Deep Space Network Software Cost Estimation Model

R. C. Tausworthe
DSN Data Systems Section

This article presents a parametric software cost estimation model prepared for JPL Deep Space Network (DSN) Data Systems implementation tasks. The resource estimation model incorporates principles and data from 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 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 skeleton, which is then input into a PERT/CPM system, producing a detailed schedule and resource budget for the project being planned.

I. Introduction

The early-on estimation of required resources and schedule for the development and maintenance of software has been one of the least precise aspects of the software life cycle and, yet, an orderly and rational attempt is necessary to plan and organize an implementation effort. Such an approach implies the need for a resource and schedule model that accepts as input the technical requirements to be achieved; the enormity 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 appelation is upgraded to "Scientific WAG," or "SWAG."

The prediction of human programming behavior is a problem in estimating a series of 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, management, 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 were evaluated for use by the Jet Propulsion Laboratory in Ref. 2. None on these, by itself, seemed to the author to contain sufficient range of application and adaptability to the diverse kinds of software being produced at the Jet Propulsion Laboratory as 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 (Ref. 3) reported the analysis of 50 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 other models (General Research Corp., Doty Associates, TRW, Air Force Electronic Systems Division, Tecolote, Aerospace Corp., et al., reported in Ref. 1, and 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 and 6) presuppose a few-parameter, specific mathematical model, which is then calibrated using available industry data to provide trade-offs among effort, duration, and risk.

Several models (Ref. 7) proceed to detail resource expenditures into the various phases of activity. This TRW model additionally compensates for task difficulty and some environmental factors.

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, to be used by project planning staffs.

The software cost model described in this article 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. It utilizes the PERT estimation technique to estimate the expected size and variance of the software elements to be produced. It utilizes a modification of the Rayleigh-Norden-Putnam model to check on the confidence factors associated with the resultant resource estimates. It applies the estimated effort, staff, and duration to a standard Work Breakdown Structure (WBS) developed for JPL Deep Space Network (DSN) software tasks, and automatically produces a task budget and schedule to be used at either the initial system/subsystem planning, software implementation planning, or software maintenance planning stages of a project.

There are about 70 parameters within the model which relate to productivity, duration, staffing level, documentation, and computer resources. Another 70 parameters divide the total estimated effort among WBS subtasks; an additional 70 relate total duration to subtask durations. Subtask precedences are preset (but 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 confidence level checking, a complete scheduling early/late start/finish and float-time budget, and 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 article describe the model in more detail. The values of parameters used currently are subject to modification and refinement as further calibration and usage of the model proceeds.

Concerning accuracy: at this writing, insufficient data have 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 prediction. 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.

The model is currently being used to estimate and plan all new programming activities involving DSN Data Systems implementations.

II. Model Description

The Software Cost and Resource Estimation Model (the acronum 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, adding sufficient margin 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. A confidence factor is computed to the resources entered. 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 statement occupying one physical line in the source 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 efforts to specify, produce, document, and test a new line of code are 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 equivalent lines of code produced is then defined to be a weighted linear sum of the lines of code in each category.

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 preliminary design 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 reduced 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 clean up. 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.

Each of the size parameters is estimated by the PERT technique (Ref. 11) which produces a maximum likelihood value and variance.

The estimated value for the "equivalent lines of code" is composed of the weighted sum of the maximum likelihood estimate for each parameter, and its standard deviation is the weighted root-sum-square of the individual deviations. The weights are determined as the relative effort required for each category of code.

B. Estimation of Productivity

In this model, the productivity P is defined as total equivalent KSLEC (here denoted L) produced divided by the total staff work effort (here denoted W):

$$P = \frac{L}{W}, \frac{\text{KSLEC}}{\text{staff-month}}$$

A number of data bases (Refs. 3, 12, and 13) have shown that L and W are well correlated through a power-law relationship:

$$W = \frac{L^a}{P_1}$$

where P_1 is the average 1-KSLEC productivity rate.

The value of P_1 is primarily set by technology and environment. In fact, industry studies show that P_1 may vary by as much as 50:1 (or more!) 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 productivity cannot increase as the program size rises; however, the least-square-power-law fits to data bases yield a-values of 0.91 (IBM), 0.991 (Doty), 0.986 (University of Maryland), and 0.975 (RADC). The consistency of these figures therefore seems to indicate that utilization of tools and technology takes place on larger projects (where it counts) more than on smaller projects.

The value for a currently being used is unity. Thus, the model may tend to be somewhat conservative.

Several models have contributed to the formula by which P_1 is calculated, principally those of GRC (Ref. 14) and IBM (Ref. 3). The form of P_1 is

$$P_1 = P_0 A$$

where P_0 is a constant factor and A is a technology and environmental adjustment factor. The value of A is computed from factors judged to be significant,

$$A = A_1 A_2 \cdots A_n$$

where each factor A_i ranges between a maximum value and a minimum value for that factor, depending on the extent to which the factor is present in the software task. The ratio of A_{max} to A_{min} is currently adjusted to span a 50:1 ratio of productivity.

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

$$T = T_1 W^b$$

where T_1 is the 1-KSLEC average duration, and b is a time-factor exponent found from industry statistics. The value used in the DNS model, $T_1 = 4.8$, was adjusted to fit limited available DSN data, and $b = 0.356$ 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

$$S = \frac{W}{T} = \left(\frac{1}{T_1}\right) W^{1-b}$$

The staffing exponent, $1 - b = 0.644$, 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 maintenance user needs (Ref. 15) recommended that documentation be about 40 to 50 pages per KSLEC for programs in the 30 KSLEC vicinity. The formula used in the model for the number of pages of documentation is

$$D = D_1 L^d$$

The model currently uses $D_1 = 90$ and $d = 0.83$ to match the DSN experience and guidelines.

The documentation cost is found by a straight dollar-per-page rate; a figure of $30 per 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 25.2 CPU hours per thousand instructions.

The DSN model computer CPU resources as

$$C = C_1 L^c$$

The exponent value $c = 0.96$, 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. Confidence Level Computation

The values predicted by the SWAG model are average values based on statistics taken over many projects. The estimated values represent but one set of resources that, on the average, produce the intended success. However, effort and time can be traded (to some extent) to produce other equally valid project scenarios.

The Rayleigh-Norden-Putnam model has a time-effort relationship for checking the reasonability of resource values other than the average values produced by the SWAG. However, in order to use this model, several adaptations were felt to be indicated.

First, the basic differential equation was modified to accommodate a nonlinear "pace" factor (or learning curve). The model assumes the work equation to be of the form

$$w' = \alpha t^r (K - w)$$

in which $w = w(t)$ is the cumulative work effort up to time t, K is the total life-cycle effort, and r is the "pace-of-work" exponent.

Second, the model assumes the following parametric form of the software equation:

$$L = c_p w^p t^q$$

The factor $f = q/p$ sets the time-effort trade-off law. Putnam uses $p = 1/3$, $q = 4/3$ ($f = 4$), and such a value may well be valid for large projects in his data base. However, for the smaller projects typifying the DSN, an f factor which would require only 1.5 times the effort (three times the staff) to reduce schedule by a factor of 2 seems to be more within the DSN experience (more data is needed here).

Based on these modifications, it is possible to solve for p, q, and r, and c_p in terms of the parameters of observed average power-law relationships between L, w, and t.

$$f = 0.585$$

$$p = 0.828$$

$$q = 0.484$$

$$r = 1.81$$

$$c_p = \frac{0.0621}{A}$$

This model is used to compute confidence for any values of L, W, and T proposed for the project. The software equation is used to estimate the margin over or under the average project figures. By assuming that the statistics are log-normal (verified by the RADC data base), the confidence factor in producing L KSLEC in W staff-months effort and T months duration is

$$\text{Conf} = P\{L_{\text{act}} \leqslant L, w \leqslant W, t \leqslant T\}$$

Computation of the confidence level involves finding the optimum operating point on the software-equation curve for margin calculation, and then numeric integration of the normal probability function.

One interesting revelation to the author was that the probability of success in not expending more than W staff-months of effort *and* not requiring more than T months' duration, for the *average* SWAG estimate case, is only about 25 percent. Moreover, a significant amount of bias in W and T is required to raise the confidence to 50 percent.

Management should not despair, however. What the confidence limit indicates for average projects is that one out of four will go okay; the others will require some form of management intervention, in the form of schedule or resource extension.

III. Typical Operation of the Model

Blank forms and/or CRT prompted inputs are used to specify the parameters needed by the cost model program. Outputs can be selected and include a WBS task file, PERT plan, and schedule. The WBS task file can be edited to modify task titles, precedences, allocated resources, or durations; in addition, new tasks may be entered and actual completion or need dates may be affixed, so that the PERT and schedule portions of the program form a project detailed planning and control tool. Typical inputs and outputs are shown in Appendix A.

IV. Summary and Conclusion

The Software Cost Model reported here is the first of a series of planned refinements. As the model is used and as performance data are collected, no doubt changes will be made: adjustments of parameters, alteration of formulas, modifications of formats, new input data types, and additional kinds of outputs. Extensions currently envisioned are the automated transfer of the WBS data base generated by the model into the project control system currently being used in DSN Data Systems implementation projects, and the refinement of the model to include nonlinear scaling of overall effort and duration into individual task requirements.

If the model, even now, seems complex, then 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 made by any model.

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

References

1. *Quantitive Software Models,* Report SRR-1, Data and Analysis Center for Software, RADC, Griffiss Air Force Base, New York, 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. Freburger, K., and Basili, V. R., *The Software Engineering Laboratory: Relationship Equations*, Technical Report TR-764, SEL-3, NSG-5123, University of Maryland, Computer Science Center, College Park, Md., May 1979.
5. Norden, P. V., "Project Life Cycle Modeling," *Software Life-Cycle Management Workshop*, United States Army Computer Systems Command, pp. 217–306, August 1977.
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, pp. 307–312, August 1977.
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, Morestown, 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. The author is unable to trace the definitive reference for this work. The usage cited is exposed in Putnam, L. H., "Example of an Early Sizing, Cost and Schedule Estimate for an Application Software System," *COMPSAC '78*, November 13–16, 1978.

12. 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.

13. Nelson, R., *Software Data Collection and Analysis, Partial Report*, Rome Air Force Development Center, Data and Analysis Center for Software, Griffiss Air Force Base, New York, Sept. 1978.

14. 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.

15. Tausworthe, R. C., *Preparation Guide for Class B Software Specification Documents*, External Publication 79-56, Jet Propulsion Laboratory, Pasadena, Calif., October 1, 1979.

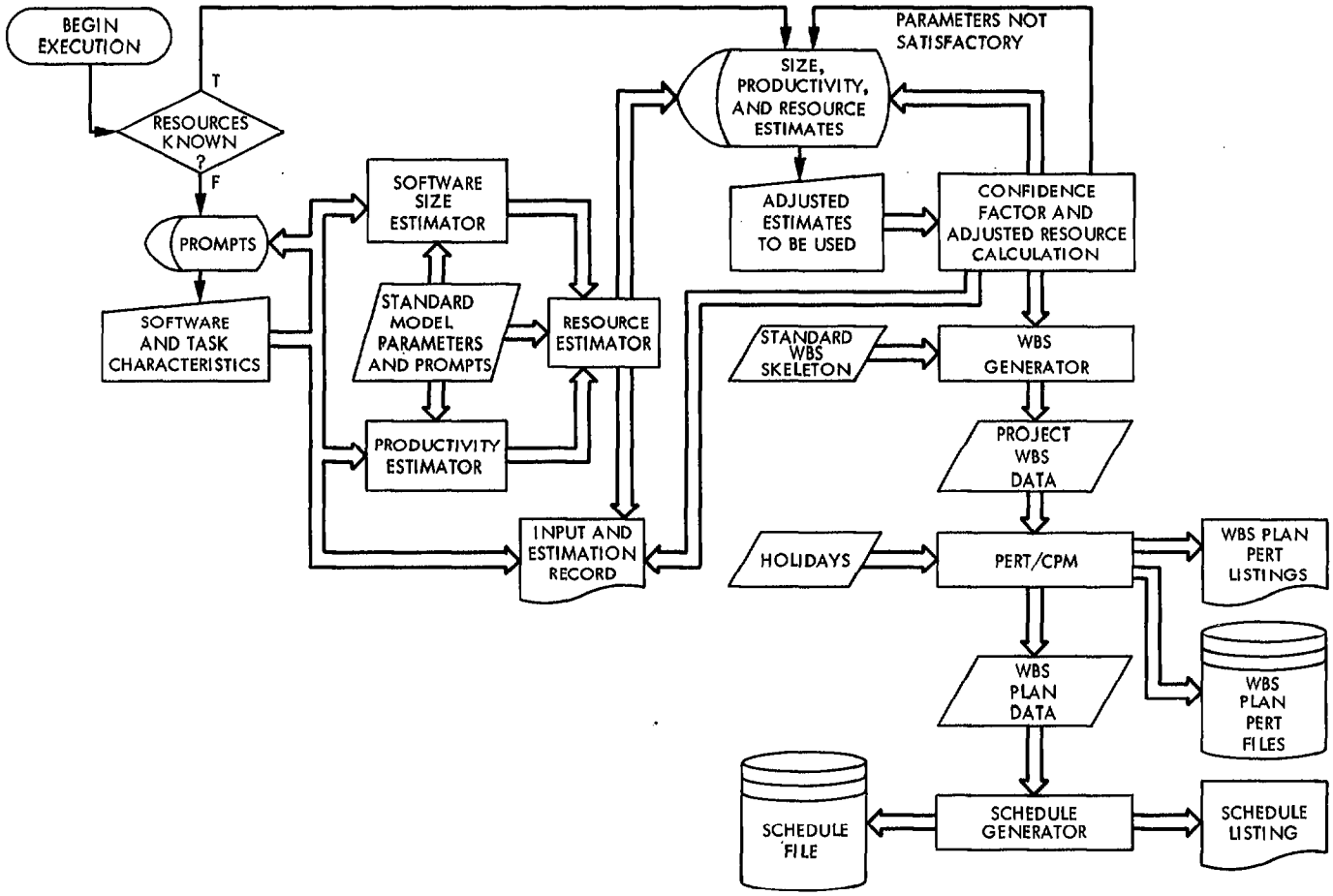

Fig. 1. DSN Software cost estimation model data flow diagram

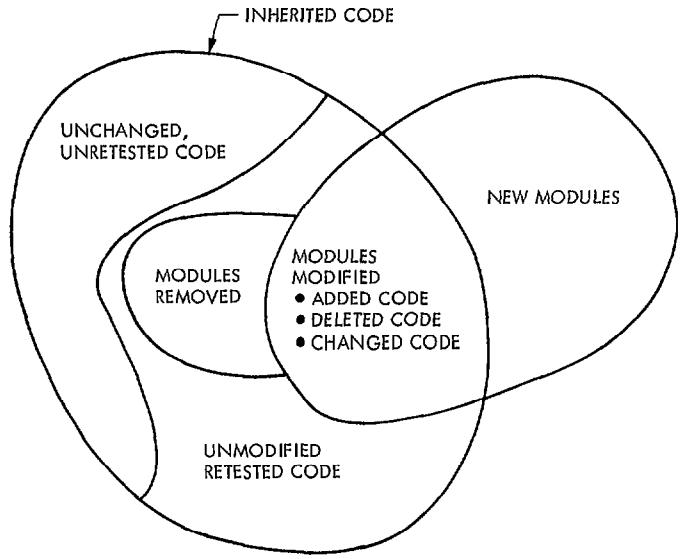

Fig. 2. Software implementation tasks related to new and inherited code activities

Appendix
Example of Operation

This Appendix contains a sample of the sequence of inputs and outputs from the Deep Space Network Software Cost Estimation Model.

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. A-1. 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. A-1 (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)?	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. A-1 (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
 19.0            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. A-2. Output of cost model using the parameters from the previous figure

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 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	20NOV80	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	SOM		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. A-3. Full form 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	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
* 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		2	271	18DEC81	271	18DEC81	273	22DEC81	273	22DEC81	0
*5.	SW Test and Transfer		0	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	18MAR82	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. A-3 (contd)

PAGE 1

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. A-4. Short form PERT/CPM work breakdown structure schedule data

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
* 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. A-4 (contd)

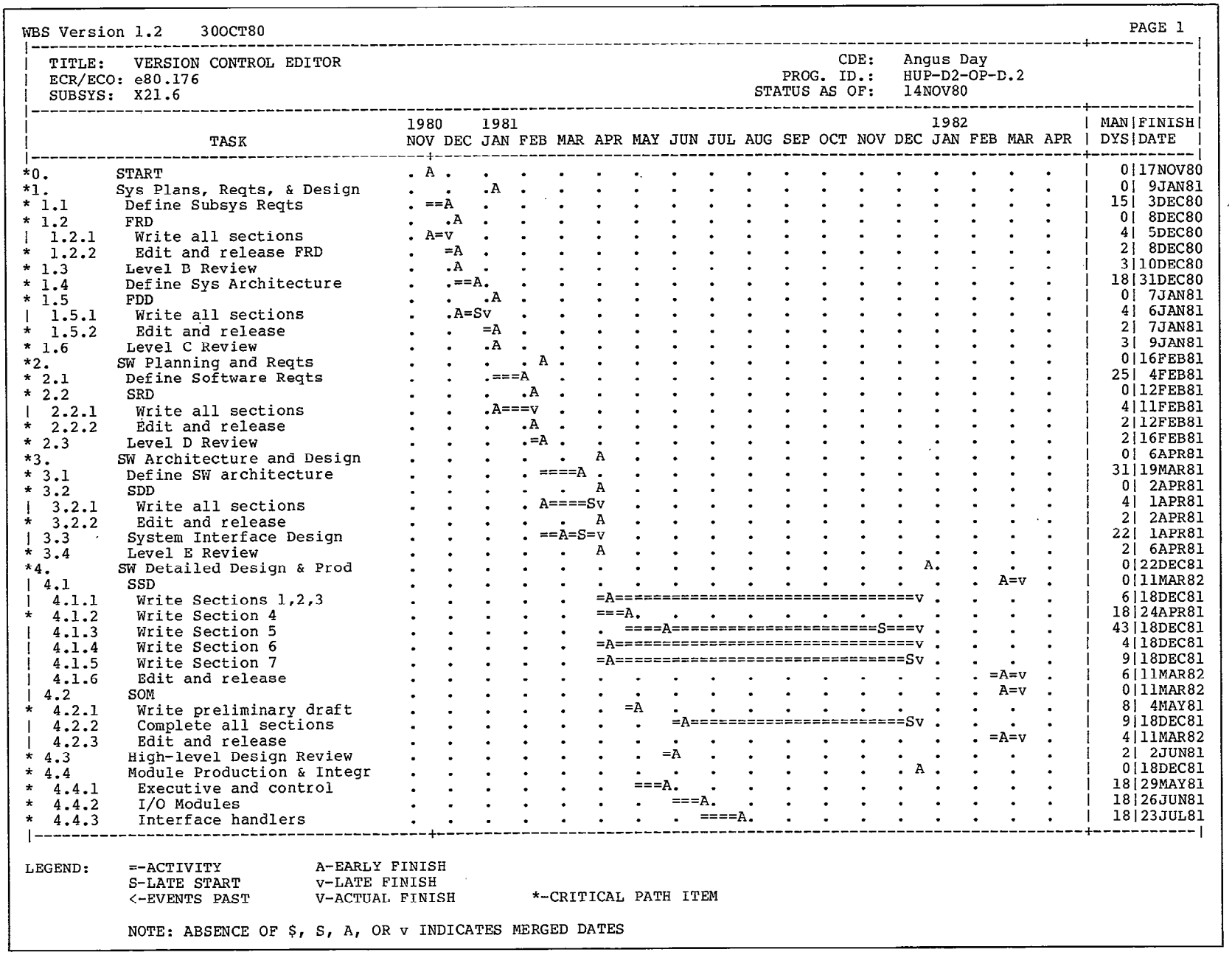

Fig. A-5. Gantt chart output of the software cost model

WBS Version 1.2 30OCT80 PAGE 2

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

	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
*	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	11FEB81
*	6.5 Revise Impl Plan	==A																		12	1APR81
	6.6 QA Audit																=S=Av			26	18MAR82
*	FINISH																		A	0	18MAR82

WBS Version 1.2 30OCT80 PAGE 3

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

	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	=======A																		51	9JAN81
*	2. SW Planning and Reqts	.=====A																		33	16FEB81
*	3. SW Architecture and Design	======A																		61	6APR81
*	4. SW Detailed Design & Prod	==========A=v																		287	11MAR82
*	5. SW Test and Transfer	.==========S==========A																		118	18MAR82
	6. Mgt Tasks and Milestones	==S==========Av																		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. A-5 (contd)