AUG81	0
* 4.4.5	Function B		18	186	17AUG81	186	17AUG81	203	11SEP81	203	11SEP81	0
* 4.4.6	Function C		17	203	11SEP81	203	11SEP81	220	6OCT81	220	6OCT81	0
* 4.4.7	Function D		17	220	6OCT81	220	6OCT81	237	29OCT81	237	29OCT81	0
* 4.4.8	Function E		17	237	29OCT81	237	29OCT81	254	23NOV81	254	23NOV81	0
* 4.4.9	Function F		17	254	23NOV81	254	23NOV81	271	18DEC81	271	18DEC81	0
4.5	Special Tasks		0	139	10JUN81	271	18DEC81	139	10JUN81	271	18DEC81	132
4.5.1	Support software		12	133	2JUN81	265	10DEC81	139	10JUN81	271	18DEC81	132
4.5.2	Other		6	133	2JUN81	268	15DEC81	136	5JUN81	271	18DEC81	135
* 4.6	Acceptance Readiness Rvw		[illegible]	271	18DEC81	271	18DEC81	273	22DEC81	273	22DEC81	0
*5.	SW Test and Transfer		[illegible]	332	18MAR82	332	18MAR82	332	18MAR82	332	18MAR82	0
* 5.1	Verification tests		28	273	22DEC81	273	22DEC81	299	1FEB82	299	1FEB82	0
5.2	Contingency		25	96	9APR81	314	22FEB82	109	28APR81	327	11MAR82	218
5.3	STT		0	313	19FEB82	332	16MAR82	313	19FEB82	332	18MAR82	19
5.3.1	Write all sections		14	133	2JUN81	292	21JAN82	140	11JUN81	299	1FEB82	159
5.3.2	Edit and release		2	312	18FEB82	326	10MAR82	313	19FEB82	327	11MAR82	14
* 5.4	Acceptance tests		20	299	1FEB82	299	1FEB82	312	18FEB82	312	18FEB82	0
* 5.5	Demonstration tests		22	312	18FEB82	312	18FEB82	327	11MAR82	327	11MAR82	0
* 5.6	Transfer, CDE to COE		7	327	11MAR82	327	11MAR82	332	18MAR82	332	18MAR82	0
6.	Mgt Tasks and Milestones		0	19	16DEC80	332	18MAR82	19	16DEC80	332	18MAR82	313
6.1	CDE Activities		37	0	17NOV80	313	19FEB82	19	16DEC80	332	18MAR82	313
* 6.2	Develop prelim budget		3	10	3DEC80	10	3DEC80	12	5DEC80	12	5DEC80	0
* 6.3	Develop Sys Impl Plan		3	27	31DEC80	27	31DEC80	29	6JAN81	29	6JAN81	0
* 6.4	Draft Software Impl Plan		6	50	4FEB81	50	4FEB81	55	11FEB81	55	11FEB81	0
* 6.5	Revise Impl Plan		12	81	19MAR81	81	19MAR81	90	1APR81	90	1APR81	0
6.6	QA Audit		26	312	18FEB82	318	26FEB82	326	10MAR82	332	18MAR82	6
*FINISH			0	332	18MAR82	332	18MAR82	332	18MAR82	332	18MAR82	0

Fig. 10 (contd)

TITLE: VERSION CONTROL EDITOR — CDE: Angus Day
ECR/ECO: e80.176 — PROG. ID.: HUP-D2-OP-D.2
SUBSYS: X21.6 — STATUS AS OF: 14NOV80

CODE	TASK	WHO	EFF	E-START	L-FINSH	FLT
*0.	START		0	17NOV80	17NOV80	0
*1.	Sys Plans, Reqts, & Design		0	9JAN81	9JAN81	0
* 1.1	Define Subsys Reqts		15	17NOV80	3DEC80	0
* 1.2	FRD		0	8DEC80	8DEC80	0
1.2.1	Write all sections		4	17NOV80	5DEC80	9
* 1.2.2	Edit and release FRD		2	5DEC80	8DEC80	0
* 1.3	Level B Review		3	8DEC80	10DEC80	0
* 1.4	Define Sys Architecture		18	10DEC80	31DEC80	0
* 1.5	FDD		0	7JAN81	7JAN81	0
1.5.1	Write all sections		4	10DEC80	6JAN81	12
* 1.5.2	Edit and release		2	6JAN81	7JAN81	0
* 1.6	Level C Review		3	7JAN81	9JAN81	0
*2.	SW Planning and Reqts		0	16FEB81	16FEB81	0
* 2.1	Define Software Reqts		25	9JAN81	4FEB81	0
* 2.2	SRD		0	12FEB81	12FEB81	0
2.2.1	Write all sections		4	9JAN81	11FEB81	21
* 2.2.2	Edit and release		2	11FEB81	12FEB81	0
* 2.3	Level D Review		2	12FEB81	16FEB81	0
*3.	SW Architecture and Design		0	6APR81	6APR81	0
* 3.1	Define SW architecture		31	16FEB81	19MAR81	0
* 3.2	SDD		0	2APR81	2APR81	0
3.2.1	Write all sections		4	16FEB81	1APR81	30
* 3.2.2	Edit and release		2	1APR81	2APR81	0
3.3	System Interface Design		22	16FEB81	1APR81	20
* 3.4	Level E Review		2	2APR81	6APR81	0
*4.	SW Detailed Design & Prod		0	22DEC81	22DEC81	0
4.1	SSD		0	23FEB82	11MAR82	12
4.1.1	Write Sections 1,2,3		6	6APR81	18DEC81	175
* 4.1.2	Write Section 4		18	6APR81	24APR81	0
4.1.3	Write Section 5		43	24APR81	18DEC81	142
4.1.4	Write Section 6		4	6APR81	18DEC81	176
4.1.5	Write Section 7		9	6APR81	18DEC81	173
4.1.6	Edit and release		6	18FEB82	11MAR82	12
4.2	SOM		0	22FEB82	11MAR82	13
* 4.2.1	Write preliminary draft		8	24APR81	4MAY81	0
4.2.2	Complete all sections		9	2JUN81	18DEC81	133
4.2.3	Edit and release		4	18FEB82	11MAR82	13
* 4.3	High-level Design Review		2	29MAY81	2JUN81	0
* 4.4	Module Production & Integ		0	18DEC81	18DEC81	0
* 4.4.1	Executive and control		18	4MAY81	29MAY81	0
* 4.4.2	I/O Modules		18	2JUN81	26JUN81	0
* 4.4.3	Interface handlers		18	26JUN81	23JUL81	0

Fig. 11. Short-form output of the PERT/CPM WBS schedule data

TITLE: VERSION CONTROL EDITOR
ECR/ECO: e80.176
SUBSYS: X21.6

CDE: Angus Day
PROG. ID.: HUP-D2-OP-D.2
STATUS AS OF: 14NOV80

	CODE	TASK	WHO	EFF	E-START	L-FINSH	FLT
*	4.4.4	Function A		18	23JUL81	17AUG81	0
*	4.4.5	Function B		18	17AUG81	11SEP81	0
*	4.4.6	Function C		17	11SEP81	6OCT81	0
*	4.4.7	Function D		17	6OCT81	29OCT81	0
*	4.4.8	Function E		17	29OCT81	23NOV81	0
*	4.4.9	Function F		17	23NOV81	18DEC81	0
	4.5	Special Tasks		0	10JUN81	18DEC81	132
	4.5.1	Support software		12	2JUN81	18DEC81	132
	4.5.2	Other		6	2JUN81	18DEC81	135
*	4.6	Acceptance Readiness Rvw		2	18DEC81	22DEC81	0
*	5.	SW Test and Transfer		0	18MAR82	18MAR82	0
*	5.1	Verification tests		28	22DEC81	1FEB82	0
	5.2	Contingency		25	9APR81	11MAR82	218
	5.3	STT		0	19FEB82	18MAR82	19
	5.3.1	Write all sections		14	2JUN81	1FEB82	159
	5.3.2	Edit and release		2	18FEB82	11MAR82	14
*	5.4	Acceptance tests		20	1FEB82	18FEB82	0
*	5.5	Demonstration tests		22	18FEB82	11MAR82	0
*	5.6	Transfer, CDE to COE		7	11MAR82	18MAR82	0
	6.	Mgt Tasks and Milestones		0	16DEC80	18MAR82	313
	6.1	CDE Activities		37	17NOV80	18MAR82	313
*	6.2	Develop prelim budget		3	3DEC80	5DEC80	0
*	6.3	Develop Sys Impl Plan		3	31DEC80	6JAN81	0
*	6.4	Draft Software Impl Plan		6	4FEB81	11FEB81	0
*	6.5	Revise Impl Plan		12	19MAR81	1APR81	0
	6.6	QA Audit		26	18FEB82	18MAR82	6
*	FINISH			0	18MAR82	18MAR82	0

Fig. 11 (contd)

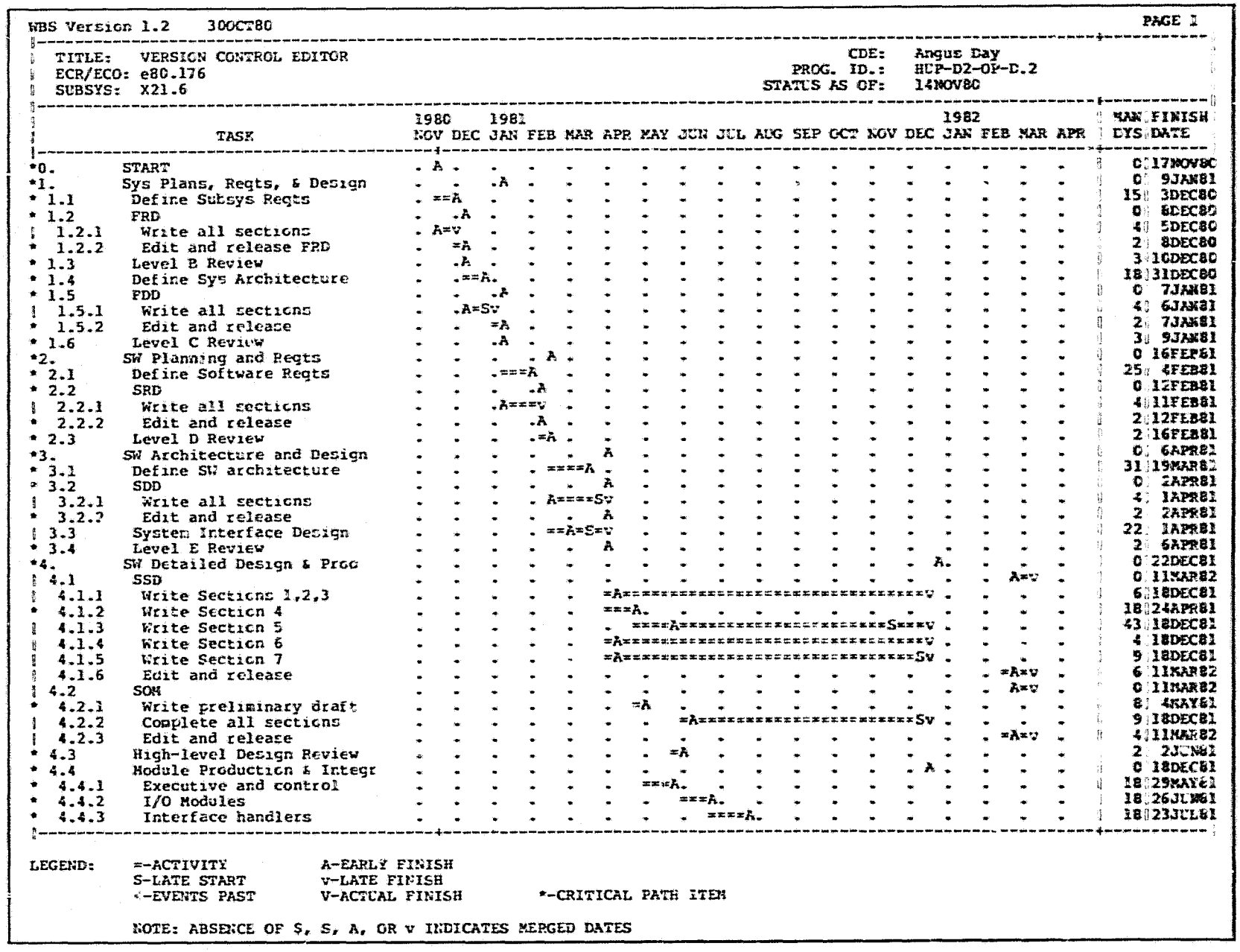

Fig. 12. Gantt chart output of the software cost model

```
WBS Version 1.2     30OCT80                                                                                                    PAGE 2
|-----------------------------------------------------------------------------------------------------------------+-----------|
| TITLE:   VERSION CONTROL EDITOR                                                  CDE:       Angus Day                         |
| ECR/ECO: e80.176                                                             PROG. ID.:     HUP-D2-CP-D.2                     |
| SUBSYS:  X21.6                                                            STATUS AS OF:     14NOV80                           |
|-----------------------------------------------------------------------------------------------------------------+-----------|
|                                      1980    1981                                                1982          | MAN FINISH|
|                TASK                  NOV DEC JAN FEB MAR APR MAY JUN JUL AUG SEP OCT NOV DEC JAN FEB MAR APR   | DYS DATE  |
|--------------------------------------+--------------------------------------------------------------------------+-----------|
*  4.4.4     Function A                 .   .   .   .   .   .   .   .   . ===A .   .   .   .   .   .   .   .       18 17AUG81
*  4.4.5     Function B                 .   .   .   .   .   .   .   .   .   . ===A .   .   .   .   .   .   .       18 11SEP81
*  4.4.6     Function C                 .   .   .   .   .   .   .   .   .   .   .===A  .   .   .   .   .   .       17  6OCT81
*  4.4.7     Function D                 .   .   .   .   .   .   .   .   .   .   .   ===A.  .   .   .   .   .       17 29OCT81
*  4.4.8     Function E                 .   .   .   .   .   .   .   .   .   .   .   .  ====A.  .   .   .   .       17 23NOV81
*  4.4.9     Function F                 .   .   .   .   .   .   .   .   .   .   .   .   .  ===A .   .   .   .       17 18DEC81
| 4.5      Special Tasks                .   .   .   .   .   .   .   .A=======================v .   .   .   .        0 18DEC81
|  4.5.1     Support software           .   .   .   .   .   .   .   =A======================Sv .   .   .   .       12 18DEC81
|  4.5.2     Other                      .   .   .   .   .   .   .   A========================v .   .   .   .        6 18DEC81
* 4.6      Acceptance Readiness Rvw     .   .   .   .   .   .   .   .   .   .   .   .   .   . *A.  .   .   .        2 22DEC81
*5.       SW Test and Transfer          .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   . A .        0 18MAR82
* 5.1      Verification tests           .   .   .   .   .   .   .   .   .   .   .   .   .   .   =====A  .   .       28  1FEB82
| 5.2      Contingency                  .   .   .   .   .   .==A=====================================S=v  .       25 11MAR82
| 5.3      STT                          .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   . A===v .        0 18MAR82
|  5.3.1     Write all sections         .   .   .   .   .   .   .   =A============================Sv .   .       14  1FEB82
|  5.3.2     Edit and release           .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   . A==v .        2 11MAR82
* 5.4      Acceptance tests             .   .   .   .   .   .   .   .   .   .   .   .   .   .   .  ==A .   .       20 18FEB82
* 5.5      Demonstration tests          .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   . ===A .       22 11MAR82
* 5.6      Transfer, CDE to COE         .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .=A .        7 18MAR82
|6.       Mgt Tasks and Milestones      .   . A=========================================================v .        0 18MAR82
| 6.1      CDE Activities               . ====A=====================================================S===v .       37 18MAR82
* 6.2      Develop prelim budget        .   A   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .        3  5DEC80
* 6.3      Develop Sys Impl Plan        .   .  =A   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .        3  6JAN81
* 6.4      Draft Software Impl Plan     .   .   .  =A   .   .   .   .   .   .   .   .   .   .   .   .   .   .        6  1FEB81
* 6.5      Revise Impl Plan             .   .   .   .   . ==A   .   .   .   .   .   .   .   .   .   .   .   .       12  1APR81
| 6.6      QA Audit                     .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   . =S=Av .       26 18MAR82
*FINISH                                 .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   . A .        0 18MAR82
|--------------------------------------+--------------------------------------------------------------------------+-----------|

LEGEND:    =-ACTIVITY          A-EARLY FINISH
           S-LATE START        v-LATE FINISH
           <-EVENTS PAST       V-ACTUAL FINISH       *-CRITICAL PATH ITEM

           NOTE: ABSENCE OF $, S, A, OR v INDICATES MERGED DATES
```

Fig. 12 (contd)

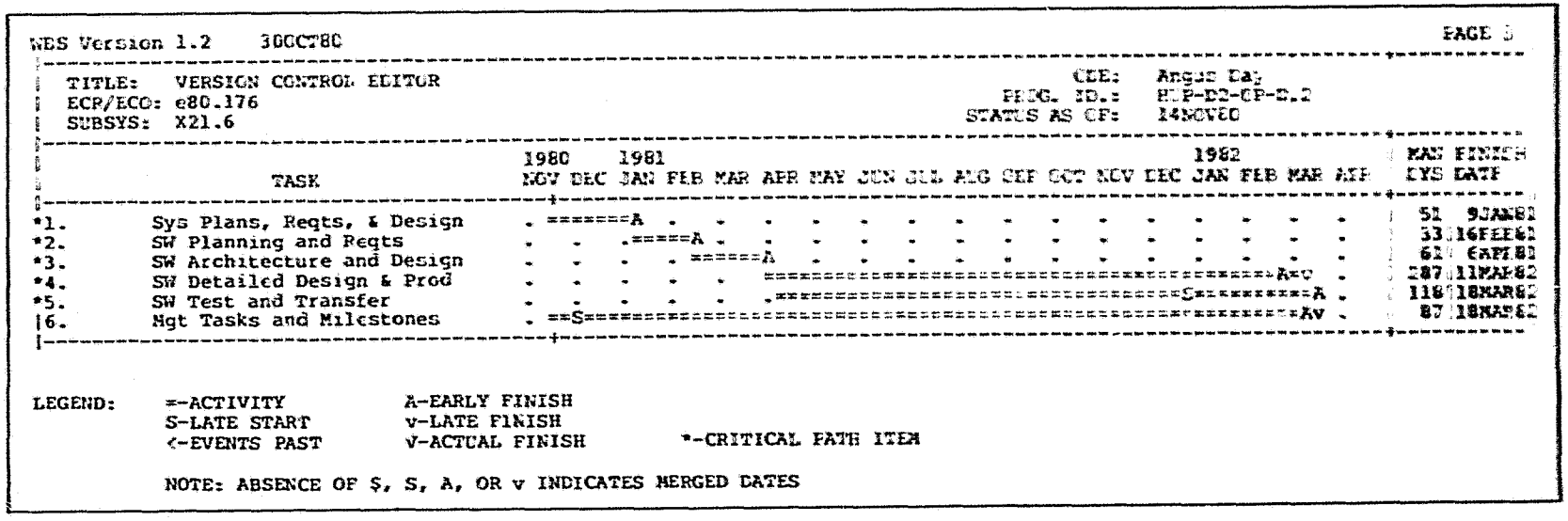
WBS Version 1.2 30OCT80
PAGE 5
TITLE: VERSION CONTROL EDITOR
ECP/ECO: e80.176
SUBSYS: X21.6
CDE: Angie Day
PROG. ID.: HLP-D2-CP-D.2
STATUS AS OF: 14NOV80
TASK
1980 NOV DEC
1981 JAN FEB MAR APR MAY JUN JUL AUG SEP OCT NOV DEC
1982 JAN FEB MAR APR
MAN DYS
FINISH DATE
*1. Sys Plans, Reqts, & Design 51 9JAN81
*2. SW Planning and Reqts 33 16FEB81
*3. SW Architecture and Design 61 6APR81
*4. SW Detailed Design & Prod 287 11MAR82
*5. SW Test and Transfer 116 18MAR82
6. Mgt Tasks and Milestones 87 18MAR82
LEGEND: =-ACTIVITY A-EARLY FINISH
S-LATE START v-LATE FINISH
<-EVENTS PAST V-ACTUAL FINISH *-CRITICAL PATH ITEM
NOTE: ABSENCE OF $, S, A, OR v INDICATES MERGED DATES

Fig. 12 (contd)

References

1. "Quantitive Software Models," Report SRR-1, Data and Analysis Center for Software, RADC, Griffiss AFB, N.Y., March 1979.

2. "Galorath, D. D., and Reifer, D. J., "Analysis of the State-of-the-Art of Parametric Software Cost Modeling," Report SMC-7R-006, Software Management Consultants, Torrance, Calif., August 1980.

3. Walston, C. E., and Felix, C. P., "A Method of Programming Measurement and Estimation," *IBM System Journal*, Vol. 16, No. 1, 1977.

4. Freberger, K., and Basili, V. R., "The Software Engineering Laboratory: Relationship Equations," TR-764, SEL-3, NSG-5123, University of Maryland, College Park, Md., May 1979.

5. Norden, P. V., "Project Life Cycle Modeling," Software Life-Cycle Management Workshop, United States Army Computer Systems Command, August 1977, pp. 217-306.

6. Putnam, L. H., "Influence of the Time-Difficulty Factor in Large-Scale Software Development," Software Life-Cycle Management Workshop, United States Army Computer Systems Command, August 1977, pp. 307-312.

7. Wolverton, R. W., "The Cost of Developing Large Scale Software," *IEEE Transactions on Computers*, Vol. C-23, No. 6, June 1974.

8. Freiman, F. R., "PRICE Software Model," RCA PRICE Systems Publication, Morristown, N. J., November 1977.

9. "Software Life-Cycle Management (SLIM) Estimating Model," Quantitative Software Management, Inc., McLean, Va., 1978.

10. Kustanowitz, A. L., "System Life Cycle Estimation (SLICE)," IEEE First International Computer Software and Applications Conference, Chicago, Ill.

11. Tausworthe, Robert C., "Software Specification Document, DSN Software Cost Model," Jet Propulsion Laboratory, Pasadena, California, 1981 (internal document).

12. The author is unable to trace the definitive reference for this work. The usage cited is given in Putnam, L. H., "Example of an Early Sizing, Cost and Schedule Estimate for an Application Software System," COMPSAC 1978, November 13-16, 1978.

13. Papoulis, A., *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill Book Co., N.Y., 1965, pp. 104, 147.

14. *Handbook of Mathematical Functions*, AMS-55, National Bureau of Standards, U.S. Government Printing Office, Washington D.C., 1964, p. 258.

15. Graver, C.A., et al., "Cost Reporting Elements and Activity Cost Tradeoffs for Defense System Software," CR-1-72/1, General Research Corp., Santa Barbara, Calif., May 1977.

16. Nelson, Richard, "Software Data Collection and Analysis, (Partial Report)," Rome Air Force Development Center, Data and Analysis Center for Software, Griffiss AFB, N.Y., September 1978.

17. Carriere, N. M., and Thibodeau, R., "Development of a Logistics Software Cost Estimating Technique for Foreign Military Sales," General Research Corp., CR-3-839, Santa Barbara, Calif., June 1979.

18. Tausworthe, Robert C., "Preparation Guide for Class B Software Specification Documents," Publication 79-56, Jet Propulsion Laboratory, Pasadena, Calif., October 1979.

19. Basili, V.R., and Zelkowitz, M.V., "Analyzing Medium-Scale Software Development," IEEE Proceedings, 3rd International Conference on Software Engineering, May 1978.

20. As of this writing, there are no externally referenceable documents describing the AODC. User Manuals are available, however, from the AODC User's Library, Information Systems Engineering Section, Jet Propulsion Laboratory, Pasadena, Calif.

Appendix A
DSN Software Cost Model Factors

This Appendix contains a listing of the standard factor file accessed by the software model. A correspondence between factors on the file and those appearing in formulas herein appears below:

FILE AND PROGRAM MNEMONIC	REPORT SYMBOL
NEWFAC	1 (normalized)
CHMFAC	b
ADLFAC	c
CHLFAC	d
DLNFAC	e
DMDFAC	g
RTSFAC	h
LLEV	f_{HOL}
LADJ	f_{assy}
CADJ	$f_{c\text{-}crit}$
TADJ	$f_{t\text{-}crit}$
EADJ	f_{easy}
HADJ	f_{hard}
BADJ	f_{base}
MADJ	f_{many}
FADJ	f_{free}

---

* Factors in the file used by the model not given a specific symbol in this report.

FILE AND PROGRAM MNEMONIC	REPORT SYMBOL
SADJ	f_{exp}
FULLUP	$f_{full-up}$
KSLECPM	P_1
A	a
AWFFAC	f_{wf}
E1FAC	Λ_{sigma}
DFAC	T_1
DEXP	f_t
T1FAC	*
PPKSLEC	D_1
AD	c_{doc}
PGCOST	*
D1FAC	*
CFAC	C_1
AC	c_{cpu}
C1FAC	*
CPTRCOST	*
S1FAC	*
WTFAC	$W_{1/2}$
ATFAC	W/K
LOGRATIO**	$-\log[P_{hi(i)}/P_{lo(i)}]$

* Factors in the file used by the model not given a specific symbol in this report.

** The LOGRATIO designator does not appear in the file; however it is found as the numeric field following the 3 ANSWER entries.

```
VERSION  (VERS$)                ,"Version 1.3    31OCT80"
WORK DAYS PER MO  (DAYSPERMO)   ,20
HDR FIELD 1  (HFL1$)            ,"TITLE:  "
HDR FIELD 2  (HFL2$)            ,"CDE:  "
HDR FIELD 3  (HFL3$)            ,"ECR/ECO:  "
HDR FIELD 4  (HFL4$)            ,"PROG. ID.:  "
HDR FIELD 5  (HFL5$)            ,"SUBSYS:  "
CLEAR SCREEN  (CLS$)            ,2,27,43
NEW CODE FACTOR  (NEWFAC)       ,1
CHG MODULE FAC  (CHMFAC)        ,.27
ADD LINES FAC  (ADLFAC)         ,.80
CHG LINES FAC  (CHLFAC)         ,.24
DEL LINES FAC  (DLNFAC)         ,-.15
DEL MODULE FAC  (DMDFAC)        ,.11
RETEST FACTOR  (RTSFAC)         ,.17
LANGUAGE LEVEL  (LLEV)          ,2.4
LANG ADJ FAC  (LADJ)            ,.82
CAP ADJ FAC  (CADJ)             ,1.01
TIME ADJ FAC  (TADJ)            ,.72
EASY ADJ FAC  (EADJ)            ,.8
HARD ADJ FAC  (HADJ)            ,1.2
BASLINE CHG FAC  (BADJ)         ,1.35
MANY CHG BL FAC  (MADJ)         ,1.9
FREE CHG FAC  (FADJ)            ,2.3
STAFF EXP ADJ FAC  (SADJ)       ,.06
FULL-UP TRNG FAC  (FULLUP)      ,.4
KSLEC/MO  (KSLECPM)             ,.237
DEFAULT PRODUCTIVITY  (PDEFLT)  ,.2
EFFORT EXPONENT  (A)            ,1
WF MODEL EXPONENT  (AWFFAC)     ,1.95
EFFORT STD DEV FAC  (E1FAC)     ,1.73
DURATION FAC  (DFAC)            ,4.88
DURATION EXP  (DEXP)            ,.356
DUR STD DEV FAC  (T1FAC)        ,1.49
PAGES DOC/KSLEC  (PPKSLEC)      ,120
DOC EXPONENT  (AD)              ,.823
COST PER PAGE  (PGCOST)         ,30
DOC STD DEV FAC  (D1FAC)        ,1.2
COMPTR HRS FAC  (CFAC)          ,25.2
CPTR TIME EXPNT  (AC)           ,.96
CPTR TIM STD DEV FAC  (C1FAC)   ,1.5
COMPTR HR COST  (CPTRCOST)      ,0
STAFF STD DEV FAC  (S1FAC)      ,1.53
HALF-TIME EFF. FACTOR (WTFAC)   ,1.5
PUTNAM KA/K FACTOR (ATFAC)      ,.88
WBS TASK SKIPS(SKIPS)           ,0, 14, 21
PROMPT1 ,"    Expected value, kilo-lines executable source"
PROMPT2 ,"    Maximum value, kilo-lines executable source(99% confidence level)"
PROMPT3 ,"    Minimum value, kilo-lines executable source(99% confidence level)"
PROMPT4 ,"1.  How much new code is to be produced (completely new modules)?"
PROMPT5 ,"2.  How much code exists in modules requiring modification?"
```

```
PROMPT6 ,"3.  How much code will be deleted from these existing modules?"
PROMPT7 ,"4.  How much code will be added to these existing modules?"
PROMPT8 ,"5.  How much code will be changed in other ways in these modules?"
PROMPT9 ,"6.  How much code will be deleted as entire modules from existing code?"
PROMPT10,"7.  How much of the remaining existing code must be retested?"
PROMPT11,"12. What percent of the programming task is in Assembly language"
PROMPT12,"13. What percent of the new or modified code must be storage-optimized"
PROMPT13,"14. What percent of the new or modified code must be timing-optimized"
PROMPT14,"15. What percent of the total programming task is 'easy'"
PROMPT15,"16. What percent of the total programming task is 'hard'"
PROMPT16,"18. What percent of the total program requirments will be established
    and stable before design, and will not be altered before delivery"
PROMPT17,"19. What percent of the requirements are likely to change slightly before
    delivery, but will do so under baseline change control"
PROMPT18,"20. What percent of the requirements are likely to change more drastically
    before delivery, but will do so under baseline control"
PROMPT19,"30. What is the average staff experience, in years, obtained from work
    similar to that required in the task being estimated"
PROMPT20,"8.  Expected percentage of code to be developed actually delivered
    "
ANSWERS,0-90,91-99,100,-.511
PROMPT21,"9.  How many different kinds of input/output data items per 1000 lines of
    new or modified code"
ANSWERS,>80,16-80,0-15,-.548
PROMPT22,"10. Overall complexity of program and data base architecture
    "
ANSWERS,high,medium,low,-.529
PROMPT23,"11. Complexity of code logical design"
ANSWERS,high,medium,low,-.324
PROMPT24,"17. When is work to start, on the"
ANSWERS,FRD/FDD,SRD,SDD,-1.386
PROMPT25,"21. Complexity of program functional requirements"
ANSWERS,high,medium,low,-.731
PROMPT26,"22. Expected user involvement in requirements definition
    "
ANSWERS,much,some,none,-.873
PROMPT27,"23. Customer experience in application area"
ANSWERS,much,none,some,-.501
PROMPT28,"24. Customer/implementor organizational interface complexity
    "
ANSWERS,high,normal,low,-1.394
PROMPT29,"25. Interfaces with other SW development projects or organizations
    "
ANSWERS,many,few,none,-.405
PROMPT30,"26. Efficiency of implementing organization"
ANSWERS,poor,ok,good,-.693
PROMPT31,"27. Overall implementation personnel qualifications and motivation
    "
ANSWERS,low,average,high,-1.133
PROMPT32,"28. Percentage of programmers doing functional design who will also
    be doing development"
ANSWERS,<25,25-50,>50,-.938
PROMPT33,"29. Previous programmer experience with application of similar or greater
```

```
    size and complexity"
ANSWERS,minimal,average,extensive,-1.033
PROMPT34,"31. Previous experience with operational computer to be used
    "
ANSWERS,minimal,average,extensive,-.759
PROMPT35,"32. Previous experience with programming language(s) to be used
    "
ANSWERS,minimal,average,extensive,-1.149
PROMPT36,"33. Use of top-down methodology"
ANSWERS,low,medium,high,-.493
PROMPT37,"34. Use of structured programmer team concepts"
ANSWERS,low,medium,high,-.622
PROMPT38,"35. Use of Structured Programming"
ANSWERS,low,medium,high,-.577
PROMPT39,"36. Use of design and code inspections"
ANSWERS,low,QA,peer,-.432
PROMPT40,"37. Classified security environment for computer"
ANSWERS,yes,,no,-.617
PROMPT41,"38. Hardware under concurrent development"
ANSWERS,much,some,none,-.518
PROMPT42,"39. Percent of work done at primary development site
    "
ANSWERS,<70,70-90,>90,-1.147
PROMPT43,"40. Development computer access mode"
ANSWERS,remote,scheduled,demand,-.742
PROMPT44,"41. Percent of development computer access availability"
ANSWERS,<30,30-60,>60,-.718
PROMPT45,"42. Quality of SW development tools and environment"
ANSWERS,poor,ok,good,-.693
PROMPT46,"43. Maturity of system and support software"
ANSWERS,buggy,ok,good,-.693
PROMPT47,"44. Overall adverse constraints on program design
    "
ANSWERS,severe,average,minimal,-.568
PROMPT48,"45. Is the program real-time, multi-task"
ANSWERS,chiefly,some,no,-2.3
PROMPT49,"46. SW to be adaptable to multiple computer configurations or environments
    "
ANSWERS,yes,,no,-.405
PROMPT50,"47. Adaptation required to change from development to operational
    environment"
ANSWERS,much,some,minimal,-.405
```

Appendix B
DSN Software Cost Model Standard
Work Breakdown Structure Skeleton

The standard Work Breakdown Structure (WBS) skeleton is contained in a file defining a PERT plan of a typical software project. Effort and duration of each task in the WBS skeleton are given in fractions of the total W and T determined by the program.

The first record in the file is merely an identifier, announcing the WBS version number (displayed on the screen when the program starts). The remaining records are of the form

⟨task code⟩, ⟨task title⟩, ⟨who field⟩, ⟨effort⟩, ⟨duration⟩, ⟨predecessor list⟩, . ⟨date⟩

The ⟨task code⟩ and ⟨task title⟩ fields are strings, while ⟨effort⟩ and ⟨duration⟩ are numeric (fractional values). The ⟨who field⟩ is not used in this version (it appears so as to be compatible with the PERT program). The ⟨predecessor list⟩ is of the form

⟨task code⟩

or

⟨task code⟩, ⟨predecessor list⟩

The ⟨predecessor list⟩ may extend over several lines; the signal for the end of the list is the period. The ⟨date⟩ field is null; it also appears here to be compatible with the PERT program. When not null, it announces to the PERT program an actual date that the task must complete.

```
STANDARD WORK BREAKDOWN STRUCTURE,Version 1.2    30OCT80
0.,START,, 0, .922,.
1.,"Sys Plans, Reqts, & Design",, 0, .96,1.6,.
    1.1,  Define Subsys Reqts,, .024, .748,0.,.
    1.2,  FRD,, 0, .96,1.2.2,.
          1.2.1,  Write all sections,, .00672, .748,0.,.
          1.2.2,  Edit and release FRD,, .00288, .748,1.2.1,6.2,1.1,.
    6.2,  Develop prelim budget,, .0048, .748,1.1,.
    1.3,  Level B Review,, .0048, .748,1.2,.
    1.4,  Define Sys Architecture,, .0288, .748,1.3,.
    1.5,  FDD,, 0, .96,1.5.2,.
          1.5.1,  Write all sections,, .00672, .96,1.3,.
          1.5.2,  Edit and release,, .00288, .748,1.5.1,6.3,.
    6.3,  Develop Sys Impl Plan,, .0048, .748,1.4,.
    1.6,  Level C Review,, .0048, .748,1.5,.
2.,  SW Planning and Reqts,, 0, .96,2.3,.
    2.1,  Define Software Reqts,, .0384, .690,1.,.
    2.2,  SRD,, 0, .96,2.2.2,.
          2.2.1,  Write all sections,, .00672, .96,1.,.
          2.2.2,  Edit and release,, .00288, .675,2.2.1,6.4,.
    6.4,  Draft Software Impl Plan,, .0096, .675,2.1,.
    2.3,  Level D Review,, .00384, .675,2.2,.
```

DSN SOFTWARE COST MODEL STANDARD WBS SKELETON

```
3.,  SW Architecture and Design Def,, 0, .96,2.,3.4,.
     3.1,  Define SW architecture,, .048, .666,2.,.
     3.2,  SDD,, 0, .96,3.2.2,.
          3.2.1,  Write all sections,, .00576, .96,2.,.
          3.2.2,  Edit and release,, .00288, .654,3.2.1,3.3,6.5,.
     3.3,  System Interface Design,, .0346, .96,2.,.
     6.5,  Revise Impl Plan,, .0192, .655,3.1,.
     3.4,  Level E Review,, .00384, .655,3.2,3.3,6.5,.
4.,  SW Detailed Design & Prod,, 0, .96,4.6,.
     4.1,  SSD,, 0, .96,4.1.6,.
          4.1.1,  "Write Sections 1,2,3",, .0096, .96,3.,.
          4.1.2,  Write Section 4,, .0288, .654,3.,.
          4.1.3,  Write Section 5,, .0672, .96,4.1.2,.
          4.1.4,  Write Section 6,, .00672, .96,3.,.
          4.1.5,  Write Section 7,, .0144, .96,3.,.
          4.1.6,  Edit and release,, .00874, .96,4.1.1,4.1.2,4.1.3,4.1.4,4.1.5,5.4,.
     4.2,  SOM,, 0, .96,4.2.3,.
          4.2.1,  Write preliminary draft,, .0125, .654,4.1.2,.
          4.2.2,  Complete all sections,, .0144, .96,4.3,.
          4.2.3,  Edit and release,, .00605, .96,4.2.2,5.4,.
     4.3,  High-level Design Review,, .00384, .582,4.1.2,4.4.1,4.2.1,.
     4.4,  Module Production & Integration,, 0, .96,4.4.9,.
          4.4.1,  Executive and control,, .0278, .498,4.1.2,4.2.1,.
          4.4.2,  I/O Modules,, .0278, .498,4.4.1,4.3,.
          4.4.3,  Interface handlers,, .0278, .498,4.4.2,.
          4.4.4,  Function A,, .0278, .515,4.4.3,.
          4.4.5,  Function B,, .0278, .515.4.4,.
          4.4.6,  Function C,, .0269, .515,4.4.5,.
          4.4.7,  Function D,, .0269, .515,4.4.6,.
          4.4.8,  Function E,, .0269, .515,4.4.7,.
          4.4.9,  Function F,, .0269, .515,4.4.8,.
     4.5,  Special Tasks,, 0, .96,4.5.2,4.5.1,.
          4.5.1,  Support software,, .019?, .963,4.3,.
          4.5.2,  Other,, .0096, .96,4.3,.
     4.6,  Acceptance Readiness Rvw,, .00384, .582,4.2.2,4.5,4.1.1,4.1.3,4.1.4,4.1.5,4.4
     ,.
5.,  SW Test and Transfer,, 0, .96,5.6,.
     5.1,  Verification tests,, .0432, .535,4.,.
     5.2,  Contingency,, .0384, .96,4.1.1,.
     5.3,  SIT,, 0, .96,5.3.2,.
          5.3.1,  Write all sections,, .0221, .96,4.3,.
          5.3.2,  Edit and release,, .00384, .96,5.4,.
     5.4,  Acceptance tests,, .0317, .776,5.1,5.3.1,.
     5.5,  Demonstration tests,, .0336, .704,5.4,.
     5.6,  "Transfer, CDE to COE",, .0115, .704,5.5,5.3.2,5.2,4.2,4.1,.
6.,  Mgt Tasks and Milestones,, 0, .96,6.1,.
     6.1,  CDE Activities,, .0576, .96,0.,.
     6.6,  QA Audit,, .04, .922,5.4,.
```

NASA—JPL—Coml., L.A., Calif.