A Generic Life-Cycle Cost Model for an Embedded Controller

by Kevin R. Parker

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science
at Texas Tech University

December 1991

CONTENTS

ABSTRACT

The need for a generic life cycle cost model surfaced during the course of research work which was being conducted by the Department of Computer Science at Texas Tech University for FSI International, Inc. The purpose of the research was to develop a requirements specification for a Module Level Controller, which controls "all operations of the semiconductor processing equipment" [FSII90, 2]. A conference involving research staff from the Department of Computer Science and an FSI management team revealed that no accurate means of estimating the cost of a hardware/software project from its inception to its completion was being utilized by FSI. Costs associated with activities such as research, design, testing, and support were viewed separately and addressed at various stages in the system life cycle, not viewed on an integrated basis [BLAN90]. Furthermore, preliminary investigation by the staff at Texas Tech concluded that none of the available life cycle costing models accurately reflected the life cycle of an embedded controller project conducted by FSI.

FIGURES

CHAPTER I: INTRODUCTION

Life cycle costing is defined as "a method of calculating the total cost of ownership over the life span of the asset" [BROW85, 1]. A life cycle is a series of phases used to coordinate and control the development, production, maintenance, and retirement of a system. Each phase consists of both a set of activities to accomplish the goals of the phase, and the associated products resulting from those activities [NASA89]. The life cycle cost of a system is determined by identifying the functions that constitute each phase of the life cycle, determining the cost of those functions, applying those costs on a year-to-year basis, and ultimately accumulating those costs over the entire life span of the system [BLAN78]. Life cycle cost models can be used as a tool in the "systematic analytical process of evaluating various alternative courses of action with the objective of choosing the best way to employ scarce resources" [BLAN90, 505].

The goal of this research project was the development of a generic life cycle cost model for an embedded controller. The model was structured to represent the life cycle of FSI's embedded controller projects. The steps which were taken to accomplish this goal were:

(1) the development of a cost breakdown structure to serve as a basis for the proposed life cycle cost model;

(2) the investigation of existing life cycle cost tools to evaluate their ability to calculate either the overall life cycle cost or the costs associated with a subsystem of the proposed model.

One of the initial steps in developing an appropriate life cycle cost model was the design of a cost breakdown structure (CBS). A cost breakdown structure provides a framework for defining life cycle costs and is the basic mechanism used for cost allocation. The CBS reflects the many different types of activities which make up the life cycle of a system, and life cycle costing is a compilation of a variety of cost factors resulting from the categories which make up the CBS. The cost breakdown structure developed during the course of this research includes all functions which are performed during the course of an embedded controller project such as those undertaken by FSI. In order to provide

clarification, a table accompanies the CBS to supply a description of each cost category included in the CBS, as well as the quantitative relationships used to determine costs.

After establishing a realistic cost breakdown structure, various life cycle cost models were investigated. In many applications, the life cycle cost model lends itself to being partitioned into a series of subsystems. For example, the life cycle of the application under consideration involves alternative approaches, and thus the analysis was made more manageable through the use of individual models for analysis at the subsystem or alternative level. In addition, some of the existing life cycle cost tools appeared to be suitable for calculating the overall life cycle cost or the costs of a single subsystem. If such a tool could be utilized as is or with slight modifications, it would have been more feasible to adopt that tool rather than to develop a new tool [BLAN78].

The scope of the tool selection process was restricted to those tools which lend themselves to the life cycle costing of an embedded controller. This eliminated special purpose tools such as those intended for avionics or other specific applications. In addition, the cost breakdown structure devised in the earlier phase of this project provided a guide for tool selection, leading to the selection of those tools which offered a way of modeling a particular subsystem. The subsystems for which tools were available were the software subsystem and the hardware subsystem.

The list of tools selected for evaluation included:

(a) *Life-Cycle Cost Calculator* (LCCC) developed at Virginia Polytechnic Institute,

(b) *Software Cost Model* (SOFTCOST) developed by the National Aeronautics and Space Administration,

(c) *Before You Leap* (BYL) developed by the Gordon Group,

(d) *Life Cycle Cost Model, Version H* (LCCH) developed at Wright-Patterson Air Force Base, and

(e) *Programmed Review of Information for Costing and Estimation--Hardware* (PRICE "H") developed by RCA.

The basis for the evaluation criteria was the degree of a tool's applicability to a subsystem. Any tool not directly applicable to either the hardware or software subsystem was not considered. Tools which could not be acquired could not be thoroughly evaluated

and thus were dropped from consideration. The focus of the research was the selection of the tools with the greatest applicability and the determination of what modifications were necessary so that they more accurately reflected actual costs generated by a sample FSI project.

Need for Life Cycle Costing

The need for life cycle costing exists because too often, when budgeting, contracting, or evaluating a system, only the procurement costs or the design and production costs are considered. Total system cost is not apparent and hidden costs, such as those costs associated with maintenance and distribution, inflate that total cost well beyond the obvious costs. The total life cycle cost encompasses all present and future costs associated with research and development, production, installation and checkout, maintenance, and ultimate system phase-out [BLAN81].

An analogy can be drawn between the cost visibility problem and an iceberg, as shown in Figure 1.1 [BLAN90]. Life cycle costing models provide a tool to address not only the acquisition costs, but the underlying costs as well.

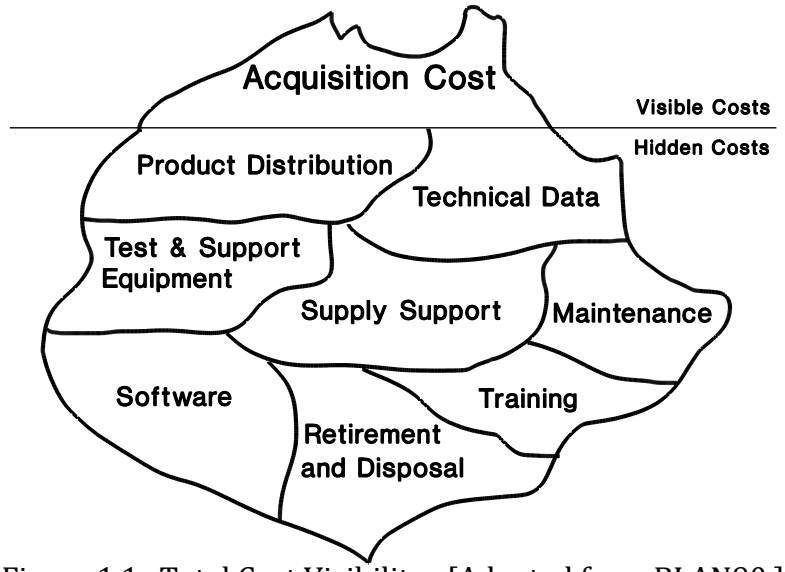


Figure 1.1. Total Cost Visibility. [Adapted from BLAN90.]

Life cycle cost analysis provides multiple benefits in addition to the obvious advantage of providing a more accurate view of the overall cost of a system. Life cycle cost

analysis can be used in the evaluation of alternative system design configurations such as hardware versus software approaches, component selection and standardization, reliability versus maintainability, levels of repair versus discard decisions, diagnostic routines, built-in test versus external test, and so forth. It can be used in the evaluation of alternative system maintenance concepts, logistic support policies, and procurement sources [BLAN90]. It is anticipated that its major attraction to a corporation such as FSI will be its usefulness in evaluating alternative sources for both hardware and software products, i.e., assisting in the decision of whether to design and produce various components in-house or to procure those components from a vendor.

Life cycle costing encompasses multiple disciplines, requiring knowledge of accounting, budgeting, computer science, contracting, engineering, financial estimating, finance, logistics planning, maintainability engineering, management, manufacturing engineering, quality control, reliability, and statistical analysis [SELD79; DHIL89]. While expertise in all of these areas is not essential, a basic familiarity with each is required for the design of an accurate life cycle cost model.

Historical Perspective

Life cycle costing has a long history of use throughout industry. Used as a tool for capital budgeting, life cycle costing has been applied to a wide variety of projects [BROW85]. It dates back to 1887, when civilian engineer A. M. Wellington analyzed the trade-offs between the initial cost of railway location and subsequent operating costs [SHUP80].

Life cycle cost analysis has long been utilized by the United States Department of Defense. Applied to virtually every new weapon system proposed or under development [BROW85], life cycle costing was introduced into the Department of Defense by "the aircraft industry, where cost optimizing techniques evolved as part of the competitive effort to sell commercial and military aircraft" [KNUS81,2]. Its importance to the defense industry was emphasized by the finding that operation and support costs for a typical weapon system accounts for as much as seventy-five percent of the total cost. In addition,

operation and support costs over a ten- to fifteen-year life span often equal or exceed the acquisition cost, thus requiring a larger portion of the budget and reducing the funds available for other projects [GUPT83]. In 1970, the Department of Defense released the "Life Cycle Costing Procurement Guide" (LCC-1) and the "Life Cycle Costing Casebook" (LCC-2) to assist in implementing the concept of life cycle costing in the acquisition of equipment below the level of complete systems [TAYL74]. In 1971, the Department of Defense issued Directive 5000.1: Acquisition of Major Defense Systems, which established the requirement for life cycle costing procurement for major defense systems acquisitions [DHIL89]. In 1973, the "Life Cycle Costing Guide for Systems Acquisitions" (LCC-3) was released [TAYL74]. Since the release of those documents, both the defense and aerospace industries design their products in terms of life cycle cost objectives [BROW85].

Other government entities require the use of life cycle costing for a variety of purposes, not only because of its contributions to cost effectiveness, but also for energy conservation programs. Since 1974, many states have passed legislation requiring the incorporation of life cycle cost analysis into the planning, design, and construction of state buildings [DHIL89]. The State Energy Conservation Program, which was established by the Energy Policy and Conservation Act of 1975, made it mandatory that states develop energy-efficient procurement procedures in order to qualify for program funds. Life cycle costing is one method of validating the energy efficiency of such procurement procedures [BROW85]. The National Energy Conservation Policy Act of 1978 requires that every new federal building be life cycle cost effective [DHIL89].

The health care field also utilizes life cycle costing. A study released by the General Accounting Office in 1972 revealed that "the operating costs of a hospital in its first three to five years of existence typically exceeded the entire cost of construction" [BROW85, 2]. That study served to stimulate interest in cost effective practices throughout the health care industries.

Several other industries make use of life cycle costing. Models exist for use in the utilities industries, petroleum industry, manufacturing industry, chemical plants, radar systems, appliances, and electronic equipment [DHIL89].

CHAPTER II - MODEL DEVELOPMENT

Life cycle cost models can be classified into a variety of categories. The classifications encountered most often in literature include: financial, analytical/heuristic/ conceptual, parametric/accounting, and phased.

Financially-oriented models are discussed in both Shupe and Brown [SHUP80; BROW85]. This classification includes such models as the present worth method, annual worth method, benefit/cost method, and return-on-investment method. Each of these methods is based on the discount rate, which is the minimum acceptable rate of return on an investment. The present worth method uses the discount rate to convert future expenditures for each alternative to their equivalent present values, and then compares the present worth of each alternative. The preferred alternative is that with the highest present worth. The annual worth method uses the discount rate to convert the cash flow from each alternative into equivalent uniform annual amounts, and then compares each amount. The best economic choice is the one with the greatest equivalent annual worth. The benefit/cost method separates costs from benefits, and using the discount rate converts the cash flows to their equivalent annual (or present) values. The equivalent benefits are compared to the equivalent costs for each alternative using the ratio of benefits to costs. If the benefit/cost ratio is greater than one, then the alternative is economically sound. The return-on-investment method determines a percentage rate of return. If the rate of return for a particular alternative exceeds the discount rate, then that alternative is acceptable.

Both Dhillon and Gupta prefer to classify life cycle cost models into analytical, heuristic, or conceptual categories [DHIL89; GUPT83]. Analytical models are based on mathematical relationships designed to describe certain aspects of the system. The subcategories of analytical models are design trade models, which minimize cost to meet a specified design parameter such as reliability; total cost models, which minimize the total life cycle cost of a system while maximizing its effectiveness; and logistic support models, which determine costs of alternative support plans. Heuristic models are not as general as analytical models and are usually suitable only for the specific situation for which they

were developed. Conceptual models are based on the hypothesized relationships of variables given in a qualitative fashion and are extremely flexible, accommodating a wide range of systems.

Knust and Priest classify models as parametric models and accounting models [KNUS81; PRIE88]. Parametric models are based on a form of regression analysis in which cost experience and performance level of past similar systems provide a baseline for estimating the cost of future systems based on their projected level of performance [PRIE88]. Parametric models are most useful early in the program before many of the design decisions have been made. This type of model employs a forecasting approach in which a product's cost is regressed against physical or performance parameters of past products similar to the new system. The relationships between historical data and certain design parameters available during preliminary design are analyzed and comparisons are made using regression analysis [PRIE88]. Those parameters are independent variables chosen from among the set of a product's characteristics believed to be estimable with reasonable accuracy. Predicted values for the relevant parameters of the new system are then substituted into an equation whose coefficients are derived from the historical data [PRIE88].

Parametric cost models are used to gain a rapid assessment of the overall life cycle costs of a proposed system. The purpose of such models is to identify relative magnitudes of cost. The results are used to disqualify proposal alternatives that generate cost extremes. This allows the company to concentrate on those alternatives which appear to be most cost effective [KNUS81].

Accounting models use detailed algorithms to directly relate design decisions to their effects on cost [PRIE88]. Factors related to design, such as part quality, producibility, redundancy, reliability, and maintenance concepts, can be taken into account by the engineer. As the design progresses and becomes more defined, cost trade-offs become more exact. The major emphasis of accounting models is on design trade-offs [PRIE88].

Accounting models are concerned with far greater detail than parametric models. This detail involves both the financial planning aspect and the cost optimization of the selected alternative [KNUS81]. This method has more potential for prediction accuracy and design trade-offs than parametric models. It also provides more detailed visibility of

the various data sensitivities [PRIE88].

Phased models include both a two phase model and a four phase model. The models presented by Taylor and Priest are partitioned into acquisition costs and sustaining costs [TAYL74; PRIE88]. Acquisition costs are those costs associated with research, design, production, and test, or with procurement. Sustaining costs involve such items as maintenance, training, and other operating expenses.

Seldon and Blanchard present a life cycle cost model which is partitioned into four phases [SELD79; BLAN78; BLAN81; BLAN90]. Those four phases are the research and development phase, the production phase, the maintenance phase, and the retirement and disposal phase. Blanchard explains each of the partitions as follows:

1. *Research and development (R&D) cost*--the cost of feasibility studies; system analysis; detail design and development, fabrication, assembly, and test of engineering models; initial system test and evaluation; and associated documentation.

2. *Production and construction cost*--the cost of fabrication, assembly, and test of operational systems (production models); operation and maintenance of the production capability; and associated *initial* logistic support requirements (e.g., test and support equipment development, spare/repair parts provisioning, technical data development, training, entry of items into the inventory, facility construction, etc.).

3. *Operation and maintenance cost*--the cost of sustaining operation, personnel and maintenance support, spare/repair parts and related inventories, test and support equipment maintenance, transportation and handling, facilities, modifications and technical data changes, and so on.

4. *System retirement/phase-out cost*--the cost of phasing the system out of the inventory due to obsolescence or wearout, and subsequent equipment item recycling and reclamation as appropriate. [BLAN81, 21]

Model Design

The process of developing a life cycle cost model requires that the designer first select the type of model which can most closely replicate the life cycle of the system under consideration. The analyst must next generate a comprehensive list of the factors which are pertinent to the system life cycle being modeled. Next, the cost breakdown structure

must be established. The cost breakdown structure illustrates "the numerous and varied segments of cost that are combined to provide the total system/product cost" [BLAN78, 191]. Seldon notes that a mathematical structure, such as the cost breakdown structure, describes how the life cycle cost elements are summed, and is an accounting model [SELD79]. It is further noted that the "addition of the cost-estimating relations for each element to the mathematical structure yields an estimating model" [SELD79, 158].

After considering the exposition of life cycle cost model development in Seldon, as well as the advantages of the accounting model over the parametric model as cited in Priest and Knust, it was decided that the appropriate model would be an accounting model [SELD79; PRIE88; KNUS81]. Rather than rejecting outright the idea of parametric models, it was decided that parametric methods would be more appropriately utilized to determine valid cost-estimating relationships during the cost determination phase of life cycle costing.

Extensive research evaluating model classifications led to the conclusion that the four phase model would serve as the most appropriate guide for the development of a life cycle cost model for an embedded controller. The four phase model is an accounting model, and can also be classified as an analytical model, more specifically a total cost model.

Many factors influence both life cycle cost and system effectiveness. Reiche writes that:

> Probably one of the most controversial and difficult subjects of LCC studies is the type of factors which one should include in the evaluation, analysis, estimation, etc. The reason for the controversy lies in the fact that almost everyone has a different view of what constitutes important factors. [REIC80, 5]

Some of the major factors which come into play in the life cycle of an embedded controller include the hardware/software partition and the make/buy partition. One of the key tasks in the design of any microprocessor-based system is the determination of which functions are best performed by hardware and which are best performed by software. The purpose of the hardware/software trade-off determination is to minimize the cost of the system. Hardware implementations are generally faster but more expensive in the production phase, while software implementations are slower and more costly in the development phase, but more flexible in terms of modifications to the design [LAM88]. The make/buy partition involves which components should be designed and produced in-house

and which should be contracted out or procured from a vendor. Components designed and produced in-house require more development time, in-house expertise, and production costs, but have the advantage of being designed specifically for the task at hand. Procured components eliminate development and production costs, but if contracted out may be considerably more expensive, and if procured off-the-shelf may not be as application-specific as desired. These factors, as well as others such as integration and testing, distribution, and system support costs, must all be considered and included in model design.

The cost breakdown structure provides a mathematical framework which can be used to calculate overall life cycle costs. It serves to delineate and categorize all significant high cost contributors, and outlines the quantitative relationships which are used to derive the overall life cycle cost. Blanchard explains that:

> The cost breakdown structure... [consists] of the various elements of cost that when combined, represent total life cycle cost. The categories identified indicate cost collection points which can be summarized upward into broader categories and/or can be collected for different program functions or system elements. The intent is to incorporate a high degree of flexibility in order to provide the necessary visibility for cost allocation, cost measurement, and cost control. [BLAN78, 191]

The cost breakdown structure must be accompanied by an explanation of each cost category, the symbology used in the CBS, and the quantitative relationships used to derive the total cost.

Life cycle costing constitutes a compilation of a variety of cost factors representing many different types of activities [BLAN78]. When developing the cost breakdown structure, it sometimes becomes apparent that the model would be clearer and more meaningful, as well as more usable, if it was partitioned into subsystems. The life cycle of an embedded controller consists of various optional approaches and the associated life cycle cost model under development lends itself to the use of individual models for analysis at the alternative level.

If the life cycle cost model is composed of various sub-routines which address the major areas where high cost visibility is desired, then the system can be evaluated in terms of total life cycle cost as well as the various individual segments of cost [BLAN78].

Individual models can be utilized for analyses at the subsystem or alternative level and still provide the necessary data for the life cycle cost model [BLAN78]. Thus, partitioning the model into subsystems provides flexibility such that the analyst may evaluate either the system as an entity or any major segment thereof [BLAN78].

Before developing a life cycle cost model in its entirety from the ground up, existing tools should be considered. Blanchard notes that:

> In selecting a model for evaluation purposes, it is desirable to first investigate those tools that are currently available. If a model already exists and is proven, then it may be feasible to adopt that model. [BLAN78, 87]

Models which are capable of calculating life cycle costs for a specific segment, or even the entire entity, may have already been developed and tested. If an existing model can be utilized, much time and effort can saved. In the case of the embedded controller, where both hardware and software components are essential, potential models exist and will be considered for the hardware subsystem as well as the software subsystem.

The Life Cycle Cost Model for an Embedded Controller

The life cycle cost model is presented in the form of a cost breakdown structure in this chapter, accompanied by a table in the Appendix which explains each cost category, the symbology used, and the quantitative relationships used to derive costs. As noted earlier, the mathematical structure embodied in a cost breakdown structure details how the various categories are summed and serves as an accounting model [SELD79]. The cost breakdown structure is illustrated on the following pages. It is presented in both diagram and outline form. The overall cost breakdown structure (Figure 2.1) is partitioned into research and development (Figure 2.2), production (Figure 2.3), maintenance and support (Figure 2.4), and retirement and disposal (Figure 2.5).

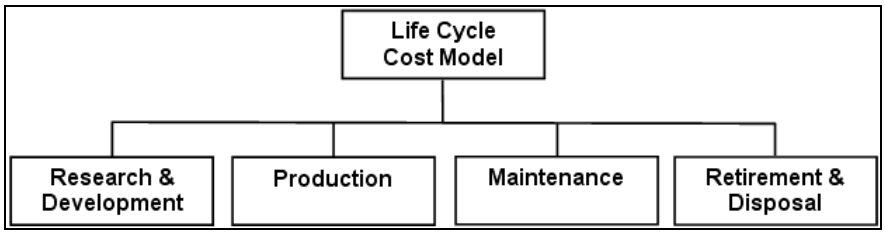


Figure 2.1. Overview of Life Cycle Cost Model.

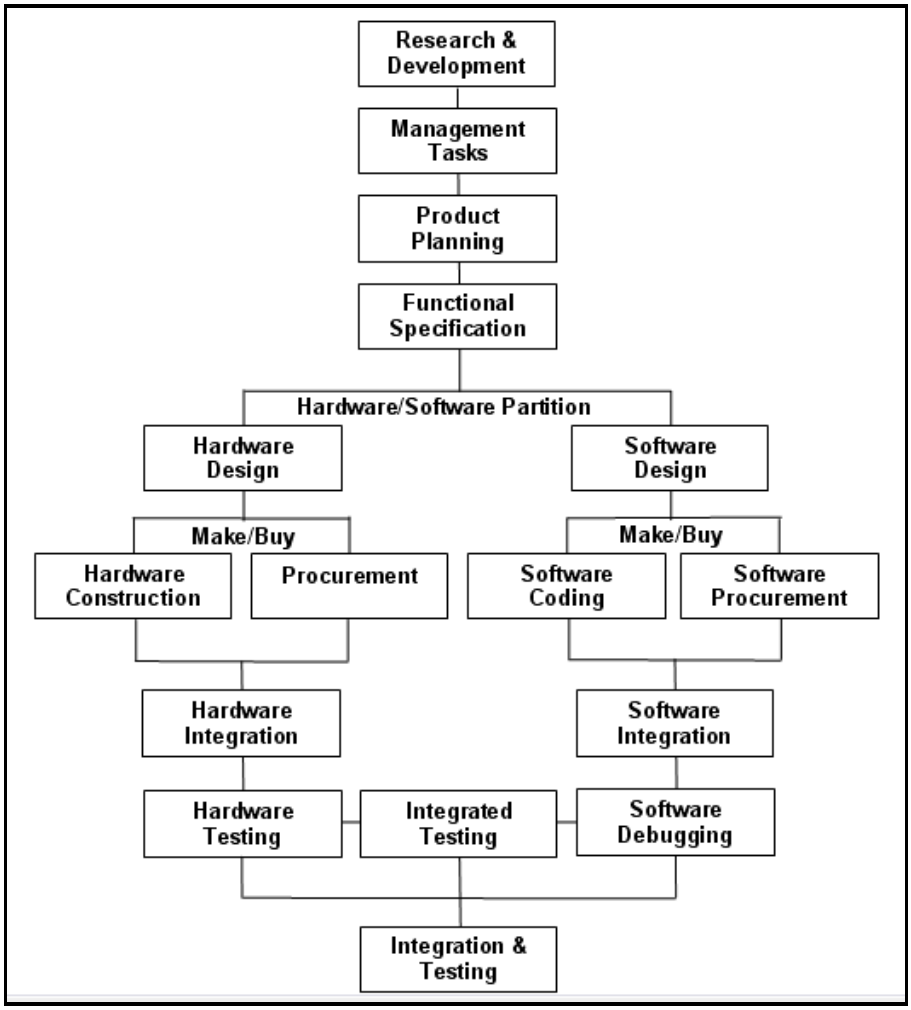


Figure 2.2. Detailed view of the R&D Branch of the Life Cycle Cost Model.

I. Research and Development Costs

A. System/Product Management

B. Product Planning
1. Market Analysis
2. Feasibility Studies
3. Program Planning

C. Functional Specification of Microprocessor System
1. System Engineering
2. Conceptual Design
3. Preliminary Design

D. Hardware/Software Partition
1. Hardware
a) Hardware Design
(1) Detailed Design
(2) Design Support
(3) Design Review

b) Make/Buy Partition
(1) Hardware construction in-house
(a) Prototype Fabrication
(2) Hardware procured from vendor
(a) Evaluation
i) Product Evaluation
ii) Vendor Evaluation
(b) Unit Costs (Vendor price)

c) Hardware Integration
(1) Integration of components
(2) Design Documentation
(a) Compile In-house Documentation
(b) Compile Vendor-provided Documentation
(c) Establish Documentation Library

d) Hardware Test and Evaluation
(1) Test Planning
(2) Test and Evaluation
(3) Test Data/Reports

2. Software
a) Software Design
(1) Software Design
(2) Design Support
(3) Design Review
(4) Software Specifications

b) Make/Buy Partition
(1) Software developed in-house
(a) Software Engineering Support Tools
(b) Software Development Tools
(c) Implementation
(d) Integration
(e) Testing
(f) Documentation
(g) Verification and Validation
(h) Rework (Debugging)
(i) Quality Assurance
(2) Software procured from vendor
(a) Evaluation
i) Vendor Evaluation
ii) Product Evaluation
(b) Vendor Fees
i) Procurement Costs
ii) Site License Agreement
iii) Networking Capabilities
(c) Training costs

c) Software Integration
(1) Integrate procured/produced subsystems
(2) Software Documentation
(a) Compile In-house Documentation
(b) Compile Vendor-provided Documentation
(c) Establish Documentation Library

d) Software Test and Evaluation
(1) Test Planning
(2) Test and Evaluation/System Testing
(3) Verification and Validation
(4) Test Data/Reports

E. Integrated Testing
1. Test Planning
2. Test and Evaluation/System Testing
3. Verification and Validation
4. Test Data/Reports

F. Integration and Testing
1. Integration of HW/SW
2. Integration and Archival of All Documentation
3. System/Product Test and Evaluation
a) Test Planning
b) Test and Evaluation
c) Test Data/Reports

Figure 2.2. (continued)

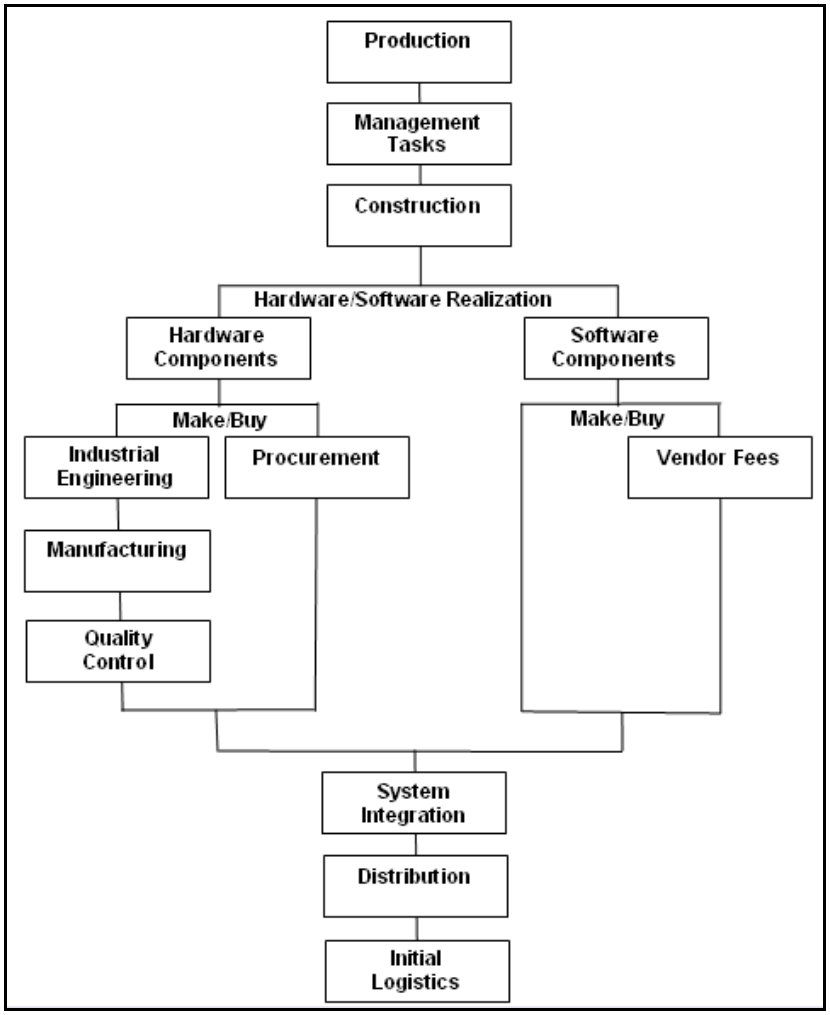


Figure 2.3. Detailed view of the Production Branch of the Life Cycle Cost Model.

II. Production Costs

A. Production/Construction Management

B. Construction
1. Production Facilities
2. Test Facilities
3. Maintenance Facilities (Acquisition)
4. Inventory Warehouses

C. System Production
1. Hardware Components
a) Make/Buy Partition
(1) Hardware Produced in-house
(a) Industrial Eng./Operations Analysis
i) Manufacturing Engineering
ii) Methods Engineering
iii) Production Control
(b) Manufacturing
i) Recurring manufacturing costs
(i) Fabrication
(ii) Subassembly/Assembly
(iii) Material (Inventories)
(iv) Inspection and Test
(v) Manufacturing Rework
ii) Nonrecurring manufacturing costs
(i) Tooling/Test Equipment
(c) Quality Control
(2) Hardware Procured from vendor
(a) Procurement
i) Unit Cost
ii) Maintenance Contract
b) Hardware Integration
(1) Integration of components
(2) Inspection and Test
(3) Quality Control

2. Software Components
a) Make/Buy Partition
(1) Produced In-House
(2) Procured from vendor
(a) Vendor Fees
i) Procurement
ii) Site License Agreement
iii) Networking Capabilities
b) Software Integration
3. System Integration

D. System Documentation Printing

E. System/Product Distribution
1. Marketing and Sales
2. Packaging
3. Transportation and Handling
4. Warehousing (Product)

F. Initial Logistics Support
1. Program Management
2. Provisioning
3. Initial Spare/Repair Parts
4. Initial Inventory Management
5. Technical Data Preparation
6. Initial Training and Training Equipment
7. Test and Support Equipment Acquisition
8. Initial Transportation of Logistics Support

Figure 2.3. (continued)

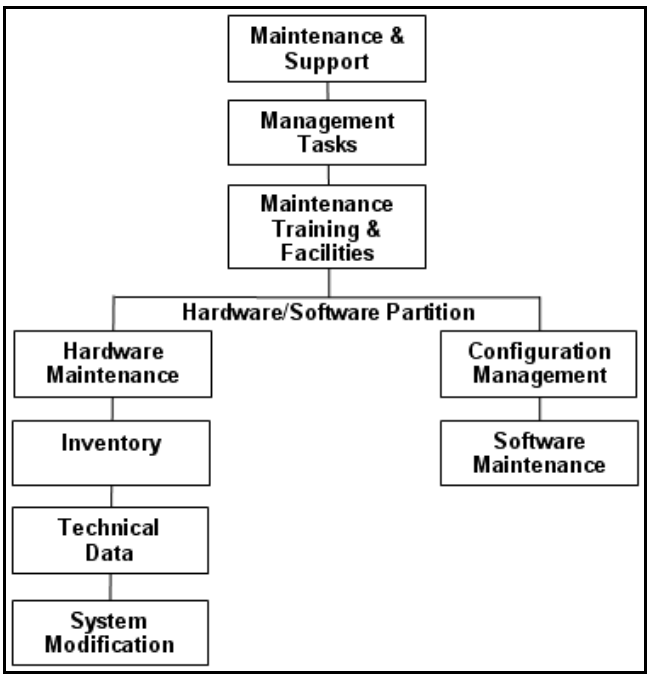


Figure 2.4. Detailed view of the Maintenance and Support Branch of the Life Cycle Cost Model.

III. Maintenance Support Cost
- A. System/Product Life Cycle Management
- B. Maintenance Training and Facilities
  - 1. Maintenance Training
    - a) Maintenance Training
    - b) Training Facilities (upkeep)
    - c) Training Data
  - 2. Maintenance Facilities (Upkeep)
- C. System Maintenance
  - 1. Hardware Maintenance
    - a) Hardware Maintenance
      - (1) Field Maintenance
      - (2) Factory Maintenance
      - (3) Test and Support Equipment
    - b) Inventory - Spares and Material Support
      - (1) Spare/Repair Parts (For Maintenance)
      - (2) Storage and Handling
      - (3) Inventory Management
    - c) Technical Data
    - d) System/Product Modifications
  - 2. Software Maintenance
    - a) Configuration Management
    - b) Software Maintenance
      - (1) Perfective Maintenance (Upgrades)
      - (2) Corrective Maintenance
      - (3) Adaptive Maintenance (Modifications)
      - (4) Debugging and Diagnostic Equipment

Figure 2.4. (continued)

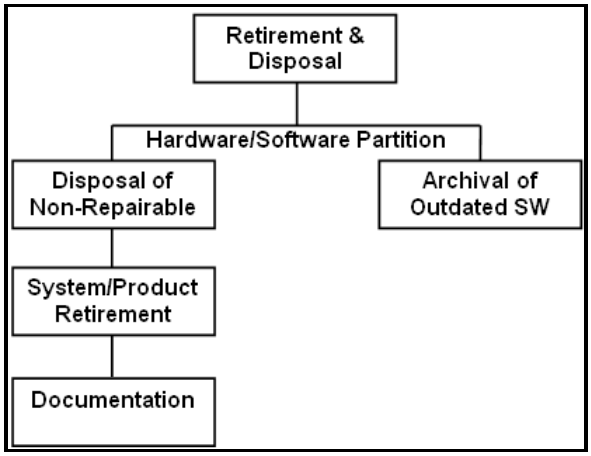


Figure 2.5. Detailed view of the Retirement and Disposal Branch of the Life Cycle Cost Model.

IV. Retirement and Disposal A. Hardware 1. Disposal of Non-Repairable Elements 2. System/Product Retirement a) Personnel b) Support Equipment c) Transportation and Handling	3. Documentation B. Software 1. Archive Outdated Software Upon Upgrade Release

Figure 2.5. (continued)

This cost breakdown structure is intended to reflect all of the cost categories associated with the life cycle of an embedded controller. To get a complete view of each phase, each of the subsections must be considered. For example, the research and development phase must take into account management tasks, product planning, functional specifications, the hardware/software partition, integrated testing, and integration and testing. This cost breakdown structure is loosely based on a general cost breakdown structure presented by Blanchard [BLAN78].

CHAPTER III - COST DETERMINATION

After the cost breakdown structure has been completed, it is necessary to generate the cost data which will serve as input to the quantitative relationships specified for the various cost categories. "The estimation of future costs is probably one of the most difficult tasks in the accomplishment of a life cycle cost analysis" [BLAN78, 35]. Sources of cost estimation data include historical data, bids and proposals from suppliers, forecasting accomplished through the use of parametric methods, and practical knowledge of engineering techniques based on experience [BLAN78].

Different methods for predicting costs are applicable during different phases in the system life cycle. During the early design stages, available data are scarce. Therefore, costs must be estimated primarily through projections based on past experience with similar systems, the use of parametric cost estimating relationships, and knowledge based on experience. As the design becomes more advanced, improved data, such as drawings, specifications, parts lists, predictions, etc., become available and the analyst is able to utilize them to perform a more thorough analysis [BLAN78]. Finally, after the system is produced and put into operation, test and field data become available for assessment purposes. Using this real world data as input into the life cycle cost model, more accurate life cycle cost figures can be obtained [BLAN78].

Cost estimates are derived from:

(a) Extant data,

(b) Estimating relationships, and

(c) Practical knowledge.

The initial step in performing cost estimation is the investigation of all possible data sources to determine what data is available and to evaluate its applicability. The analyst should research available data in existing databases, supplier documentation, system planning data, reliability and maintainability predictions, engineering test and field data, etc. [BLAN90].

One method of acquiring cost estimation data is by establishing a correlation between existing costs and new costs [REIC80]. Historical data can often be used as the basis for such comparisons. In many cases, systems which are similar in configuration and function to the system under consideration are already in existence. If cost data has been recorded for such systems, then that information can serve as a database for the current project. Thus, actual historical information can be employed in deriving future estimates on the basis of similarity. Some data can be used directly, but other data may make it necessary to apply adjustment factors to compensate for any differences in technology, configuration, etc. This technique is sometimes referred to as analogous cost estimating [BLAN90].

Supplier documentation such as proposals, catalogs, design data, and reports covering special studies conducted by suppliers may qualify as a data source. Potential suppliers may submit proposals for consideration, and these proposals may include not only procurement costs but sometimes life-cycle cost projections [BLAN90]. Reiche notes the usefulness of vendor catalogs:

> Although the price for a single item in a catalogue may not reflect the actual market price, it is a base from which one can operate. In a sense it is historical information which once must have been correct and used. The comparison of one catalogued item and the same from another catalogue may yield a good estimated price. [REIC80, 5]

Another source of cost data is advanced planning data such as market analysis data, system operational requirements, the maintenance concept, and the results of technical feasibility studies. Information pertaining to the proposed physical configuration, major performance features of the system, system effectiveness, and maintenance and logistic support provides essential input to the cost estimation process [BLAN90].

Individual cost estimates, predictions, and analyses, which take place throughout the research and development, production, and maintenance phases of the system life cycle, should be considered as another source of cost estimation information. Estimates associated with research and development include initial engineering cost estimates or cost-to-complete projections. Such projections deal primarily with labor costs. Production cost estimates are often based on individual manufacturing cost standards, value engineering data, industrial engineering standards, etc. Maintenance cost estimates are

based on the predicted frequency of maintenance or the mean time between failures (MTBF) factor, since support costs are basically a function of the inherent reliability and maintainability characteristics in the system design [BLAN90].

More cost estimation data becomes available during the later stages of system development and production. When the system is being tested or is in operational use, engineering test and field data may be used to access the impact on life cycle cost that may result from any proposed modifications to hardware, software, and/or the elements of logistic support [BLAN90].

If no suitable data is available, then the use of parametric cost estimating methods may be called for. The use of parameters involves developing a generalized relationship between one or more program characteristics and the cost [SELD79]. These parametric relationships are called cost estimating relationships, or CERs.

Dhillon defines a cost estimating relationship as "an equation relating cost as the dependent variable to one or more independent variables" [DHIL89,4]. These CERs relate various cost categories to cost generating variables. These variables usually represent characteristics of system performance, physical features, system effectiveness, etc. [BLAN78]. Thus, the CER transforms the problem from one of estimating dollars to one of estimating more familiar and more accessible variables [SELD79].

Blanchard discusses the formulation of valid parameters:

> Cost estimating relationships may assume numerous forms, varying from informal rules of thumb or simple analogies to a more formal mathematical relationship derived through a statistical analysis of empirical data. Generally, cost and related data are collected on existing systems in the inventory, analyzed, converted to some form of relationship, and applied to a new system (which is similar in form and function) as a predicting tool. Given an identifiable database, the analyst assumes some theoretical relationship and then proceeds to test that relationship for validity.... [BLAN81, 373]

Cost estimating relationships may be expressed in various ways, for example, linear functions, nonlinear functions, or multivariate functions.

The estimating relationship may take the form of a linear function expressed by the equation

cost = (some constant)(variable X)

Using a plot of several data points, a linear relationship can be established through curve fitting techniques or through linear regression analysis [BLAN81].

Nonlinear cost relationships may be normal, log-normal, exponential, or hyperbolic in nature. Some cost relationships are discontinuous in nature. A step function assumes one form between two discrete values of a variable and another form between other values of the same variable. In other words, the function is constant over a certain range, then suddenly shifts to another level before becoming once again constant over another range of values [BLAN81].

Multivariate functions may be used to express cost estimating relationships in situations in which multiple variables are required to express cost. The cost function may be expressed in the form

$$cost = 100 + (K_1)(X_1) + (K_2)(X_2)$$

where K is some fixed constant and X is some variable [BLAN81].

When multiple parameters are involved in cost estimating, each parameter should be evaluated from the standpoint of degree of importance, that is, how greatly it influences the cost of the system. The degree of importance can be incorporated by assigning a weighting factor to each parameter.

The following section is devoted to the discussion of various parameters which should be considered when performing cost estimation for an embedded controller. Both hardware and software parameters are dealt with.

Parameters should be selected with discretion. Selection of invalid or unsuitable parameters may introduce a great deal of error into the cost estimation process. Cost estimating relationships are highly significant factors in cost estimation and, if improperly applied, can invalidate the entire life cycle cost model.

The final cost estimating technique is the use of estimates based on practical experience. Individuals with extensive experience in their field can generally accurately estimate the time and cost associated with a job. Seldon discusses this estimating technique:

> The most acceptable and time-honored approach to engineering cost estimates is to ask the people who will have to do the job how much it will cost. They certainly ought to know and they do. That is, they do if they receive the proper directions. They must have a clear description of the task

> (a statement of work with performance specifications, data requirements, test requirements, and so on), a definition of which organization does each task, and a schedule. Each part of the organization is then requested to estimate the cost of doing the task assigned to it. It does this estimating generally by analogy, by its own CERs, or by even more esoteric methods. [SELD79, 31]

Regardless of which cost determination method is used, it is clear that life cycle cost analysis is an iterative process which is carried out throughout the life cycle of the system. The analyst uses the best data available at each stage. Although cost estimates based on analogous cost estimating techniques and/or data gained from practical experience are easier to develop, the usefulness of parametric methods for generating cost estimates should not be underestimated.

Potential Parameters

In the development of cost estimating relations for an embedded controller, there are several cost generating variables which must be considered. The major cost drivers are, of course, not only directly influenced by the hardware/software trade-off as well as the make/buy decision, but also by a myriad of system characteristics which must be considered as well.

As noted previously, a hardware implementation of a function is more expensive in terms of production costs, while a software implementation is more expensive in terms of development costs but is more flexible in terms of modifications to the design. Lam reaches the conclusion that:

> Consequently, for a μP-based system with a projected large production volume, hardware components should be kept at a minimum, and functions should be implemented with software modules whenever performance requirements permit. The increase in initial (software) development cost will be averaged over the large number of units produced.... On the other hand, if the production volume will be small, then the initial (software) development cost dominates the production cost. In this case, hardware implementations of functions can be more cost-effective. [LAM88, 462]

The decision to make or buy a component can also be a major cost driver. As noted in a previous chapter, components designed and produced in-house require extensive

development time and costs, the cost and availability of in-house experts, and related production costs. Such components, however, are tailored for their specific application. Procured components bypass development and production costs, but procurement costs, which include a vendor profit margin, must be considered. Although procured components are generally available more quickly, they have the drawback of being designed for generic applications and may include unnecessary features or lack desired ones. Fabrycky explains how to assess the make or buy situation:

> The question of whether to manufacture or purchase a needed item may be resolved by the application of minimum cost alternatives for multiple alternatives. The alternative of producing may be compared with the alternative of purchasing if the minimum cost procurement quantity for each is computed and used to find the respective cost values. Choice of the total cost value that is minimum identifies the better of the two alternatives. [FABR80, 337]

Aside from the hardware/software trade-off and the make/buy decision, there are additional parameters which must be considered, parameters associated with both the hardware components and with the software components.

Various parameters directly influence the cost of the hardware components which make up an embedded controller. All phases of the system life cycle, development, production, maintenance, and disposal, are affected. Cutaia, Gibson, and Lam discuss several parameters which should be investigated when developing accurate cost estimating relationships for an embedded controller [CUTA90; GIBS79; LAM88].

Parameters which influence the complexity, and therefore the cost, of design include microprocessor-related factors, bus-related details, and miscellaneous features (Figure 3.1).

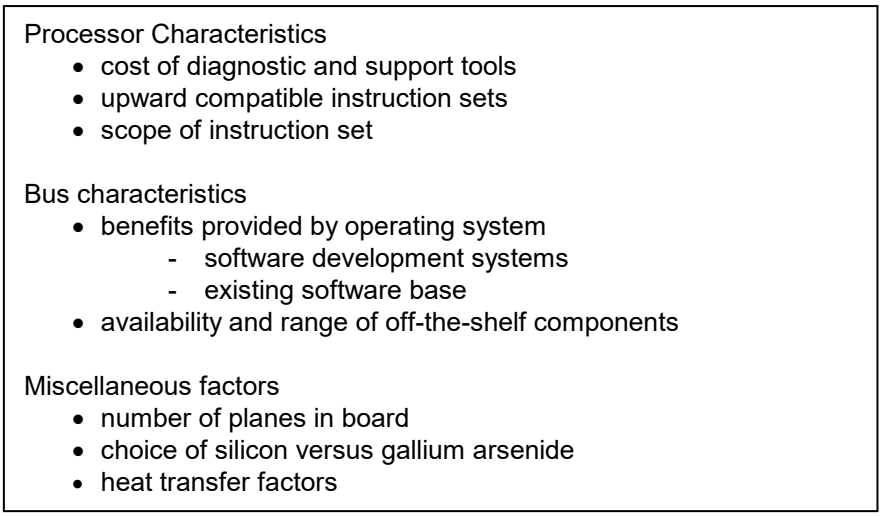


Figure 3.1. Design Parameters.

The cost of the development and diagnostic support tools associated with the selected microprocessor must be considered. In addition, the effort required to develop and debug the microprocessor-based system is a major cost, and is dependent on such factors as the scope and flexibility of the instruction set as well as the upward compatibility of the instruction set.

Various details associated with the selected bus also affect the development costs. The benefits provided by a particular operating system in terms of the quality of software development systems and the extent of existing software can greatly influence development time and costs.

Miscellaneous factors such as the number of planes in each board, the choice of silicon versus gallium arsenide integrated circuits, and heat transfer factors all influence the complexity of the design process.

Production costs are influenced by the cost and availability of components (Figure 3.2). The cost of components is influenced by their complexity and capabilities, as well as their position relative to the leading edge of technology.

Factors which influence the complexity of a microprocessor include the execution speed. If the application is computation-intensive, then the instruction cycle time of arithmetic instructions or the availability of a floating point unit is significant. If the application is input/output-intensive, then the instruction cycle time of I/O instructions or the availability of an I/O processor is important. Another factor is the size of the address

bus. The capacity of the address space influences the complexity of the microprocessor system that it can support. The word size of a processor is determined by the width of its internal registers and the width of the data bus. Processors with larger word sizes are more powerful. Multitasking support hardware allows the system to process multiple tasks or users. The presence of such hardware increases the complexity and costs of the processor. Other factors, such as the processor's position in the processor family and the length of time that the processor has been on the market also contribute to costs. Finally, the cost of the processor is sometimes reduced by the availability of second sources.

The costs related to bus selection also influence the cost of each unit produced. Factors such as the relative capabilities of the bus greatly affect the cost of off the shelf boards. For example, boards which have been designed for the Futurebus are considerably more expensive than boards for the VMEbus. As was noted with respect to the microprocessor, the availability of second sources tends to reduce prices.

Processor characteristics
- execution speed
  - instruction cycle time of arithmetic instructions or FPU availability for computationally intensive applications
  - instruction cycle time of I/O instructions or IOP availability for I/O intensive applications
- address space--size of address bus
- word size--size of data bus and internal registers
- multitasking support hardware
- position in processor family
- length of time on the market
- second sources

Bus characteristics
- capabilities of bus
- availability of second sources

Miscellaneous factors
- number of planes in board
- mean time between failures
- choice of silicon versus gallium arsenide
- heat transfer requirements
- need to supercool chip to enhance superconductivity
- special power requirements
- extensive memory requirements
- EPA regulations on disposal of etching chemicals

Figure 3.2. Production Parameters.

There are other factors which influence the costs of the production of an embedded controller. An increase in the number of planes in the board, higher mean time between failures, the choice of gallium arsenide rather than silicon, stringent heat transfer requirements, the need to supercool the chip to enhance superconductivity, special power requirements, extensive memory requirements, and Environmental Protection Agency regulations on the disposal of etching chemicals all contribute to increased production costs.

Maintenance costs are influenced by continued vendor support of chips, boards, bus standards, and operating system software, as well as the availability of second sources for those items (Figure 3.3). If any of the system components are no longer supported and no alternate vendor is available, maintenance costs will escalate drastically. Maintenance costs are also influenced by the mean time between failures (MTBF) related to the various components. The mean time between failures influences many phases of the life cycle. Perrigo notes that:

> ...the cost of the hardware, that is the purchase price of the electronic equipment, increases as higher and higher MTBF are demanded....It can also be seen that the [overall] cost of maintenance labor and materials decreases with increasing MTBF, as a result of the fact that fewer failures occur, and thus fewer maintenance actions are required. [PERR74, 522]

de Neumann notes that reliability is the most important driver of life cycle costs, closely followed by maintainability [DENE83].

Factors which influence the disposal costs include the disposal of contaminated equipment in accordance with EPA regulations (Figure 3.3).

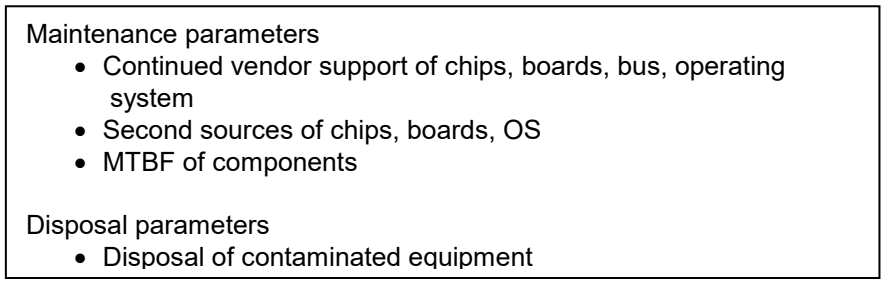


Figure 3.3. Maintenance and Disposal Parameters.

Both the cost of producing software and the reliability of that software are becoming major influences in life cycle costing [DENE83]. In fact, it appears that software costs are becoming increasingly greater relative to hardware costs [DENE83]. It is expected that the software production costs will eventually become the dominant feature of life cycle costing and software reliability a major parameter of a system's effectiveness [DENE83].

Boehm discusses several parameters which influence life cycle costs [BOEH81]. These parameters are included in Boehm's Constructive Cost Model (COCOMO), and apply to both the development phase and the maintenance phase [BOEH81]. The parameters can be classified as product attributes, computer attributes, personnel attributes, and project attributes. Sommerville summarizes the parameters as follows [SOMM89].

Product attributes reflect such factors as software reliability, database size, and product complexity (Figure 3.4). Required software reliability can be gauged by the repercussions associated with a software failure, ranging from minor inconvenience to loss of life. The database size parameter is based on a comparison of the size of the database to the number of delivered source instructions (DSIs), ranging from ten times less than the number of DSIs to one thousand times more than the program size. Product complexity rates the complexity of the code. Low complexity code involves simple input/output operations, simple data structures, and sequential code. Nominal complexity uses multi-file input/output, use of library routines, and inter-module communication. Very high complexity code might involve re-entrant or recursive code, complex file handling, parallel processing, etc.

Figure 3.4. Product Attributes.

Computer attributes refer to hardware constraints such as speed and space restrictions which affect software productivity (Figure 3.5). Execution time constraints refer to the percentage of available execution time which is used. Similarly, storage constraints refer to the percentage of available storage which is used. That storage

pertains to direct random access storage devices such as disk drives. Virtual machine volatility refers to the frequency at which the virtual machine is altered. The virtual machine is the combination of hardware and software that the system utilizes to complete its tasks. Computer turnaround time indicates the level of computer response time experienced by the software development team. This factor is no longer as much of a cause for concern as it was when COCOMO was first developed.

Figure 3.5. Computer Attributes.

Personnel attributes reflect the capabilities and experience of the software development team (Figure 3.6). Analyst capability includes such concerns as analyst abilities, efficiency and thoroughness, and the ability to communicate and cooperate. Applications experience refers to the length of applications experience of the software development team. Programmer capability rates the capabilities of the programmers in the same manner that the analyst was rated. Virtual machine experience refers to the length of experience that the software development team has with the particular virtual machine. Programming language experience gauges the experience level of the team with respect to the programming language to be used.

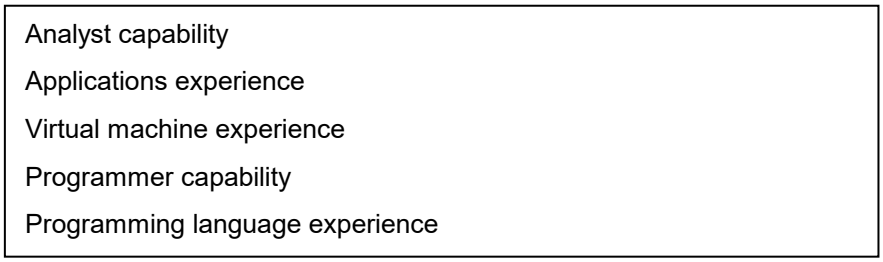


Figure 3.6. Personnel Attributes.

Project attributes refer to such factors as the use of modern programming practices, use of software tools, and the scheduling constraints (Figure 3.7). Modern programming practices include such techniques as top-down design, structured programming

techniques, design and code inspections, etc. This parameter measures the degree to which these practices are utilized. The use of software tools refers to the extent that software tools are utilized. A low value assessment of this parameter indicates that only basic tools such as an assembler are utilized, while a high value means that tools are available to support all development phases. The schedule constraint parameter refers to the level of schedule constraint imposed on the software development team. This parameter can reflect a range of schedule durations from an accelerated schedule to an extended schedule.

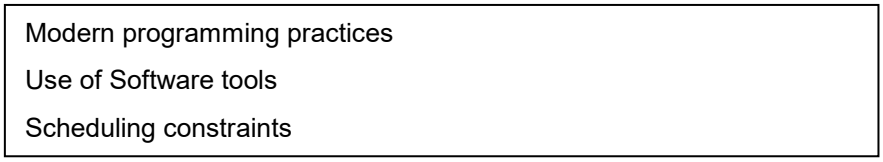


Figure 3.7. Project Attributes.

Obviously not all of these parameters will come into play for every embedded controller project. The analyst using the life cycle cost model must select those parameters which appear to be the major cost drivers for the project under consideration, and weight them accordingly.

CHAPTER IV - VALIDATION AND USES

Validation refers to the process of insuring that the selected model incorporates the necessary characteristics to perform an accurate cost estimation. Model validation is difficult because the model is designed to estimate future costs which are impossible to verify. It may be possible to select as a reference an existing system which has been in use for a substantial period of time and which is similar to the proposed system. The model is then applied to the reference system by using existing data gathered from the reference system as an input to the model. The output results are compared with actual cost figures associated with that system. A close correlation between the results and actual figures indicates that the model is valid in terms of the target application. On the other hand, if the results are significantly different, then the analyst must determine the source of the incongruities and the reasons for such [BLAN78].

It should be possible to validate the life cycle model for an embedded controller by applying it to a similar FSI project which has been completed. If thorough records were maintained, the project should yield accurate cost figures relating to its research and design, production, maintenance, and anticipated disposal. Those figures can be applied to the model to determine an overall life cycle cost. This figure can then be compared to the actual life cycle cost as determined by project leaders. As with any validation process, the success of the validation depends not only on the validity of the model, but also on the validity of the test data [PRIE88].

Applications of Life Cycle Costing

Life cycle costing should be used within the framework of cost effectiveness. Cost effectiveness is the measure of how much performance can be obtained for a given cost, that is, the estimated effectiveness of a system as a function of its cost in dollars [BOEH81].

Cost effectiveness encompasses both system effectiveness and total life cycle cost (Figure 4.1). System effectiveness refers to how well a system fulfills a need and may be expressed by various figures-of-merit such as speed, accuracy, or the probability of system success [BLAN78].

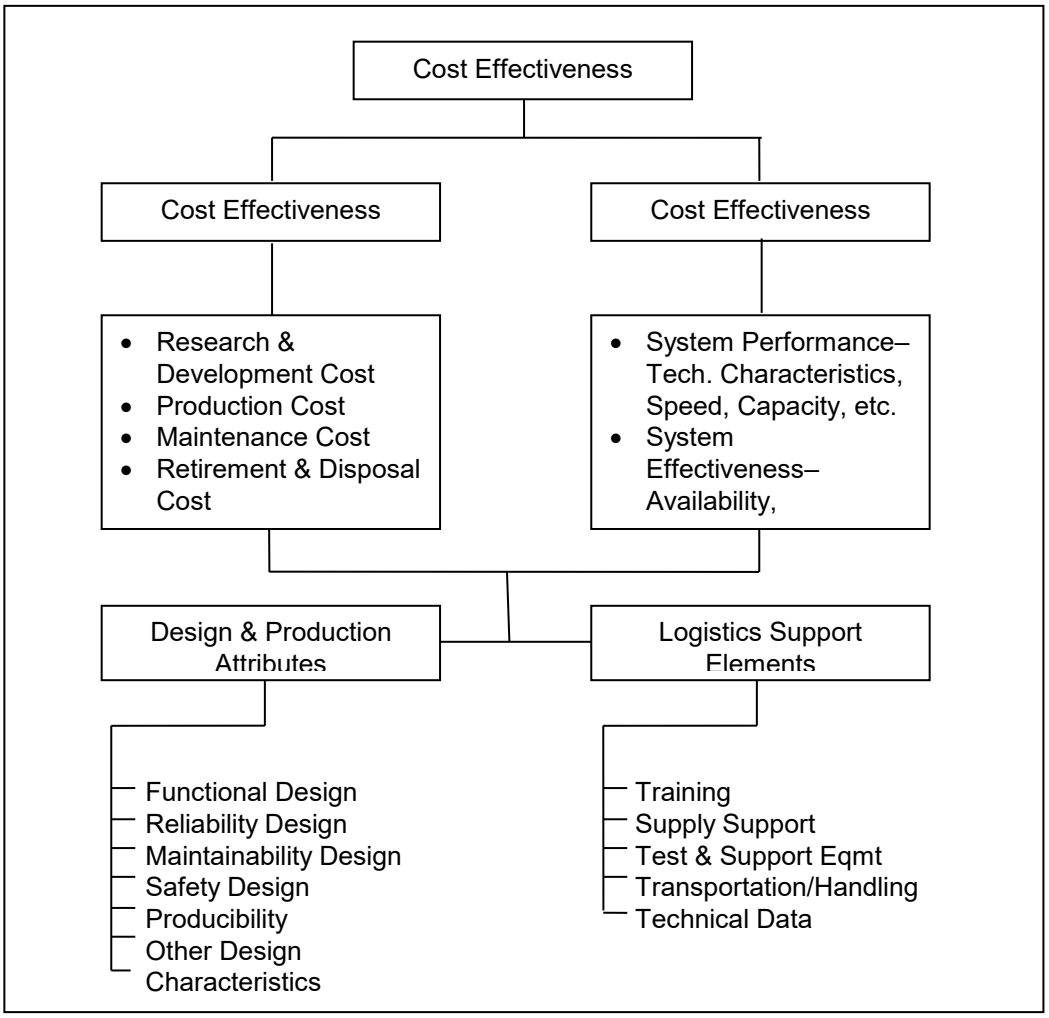


Figure 4.1. Components of Cost Effectiveness. [Adapted from BLAN78.]

Cost effectiveness is not only influenced by the characteristics of system design, but also by the effectiveness of the logistic support capability. For example, an embedded controller may be an excellent performer when it is operating correctly; however, unless there is adequate logistic support available when it fails, its overall usefulness is severely impaired.

When evaluating various design alternatives, not only should system effectiveness and performance parameters serve as criteria for the determination of a preferred approach, but life cycle cost must be considered as well.

One of the primary applications of life cycle cost models is to evaluate various alternative approaches in order to determine the most cost effective system configuration. A life cycle cost analysis is conducted to evaluate each possible candidate to determine which of the various candidates being considered is preferred from an overall cost effectiveness standpoint [BLAN78].

The evaluation process is conducted with the goal of selecting the preferred design alternative, and involves a comparison of all possible alternatives. Blanchard explains the complexity of the process:

> ...there may be many top level candidates for consideration, many different configurations of each major candidate, numerous variations within each configuration, and so on. Initially, major candidates are considered and a preferred approach is selected. Then, different configurations of the selected approach are evaluated and a specific configuration is chosen from that group of alternatives. This is an iterative process working from the top level down to the depth necessary to support a given decision. [BLAN78, 78]

The cost breakdown structure serves as a starting point for conducting a life cycle cost analysis of the proposed alternative configurations. Although some of the cost categories of the generic model may be irrelevant or insignificant in terms of magnitude of cost, the overall cost breakdown structure should provide a guide for cost accumulation and comparison of alternatives.

As Figure 4.2 indicates, the analysis involves the evaluation of alternative configurations, and the selection of a preferred approach. In each instance, there will be required activities involving planning, management, engineering design, test and evaluation, production, distribution, system operations, maintenance and support, and ultimate equipment disposal.

In an attempt to determine specific costs, the analyst may wish to perform the following steps:

(a) Identify all anticipated program activities that will generate costs in the life cycle for each of the alternatives.

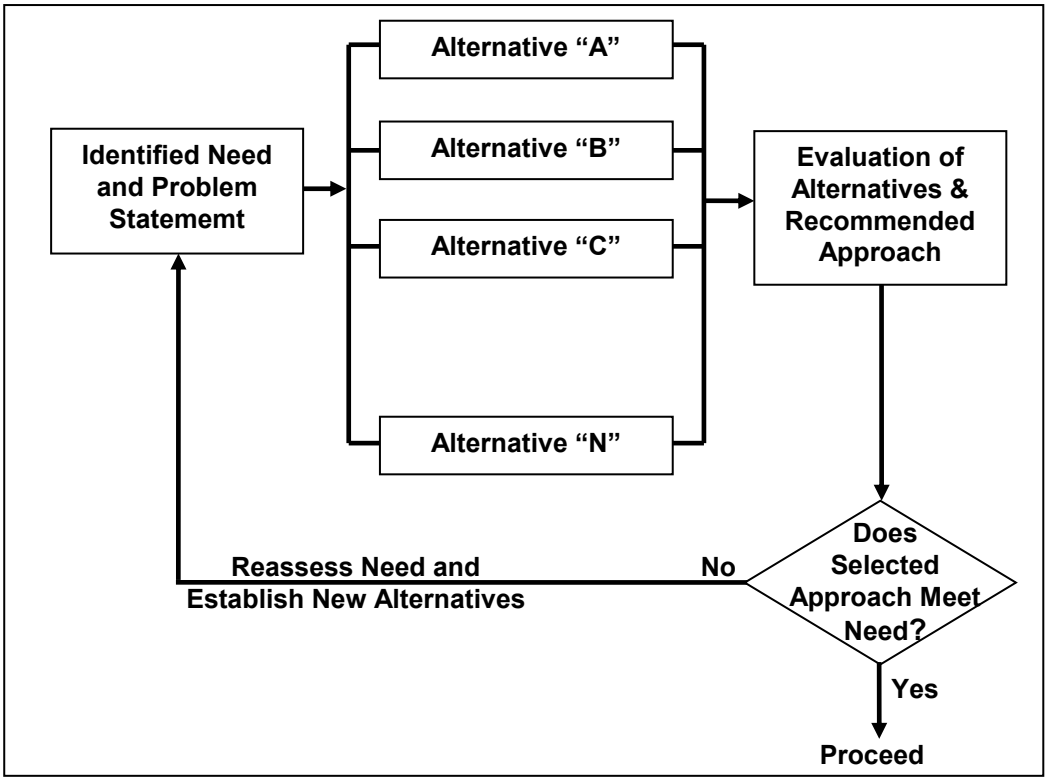


Figure 4.2. The Basic Evaluation Process. [Adapted from BLAN78.]

(b) Relate each activity to a specific cost category in the CBS. Each activity should fall into one or more of the categories of the CBS. If not, the CBS should be modified to include overlooked expenditures.

(c) Develop a cost matrix-type worksheet for the purposes of recording costs for each applicable category by year in the life cycle (Figure 4.3).

(d) Determine cost input data for each activity included in the matrix, and record that data in the matrix.

(e) Accumulate the costs in the matrix to arrive at a total life cycle cost for each configuration.

Before making a recommendation based on the figure obtained from life cycle cost analysis, the analyst should review the process to verify its accuracy. A sample checklist is presented in Figure 4.4 as an aid in assessing the final results. The checklist is made up of questions pertinent to any analysis, and which should all receive an affirmative response. Items such as the problem definition, stated assumptions, selected parameters, and data input should be verified for accuracy and applicability.

Cost Category	Alternative "A"		Alternative "B"	
	Cost ($)	% of Total	Cost ($)	% of Total
1. Research & Development Cost (C_R)				
(a) System/Product Management	132,563	20.3	131,928	21.4
(C_{RM})	50,100			
(b) Product Planning (C_{RP})	165,717	7.7	46,997	7.6
(c) Functional Specification of	153,292	25.4	115,389	18.7
(d) Microprocessor System (C_{RF})	51,496	23.5	140,710	22.8
(e) Hardware/Software Partition	99,836			
(C_{RD})		7.9	72,664	11.8
(f) Integrated Testing (C_{RT})	653,004	15.3	109,095	17.7
(g) Integration and Testing (C_{RI})				
Sub-Total		33.6	616,783	32.4
2. Production Cost (C_P)				
(a) Production/Construction	95,500			
Management(CPM)	250,000	15.2	75,000	13.3
(b) Construction (CPC)	175,000	39.7	135,500	24.0
(c) System Production (CPP)	12,000	27.8	223,750	39.6
(d) System/Product Documentation	32,000			
(CPT)	65,103	1.9	11,000	1.9
(e) System/Product Distribution	629,603			
(CPD)		5.1	45,000	8.0
(f) Initial Logistics Support (CPL)		10.3	74,525	13.2
Sub-total		32.4	564,775	29.6
3. Maintenance Support Cost (C_M)				
(a) System/Product Life Cycle Mgmt	75,435			
(CML)	125,350	11.7	67,935	9.6
(b) Maintenance Training & Facilities	443,720			
(CMT)	644,505	19.4	154,310	21.8
(c) System Maintenance (CMS)		68.8	484,940	68.6
Sub-Total		33.2	707,185	37.1
4. Retirement & Disposal Cost (C_D)	15,000	0.8	17,000	0.9
GRAND TOTAL	1,942,112	100.0	1,905,743	100.0

Figure 4.3. Life Cycle Cost Analysis Breakdown. [Adapted from BLAN78.]

When parametric cost estimating methods have been employed, a sensitivity analysis may be conducted in order to determine the validity of the recommended approach. A

sensitivity analysis can be used in conjunction with parametric methods to determine the sensitivity of the model to input variations. One method of performing such an analysis is to apply the model to a baseline system configuration. A baseline configuration is an assumed configuration for the system being evaluated and does not necessarily reflect the final configuration selected [BLAN78]. The model can be applied multiple times while varying different key parameters to determine their impact on the results. Variation may be accomplished by substituting various values for the input parameter(s) under scrutiny [BLAN78].

The sensitivity analysis should test those parameters which are directly related to the high cost categories and the major cost drivers. Blanchard notes some concerns which must be taken into account when observing the variations which result from a sensitivity analysis:

> ...the analyst should be concerned not only with the delta effects of these variations on total life cycle cost, but the degree of variation that can occur without introducing an unnecessary risk in decisions pertaining to the selection of alternatives. The degree of variation that can be tolerated will relate directly to the accuracy of the input data requirements necessary for the life cycle cost analysis. If the allowable output variation is relatively small and the input data factors vary over a wide range, then the analyst may which to expend some additional effort to acquire better input data. [BLAN78, 98]

The analyst can conduct a sensitivity analysis on a particular life cycle cost model to reveal cause and effect relationships, to predict trends, and to respond to "what if" questions [BLAN78].

A. Assumptions
   1. Are all assumptions adequately identified and documented?
   2. Do all assumptions avoid treating quantitative or qualitative uncertainties as facts?
   3. Are major assumptions reasonable?
B. Alternatives
   1. Are all current capabilities adequately considered among alternatives?
   2. Have all feasible alternatives been considered?
   3. Have the inadequacies of those alternatives which have been rejected been justified and documented?
C. Documentation
   1. Is the study adequately documented?
   2. Are facts stated correctly and with proper qualification?
   3. Are the applicable references documented?
D. Model relationships
   1. Does the model adequately address the problem?
   2. Are cost and effectiveness parameters linked logically?
   3. Is the model designed in such a manner as to allow for the evaluation of specific elements of the system independent of other elements?
   4. Does the model allow for a timely response?
   5. Does the model provide valid (comprehensive) and reliable (repeatable) results?
E. Cost
   1. Has the overall cost breakdown structure been defined?
   2. Are all cost categories in the cost breakdown structure adequately defined?
   3. Are all life cycle costs being considered?
   4. Are all cost estimating relationships relevant and realistic?
   5. Are variable and fixed costs separately identifiable?
   6. Are all cost elements considered?
      (a) Feasibility studies
      (b) Design and development
      (c) Production and test
      (d) Installation and checkout
      (e) Personnel and training
      (f) Technical data
      (g) Facility construction and maintenance
      (h) Spare/repair parts
      (i) Support equipment/tools
      (j) Inventory maintenance
      (k) Customer support (field service)
      (l) Program management
   7. Are the cost aspects of alternatives treated in a consistent and comparable manner?
   8. Are the cost estimating relationships reasonably accurate?
   9. Has the sensitivity of parametric cost estimates been properly addressed through a sensitivity analysis?
F. Conclusions and recommendations
   1. Are the conclusions and recommendations logically derived from the material contained in the study?
   2. Have all the significant ramifications been considered in arriving at the conclusions and recommendations presented?
   3. Are the conclusions and recommendations free of bias?
   4. Do the conclusions and recommendations appear to be independent of any external influences?
   5. Are the conclusions and recommendations based on more than insignificant differences?

Figure 4.4. Sample Analysis Checklist.

CHAPTER V - EVALUATION PHASE

After the completion of the cost breakdown structure, existing life cycle cost tools must be evaluated in order to determine their ability to calculate accurate life cycle costs. The cost breakdown structure which was produced by this research indicates that the life cycle cost model can be partitioned into a hardware subsystem and a software subsystem. Tools which offer a way of modeling each of these subsystems are currently available.

The packages which were chosen for study regarding the software subsystem were:

(a) *Software Cost Model* (SOFTCOST) developed by the National Aeronautics and Space Administration, and

(b) *Before You Leap* (BYL) provided by the Gordon Group.

The tools which were selected for evaluation with respect to the hardware subsystem were:

(a) *Life Cycle Cost Model, Version H* (LCCH) provided by the Headquarters Acquisition Logistics Division of the United States Air Force, and

(b) *Programmed Review of Information for Costing and Estimation--Hardware* (PRICE "H") developed by RCA.

An additional package, designed to estimate overall life cycle costs, was also selected for evaluation: *Life-Cycle Cost Calculator* (LCCC) developed at Virginia Polytechnic Institute.

Each life cycle cost tool will be evaluated with respect to its degree of applicability to the subsystem which it is intended to model. In order to evaluate the tools associated with each subsystem, data will be gathered from existing FSI projects and applied to each tool. The suitability of each tool will be measured by how accurately that tool reflects the actual results. Those tools which are deemed to be less suitable will be examined to determine if their accuracy can be improved by minor modifications. Tools which cannot be acquired, or which have major operational errors, cannot be thoroughly evaluated and thus will be dropped from consideration.

Software Costing Tool Evaluation

Life cycle cost tools are currently available to assist in the estimation of the time and costs involved in a software project. Those packages which were selected for evaluation are:

(a) *Software Cost Model* (SOFTCOST) developed by the National Aeronautics and Space Administration.

(b) *Before You Leap* (BYL) provided by the Gordon Group.

In order to evaluate the accuracy of the estimates produced by SOFTCOST and BYL, FSI provided actual figures pertaining to the duration and expense of a recently completed software project. The FSI project for which data was gathered involved the development of software to control the operations of a cluster level controller. This software project consisted of approximately 120,000 lines of new code, written primarily in the C programming language. The input data provided for each of the software costing packages provides specific details about the software project. A listing of that input data is included in the evaluation of each tool.

SOFTCOST Evaluation

The *Deep Space Network Software Cost and Resource Estimation Model* (SOFTCOST) is designed to estimate the required resources and to provide a schedule for software development. The model incorporates features from several previously developed software cost models in order to provide an algorithm which takes into account a variety of implementation factors relative to project size, organizational environment, system environment, and project difficulty [TAUS81].

The SOFTCOST model utilizes features from several existing models. It utilizes several factors from both the General Research Corporation model and the Watson-Felix model developed for IBM. It utilizes the "PERT" technique to estimate the anticipated size and variance of the software being developed. A modification of the Rayleigh-Norden-Putnam model is used to evaluate resource estimates. Various other models have been used to provide a guide for model development, including PRICE "S," SLIM, SLICE, as well as

models developed by R. W. Wolverton, TRW (COCOMO), Air Force Electronic Systems Division, Tecolote, and Aerospace Corporation.

SOFTCOST measures lines of code in kilo-source lines of executable code, or KSLEC. A source line of executable code is defined by SOFTCOST to be a source language statement occupying one physical line in the source file that results in the generation of object code, the reservation of storage, or the definition of data types. Comments are excluded.

Model outputs include estimates and variance values for project size, staff productivity, effort, probable duration, recommended staff level, the amount and cost of documentation, and required computer resources (Figure 5.1). The user is required to evaluate the resulting values and then to enter risk-biased values for effort, duration, and staffing. The model evaluates those values for reasonableness, and determines a confidence level for the associated estimates. The model is also capable of producing complete scheduling data, including a PERT chart with subtask efforts, durations, and precedences (Figure 5.2) as well as a Gantt chart of planned activities (Figure 5.3). It applies the estimated effort, staff, and duration to a standard Work Breakdown Structure (WBS) developed for Deep Space Network software tasks, and produces a task plan to be used at the initial system planning, software implementation planning, or software maintenance planning stages of a project. Samples of the various types of output are included in this evaluation.

Type of Test

The SOFTCOST program received as input a set of parameters which were derived from a recent FSI software project. Those parameters can be seen in Figure 5.1.

Test Goals

SOFTCOST produces an estimate of project duration and required staff. It was hoped that SOFTCOST, when provided with the actual data, would produce estimates which would approximate the actual values for duration, 24 months, and staff, 8 team members.

SOFTCOST Results

When the actual FSI data was applied to SOFTCOST, the results failed to approximate the actual values. Both the estimate for duration and the estimate for staff were greater

than the actual values. SOFTCOST estimated that 9.3 team members would be required with a project duration of 40.3 months, as seen in Figure 5.1. After producing its estimates, the model requests risk-biased values for duration and staffing. The model concluded that with the data set provided by FSI the confidence level of completing the project with eight members in 24 months was 0%. Clearly, the model failed to produce estimates which reflected FSI's performance figures.

In an attempt to isolate the cause of the model's failure to perform as desired, a substitute data set was provided. This substitute data set was designed to provide a "best-case" scenario while producing the actual number of lines of source code. Improvements in the model's performance would indicate that further testing of this data set would provide insight into the sensitivity of the model to specific parameters. The next step in the testing procedure would be to alter the parameters one by one in an attempt to determine which parameter or parameters were causing the disparity between the model's estimates and the actual figures. SOFTCOST is designed so that for each multiple choice query the first response decreases productivity, the second has no effect, and the third increases productivity. Based on that fact, the "best-case" data set was structured so that every multiple choice query was provided with the third selection. This data set can be seen in Figure 5.4. Even when provided with a data set designed to improve the estimates, Figure 5.4 shows that the model did produce an accurate estimate for staff size, 8 team members, but the estimate for duration, 37 months, was still inaccurate. These estimates were an improvement, but not as great an improvement as expected. In addition, when provided with the risk-biased values of 8 team members and a duration of 24 months, the model still produced a low confidence level, only an 11%. It was concluded from these results that the modification of a limited subset of the model's productivity parameters would not be sufficient to calibrate the model's estimates.

Finally, the estimate of staff productivity produced by SOFTCOST was tested to determine if it remains constant over a particular set of parameters if the lines of code are varied. Staff productivity is the number of lines of code each staff member can produce per month, and is based on the relationship of total lines of code divided by the product of the number of staff members multiplied by the project duration. The staff productivity associated with the FSI project is 625 SLEC/staff-month. Despite varying the lines of code

from 1 line to 200,000 lines, the staff productivity estimate remained constant at 324 SLEC/staff-month. This indicates that SOFTCOST's estimate for staff productivity is based only on the parameters and is independent of the amount of code involved in the project. In addition, the estimate is far less than the actual lines of code produced by each individual working on the FSI project. The results of these tests are shown in Figure 5.5. Extensive effort was devoted to altering the model's productivity parameter set, but the proper combination of modifications to produce more accurate estimates could not be isolated.

Conclusions

The failure of SOFTCOST to produce estimates which accurately reflect FSI's performance can be attributed to several factors. Developed for use by NASA, SOFTCOST reflects NASA's more extensive planning requirements, coding differences, and enforced adherence to standards such as the Information System Life-Cycle and Documentation Standards release 4.3 developed under the direction of NASA's Software Management and Assurance Program. These standards are intended to provide a systematic, NASA-wide structure for documenting software development projects. The standards are supported by NASA Data Item Descriptions (DIDS). Each DID outlines a document, such as the Requirements Specification, required for quality software planning and development.

Another factor which might contribute to unexpectedly high estimates is the difference in the level of programmers anticipated by SOFTCOST and utilized by FSI. The FSI cluster level controller project utilized only graduate-level programmers, in most cases computer science doctoral students who had already obtained their Master's Degree in Computer Science. In addition, those students had at their disposal a faculty of computer science professionals with years of computer science knowledge and experience. While the programmers retained by NASA and familiar to the developers of SOFTCOST are no doubt competent and highly skilled, it can be theorized that their abilities are not on an equivalent level with those at FSI.

Furthermore, while the definition of lines of code does not include comments, the estimate for duration must include the time required to comment the source code, due to the extensive documentation required by NASA standards. Extensive commenting increases the duration of a project by reducing staff productivity, i.e., the lines of code

generated by each team member, because the time spent writing comments is time which cannot be spent writing code. In addition, decreasing the productivity of each team member increases the number of team members required to complete a project within a specified time period. Therefore, if the extent to which a program is commented differs greatly between NASA code and FSI code, that could contribute to the higher estimates for duration and staffing produced by SOFTCOST.

Decision

Based upon the results of the tests conducted on SOFTCOST, it was decided to reject this package. Several factors contributed to this decision, including inaccurate estimates, a difficult interface, and failure to consider the entire software life cycle.

As discussed extensively above, the estimates produced by SOFTCOST do not accurately reflect the demonstrated performance by FSI programming teams. In addition, attempts to locate and correct the causes of these inaccuracies were unsuccessful. Although the alteration of the productivity parameters is not extremely difficult, it was concluded that additional effort directed in this area would not be profitable due to the repeated failure of the model to approximate the actual figures. Indications are that wholesale changes in the productivity parameters would be required, altering the entire parameter set upon which the model is based.

The user interface consists of a sequential series of prompts for values pertaining to the various parameters. Once the user has hit the enter key after entering a value, that value cannot be changed except by restarting the program and reentering all values again. In fact, whenever the user wishes to change one or more parameters in order to test scenarios on a "what-if" basis, the entire model has to be rerun. The interface is not user friendly, and flexibility does not appear to have been a factor in its design.

Additionally, SOFTCOST fails to take into account any maintenance activities which follow the software development phase. This is a serious limitation if the entire life cycle of a software project is of interest.

The conclusion of this research is that SOFTCOST is not a suitable software costing model for use in this life cycle costing model.

Before You Leap Evaluation

Before You Leap (BYL) is an automated software cost modeling tool developed by the Gordon Group. BYL is a knowledge-based program designed to provide estimates of the duration, the number of delivered source instructions, and the average number of personnel required to plan, complete, and maintain any software project.

BYL incorporates two major modeling subsystems: an advanced implementation of the Construction Cost Model (COCOMO) algorithms developed by Boehm and other researchers at TRW, and Function Point Analysis. The COCOMO model is considered to be an excellent approach to synthesizing project scheduling and staffing. The Function Point Analysis (FPA) subsystem contains a very powerful and flexible method for measuring and estimating delivered source instructions. The merging of these two subsystems into a single software cost modeling system provides wide ranging capabilities such as the ability to determine the effect on a project schedule if an alternative compiler is utilized.

Before You Leap is a knowledge-based system which utilizes fuzzy logic. The User's Guide indicates that:

> ...BYL makes its estimates for effort, schedule, cost, productivity, and other values by drawing upon a knowledge base derived from over 150 project-years of software development history. This has been supplemented with additional studies of complete software life-cycles that provide scores of extra project-years to the knowledge base. The overall knowledge base was developed from a broad variety of projects that use an equally wide range of compilers, host environments, standards, and practices. All of this was done to ensure that BYL is applicable to any organization, regardless of the computer used, the application type, or any other identifiable factor. [GORD87, 35]

In addition, BYL allows the knowledge base to be modified so that it reflects the demonstrated productivity and abilities of a particular organization. This is accomplished simply by entering the cost driver values and the sizing estimates for one or more completed projects. According to the User's Guide, doing so:

> ...automatically calibrates the basis of the estimates: as more actuals are entered, the overall reliability of calibrated estimates increases, effectively expanding the knowledge base while biasing it towards the demonstrated capabilities of the organization. [GORD87, 35]

BYL considers the software life cycle to begin with the Preliminary Planning and Requirements Phase, to continue through the Design, Programming and Integration and Test Phases, and to conclude with the Software Maintenance Phase.

BYL's measurement of lines of code is KDSI, or thousands of delivered source instructions. This refers to any line of source text regardless of the number of actual instructions on that line. The definition of KDSI excludes comments as well as the lines of code which make up any undelivered support software [SOMM89].

BYL provides a highly interactive environment. The instantly updated screens allow the user to view the various trade-offs offered by different development scenarios. BYL's interactive environment allows the user to perform sensitivity analyses on many different levels. This allows the comparison of different software project scenarios on a "what if" basis. Options exist to allow the user to move among three separate sets of data, and extend the ability to perform sensitivity analyses by allowing the user to examine multiple scenarios. As estimates are made in one data model, other data models can be viewed in order to perform comparisons.

BYL provides a context sensitive help facility. Help is provided in the form of pop-up screens, which can be viewed during the course of estimating a software project. Help screens can be viewed from any screen and at any time.

BYL produces software development estimates pertaining to the expected effort, schedule, cost, productivity, and staffing requirements for the project. These estimates range from the initiation of software development through the integration and testing of the software. The software maintenance estimates provided by BYL encompass expected effort, cost, productivity, and staffing requirements for the project from the point at which development is complete through the operational life of the software.

BYL's output can take the form of graphs or reports. BYL provides the capability of producing pie charts, horizontal bar graphs, and vertical stacked bar and side bar graphs. BYL provides several reports pertaining to projected costs, manpower requirements, and scheduling for a software project. The available reports include the Cost Driver Report (Figure 5.6), the Maintenance Report (Figure 5.7), the Phase Distribution Report (Figure 5.8), and the Life-Cycle Report (Figure 5.9). The Cost Driver Report lists the development cost drivers and shows the sizing estimates for the current model. The Maintenance Report

lists the maintenance cost drivers and the sizing estimates for the current model. The Phase Distribution Report includes estimates for effort, scheduling, and staffing across the three main phases of software development, and assists in forecasting manpower and scheduling requirements. The Life-Cycle Report summarizes the estimates for effort, scheduling, and staffing for the software project through all phases of the software life-cycle, and assists in forecasting manpower, scheduling, and budget requirements. Additional reports which are included are the Aggregate Activity Report (Figure 5.10), the Cash Flow Report (Figure 5.11), and the Function Point Report (Figure 5.12).

Type of Test

The same data set which was applied to SOFTCOST was also applied to BYL. Those parameters can be seen in the attached BYL Cost Driver Report, Figure 5.6.

Test Goals

Like SOFTCOST, BYL produces an estimate of project duration and required staff. It was hoped that these estimates, when based on the actual data, would approximate the actual values for duration, 24 months, and staff, 8 team members.

BYL Results

The default estimates produced by *Before You Leap* differ greatly from the project actuals provided by FSI. In fact, the initial estimates of a 21.5 month duration and a 21.73 member staff were even less accurate than those produced by SOFTCOST. However, when estimates were based on the actual FSI results, the model produced calibrated results which were identical to the actuals. Due to the fact that the only FSI project providing historical background was the same project being examined, the calibrated results are inconclusive. However, if the knowledge base can indeed be biased by historical FSI data entered into the model, then data from additional FSI projects will train the model to produce calibrated estimates which accurately reflect the programming performance demonstrated by FSI.

Conclusions

While a single test set did not make it possible to verify the software's claims that it has the capability to tailor itself to a specific organization, limited testing indicated that it did, indeed, calibrate itself to provide estimates which reflected actual FSI costs. Of course, it can only reflect the performance of FSI programming teams as long as that performance remains relatively consistent.

Before You Leap appears to be a flexible and useful tool which should be investigated thoroughly by entering several additional actual projects and observing the results. It is anticipated that the figures related to project duration, staffing, and cost can be biased to more accurately reflect actual values.

Before You Leap is not without its drawbacks. The ease with which *Before You Leap* can be utilized is greatly enhanced by an understanding of the COCOMO model. Although both the help facility and the documentation provided with BYL explain the various parameters thoroughly, general knowledge of the COCOMO model is helpful. Detailed information about the COCOMO model is available in Boehm's text, and less thorough treatments are presented in many software engineering texts such those by Sommerville and Pfleeger [BOEH81; SOMM89; PFLE87].

Furthermore, the COCOMO model itself has some disadvantages. As discussed previously, undelivered support software is not reflected in the actual lines of code. In addition, some parameters are included to accommodate characteristics of computer systems in the 1970's, the time during which the model was developed. For example, the parameter TURN, computer turnaround time, reflects batch processing constraints which do not pertain to interactive systems or individual workstations. Other parameters are not clearly explained, such as TIME and STOR, which reflect the percentage of available execution time and main storage which is used by all software which is executing concurrently.

Another shortcoming is that BYL assumes that every organization involved in software development operates in a software engineering environment. This requires formal quality assurance, configuration management, test and evaluation, and verification and validation. In addition, BYL dictates the number of team members involved in product

design, programming, requirements analysis, configuration management, quality assurance, verification and validation, testing and evaluation, etc. While software engineering techniques are desirable, it is unreasonable to expect that every organization involved in software development utilizes such techniques or dedicates staff to every position specified by BYL.

Finally, BYL lacks an option to print the graphs. Although several types of graphs can be displayed on the screen, there is no feature which allows those graphs to be printed.

However, the advantages of *Before You Leap* far outweigh the disadvantages. Unlike SOFTCOST, BYL has an excellent interface, the parameter values can be altered and the results can be observed immediately, and it is intended for commercial applications. The Function Point Analysis feature for estimating lines of code is not currently needed by FSI, but is a useful feature which should be investigated more thoroughly. *Before You Leap* has the potential to be an extremely useful package for any organization which performs extensive software development.

Decision

After testing *Before You Leap*, the decision was made to accept the package conditionally. None of the test results contradicted the claims made in the BYL documentation that the knowledge base can be biased towards the capabilities of an organization. This lends credence to the assumption that the package can be adapted to provide relatively accurate estimates for the software subsystem of the embedded controller life cycle model.

Hardware Costing Tool Evaluation

A variety of life cycle cost tools is available to assist in the estimation of the time and costs involved in hardware projects. The difficulty associated with hardware costing tools is the lack of general purpose tools. Most of the tools in use are special purpose tools which are very application specific. Those packages which were selected for evaluation are:

(a) *Life Cycle Cost Model, Version H* (LCCH) provided by the United States Air Force.

(b) *Programmed Review of Information for Costing andEstimation--Hardware* (PRICE "H") developed by RCA.

Because of the lack of availability of general purpose tools, the evaluation approach was to first determine if either of these packages could be used for cost estimation of the hardware portion of the embedded controller project. Only after establishing their applicability would data be gathered with which to test the packages. As will be detailed in the individual evaluations, however, it was determined that no further study of either of the tools was necessary.

Life Cycle Cost Model, Version H Evaluation

Life Cycle Cost Model, Version H, (LCCH) release 1.3 was obtained from the Headquarters Acquisition Logistics Division (AFLC) of the United States Air Force. It was acquired in order to investigate the possibility that it could serve as a general life cycle costing tool for the hardware subsystem of the life cycle cost model for an embedded controller.

LCCH documentation refers to the model as both a logistics support model and an accounting model. It is a modification of the LCC-2A model, which is an enhanced version of the LCC-2 model, a revision of the original LCC model designed for Air Force use. Each of these life cycle cost models was designed to estimate the costs associated with acquiring and supporting an avionics system. An example of avionics spares for which the system is intended to handle include receivers, receiver interfaces, and antenna couplers.

LCCH can perform cost comparisons for use in the selection of hardware mechanization alternatives as well as in the evaluation of alternative maintenance plans for the system. The model does not utilize a work breakdown structure for a system development phase, but instead models costs based on firm bid prices for the acquisition of prime hardware, spares, support equipment, etc. [GATE76].

The fundamental entity in this particular life cycle cost model is the avionics system under investigation. That avionics system is a collection of hardware and associated software whose purpose is to perform specific functions in the unit in which it is installed. The life cycle cost is partitioned into acquisition cost and operation and maintenance cost. As indicated previously, the model is intended not only to perform cost estimation, but to

assist in logistics support planning as well. John Huff, an Operations Research Analyst with the AFLC, referred to the model as being primarily a logistics support model [HUFF91]. The extensive logistics support capabilities include costing for prime hardware, support equipment, initial spares, flight line maintenance, base level maintenance, government depot level maintenance, packing and shipping for contractor depot level maintenance, and support equipment maintenance.

Type of Test

Before performing extensive data gathering for the hardware subsystem of the embedded controller project, an attempt was made to determine the probability that the LCCH model would be applicable to that subsystem. This process involved discussing the model with Air Force representatives, examining the documentation which accompanied the model, and viewing the implementation of the model itself.

Test Goals

Air Force representatives expressed doubts that this model would prove to be useful for the intended application [HUFF91]. The goal of the testing was to verify the accuracy of their reservations. If the model was determined not to be applicable, then further testing would be unnecessary. Otherwise, testing would be conducted using actual data gathered from an FSI project.

Conclusions

The fact that LCCH is a specific life cycle cost model intended for use with avionics systems severely limits its general usefulness. Although spares lists do include such items as central processors, control units, power supplies, memory units, and arithmetic units, these items are referred to as part of a much larger avionics subsystem. Because LCCH was specifically "developed to evaluate the combined costs of acquiring an avionics system and supporting it over its operational life" [GATE76,1-1], the model has a limited scope, and therefore is not applicable to the embedded controller project. Furthermore, LCCH documentation does not indicate the built-in flexibility required to adapt the model to provide costing estimates for the hardware subsystem of an embedded controller project.

Decision

Based on the test results, the decision was made to reject the LCCH model as a potential subsystem for the embedded controller project.

PRICE "H" Evaluation

The *Programmed Review of Information for Costing and Estimation* (PRICE) model was cited by multiple sources as an excellent life cycle cost tool. Both an "H" model for hardware costing and an "S" model for software costing have been developed. However, neither a copy of the PRICE "H" model nor pertinent documentation were available for evaluation.

PRICE "H" is a proprietary model developed by the RCA Corporation to estimate the costs associated with hardware development. PRICE "H" is essentially a parts count model [GRIM74]. It requires very specific details about the hardware and the production program, but provides a prediction "which seems to be correct within plus or minus five percent of the actual cost" [GRIM74, 507]. The PRICE "H" model has been used extensively by the United States Air Force Avionics Laboratory for avionics hardware development and production [FERE74].

Type of Test

Like the testing process associated with LCCH, an evaluation of the model's applicability to the hardware subsystem of the embedded controller project was performed prior to conducting extensive data gathering for that subsystem. Because RCA requires a substantial licensing fee before providing copies of the software, the model was not available for evaluation. Although this prevented study of the model, it was decided to examine RCA documentation in an attempt to learn more about PRICE "H."

Test Goals

The goal of the testing process was to determine if RCA documentation indicated if PRICE "H" is applicable to the hardware subsystem of the embedded controller life cycle model. Because the actual model was not available, further testing was not possible.

PRICE "H" Results

According to John Huff of AFLC, because PRICE "H" is a proprietary model, it is essentially a "black box" [HUFF91]. Various parameters are input, those numbers are manipulated, and results are produced. Details about specific calculations leading to those results are not available.

Not only was the PRICE "H" model not available for examination, but all attempts to obtain pertinent documentation were unsuccessful. Consequently, the evaluation process could not be conducted.

Decision

Although PRICE "H" is reportedly a very accurate hardware costing tool, its licensing fee discouraged attempts to acquire it for study. In addition, attempts to acquire documentation were unsuccessful. As specified earlier, tools which cannot be acquired cannot be thoroughly evaluated. For that reason, the PRICE "H" model was dropped from consideration.

Overall Costing Tool Evaluation

General purpose tools which are capable of determining life cycle costs for an overall project are not widely available. Such tools are designed to calculate life cycle costs from the project's inception to its final retirement and disposal.

Only a single tool was chosen for evaluation in this category: *Life-Cycle Cost Calculator* (LCCC) provided by Virginia Polytechnic Institute and State University.

Life-Cycle Cost Calculator (LCCC)

The *Life-Cycle Cost Calculator* program, LCCC, was developed at Virginia Polytechnic Institute and State University. It is a complex, mathematics intensive program developed to allow the user to input a previously designed cost breakdown structure (CBS), enter cost data into the CBS, and determine overall life cycle cost. Both the CBS and the cost data can be modified, and the model can be displayed in either summary or detailed format. Extensive graphics options are provided. The total life span of the system under study, the interest rate, and the inflation rate are all factored in [VIRG91].

The LCCC program consists of two screens, the Cost Summary screen (Figure 5.13) and the Cost Category screen (Figure 5.14). This division permits the user to view the overall distribution of the cost breakdown structure, or to examine the depths of any category, as desired. Annual data can be reviewed for any category.

Type of Test

Before undertaking an effort to gather accurate cost data for an entire FSI project, it was decided to first determine if the package was fully functional and to confirm that the results produced were in line with the type of results required for the embedded controller project. It was determined that if the package satisfied these conditions, then data would be gathered and applied to the model.

LCCC Results

LCCC does not work properly due to the presence of errors. The model could not be properly evaluated because a software error prevented the cost breakdown structure and cost data from being saved. Due to the presence of program flaws, the model could not be thoroughly tested. Consequently, the additional effort of gathering overall cost data proved to be unnecessary.

Conclusion

When LCCC is debugged and fully capable of performing the tasks specified in its documentation, it should prove to be a useful software package. Its ability to accept a cost breakdown structure and associated data indicate that it would be a useful tool for the embedded controller life cycle model.

LCCC does have many drawbacks. Aside from the coding error, the documentation does not always reflect the actual software. The interface, while relatively easy to use, is not intuitive and requires some experience before the user is comfortable with it. For example, data entry is allowed only in specific areas of the screen, and if an entry is not active the cursor remains outside a valid data field. Pressing any key, other than a function key or an active navigation key, has no effect, and gives the appearance that the system has been locked down by the program. In addition, the program is capable only of displaying the graphical representations of the data, and fails to provide an option to print graphs.

Decision

Because the earlier phases of this research focused on the development of a cost breakdown structure for an embedded controller, this package seems to be the ideal implementation of that research. Unfortunately, this assumption could not be tested due to errors present in the software. Although the developer has been notified of the errors, a modified version could not be acquired in time to be considered during the course of this research. It is anticipated that the debugged version will prove to be a valuable life cycle cost tool, and should be investigated further in future research.

TITLE: FSI Cluster Controller	CDE:
ECR/ECO:	PROG. ID:
SUBSYS:	Model Data Version 3.0 6-02-81

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)?	150
	Expected value, kilo-lines executable source?	120
	Minimum value, kilo-lines executable source (99% confidence level)?	100
2.	How much code exists in modules requiring modification?	
	Maximum value, kilo-lines executable source (99% confidence level)?	0
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)?	HIGH
11.	Complexity of code logical design (high, medium, low)?	HIGH
12.	What percent of the programming task is in Assembly language?	10
13.	What percent of the new or modified code must be storage-optimized?	30
14.	What percent of the new or modified code must be timing-optimized?	60
15.	What percent of the total programming task is 'easy'?	10
16.	What percent of the total programming task is 'hard'?	90
17.	When is work to start, on the (FRD/FDD, SRD, SDD)?	FRD/FDD
18.	What percent of the total program requirements will be established and stable before design, and will not be altered before delivery?	30
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?	60
21.	Complexity of program functional requirements (high, medium, low)?	HIGH
22.	Expected user involvement in requirements definition (much, some, none)?	SOME
23.	Customer experience in application area (much, none, some)?	SOME
24.	Customer/Implementer organizational interface complexity (high, normal, low)?	HIGH
25.	Interfaces with other SW development projects or organizations (many, few, none)?	FEW

Figure 5.1. SOFTCOST Output--FSI Data.

26. Efficiency of implementing organization (poor, ok, good)?	POOR
27. Overall implementation personnel qualifications and motivation (low, average, high)?	AVERAGE
28. Percentage of programmers doing functional design who will also be doing development (<25, 25-50, >50)?	>50
29. Previous programmer experience with application of similar or greater size and complexity (minimal, average, extensive)?	MINIMAL
30. What is the average staff experience, in years, obtained from work similar to that required in the task being estimated?	5
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)?	AVERAGE
33. Use of top-down methodology (low, medium, high)?	LOW
34. Use of structured programmer team concepts (low, medium, high)?	LOW
35. Use of Structured Programming (low, medium, high)?	MEDIUM
36. Use of design and code inspections (low, QA, peer)?	PEER
37. Classified security environment for computer (yes, , no)?	NO
38. Hardware under concurrent development (much, some, none)?	MUCH
39. Percent of work done at primary development site (<70, 70-90, >90)?	>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)?	POOR
43. Maturity of system and support software (buggy, ok, good)?	OK
44. Overall adverse constraints on program design (severe, average, minimal)?	SEVERE
45. Is the program real-time, multi-task (chiefly, some, no)?	CHIEFLY
46. SW to be adaptable to multiple computer configurations or environments (yes.. no)?	YES
47. Adaptation required to change from development to operational environment (much, some, minimal)?	SOME

Figure 5.1. (continued)

```
Estimated Overall Parameters:
                                        =average value
                                  +1-sigma            -1-sigma
Adjusted Lines of code= 121667 SLEC, +/- 9365 SLEC
 131031       112302
Effort=375.5 person-months
652.9         216.0
Staff productivity= 324 SLEC/staff-month
 563          186
Duration= 40.3 months
 49.0          33.1
Avg. Staff=  9.3
 16.2          5.4
Documentation= 6241 pages   $187.2K
 7564          5150          $226.9K          $154.5K
Computer CPU time= 5535 hours     $0.0K
 8354          3667           $0.0K           $0.0K

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


Values specified are:
  Kilo-lines of code:                          121.67
  Effort (person-months):                        192.0
  Duration (months):                             24.0
  Average staff (persons):                        8.0

   For the numbers entered, a reasonableness check indicates that
the average project would produce 54356 lines of code, using 192 staff-months
of resources and 24 months of duration, with an average staff of 8 persons,
for a productivity of 283 SLEC/staff-month.

   The level of confidence in delivering 121667 lines of code,
on-time and within resources= 0 %.

Is output to be saved in a file?                    YES

Name of output file to be created:                   FSI1

Schedule start date:                           01SEP89


Select desired outputs and output media.  Defaults are 1B, 3B. Choices are:

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

Enter 0 (zero) if none are wanted.

CHOICE(S):                                     1B,3B
```

Figure 5.1. (continued)

TITLE: FSI Cluster Controller CDE:
ECR/ECO: PROG. ID.:
SUBSYS: STATUS AS OF: 31OCT91

CODE	TASK	DUR	EFF	E-START	L-FINSH	FLT
0.	START	0.0	0.0	1SEP89	1SEP89	0
1.	Mgt Tasks & Milestones	0.0	0.0	13OCT89	13OCT89	451
1.1	CDE	30.0	153.6	1SEP89	13OCT89	451
2.	SW Planning and Reqs	0.0	0.0	13NOV89	13NOV89	24
2.1	SRD	0.0	0.0	1NOV89	1NOV89	0
2.1.1	Write & Release SRD	43.0	218.9	1SEP89	1NOV89	0
2.1.2	Software Cost Model	2.0	11.5	1SEP89	5SEP89	41
2.2	Level D Review	8.0	38.4	1NOV89	13NOV89	0
3.	SW Design Def & Arch	0.0	0.0	6FEB90	6FEB90	0
3.1	SDD	0.0	0.0	11JAN90	11JAN90	0
3.1.1	Write & Release SDD	43.0	218.9	13NOV89	11JAN90	0
3.1.2	Detailed WBS	19.0	96.0	13NOV89	8DEC89	24
3.2	SOM, First Draft	15.0	76.8	13NOV89	4DEC89	38
3.3	Devel Test Plan (DTP)	10.0	49.9	11JAN90	25JAN90	0
3.4	Level E Review	8.0	38.4	25JAN90	6FEB90	0
4.	SW Detail Design & Prod	0.0	0.0	22NOV90	22NOV90	0
4.1	SSD (Upgrade SDD)	0.0	0.0	12NOV90	12NOV90	0
4.1.1	Write Sects 1,2,3	8.0	38.4	6FEB90	16FEB90	186
4.1.2	Write Section 4	23.0	115.2	6FEB90	9MAR90	0
4.1.3	Write Section 5	53.0	268.8	9MAR90	23MAY90	114
4.1.4	Write Section 6	4.0	19.2	23MAY90	29MAY90	114
4.1.5	Write Section 7	15.0	76.8	6FEB90	27FEB90	179
4.1.6	Edit & Distrib SSD	5.0	23.0	5NOV90	12NOV90	0
4.2	SOM (Upgrade Draft SOM)	38.0	192.0	6FEB90	30MAR90	161
4.3	Prod, Integ, & Test	0.0	0.0	5NOV90	5NOV90	0
4.3.1	Function 1	0.0	0.0	5APR90	5APR90	0
4.3.1.1	Mod Prod/Integ	15.0	76.8	9MAR90	30MAR90	0
4.3.1.2	Function 1 Demo	4.0	19.2	30MAR90	5APR90	0
4.3.2	Function 2	0.0	0.0	2MAY90	2MAY90	0
4.3.2.1	Mod Prod/Integ	15.0	76.8	5APR90	26APR90	0
4.3.2.2	Function 2 Demo	4.0	19.2	26APR90	2MAY90	0
4.3.3	Function 3	0.0	0.0	29MAY90	29MAY90	0

4.3.3.1	Mod Prod/Integ	15.0	76.8	2MAY90	23MAY90	0
4.3.3.2	Function 3 Demo	4.0	19.2	23MAY90	29MAY90	0
4.3.4	Function 4	0.0	0.0	25JUN90	25JUN90	0
4.3.4.1	Mod Prod/Integ	15.0	76.8	29MAY90	19JUN90	0
4.3.4.2	Function 4 Demo	4.0	19.2	19JUN90	25JUN90	0
4.3.5	Function 5	0.0	0.0	20JUL90	20JUL90	0
4.3.5.1	Mod Prod/Integ	15.0	76.8	25JUN90	16JUL90	0
4.3.5.2	Function 5 Demo	4.0	19.2	16JUL90	20JUL90	0
4.3.6	Function 6	0.0	0.0	16AUG90	16AUG90	0
4.3.6.1	Mod Prod/Integ	15.0	76.8	20JUL90	10AUG90	0

Figure 5.2. SOFTCOST Pert Chart.

TITLE: FSI Cluster Controller CDE:
ECR/ECO: PROG. ID.:
SUBSYS: STATUS AS OF: 31OCT91

CODE	TASK	DUR	EFF	E-START	L-FINSH	FLT
4.3.6.2	Function 6 Demo	4.0	19.2	10AUG90	16AUG90	0
4.3.7	Function 7	0.0	0.0	12SEP90	12SEP90	0
4.3.7.1	Mod Prod/Integ	15.0	76.8	16AUG90	6SEP90	0
4.3.7.2	Function 7 Demo	4.0	19.2	6SEP90	12SEP90	0
4.3.8	Function 8	0.0	0.0	9OCT90	9OCT90	0
4.3.8.1	Mod Prod/Integ	15.0	76.8	12SEP90	3OCT90	0
4.3.8.2	Function 8 Demo	4.0	19.2	3OCT90	9OCT90	0
4.3.9	Function 9	0.0	0.0	5NOV90	5NOV90	0
4.3.9.1	Mod Prod/Integ	15.0	76.8	9OCT90	30OCT90	0
4.3.9.2	Function 9 Demo	4.0	19.2	30OCT90	5NOV90	0
4.4	STT, Working Draft	15.0	76.8	6FEB90	27FEB90	179
4.5	Special Tasks	0.0	0.0	26FEB90	26FEB90	185
4.5.1	Support software	14.0	73.0	6FEB90	26FEB90	185
4.5.2	Other	8.0	38.4	6FEB90	16FEB90	191
4.6	Final Demo Test Rvw	8.0	38.4	12NOV90	22NOV90	0
5.	Combined Subsys Tests	0.0	0.0	2APR91	2APR91	0
5.1	Sec 338 Lab CST	30.0	153.6	22NOV90	3JAN91	0
5.2	CTA-21 or SIF CST	30.0	153.6	3JAN91	14FEB91	0
5.3	DSCC-10/11 CST	30.0	153.6	14FEB91	28MAR91	0
5.4	Preliminary STT	18.0	92.2	22NOV90	18DEC90	42
5.5	Prelim Accept Tests	30.0	153.6	18DEC90	29JAN91	42
5.6	Acc Readiness Rvw	3.0	15.4	28MAR91	2APR91	0
6.	SW Test and Transfer	0.0	0.0	8JUL91	8JUL91	0
6.1	Complete STT	8.0	42.2	2APR91	12APR91	0
6.2	Complete SSD	8.0	42.2	2APR91	12APR91	53
6.3	Acceptance Tests	23.0	115.2	12APR91	15MAY91	0
6.4	Soak Tests	30.0	153.6	15MAY91	26JUN91	0
6.5	Transfer Review	8.0	38.4	26JUN91	8JUL91	0
6.6	Software Transfer	0.0	0.0	8JUL91	8JUL91	0
FINISH		0.0	0.0	8JUL91	8JUL91	0

Figure 5.2. (continued)

Figure 5.3. SOFTCOST Gantt Chart.

Figure 5.3. (continued)

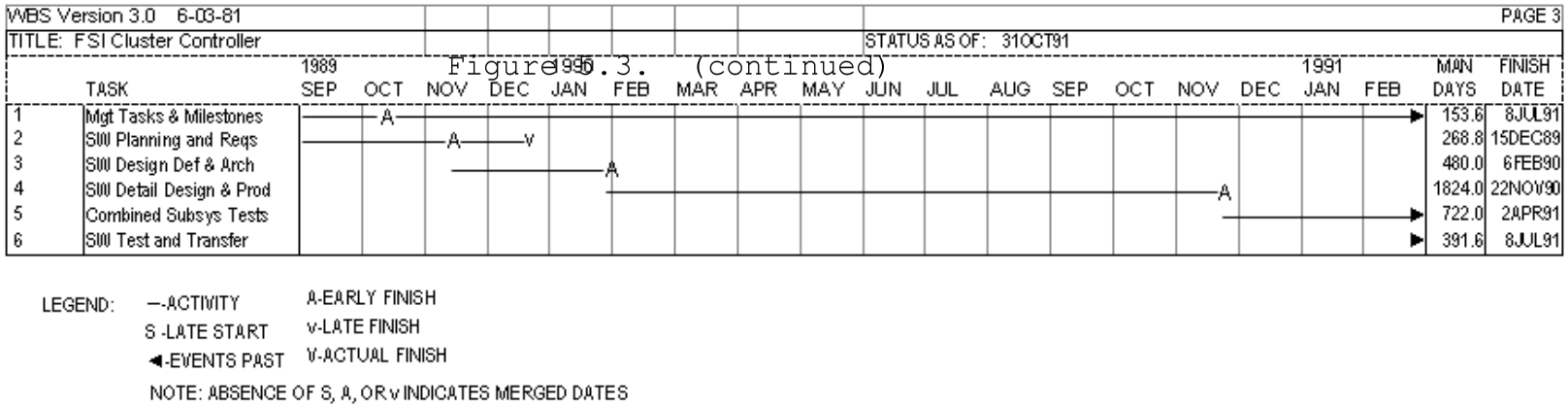
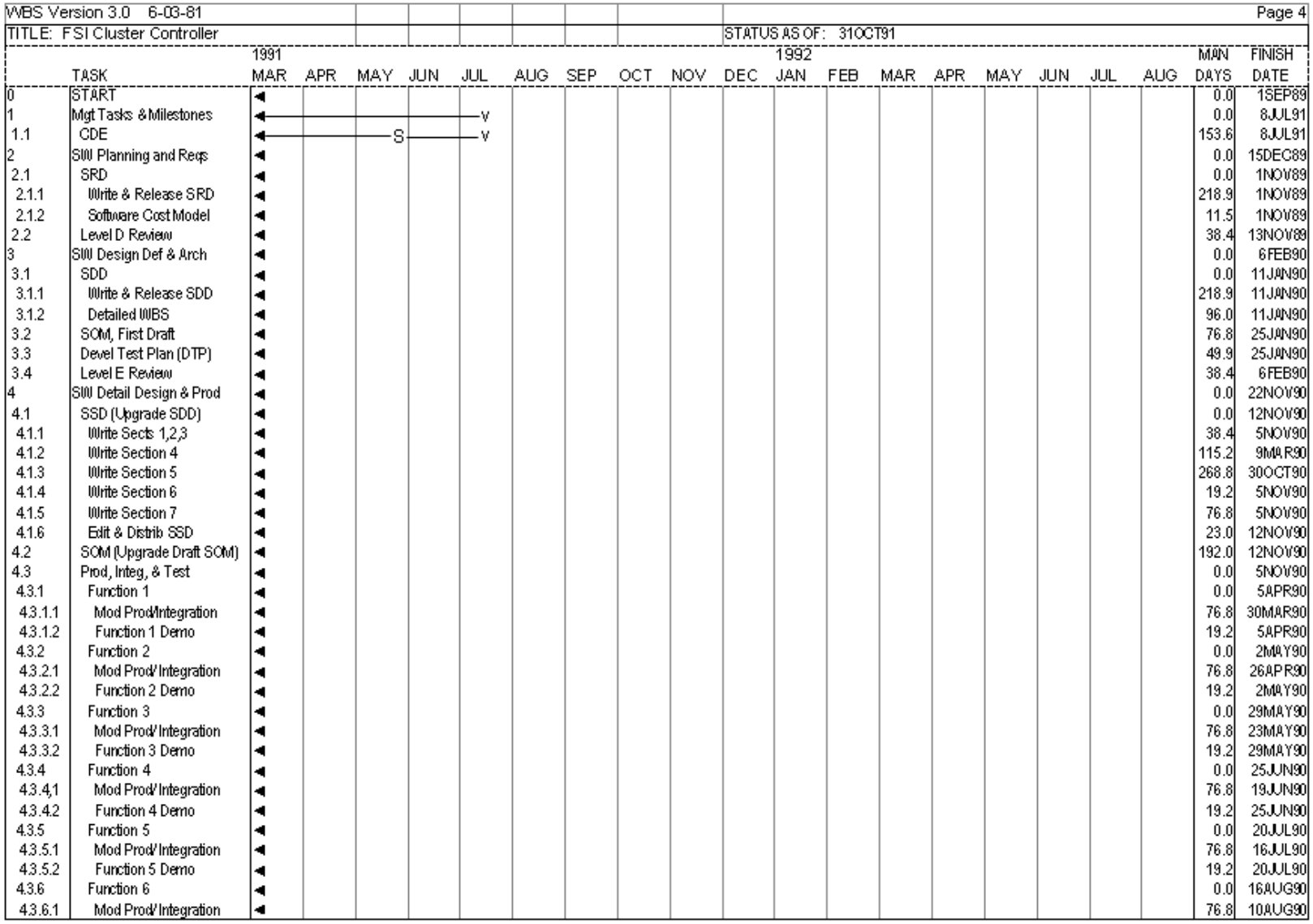


Figure 5.3. (continued)

Figure 5.3. (continued)



WBS Version 3.0 6-03-81 — Page 4

TITLE: FSI Cluster Controller — STATUS AS OF: 31OCT91

	TASK	1991 MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	1992 JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	MAN DAYS	FINISH DATE
0	START	◄																		0.0	1SEP89
1	Mgt Tasks & Milestones	◄				v														0.0	8JUL91
1.1	CDE	◄		S		v														153.6	8JUL91
2	SW Planning and Reqs	◄																		0.0	15DEC89
2.1	SRD	◄																		0.0	1NOV89
2.1.1	Write & Release SRD	◄																		218.9	1NOV89
2.1.2	Software Cost Model	◄																		11.5	1NOV89
2.2	Level D Review	◄																		38.4	13NOV89
3	SW Design Def & Arch	◄																		0.0	6FEB90
3.1	SDD	◄																		0.0	11JAN90
3.1.1	Write & Release SDD	◄																		218.9	11JAN90
3.1.2	Detailed WBS	◄																		96.0	11JAN90
3.2	SOM, First Draft	◄																		76.8	25JAN90
3.3	Devel Test Plan (DTP)	◄																		49.9	25JAN90
3.4	Level E Review	◄																		38.4	6FEB90
4	SW Detail Design & Prod	◄																		0.0	22NOV90
4.1	SSD (Upgrade SDD)	◄																		0.0	12NOV90
4.1.1	Write Sects 1,2,3	◄																		38.4	5NOV90
4.1.2	Write Section 4	◄																		115.2	9MAR90
4.1.3	Write Section 5	◄																		268.8	30OCT90
4.1.4	Write Section 6	◄																		19.2	5NOV90
4.1.5	Write Section 7	◄																		76.8	5NOV90
4.1.6	Edit & Distrib SSD	◄																		23.0	12NOV90
4.2	SOM (Upgrade Draft SOM)	◄																		192.0	12NOV90
4.3	Prod, Integ, & Test	◄																		0.0	5NOV90
4.3.1	Function 1	◄																		0.0	5APR90
4.3.1.1	Mod Prod/Integration	◄																		76.8	30MAR90
4.3.1.2	Function 1 Demo	◄																		19.2	5APR90
4.3.2	Function 2	◄																		0.0	2MAY90
4.3.2.1	Mod Prod/Integration	◄																		76.8	26APR90
4.3.2.2	Function 2 Demo	◄																		19.2	2MAY90
4.3.3	Function 3	◄																		0.0	29MAY90
4.3.3.1	Mod Prod/Integration	◄																		76.8	23MAY90
4.3.3.2	Function 3 Demo	◄																		19.2	29MAY90
4.3.4	Function 4	◄																		0.0	25JUN90
4.3.4.1	Mod Prod/Integration	◄																		76.8	19JUN90
4.3.4.2	Function 4 Demo	◄																		19.2	25JUN90
4.3.5	Function 5	◄																		0.0	20JUL90
4.3.5.1	Mod Prod/Integration	◄																		76.8	16JUL90
4.3.5.2	Function 5 Demo	◄																		19.2	20JUL90
4.3.6	Function 6	◄																		0.0	16AUG90
4.3.6.1	Mod Prod/Integration	◄																		76.8	10AUG90

Figure 5.3. (continued)

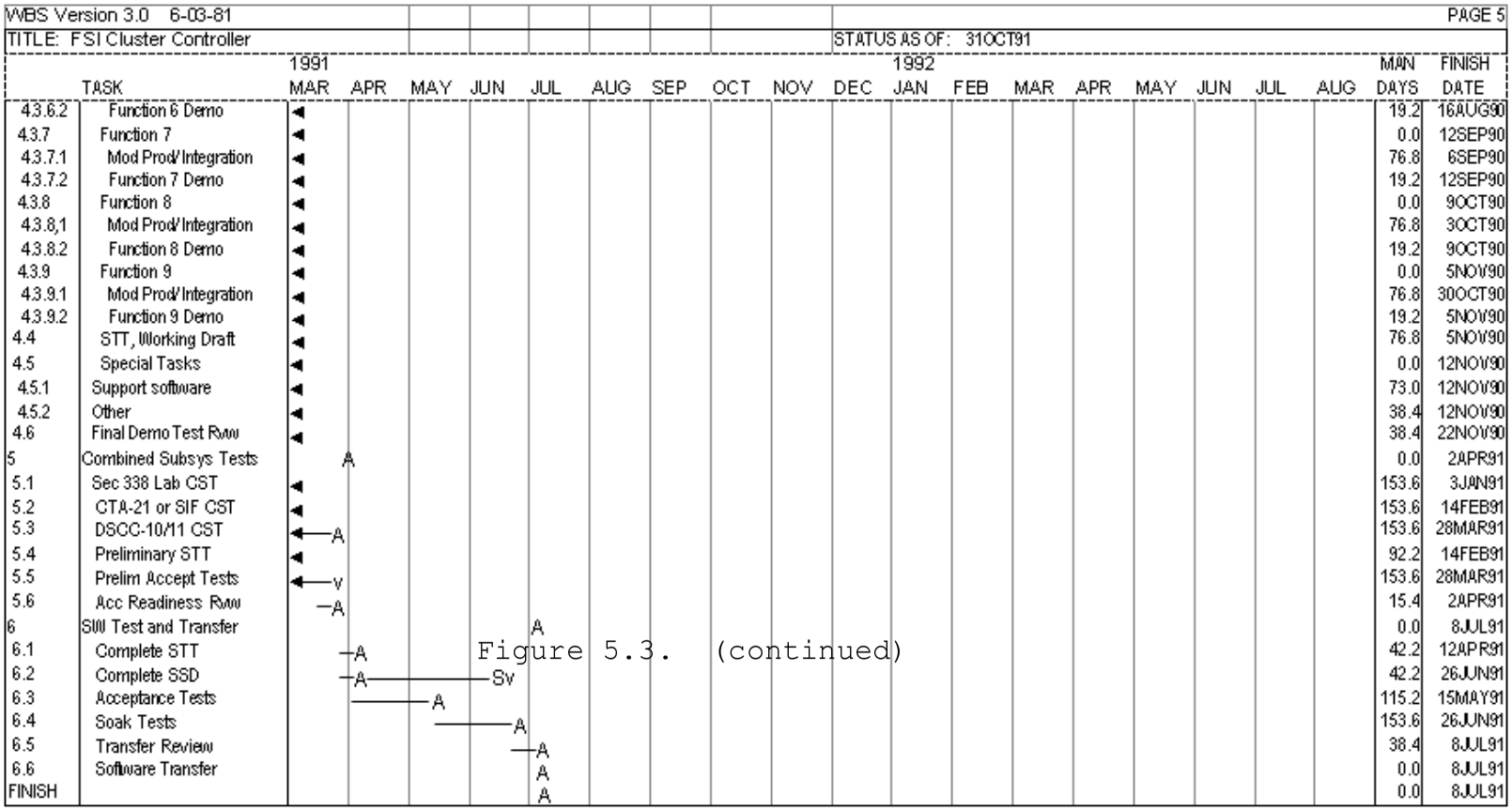


Figure 5.3. (continued)

Figure 5.3. (continued)

Figure 5.3. (continued)

TITLE: FSI Data -- Biased CDE:
ECR/ECO: PROG. ID.:
SUBSYS: Date Estimated: 01NOV91
Model Data Version 3.0 6-02-81

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)?	150
	Expected value, kilo-lines executable source?	120
	Minimum value, kilo-lines executable source (99% confidence level)?	100
2.	How much code exists in modules requiring modification?	
	Maximum value, kilo-lines executable source (99% confidence level)?	0
8.	Expected percentage of code to be developed actually delivered (0-90, 91-99, 100)?	100
9.	How many different kinds of input/output data items per 1000 lines of new or modified code (>80, 16-80, 0-15)?	0-15
10.	Overall complexity of program and data base architecture (high, medium, low)?	LOW
11.	Complexity of code logical design(high, medium, low)?	LOW
12.	What percent of the programming task is in Assembly language?	10
13.	What percent of the new or modified code must be storage-optimized?	30
14.	What percent of the new or modified code must be timing-optimized?	60
15.	What percent of the total programming task is 'easy'?	10
16.	What percent of the total programming task is 'hard'?	90
17.	When is work to start, on the (FRD/FDD, SRD, SDD)?	SDD
18.	What percent of the total program requirements will be established and stable before design, and will not be altered before delivery?	30
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?	60
21.	Complexity of program functional requirements (high, medium, low)?	LOW
22.	Expected user involvement in requirements definition (much, some, none)?	NONE
23.	Customer experience in application area (much, none, some)?	SOME
24.	Customer/implementer organizational interface complexity (high, normal, low)?	LOW

Figure 5.4. SOFTCOST Output--Biased Data.

25.	Interfaces with other SW development projects or organizations (many, few, none)?	NONE
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)?	>50
29.	Previous programmer experience with application of similar or greater size and complexity (minimal, average, extensive)?	EXTENSIVE
30.	What is the average staff experience, in years, obtained from work similar to that required in the task being estimated?	5
31.	Previous experience with operational computer to be used (minimal, average, extensive)?	EXTENSIVE
32.	Previous experience with programming language(s) to be used (minimal, average, extensive)?	EXTENSIVE
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
36.	Use of design and code inspections (low, QA, peer)?	PEER
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)?	>90
40.	Development computer access mode (remote, scheduled, demand)?	DEMAND
41.	Percent of development computer access availability (<30, 30-60, >60)?	>60
42.	Quality of SW development tools and environment (poor, ok, good)?	GOOD
43.	Maturity of system and support software (buggy, ok, good)?	GOOD
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

Figure 5.4. (continued)

```
Estimated Overall Parameters:
                                                =average value
                                      +1-sigma                  -1-sigma
Adjusted Lines of code= 121667 SLEC, +/- 9365 SLEC
  131031       112302
Effort=175.0 person-months
304.3          100.7
Staff productivity= 695 SLEC/staff-month
 1208          400
Duration= 30.7 months
 37.4          25.2
Avg. Staff=  5.7
  9.9          3.3
Documentation= 6241 pages   $187.2K
 7564          5150          $226.9K          $154.5K
Computer CPU time= 5535 hours     $0.0K
 8354          3667           $0.0K           $0.0K

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

Values specified are:
  Kilo-lines of code:                          121.67
  Effort (person-months):                        192.0
  Duration (months):                             24.0
  Average staff (persons):                        8.0

   For the numbers entered, a reasonableness check indicates that
the average project would produce 116601 lines of code, using 192 staff-months
of resources and 24 months of duration, with an average staff of 8 persons,
for a productivity of 607 SLEC/staff-month.

   The level of confidence in delivering 121667 lines of code,
on-time and within resources= 8 %.

Is output to be saved in a file?                          NO

Name of output file to be created:                    SCRATCH

Schedule start date:                                  01SEP89

 Select desired outputs and output media.  Defaults are 1B, 3B. Choices are:
          1=Gantt Chart                 A=file
          2=PERT data, 132 width        B=line printer
          3=PERT data, 80 width
          4=PERT output interface file only
Enter 0 (zero) if none are wanted.

CHOICE(S):                                            0
```

Figure 5.4. (continued)

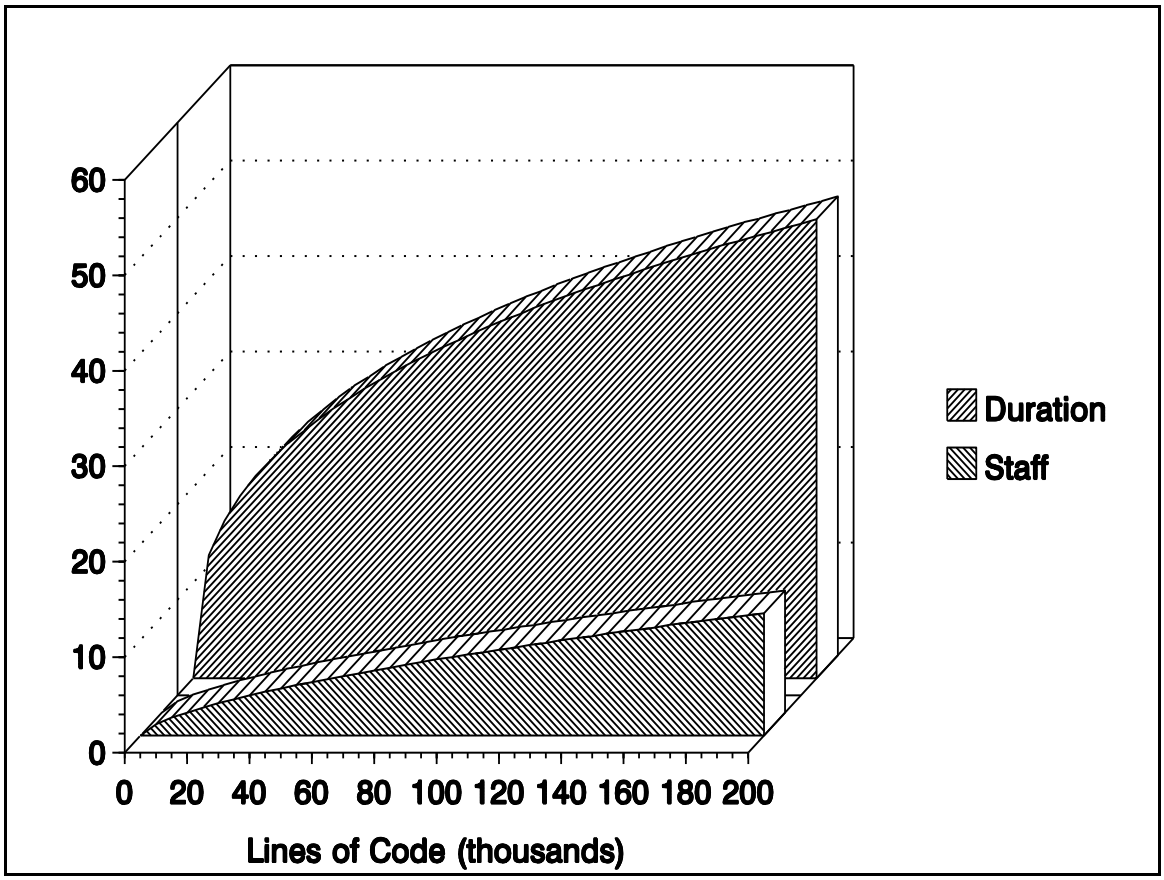


Figure 5.5. FSI Parameters Applied to SOFTCOST with Varying Lines of Code.

```
**********       SOFTWARE COST MODEL       **********
            copyright 1986, Gordon Group

Description: FSI Sample Data
Basis of Est.: Calibrated from Actuals; Default Cost Driver Values
Development Mode: SEMIDETACHED
Thousands of New Source Instructions (KDSI):     120.00

Thousands of Adapted Source Instructions (KDSI):      0.00
   Percentage Requiring Design Modification:       0%
   Percentage Requiring Code Modification:         0%
   Percentage Requiring Integration Modification:  0%

Thousands of Converted Source Instructions (KDSI):      0.00
   Percentage Requiring Design Modification:       0%
   Percentage Requiring Code Modification:         0%
   Percentage Requiring Integration Modification:  0%
   Conversion Analysis and Planning (0-LOW ... 5-HIGH): 0

PRODUCT ATTRIBUTES
RELY  Required Software Reliability:  HIGH
DATA  Database Size:                  NOMINAL
CPLX  Product Complexity:             XHIGH

COMPUTER ATTRIBUTES
TIME  Execution Time Constraint:      HIGH
STOR  Main Storage Constraint:        NOMINAL
VIRT  Virtual Machine Volatility:     LOW
TURN  Computer Turnaround Time:    LOW

PERSONNEL ATTRIBUTES
ACAP  Analyst Capability:                VHIGH
AEXP  Applications Experience:           VHIGH
PCAP  Programmer Capability:             NOMINAL
VEXP  Virtual Machine Experience:        NOMINAL
LEXP  Programming Language Experience:  HIGH

PROJECT ATTRIBUTES
MODP  Use of Modern Programming Practices:   HIGH
TOOL  Use of Software Tools:                 HIGH
SCED  Required Development Schedule:         NOMINAL

OUTPUTS
 EFFORT:
   192.00 man-months

 PRODUCTIVITY:
   625.00 new-equivalent delivered source instructions/man-month

 SCHEDULE:
    24.00 months

 AVERAGE STAFFING:
     8.00 full-time equivalent software personnel, FSP
```

Figure 5.6. BYL Cost Driver Report.

BYL Maintenance Report

********** SOFTWARE COST MODEL **********
copyright 1987, Gordon Group

Description: FSI Sample Data

Basis of Estimate: Calibrated from Actuals; Default Cost Driver Values
Maintenance Mode: SEMIDETACHED
Thousands of Source Instructions in Product (KDSI): 120.00
Expected Annual Change Traffic (ACT): 20%
Equiv. Annual Conversion of Deliv. Source Instr. (KDSI): 24.00
Expected Operational Product Life (Months): 36

PRODUCT ATTRIBUTES
RELY Required Software Reliability: HIGH
DATA Database Size: NOMINAL
CPLX Product Complexity: XHIGH

COMPUTER ATTRIBUTES
TIME Execution Time Constraint: HIGH
STOR Main Storage Constraint: NOMINAL
VIRT Virtual Machine Volatility: LOW
TURN Computer Turnaround Time: LOW

PERSONNEL ATTRIBUTES
ACAP Analyst Capability: VHIGH
AEXP Applications Experience: VHIGH
PCAP Programmer Capability: HIGH
VEXP Virtual Machine Experience: HIGH
LEXP Programming Language Experience: HIGH

PROJECT ATTRIBUTES
MODP Use of Modern Programming Practices: HIGH
TOOL Use of Software Tools: HIGH
SCED Required Development Schedule: NOMINAL

OUTPUTS
Expected Annual Effort:
23.66 man-months

Maintenance Productivity:
1014.46 changed delivered source instructions/man-month

Average Maintenance Staffing Requirements:
1.97 full-time equivalent software personnel, FSP

Life-Cycle Maintenance Effort:
70.97 man-months

Figure 5.7. BYL Maintenance Report.

BYL Phase Distribution Report

********** SOFTWARE COST MODEL **********
copyright 1986, Gordon Group

Description: FSI Sample Data
Basis of Est.: Calibrated from Actuals; Default Cost Driver Values
PRODUCT SIZE:
120.00 thousand new-equivalent delivered source instructions
PROJECT SCHEDULE:
24.00 months
ESTIMATED EFFORT:
192.00 man-months

Phase	Product Design	Programming	Integration and Test
DISTRIBUTION	17.00%	55.25%	27.75%
Activity percentage			
Requirements analysis	12.50	4.00	2.50
Product design	41.00	8.00	5.00
Programming	13.46	56.50	38.83
Test planning	5.96	5.46	3.00
Verification & validation	7.46	8.46	28.58
Project office	10.08	6.04	7.04
Configuration mgmt/QA	2.50	6.50	8.00
Manuals	7.04	5.04	7.04
EFFORT	32.64 MM	106.08 MM	53.28 MM
Activity man-months			
Requirements analysis	4.08	4.24	1.33
Product design	13.38	8.49	2.66
Programming	4.39	59.94	20.69
Test planning	1.94	5.79	1.60
Verification & validation	2.43	8.97	15.23
Project office	3.29	6.41	3.75
Configuration mgmt/QA	0.82	6.90	4.26
Manuals	2.30	5.35	3.75
SCHEDULE	26.92%	44.33%	28.75%
Duration schedule months	6.46	10.64	6.90
AVERAGE STAFFING	5.05	9.97	7.72
Full-time software personnel			
Requirements analysis	0.63	0.40	0.19
Product design	2.07	0.80	0.39
Programming	0.68	5.63	3.00
Test planning	0.30	0.54	0.23
Verification & validation	0.38	0.84	2.21
Project office	0.51	0.60	0.54
Configuration mgmt/QA	0.13	0.65	0.62
Manuals	0.36	0.50	0.54

Figure 5.8. Phase Distribution Report.

********** SOFTWARE COST MODEL **********
copyright 1987, Gordon Group

Description: FSI Sample Data
Basis of Est.: Calibrated from Actuals; Default Cost Driver Values

Life-Cycle Effort Distribution	Activity Man-Months
PRELIMINARY PLANNING & REQUIREMENTS ANALYSIS	
Schedule duration: 5.24 months;	
Requirements analysis	6.06
Product design	2.35
Programming	0.80
Test planning	0.53
Verification & validation	0.48
Project office	1.69
Configuration mgmt/QA	0.40
Manuals	0.68
Total Effort	12.98MM
AGGREGATE SOFTWARE DEVELOPMENT	
Schedule duration: 24.00 months	
Requirements analysis	9.66
Product design	24.53
Programming	85.02
Test planning	9.33
Verification & validation	26.64
Project office	13.45
Configuration mgmt/QA	11.97
Manuals	11.40
Total Effort	192.00MM
SOFTWARE MAINTENANCE	
Operational life: 36 months	

	Annually	Operational Life
Requirements analysis	1.43	4.29
Product design	2.84	8.52
Programming	9.70	29.10
Test planning	0.94	2.81
Verification & validation	3.18	9.55
Project office	1.67	5.00
Configuration mgmt/QA	1.42	4.26
Manuals	2.48	7.45
Annual Effort	23.66MM	
Total Effort		70.97MM
Total Life-Cycle Effort		275.95MM

Figure 5.9. BYL Life-Cycle Report.

********** SOFTWARE COST MODEL **********
copyright 1987, Gordon Group

Description: FSI Sample Data
Basis of Est.: Calibrated from Actuals; Default Cost Driver Values

Life-Cycle Costs Distribution	Current Dollars
PRELIMINARY PLANNING & REQUIREMENTS ANALYSIS	
Schedule duration: 5.24 months;	
Requirements analysis	32192.53
Product design	6619.19
Programming	1093.40
Test planning	532.00
Verification & validation	952.00
Project office	1902.60
Configuration mgmt/QA	2142.20
Manuals	762.30
Total Cost $	46196.23
AGGREGATE SOFTWARE DEVELOPMENT	
Schedule duration: 24.00 months	
Requirements analysis	51298.08
Product design	69207.03
Programming	116900.30
Test planning	9333.40
Verification & validation	53272.40
Project office	15133.50
Configuration mgmt/QA	63615.74
Manuals	12823.20
Total Cost $	391583.64
SOFTWARE MAINTENANCE	
Operational life: 36 months	

	Annually	Operational Life
Requirements analysis	7594.07	22782.21
Product design	8008.70	24026.11
Programming	13337.20	40011.59
Test planning	936.46	2809.39
Verification & validation	6367.94	19103.83
Project office	1874.16	5622.47
Configuration mgmt/QA	7541.70	22625.09
Manuals	2794.60	8383.80
Annual Cost $	48454.83	
Total Cost		$ 145364.49
Total Life-Cycle Cost	$ 583144.36	

Figure 5.9. (continued)

********** SOFTWARE COST MODEL **********

Description: FSI Sample Data
Basis of Est.: Calibrated from Actuals; Default Cost Driver Values
PRODUCT SIZE:
120.00 thousand new-equivalent delivered source instructions
PROJECT SCHEDULE:
24.00 months
ESTIMATED EFFORT:
192.00 man-months

Aggregate Development		Preliminary Planning & Requirements Analysis (not included in Aggregate)
100.00%	DISTRIBUTION	7.00%
	Activity percentage	
5.03	Requirements analysis	45.08
12.78	Product design	17.46
44.28	Programming	5.92
4.86	Test planning	3.96
13.87	Verification & validation	3.54
7.01	Project office	12.58
6.24	Configuration mgmt/QA	3.00
5.94	Manuals	5.04
192.00 MM	EFFORT	12.98 MM
	Activity man-months	
9.66	Requirements analysis	6.06
24.53	Product design	2.35
85.02	Programming	0.80
9.33	Test planning	0.53
26.64	Verification & validation	0.48
13.45	Project office	1.69
11.97	Configuration mgmt/QA	0.40
11.40	Manuals	0.68
100.00%	SCHEDULE	21.83%
24.00	Duration schedule months	5.24
8.00	AVERAGE STAFFING	2.56
	Full-time software personnel	
0.40	Requirements analysis	1.16
1.02	Product design	0.45
3.54	Programming	0.15
0.39	Test planning	0.10
1.11	Verification & validation	0.09
0.56	Project office	0.32
0.50	Configuration mgmt/QA	0.08
0.47	Manuals	0.13

Figure 5.10. Aggregate Activity Report.

********** SOFTWARE COST MODEL **********
copyright 1986, Gordon Group

Description: FSI Sample Data
Basis of Est.: Calibrated from Actuals; Default Cost Driver Values
COSTS PER EMPLOYEE-TYPE PER MAN-MONTH ********************************

Requirements Analysts :	5313.00
Product Designers :	2821.00
Computer Programmers, Programmer Analysts :	1375.00
Test Planners & Test Engineers :	1000.00
Value Engineers & Product Analysts :	2000.00
Project Office Personnel (non-clerical) :	1125.00
QA Specialists & Config Mgmt/Librarians :	5313.00
Software Manual Writers/Technical Writers :	1125.00

PHASE DISTRIBUTION OF DEVELOPMENT **********
COSTS PER EMPLOYEE-TYPE

Phase	Product Design	Programming	Integration and Test	
PHASE COSTS	82906.20	202491.60	106185.84	
Activity costs				
Requirements analysis		21677.04	22544.12	7076.92
Product design		37751.75	23940.13	7515.14
Programming		6040.10	82410.90	28449.30
Test planning		1944.80	5790.20	1598.40
Verification & validation		4868.80	17945.20	30458.40
Project office		3702.60	7210.12	4220.77
Configuration mgmt/QA		4335.41	36634.20	22646.13
Manuals		2585.70	6016.72	4220.77
Duration schedule months		6.46	10.64	6.90
Average cost per month		12833.78	19031.17	15389.25

AGGREGATE DISTRIBUTION OF COSTS **********

Aggregate Development		Preliminary Planning and Requirements (not in Aggregate)
391583.64	COSTS	46196.23
	Activity costs	
51298.08	Requirements analysis	32192.53
69207.03	Product design	6619.19
116900.30	Programming	1093.40
9333.40	Test planning	532.00
53272.40	Verification & validation	952.00
15133.50	Project office	1902.60
63615.74	Configuration mgmt/QA	2142.20
12823.20	Manuals	762.30
24.00	Duration schedule months	5.24
16315.99	Average cost per month	8816.07

PROJECTED COSTS **********

Software Development : $ 391584
Preliminary Planning & Requirements Analysis : $ 46196
>>> Overall Project Costs : $ 437780

Figure 5.11. BYL Cash Flow Report.

BYL Function Point Report

********** SOFTWARE COST MODEL **********
copyright 1986, Gordon Group

Description: FSI Sample Data

FUNCTION COUNT & COMPLEXITY **********

	Simple	Average	Complex	Unadjusted Function Points
External Input/Inquiry	5x3	10x4	14x6	139
External Output	5x4	10x5	14x7	168
Logical Internal File	5x7	9x10	11x15	290
External Interface File	4x5	9x7	11x10	193

Total unadjusted function points: 790

PROCESSING COMPLEXITIES **********

Characteristic	Rating	Adjustment Factor
Data Communications	Significant	+1.5%
Distributed Functions	Significant	+1.5%
Performance	Strong	+2.5%
Heavily Used Config	Strong	+2.5%
Transaction Rate	Significant	+1.5%
Online Data Entry	Significant	+1.5%
End User Efficiency	Average	+0.5%
Online Update	Moderate	-0.5%
Complex Processing	Significant	+1.5%
Reusability	Strong	+2.5%
Installation Ease	Insignificant	-1.5%
Operational Ease	Significant	+1.5%
Multiple Sites	Significant	+1.5%
Facilitate Change	Strong	+2.5%

Net adjustment factor : +19.0%
Total adjusted function points : 940.10

Compiler : C Coefficient : 128
Estimated delivered source instructions : 120333

Figure 5.12. BYL Function Point Report.

<<<COST SUMMARY>>>

Project: ALTERNATIVE A Categories: 33 Interest Rate: 10.0% Page
File Name: SEA19-A.DBF Periods: 13 Inflation Rate: 0.0% 1

TYPE	CATEGORY	REAL DOLLARS%	DISCOUNTED%
P	ALTERNATIVE A	2513453.00	1663224.37
C	R&D	637250.00	519049.75
C	Customer Costs	149181.00	124276.46
C	System Management	133221.00	109767.37
C	Production Planning	15960.00	14509.09
C	Supplier Costs	488069.00	394773.28
C	System Management	113850.00	94022.04
C	Product Planning	21130.00	18294.37
C	Engineering Design	237016.00	192293.37
C	Design Docs	48425.00	39338.54
C	System Test & Eval	67648.00	50824.94
C	Prod & Const	1112699.00	770851.06
C	Customer Costs	112497.00	73178.06
C	System/Product Mgmt	112497.00	73178.06
C	Supplier Costs	1000202.00	697673.00
C	I.E. & Op Analysis	68200.00	47991.00
C	Manufacturing	684000.00	471512.68
C	Recurring	635100.00	431099.43
C	Non-Recurring	48900.00	40413.22
C	Quality Control	78502.00	53669.95
C	Initial Log Supp	169500.00	124499.38
C	Supply Support	48000.00	32248.16
C	Test & Support Equip	70000.00	52967.69
C	Technical Data	5100.00	4214.87
C	Personnel Training	46400.00	35068.64
C	Ops & Support	763504.00	373323.40
C	Operating Personnel	24978.00	12076.83
C	Transportation	268000.00	131059.27
C	Unscheduled Maint	123130.00	60314.31
C	Maint Facilities	4626.00	2266.19
C	Supply Support	295920.00	144959.71
C	Maint Pers Training	9100.00	4322.75
C	Test & Support Equi	32500.00	15753.82
C	Transport & Handlin	5250.00	2570.51

Figure 5.13. LCCC Cost Summary Report.

11/01/91

<<<COST CATEGORY>>>
ALTERNATIVE A

Sub-Categories: 4 PROJECT Interest Rate: 10.0% Page
Next Higher: ALTERNATIVE A Inflation Rate: 0.0% 1
Life Cycle Cost: 2513453.00 Discounted Cost: 1663224.37

PERIOD	REAL DOLLARS	DISCOUNTED	PERIOD	REAL DOLLARS	DISCOUNTED
1	162712.00	147920.00			
2	279757.00	231204.12			
3	610569.00	458729.53			
4	417062.00	284858.96			
5	450983.00	280024.96			
6	90954.00	51341.16			
7	90954.00	46673.78			
8	90954.00	42430.71			
9	90954.00	38573.37			
10	90954.00	35066.70			
11	90954.00	31878.82			
12	34884.00	11115.11			
13	11762.00	3407.03			

Figure 5.14. LCCC Cost Category Report.

CHAPTER VI - CONCLUSIONS

As earlier research indicated, while excellent software cost modeling tools are available, hardware cost modeling tools which are not specific in nature are difficult to find.

The evaluation process led to the conclusion that *Before You Leap* is an excellent software cost modeling tool which merits further investigation. Such future research should focus on verifying the indications stemming from this research: that the inclusion of multiple FSI projects into the knowledge base will ultimately enable the package to produce accurate results pertaining to FSI software development. If such accurate results are forthcoming, they will serve as data for the "make" portion of the software partition.

The lack of a suitable hardware cost modeling tool would seem to indicate that a special purpose tool will have to be developed during the course of future research. It is recommended that the new tool be patterned after *Before You Leap*. For example, the use of development cost drivers, the use of maintenance cost drivers, the built in ability to modify cost driver values, and the overall interface scheme should be incorporated into the new package. Cost drivers are discussed in an earlier chapter in this paper which explores potential parameters. That discussion should assist in selecting the proper parameters for the hardware cost modeling tool. If such a model is successfully developed, it should serve as an excellent source of data for the "make" portion of the hardware partition.

Based on the assumption that a fully functional version of the *Life-Cycle Cost Calculator* will be made available by Virginia Polytechnic Institute, it is recommended that the cost breakdown structure for an embedded controller, which is detailed earlier in this paper, be entered into the *Life-Cycle Cost Calculator*. Using data gathered from various sources, including *Before You Leap* and the new hardware cost modeling tool, the *Life-Cycle Cost Calculator* can determine overall costs projected over the entire life span of the project, incorporating such important factors as the discount rate.

The conclusion resulting from this research is that future FSI planning involving the generic life cycle cost model for an embedded controller can best be implemented by utilizing the *Life-Cycle Cost Calculator* and *Before You Leap*, and developing locally a hardware costing package. LCCC should be provided with the cost breakdown structure

resulting from this research. Costs for the software subsystem can be obtained from *Before You Leap*. A hardware costing package, similar in function and form to *Before You Leap*, will provide the costs for the hardware subsystem.

Other costs not included in these subsystems can be determined using the methods outlined in the chapter dealing with cost determination. When these costs are entered into the cost breakdown structure in the *Life-Cycle Cost Calculator*, the final results will be the life cycle cost estimate for the complete embedded controller project.

REFERENCES

[BLAN78]Blanchard, Benjamin S., Design and Manage to Life Cycle Cost, M/A Press, Portland, Oregon, 1978.

[BLAN81]Blanchard, Benjamin S., Logistic Engineering and Management, 2nd ed., Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1981.

[BLAN90]Blanchard, Benjamin S. and Fabrycky, Wolter J., Systems Engineering and Analysis, 2nd ed., Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1990.

[BOEH81]Boehm, Barry W., Software Engineering Economics, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1981.

[BROW85]Brown, Robert J. and Yanuck, Rudolph R., Introduction to Life Cycle Costing, The Fairmont Press, Inc., Atlanta, Georgia, 1985.

[CUTA90]Cutaia, Al, Technology Projection Modeling of Future Computer Systems, Prentice Hall, Englewood Cliffs, NJ, 1990.

[DENE83]de Neumann, Bernard, *Life Cycle Cost Models*, Electronic Systems Effectiveness and Life Cycle Costing, ed. Skwirzynski, J.K., Springer-Verlag, Berlin, Germany, 1983.

[DHIL89]Dhillon, B.S., Life Cycle Costing, Gordon and Breach Science Publishers, New York, New York, 1989.

[FABR80]Fabrycky, W.J. and Thuesen, G.J., Economic Decision Analysis, 2nd ed., Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1980.

[FERE74]Ferens, Daniel V. and Harris, Robert L., *Avionics Computer Software Operation and Support Cost Estimation*, Proceedings of the IEEE 1979 National Aerospace and Electronics Conference. Institute of Electrical and Electronic Engineers, Inc., New York, New York, 1974.

[FSII90]FSI, International, Inc., Requirements Specification for the Module Level Controller: FSI Project No. 3509, Lubbock, Texas, 1990.

[GATE76]Gates, Robert K. and Abraham, Michael J., Program LCC Documentation: Version 2, The Analytical Sciences Corporation, Reading, Massachusetts, 1976.

[GIBS79]Gibson, Keith and Miller, Sid, *Evolution of Low Life-Cycle-Cost Inertial Navigator - The N73 Strapdown System*, Proceedings of the IEEE 1979 National Aerospace and Electronics Conference, Institute of Electrical and Electronic Engineers, Inc., New York, New York, 1979.

[GORD87]The Gordon Group, Before You Leap: User's Guide, San Jose, California, 1987.

[GRIM74]Grimm, Major Richard W., *Fire Control Radar and Airborne Computer Cost Prediction Based on Technical Parameters*, Proceedings of the IEEE 1974 National Aerospace and Electronics Conference, Institute of Electrical and Electronic Engineers, Inc., New York, New York, 1974.

[GUPT83]Gupta, Yash P., *Life Cycle Cost Models and Associated Uncertainty*, Electronic Systems Effectiveness and Life Cycle Costing, ed. Skwirzynski, J.K., Springer-Verlag, Berlin, Germany, 1983.

[HUFF91]Huff, John, telephone conversation on Life Cycle Costing as it Pertains to Embedded Controllers, Wright-Patterson AFB, Ohio, July 1991.

[KNUS81]Knust, H., Report on the Navy Life Cycle Cost Model for the Sea Nymph Project, Naval Underwater Systems Center, Newport, Rhode Island, 1981.

[LAM88]Lam, Herman and O'Malley, John, Fundamentals of Computer Engineering: Logic Design and Microprocessors, John Wiley and Sons, New York, New York, 1988.

[MANT88]Mantei, Marilyn M. and Teorey, Toby J., *Cost/Benefit Analysis for Incorporating Human Factors in the Software Lifecycle*, Communications of the ACM, 31:428-439, April 1988.

[NASA89]National Aeronautics and Space Administration, Office of Safety, Reliability, Maintainability, and Quality Assurance, Information System Life-Cycle and Documentation Standards release 4.3, Washington, D.C., 1989.

[PERR74]Perrigo, W.R. and Easterday, J.L., *Avionic Equipment Reliability and Low Life Cycle Cost*, Proceedings of the IEEE 1974 National Aerospace and Electronics Conference, Institute of Electrical and Electronic Engineers, Inc., New York, New York, 1974.

[PFLE87]Pfleeger, Shari Lawrence, Software Engineering: The Production of Quality Software, Macmillan Publishing Company, New York, New York, 1987.

[PHIS76] Phister, Montgomery Jr., Data Processing Technology and Economics, The Santa Monica Publishing Company, Santa Monica, California, 1976.

[PRIE88]Priest, John W., Engineering Design for Producibility and Reliability, Marcel Dekker, Inc., New York, New York, 1988.

[REIC80]Reiche, H., *Life Cycle Cost*, Reliability and Maintainability of Electronic Systems, ed. Arsenault, J.E., and Roberts, J.A., Computer Science Press, Potomoc, Maryland, 1980.

[SELD79] Seldon, Robert M., Life Cycle Costing: A Better Method of Government Procurement, Westview Press, Boulder, Colorado, 1979.

[SHUP80]Shupe, Dean S., What Every Engineer Should Know About Economic Decision Analysis, Marcel Dekker, Inc., New York, New York, 1980.

[SOMM89]Sommerville, Ian, Software Engineering, 3rd ed., Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.

[STON79]Stone, Harold S., *Life-Cycle Cost Analysis of Instruction-Set Architecture Standardization for Military Computer Systems*, IEEE Computer, 35-46, April 1979.

[TAUS81]Tausworthe, Robert C., Deep Space Network Software Cost Estimation Model, Jet Propulsion Laboratory, Pasadena, California, 1981.

[TAUS89]Tausworthe, Robert C., User's Manual--DSN Software Cost Model, Jet Propulsion Laboratory, Pasadena, California, 1989.

[TAYL74]Taylor, James H., *A Realistic Approach to System Life Cycle Cost*, Proceedings of the IEEE 1974 National Aerospace and Electronics Conference, Institute of Electrical and Electronic Engineers, Inc., New York, New York, 1974.

[VIRG91]Virginia Polytechnic Institute and State University, Life-Cycle Cost Calculator, Blacksburg, Virginia, 1991.

[WOLV84]Wolverton, R.W., *Software Costing*, Handbook of Software Engineering, ed. Vick, C.R. and Ramamoorthy, C.V., Van Nostrand Reinhold Company, New York, New York, 1984.

APPENDIX - DESCRIPTION OF COST CATEGORIES

Cost Category
Method of Determination
Cost Category Description and Justification

Total system cost (C)
$C = [C_R + C_P + C_M + C_D]$ C_R = Research and Development cost C_P = Production cost C_M = Maintenance cost C_D = Retirement and Disposal cost
Includes all future life cycle costs associated with the acquisition, maintenance and subsequent disposition of the system.

Research and development (C_R)
$C_R = [C_{RM} + C_{RP} + C_{RF} + C_{RD} + C_{RT} + C_{RI}]$ C_{RM} = System/Product management cost C_{RP} = Product Planning cost C_{RF} = Functional Specification cost C_{RD} = Hardware/Software development partition C_{RT} = Integrated Testing C_{RI} = Integration and Testing
Includes all costs associated with program management, conceptual/feasibility studies, research and development, overall design, design documentation, design and implementation of system software, design and fabrication of hardware components and prototypes, and associated documentation. These costs are basically nonrecurring.

System/Program management (C_{RM})
$C_{RM} = \sum_{i=1}^{N} C_{RM_i}$ C_{RM_i} = Cost of specific activity i N = Number of management activities
Cost of management activities applicable to conceptual/feasibility studies, product research, engineering design, system development, system test and evaluation, and related documentation. Such costs cover the program manager and staff; marketing; contracts; procurement; configuration management; logistics management; data management; etc. Management functions relate to C_{RP}, C_{RR}, C_{RF}, and C_{RD}.

Product Planning (C_{RP})
$C_{RP} = \sum_{i=1}^{N} C_{RP_i}$ C_{RP_i} = Cost of specified activity i N = Number of planning activities
Product planning includes a market analysis to identify the need for a system because of deficiencies or problems, as well as feasibility studies to determine and/or justify a need for a specific requirement. This also involves the development of operational and maintenance concepts, preparation of technical and program proposals, development of program plans and specifications, development of financial plans, etc.

Functional Specification of Microprocessor System (C_{RF})
$C_{RF} = \sum_{i=1}^{N} C_{RF_i}$ C_{RF_i} = Cost of specific design activity i N = Number of design activities
Includes all initial design effort associated with system/equipment definition and development. Specific areas include system engineering; design engineering (electrical, mechanical, drafting); reliability and maintainability engineering; human factors; functional analysis and allocation; logistic support analysis; components; producibility; standardization; safety; etc. Design associated with modifications is covered in C_{ON}. Conceptual design includes effort oriented to defining mission scenarios, system operational requirements analysis and definition, preliminary maintenance concept, etc. Preliminary design includes such tasks as evaluation of alternative design configurations, evaluation of logistic support requirements, preliminary design and analysis of the chosen configuration, and the actual system specifications.

Hardware/Software Development Partition (C_{RD})
$C_{RD} = [C_{RDH} + C_{RDS}]$ C_{RDH} = Development costs associated with the hardware portion of the project C_{RDS} = Development costs associated with the software portion of the project
When portions of the system are implemented in hardware and others in software, related development costs in both areas must be taken into account.

Hardware Costs (C_{RDH})
$C_{RDH} = [C_{RDHC} + C_{RDHP} + C_{RDHI} + C_{RDHT}]$ C_{RDHC} = Design Completion C_{RDHP} = Hardware Make/Buy Partition C_{RDHI} = Hardware Integration C_{RDHT} = Hardware Test and Evaluation
Hardware costs involve the completion of the design initiated in C_{RF}, the decision to make or buy hardware components, the integration of all hardware components, and test and evaluation of the integrated hardware.

Design Completion of Hardware (C_{RDHC})
$C_{RDHC} = \sum_{i=1}^{N} C_{RDHC_i}$ C_{RDHC_i} = Cost of specific activity i N = Number of activities
The design completion phase involves development of the detailed design, design support, and design review. Detailed design includes the design and definition of units, assemblies, subassemblies, and the specifications of components and parts. Design support includes the costs of draftsmen, technical publication specialists, model builders, laboratory technicians, component-parts specialists, test technicians, and computer-aided design specialists. Design reviews include formal reviews such as the equipment design review, which is conducted during the detailed design phase, and the critical design review, which is conducted after the detailed design review to verify the adequacy and producibility of the design.

Hardware Make/Buy Partition (C_{RDHP})
$C_{RDHP} = [C_{RDHPI} + C_{RDHPP}]$ C_{RDHPI} = Development costs associated with hardware produced in-house C_{RDHPP} = Development costs associated with hardware procured from a vendor or subcontracted out
Some hardware components will be developed and produced in-house, others may be available off-the-shelf, and procured from a vendor.

Development Costs of Hardware Produced In-House (C_{RDHPI})
$C_{RDHPI} = \sum_{i=1}^{N} C_{RDHPI_i}$ C_{RDHPI_i} = Cost of specific activity i N = Number of activities
To produce hardware in-house, prototypes must be developed and tested to validate the design, and the various board level components must be constructed.

Development Costs of Hardware Procured from Vendor (C_{RDHPP})
$C_{RDHPP} = [C_{RDHPPE} + C_{RDHPPU}]$ C_{RDHPPE} = Evaluation of Hardware Procured from Vendor C_{RDHPPU} = Unit Costs of Hardware Procured from Vendor
For hardware which is procured, both categories of products and specific vendors must be evaluated, and the selected components must be purchased.

Evaluation of Hardware Procured from Vendor (C_{RDHPPE})
$C_{RDHPPE} = [C_{RDHPPEP} + C_{RDHPPEV}]$ $C_{RDHPPEP}$ = Product Evaluation $C_{RDHPPEV}$ = Vendor Evaluation
This includes the evaluation of various off-the-shelf products to determine which best suits the specifications. A list of desired features should be compiled initially, and those products which have a high percentage of those features should be considered. It also includes the evaluation of the available vendors, based on such factors as years in business, stability, warranties, customer recommendations, etc.

Unit Costs of Hardware Procured from Vendor (C_{RDHPPU})
$C_{RDHPPU} = \sum_{i=1}^{N} C_{RDHPPU_i}$ C_{RDHPPU_i} = Cost of specific component i N = Number of components
Each hardware component procured from a vendor has a vendor price associated with it.

Hardware Integration (C_{RDHI})
$C_{RDHI} = [C_{RDHII} + C_{RDHID}]$ C_{RDHII} = Costs associated with integration of hardware components C_{RDHID} = Documentation costs associated with integrated hardware
When all hardware components have been fabricated and/or procured, all components must be integrated into an overall hardware package and all documentation must be compiled and archived.

Integration of Components (C_{RDHII})
$C_{RDHII} = \sum_{i=1}^{N} C_{RDHII_i}$ C_{RDHII_i} = Cost of specific component i N = Number of components
All hardware components, whether fabricated in-house or procured, must be integrated into an overall hardware package.

Design Documentation of Hardware (C_{RDHID})
$C_{RDHID} = [C_{RDHIDI} + C_{RDHIDP} + C_{RDHIDL}]$ C_{RDHIDI} = Costs of compiling in-house documentation C_{RDHIDP} = Costs of compiling vendor documentation C_{RDHIDL} = Costs of establishing documentation library
Design documentation includes the costs of compiling in-house documentation and vendor documentation, as well as archiving all documentation.

Compilation of in-house documentation (C_{RDHIDI})
$C_{RDHIDI} = \sum_{i=1}^{N} C_{RDHIDI_i}$ C_{RDHIDI_i} = Cost of specific activity i N = Number of activities
In-house documentation includes such items as design drawings (assembly drawings, control drawings, logic diagrams, installation drawings, and schematics), material and parts lists (parts lists, material lists, provisioning lists, etc.), and analyses and reports (trade-off study reports supporting design decisions, reliability and maintainability analyses and predictions, human factors analyses, safety reports, logistic support analyses, configuration identification reports, installation and assembly procedures, etc.). This category also includes the preparation, printing, publication and distribution of all data/documentation associated with previous phases. This covers program plans; R & D reports; design data; test plans and reports; analyses; preliminary operational and maintenance procedures; and all effort related to a specific documentation requirement.

Compilation of vendor documentation (C_{RDHIDP})
$C_{RDHIDP} = \sum_{i=1}^{N} C_{RDHIDP_i}$ C_{RDHIDP_i} = Cost of specific activity i N = Number of activities
If multiple components are procured from vendors, the various pieces of documentation which accompanied each component must be gathered together into one document.

Establish Documentation Library (C_{RDHIDL})
$C_{RDHIDL} = \sum_{i=1}^{N} C_{RDHIDL_i}$ C_{RDHIDL_i} = Cost of specific activity i N = Number of activities
All documentation, whether associated with in-house or procured hardware, must be archived in a single location.

Hardware Test and Evaluation (C_{RDHT})
$C_{RDHT} = \sum_{i=1}^{N} C_{RDHT_i}$ C_{RDHT_i} = Cost of specific activity i N = Number of activities
This involves test planning, testing and evaluation, and resulting test data and reports dealing with the overall hardware package.

Software Costs (C_{RDS})
$C_{RDS} = [C_{RDSD} + C_{RDSP} + C_{RDSI} + C_{RDST}]$ C_{RDSD} = Software Design costs C_{RDSP} = Make/Buy Partition C_{RDSI} = Software Integration costs C_{RDST} = Software Test and Evaluation costs
Various functions in a microprocessor system may be implemented in software. Associated costs include software design costs, in-house and/or procured software, software integration costs, and test and evaluation costs.

Software Design (C_{RDSD})
$C_{RDSD} = \sum_{i=1}^{N} C_{RDSD_i}$ C_{RDSD_i} = Cost of specific activity i N = Number of activities
The software design phase involves development of the detailed design, design support, and design review. Detailed design includes the actual software design, which is the process of representing the functions of each software system in such a manner that they may be easily transformed into one or more programs. Design support includes the costs of tools such as compilers and debuggers, flowcharting software, programmers' toolkits, CASE tools, etc., as well as documentation specialists. Design reviews include formal reviews such as the static program verification, code inspection, and flight readiness review.

Software Make/Buy Partition (C_{RDSP})
$C_{RDSP} = [C_{RDSPI} + C_{RDSPP}]$ C_{RDSPI} = Development costs associated with software produced in-house C_{RDSPP} = Development costs associated with software procured from a vendor or subcontracted out
Some software components will be developed and produced in-house, others may be available off-the-shelf, and procured from a vendor.

Software Developed In-House (C_{RDSPI})
$C_{RDSPI} = \sum_{i=1}^{N} C_{RDSPI_i}$ C_{RDSPI_i} = Cost of specific activity i N = Number of activities
The costs associated with software developed in-house include the costs of software engineering support tools, software development tools, all costs associated with the actual implementation effort, costs of integrating subsystems, costs of test and evaluation of software components, software documentation costs, costs of verification and validation, debugging effort, and quality assurance activities.

Software Procured from Vendor (CRDSPP)
$C_{RDSPP} = [C_{RDSPPE} + C_{RDSPPV} + C_{RDSPPT}]$ C_{RDSPPE} = Evaluation C_{RDSPPV} = Vendor Fees C_{RDSPPT} = Training Costs
For software which is procured, both categories of products and specific vendors must be evaluated, software packages must be purchased, and personnel must familiarize themselves with the software packages.

Evaluation of Software Procured from Vendor (C_{RDSPPE})
$C_{RDSPPE} = [C_{RDSPPEP} + C_{RDSPPEV}]$ $C_{RDSPPEP}$ = Product Evaluation $C_{RDSPPEV}$ = Vendor Evaluation
This includes the evaluation of various off-the-shelf products to determine which best suits the specifications. A list of desired features should be compiled initially, and those products which have a high percentage of those features should be considered. It also includes the evaluation of the available vendors, based on such factors as years in business, stability, warranties, customer recommendations, etc.

Vendor Fees for Software Procured from Vendor (C_{RDSPPV})
$C_{RDSPPV} = [C_{RDSPPVP} + C_{RDSPPVS} + C_{RDSPPVN}]$ $C_{RDSPPVP}$ = Procurement Costs $C_{RDSPPVS}$ = Site License Agreement $C_{RDSPPVN}$ = Networking Capabilities
Vendor fees include such costs as the purchase price of the software package, a site license agreement to permit the company to make multiple copies of the software, and any additional fees involved in providing networking capabilities for the software if they are needed.

Training Costs for Software Procured from Vendor (C_{RDSPPT})
$C_{RDSPPT} = \sum_{i=1}^{N} C_{RDSPPT_i}$ C_{RDSPPT_i} = Cost of specific activity i N = Number of activities
The training costs associated with software procured from a vendor include seminars, training tapes, or self-instruction needed to familiarize the personnel with the capabilities of the software package.

Software Integration (C_{RDSI})
$C_{RDSI} = [C_{RDSII} + C_{RDSID}]$ C_{RDSII} = Costs associated with integration of software subsystems C_{RDSID} = Documentation costs associated with software
When all software has been implemented and/or procured, all subsystems must be integrated into an overall software package and all documentation must be compiled and archived.

Integration of Software Subsystems (C_{RDSII})
$C_{RDSII} = \sum_{i=1}^{N} C_{RDSII_i}$ C_{RDSII_i} = Cost of specific subsystem i N = Number of subsystems
All software subsystems, whether implemented in-house or procured, must be integrated into an overall software package.

Software Documentation (C_{RDSID})
$C_{RDSID} = [C_{RDSIDI} + C_{RDSIDP} + C_{RDSIDL}]$ C_{RDSIDI} = Costs of compiling in-house documentation C_{RDSIDP} = Costs of compiling vendor documentation C_{RDSIDL} = Costs of establishing documentation library
Software documentation includes the costs of compiling in-house documentation and vendor documentation, as well as archiving all documentation.

Compilation of in-house documentation (C_{RDSIDI})
$C_{RDSIDI} = \sum_{i=1}^{N} C_{RDSIDI_i}$ C_{RDSIDI_i} = Cost of specific activity i N = Number of activities
In-house documentation includes such items as user documentation and system documentation. User documentation should include such documents as a functional description which explains the system capabilities, an installation document which explains the installation procedure and configuration options, an introductory manual which explains how to get started using the system, a reference manual which describes all of the system facilities and how they can be used, and a system administrator's manual, which explains how to respond to various situations and how to perform various housekeeping chores.

Compilation of vendor documentation (C_{RDSIDP})
$C_{RDSIDP} = \sum_{i=1}^{N} C_{RDSIDP_i}$ C_{RDSIDP_i} = Cost of specific activity i N = Number of activities
If multiple software components are procured from vendors, the various pieces of documentation which accompanied each component must be gathered together into one document.

Establish Documentation Library (C_{RDSIDL})
$C_{RDSIDL} = \sum_{i=1}^{N} C_{RDSIDL_i}$ C_{RDSIDL_i} = Cost of specific activity i N = Number of activities
All documentation, whether associated with in-house or procured software, must be archived in a single location.

Software Test and Evaluation (C_{RDST})
$C_{RDST} = \sum_{i=1}^{N} C_{RDST_i}$ C_{RDST_i} = Cost of specific activity i N = Number of activities
This includes test planning, test and evaluation, system planning, verification and validation, data collection, and reports of software performance.

Integrated Testing (C_{RT})
$C_{RT} = \sum_{i=1}^{N} C_{RT_i}$ C_{RT_i} = Cost of specific activity i N = Number of activities
This step involves in-circuit emulation of the microprocessor system, in order to debug both the hardware and software prior to integration.

Integration and Testing (C_{RI})
$C_{RI} = [C_{RII} + C_{RID} + C_{RIT}]$ C_{RII} = Cost of integration of hardware and software C_{RID} = Cost of integrating and archiving all documentation C_{RIT} = Cost of System Test and Evaluation
This phase involves the integration of the hardware and software components, compilation of system documentation, and final testing of the integrated system. All hardware and software components are integrated into the final functional package. In addition, all documentation which pertains to the hardware and software components is merged into an overall set of documentation as well as being archived. Finally, the functional system is put through final tests which involve test planning, testing and evaluation, and reports of system performance.

Production (C_P)
$C_P = [C_{PM} + C_{PC} + C_{PP} + C_{PD} + C_{PL}]$ C_{PM} = Production/Construction management cost C_{PC} = Construction cost C_{PP} = System Production costs C_{PT} = System Documentation costs C_{PD} = System/Product Distribution costs C_{PL} = Cost of initial logistic support
Includes all costs associated with the acquisition and/or production of systems subsequent to the completion of the research and development phase. Specifically this covers management, construction, system realization, system distribution, and initial logistic support.

Production Management (C_{PM})
$C_{PM} = \sum_{i=1}^{N} C_{PM_i}$ C_{PM_i} = Cost of specific management activity i N = Number of activities
Cost of management oriented activity applicable to construction of facilities, system production, product distribution, and logistics management.

Construction cost (C_{PC})
$C_{PC} = [C_{PCP} + C_{PCT} + C_{PCM} + C_{PCI}]$ C_{PCP} = Production facilities cost C_{PCT} = Test facilities cost C_{PCM} = Maintenance facilities acquisition cost C_{PCI} = Inventory Warehouse acquisition cost For each item, one should consider the following. $C_{PCx} = [C_{PCxL} + C_{PCxM} + C_{PCxU} + C_{PCxE}]$ C_{PCxL} = Construction labor cost C_{PCxM} = Construction material cost C_{PCxU} = Cost of utility installation C_{PCxE} = Capital equipment cost $x = \{_{P,T,M,I}\}$
Includes all initial acquisition costs associated with production, test, maintenance, and/or warehousing facilities. Facilities constitute real property, plant, equipment, and installation of utilities (gas, electric power, water, telephone, heat, air conditioning, etc.). Facility costs cover the development of new building projects, the modification of existing facilities, and/or the occupancy of existing facilities without modification. Category costs include preliminary surveys; real estate; building construction; access roads; etc. Cost items include construction labor, construction material, capital equipment, and utility installation. (a) Production facilities support the operations described in C_{PP} and C_{PI}. Initial acquisition and sustaining maintenance costs are included. (b) Special test facilities cover any peculiar requirements (beyond that covered under existing categories such as engineering and manufacturing tests) for evaluation and test. Initial acquisition and sustaining maintenance costs are included. (c) Maintenance facilities are required to support the maintenance needs of the system throughout its life-cycle. Recurring sustaining costs are covered in C_{MTF}. (d) Inventory warehouses are required for the storage of completed systems which have not yet been distributed.

System Production (C_{PP})
$C_{PP} = [C_{PPH} + C_{PPS} + C_{PPI}]$ C_{PPH} = Cost of Hardware Components C_{PPS} = Cost of Software Components C_{PPI} = Cost of System Integration
This category involves the fabrication of hardware components, incorporation of software components, and integration of all components into a functional unit.

Hardware Components (C_{PPH})
C_{PPH} = [C_{PPHP} + C_{PPHI}] C_{PPHP} = Make/Buy Partition C_{PPHI} = Cost of Hardware Integration
This category includes the cost of hardware which is built in-house, the cost of hardware procured from a vendor or subcontracted out, and the cost of integrating the two into the overall hardware portion of the system.

Make/Buy Partition (C_{PPHP})
C_{PPHP} = [C_{PPHPI} + C_{PPHPP}] C_{PPHPI} = Development costs associated with hardware produced in-house C_{PPHPP} = Development costs associated with hardware procured from a vendor or subcontracted out
Some hardware components will be produced in-house, others may be procured from a vendor.

Hardware Produced In-House (C_{PPHPI})
C_{PPHPI} = [C_{PPHPII} + C_{PPHPIP} + C_{PPHPIQ}] C_{PPHPII} = Industrial Engineering costs C_{PPHPIP} = Production costs C_{PPHPIQ} = Quality Control costs
This category includes those costs involved in the production of those hardware components which are built in-house. This includes the costs associated with industrial engineering, production, and quality control.

Industrial Engineering costs of Hardware Built In-House (C_{PPHPII})
C_{PPHPII} = [$C_{PPHPIIM}$ + $C_{PPHPIIE}$ + $C_{PPHPIIC}$] $C_{PPHPIIM}$ = Cost of manufacturing engineering $C_{PPHPIIE}$ = Cost of methods engineering $C_{PPHPIIC}$ = Cost of production control
Includes all recurring and nonrecurring costs associated with the initial and sustaining engineering functions of construction and production. This constitutes (1) manufacturing engineering (e.g., process design, design of special tools/fixtures/test equipment, man-machine functions, etc.); (2) methods engineering (e.g., work methods, job skill requirements, standards, design of subassembly and assembly operations, etc.); and (3) production control operations (e.g., production lot quantities, economic order quantities, inventory levels, the evaluation of production operations to insure that product quality, performance, reliability, maintainability, safety, and other features are maintained throughout the production process, etc.).

Production/Manufacturing Costs (C_{PPHPIP})
$C_{PPHPIP} = [C_{PPHPIPR} + C_{PPHPIPN}]$ $C_{PPHPIPR}$ = Recurring manufacturing costs $C_{PPHPIPN}$ = Nonrecurring manufacturing costs
This covers all recurring and nonrecurring costs associated with the production and test of multiple quantities of prime systems. Facility construction, capital equipment, and facility maintenance are covered under C_{PC}. (1) Recurring manufacturing costs--fabrication and assembly labor cost, material and inventory cost, inspection and test cost, and product rework as required. Sustaining engineering support required on a recurring basis is also included. Costs are associated with the production of prime equipment. Operational test and support equipment, training equipment, and spare/repair parts material costs are included in C_{PL}. Manufacturing management cost is included in C_{PM}. (2) Nonrecurring manufacturing costs--labor and material costs associated with the installation and support of factory tools, fixtures, and test equipment. Design costs are included in $C_{PPHPIIM}$.

Quality Control (C_{PPHPIQ})
$C_{PPHPIQ} = [C_{PPHPIQA} + \sum_{i=1}^{N} C_{PPHPIQC_i} + \sum_{i=1}^{N} C_{PPHPIQS_i}]$ $C_{PPHPIQA}$ = Quality assurance cost $C_{PPHPIQC_i}$ = Cost of qualification test i $C_{PPHPIQS_i}$ = Cost of production sampling test i N = Number of activities
This category covers the recurring cost of maintaining an on-going quality assurance or quality control capability throughout production and construction, and directly supports activities in C_{PC}, C_{PPHPII}, C_{PPHPIP}, and C_{PL}. In addition, the specific nonrecurring costs associated with the initial product qualification testing and periodic sampling tests conducted throughout production are included. The inspection and acceptance testing for individual items in production is covered in C_{PPHPIP}.

Hardware Procured from Vendor (C_{PPHPP})
$C_{PPHPP} = \sum_{i=1}^{N} C_{PPHPPP_i}$ C_{PPHPPP_i} = Cost of procurement expense i N = Number of expenses
Each hardware component procured from a vendor has a procurement cost associated with it. This procurement cost may include the cost per unit, as well as a maintenance contract if provided by the vendor.

Hardware Integration (C_{PPHI})
C_{PPHI} = [C_{PPHII} + C_{PPHIT} + C_{PPHIQ}] C_{PPHII} = Integration of components C_{PPHIT} = Inspection and test of integrated unit C_{PPHIQ} = Quality control of integration process
All hardware components, fabricated in-house and procured, must be integrated into an overall hardware package. The integrated unit must then be tested, with quality assurance monitoring the integration and testing processes.

Software Components (C_{PPS})
C_{PPS} = [C_{PPSP} + C_{PPSI}] C_{PPSP} = Make/Buy Partition C_{PPSI} = Cost of Software Integration
This category includes the cost of all software components implemented in-house or procured from a vendor (or subcontracted out), and the cost of integrating the two into the overall software portion of the system.

Make/Buy Partition (C_{PPSP})
C_{PPSP} = [C_{PPSPI} + C_{PPSPP}] C_{PPSPI} = Costs associated with software implemented in-house C_{PPSPP} = Acquisition costs associated with software procured from a vendor or subcontracted out
Those software components produced in-house must be duplicated for use in the overall system, and those which will be procured have associated costs.

Software Produced In-House (C_{PPSPI})
C_{PPSPI} = Costs associated with software duplication
That software written in-house must be duplicated for use in the overall system.

Software Procured from Vendor (C_{PPSPP})
C_{PPSPP} = [C_{PPSPPP} + C_{PPSPPS} + C_{PPSPPN}] C_{PPSPPP} = Procurement Costs C_{PPSPPS} = Site License Agreement C_{PPSPPN} = Networking Capabilities
Procurement costs include such costs as the purchase price of the software package, a site license agreement to permit the company to make multiple copies of the software, and any additional fees involved in providing networking capabilities for the software if they are needed.

Software Integration (C_{PPSI})
C_{PPSI} = [C_{PPSII} + C_{PPSIT}] C_{PPSII} = Integration of components C_{PPSIT} = Inspection and test of integrated unit
All software components, implemented in-house and procured, must be integrated into an overall software system. The final software package must then be tested.

System Integration (C_{PPI})
C_{PPI} = [C_{PPII} + C_{PPIT}] C_{PPII} = Integration of components C_{PPIT} = Inspection and test of integrated unit
All hardware and software components must be integrated into an overall functional unit. The final unit must then be tested.

System Documentation (C_{PT})
C_{PT} = Printing of system documentation
All documentation pertaining to the functional unit must be printed and bound for distribution.

System/Product Distribution (C_{PD})
C_{PD} = [C_{PDM} + C_{PDP} + C_{PDT} + C_{PDI}] C_{PDM} = Cost of marketing and sales C_{PDP} = Cost of packaging C_{PDT} = Cost of transportation and traffic management C_{PDI} = Cost of inventory in warehouses
This category includes: (1) The cost of product marketing and sales--advertising, exhibits, personnel costs associated with marketing and distribution, etc. (2) The cost of packaging the product for safe shipping. (3) The cost of transportation and traffic management--initial destination transportation from the production site to warehouses, and subsequent transportation from warehouses to the consumer. Traffic management and control functions are also included. (4) The cost of inventory in various distribution warehouses.

Initial logistic support cost (C_{PL})
$C_{PL} = [C_{PLM} + C_{PLP} + C_{PLS} + C_{PLI} + C_{PLD} + C_{PLT} + C_{PLX} + C_{PLY}]$ C_{PLM} = Logistic program management cost C_{PLP} = Cost of provisioning C_{PLS} = Initial spare/repair part material cost C_{PLI} = Initial inventory management cost C_{PLD} = Cost of technical data preparation C_{PLT} = Cost of initial training and training equipment C_{PLX} = Acquisition cost of operational test and support equipment C_{PLY} = Initial transportation and handling cost
Includes all integrated logistic support planning and control functions associated with the development of system support requirements, and the transition of such requirements from supplier(s) to the applicable operational site. Elements cover (a) Logistic program management cost--management, control, reporting, corrective action system, budgeting, planning, etc. (b) Provisioning cost--preparation of data which is needed for the procurement of spare/repair parts and test and support equipment. (c) Initial spare/repair part material cost--spares material stocked at the various inventory points to support the maintenance needs of prime equipment, test and support equipment and training equipment. Replenishment spares are covered in C_{MSHI}. (d) Initial inventory management cost--cataloging, listing, coding, etc., of spares entering the inventory. (e) Technical data preparation cost--development of operating and maintenance instructions, test procedures, maintenance cards, tapes, etc. Also includes reliability and maintainability data, test data, etc., covering production and test operations. (f) Initial training and training equipment cost--design and development of training equipment, training aids/data, and the training of personnel initially assigned to operate and maintain the prime equipment, test and support equipment, and training equipment. Personnel training costs include instructor time; supervision; student pay and allowances; training facilities; and student transportation. (g) Test and support equipment acquisition cost--design, development, and acquisition of test and support equipment plus handling equipment needed to operate and maintain prime equipment in the field. The maintenance of test and support equipment throughout the system life-cycle is covered in C_{MSHHT}. (h) Initial transportation and handling cost (first destination transportation of logistic support elements from the supplier to the applicable operational site). Initial facility costs are identified in C_{PC}.

Maintenance Support Cost (C_M)
$C_M = [C_{ML} + C_{MT} + C_{MS}]$ C_{ML} = Cost of system/equipment life-cycle management C_{MT} = Cost of maintenance training and facilities C_{MP} = Cost of system maintenance
Includes all costs associated with maintenance support of the system throughout its life-cycle subsequent to equipment delivery in the field. Specific categories cover the cost of life cycle management, maintenance training facilities, and system maintenance. Costs are generally determined for each year throughout life-cycle.

System/Product Life Cycle Management (C_{ML})
$C_{ML} = \sum_{i=1}^{N} C_{ML_i}$ C_{ML_i} = Cost of specific management activity i N = Number of activities
Cost of management oriented activity applicable to system maintenance.

Maintenance Training and Facilities cost (C_{MT})
$C_{MT} = [C_{MTT} + C_{MTF}]$ C_{MTT} = Cost of maintenance training C_{MTF} = Cost of maintenance facilities
This category includes the costs associated with training maintenance personnel and the upkeep of maintenance facilities.

Maintenance Training cost (C_{MTT})
$C_{MTT} = [((Q_{MS})(T_T)(C_{MT})) + C_{MTTF} + C_{MTTD}]$ Q_{MS} = Quantity of maintenance students C_{MT} = Cost of maintenance training ($/student-week) T_T = Duration of training program (weeks) C_{MTTF} = Cost of upkeep of training facilities C_{MTTD} = Cost of training data
Initial maintenance training cost is included in CPLT. This category covers the formal training of personnel assigned to maintain the prime equipment, test and support equipment, and training equipment. Such training is accomplished on a periodic basis throughout the system life-cycle to cover personnel replacement due to attrition. Total costs include instructor time; supervision; student pay and allowances while in school; training facilities (allocation of portion of facility required specifically for formal training); training aids and data; and student transportation as applicable.

Maintenance facility costs (C_{MTF})
C_{MTF} = [(C_{MFS} + C_U) x (% allocation)] C_{MFS} = Cost of maintenance facility support ($/site) C_U = Cost of utilities ($/site)
Initial acquisition (construction) cost for maintenance facility is included in CPCM. This category covers the annual recurring costs associated with the occupancy and upkeep of maintenance facilities. If a maintenance shop supports more than one system, associated costs are allocated proportionately to each system concerned.

System Maintenance cost (C_{MS})
C_{MS} = [C_{MSH} + C_{MSS}] C_{MSH} = Hardware maintenance cost C_{MSS} = Software maintenance cost
System maintenance involves maintenance activities dealing with both the hardware and software components.

Hardware Maintenance (C_{MSH})
C_{MSH} = [C_{MSHH} + C_{MSHI} + C_{MSHT} + C_{MSHM}] C_{MSHH} = Hardware maintenance costs C_{MSHI} = Inventory--spares and material support C_{MSHT} = Technical data costs C_{MSHM} = System modifications
Includes all hardware maintenance costs including field and factory maintenance costs, maintenance personnel, spares and material support, technical data, and the costs of system modifications.

Hardware Maintenance Activities (C_{MSHH})
C_{MSHH} = [C_{MSHHF} + C_{MSHHM} + C_{MSHHT}] C_{MSHHF} = Field maintenance costs C_{MSHHM} = Factory maintenance costs C_{MSHHT} = Test and support equipment maintenance
Includes all maintenance activities performed on-site as well as those requiring special facilities at the maintenance facilities. When a system malfunctions or when a scheduled maintenance action is performed, personnel manhours are expended, spare parts and related materials are utilized, and reports are completed. Field maintenance also involves personnel travel, while factory maintenance involves transportation and handling of the system. This category also includes the annual recurring maintenance costs for the test and support equipment itself, and support equipment operation costs.

Inventory--Spare/repair parts cost (C_{MSHI})
$C_{MSHI} = [C_{IO} + C_{II} + C_{IS} + C_{IC}]$ C_{IO} = Cost of organizational spare/repair parts C_{II} = Cost of intermediate spare/repair parts C_{IS} = Cost of supplier spare/repair parts C_{IC} = Cost of consumables $C_{IO} = [(C_A)(Q_A) + (C_{I_i})(Q_{I_i}) + (C_{H_i})(Q_{H_i})]$ C_A = Average cost of material purchase order ($/order) Q_A = Quantity of purchase orders C_I = Cost of spare item i Q_I = Quantity of i items required or demand C_H = Cost of maintaining spare item i in the inventory ($/$ value of the inventory) Q_H = Quantity of i items in the inventory C_{II} and C_{IS} are determined in a similar manner.
Initial spare/repair part costs are covered in C_{PLS}. This category includes all replenishment spare/repair parts and consumable materials (e.g., oil, lubricants, fuel, etc.) that are required to support maintenance activities associated with prime equipment, transportation and handling equipment (C_{PDT}), test and support equipment (C_{MSHHT}), and training equipment. This category covers the cost of purchasing items; the actual cost of the material itself; and the cost of holding or maintaining items in the inventory. Supply support costs are assigned to the applicable level of maintenance.

Technical data cost (C_{MSHT})
$C_{MSHT} = \sum_{i=1}^{N} C_{MSHT_i}$ C_{MSHT_i} = Cost of specific data item i N = Number of data items
Initial technical data preparation costs are covered in C_{PLD}. Individual data reports covering specific maintenance actions are included in C_{MSHH}, C_{MSSS}, and C_{MSHHT}. This category includes any other data (developed on a sustaining basis) necessary to support the operation and maintenance of the system throughout its life-cycle.

System/equipment modification cost (C_{MSHM})
$C_{MSHM} = \sum_{i=1}^{N} C_{MSHM_i}$ C_{MSHM_i} = Cost of specific modification i N = Number of modifications
Throughout the system life-cycle after equipment has been delivered in the field, modifications are often proposed and initiated to improve system performance, effectiveness, or a combination of both. This category includes modification kit design (R & D); material; installation and test instructions; personnel and supporting resources for incorporating the modification kit; technical data change documentation; formal training (as required) to cover the new configuration; spares; etc. This modification may affect all elements of logistics.

Software Maintenance (C_{MSS})
$C_{MSS} = [C_{MSSC} + C_{MSSS}]$ C_{MSSC} = Configuration management costs C_{MSSS} = Software maintenance costs
Includes all software maintenance costs including configuration management, perfective, adaptive, and corrective maintenance, and software maintenance equipment.

Configuration Management (C_{MSSC})
$C_{MSSC} = \sum_{i=1}^{N} C_{MSSC_i}$ C_{MSSC_i} = Cost of specific modification i N = Number of modifications
Configuration management activities involve reviewing and processing change requests and discrepancy reports and verifying that approved modifications were made and documented. This category also involves activities associated with maintaining files of current and previous releases of software items.

Software Maintenance Activities (C_{MSSS})
$C_{MSSS} = [C_{MSSSP} + C_{MSSSC} + C_{MSSSA} + C_{MSSSD}]$ C_{MSSSP} = Perfective maintenance C_{MSSSC} = Corrective maintenance C_{MSSSA} = Adaptive maintenance C_{MSSSD} = Debugging and diagnostic equipment
Software maintenance activities take many forms. Perfective maintenance involves enhancements in the software in response to changes in the environment, as well as modifying code to improve the performance of the system. Corrective maintenance involves debugging software in response to errors which occur in the system. Adaptive maintenance is necessary when a change in one part of the system requires changes to other parts of the system. The implementation of the secondary changes is adaptive maintenance. Debugging and diagnostic equipment includes such items as test data generators, execution flow summarizers, file comparators, simulators, symbolic dump programs, trace packages, and interactive debugging environments.

System retirement and disposal cost (C_D)
$C_D = [(F_C)(C_{DIS} - C_{REC})] + C_{DR}$ F_C = Condemnation factor C_{DIS} = Cost of system/equipment disposal C_{REC} = Reclamation value C_{DR} = Cost of system/equipment ultimate retirement
As the system evolves through its life cycle, there are non-repairable items which fail and must be discarded. In addition, there are items which are beyond economic repair and are also discarded. Eventually the system will be retired due to obsolescence or wearout. The process of phase-out and disposal may involve disassembly, decomposition, reforming, reprocessing, etc. This in turn involves personnel, support equipment, and transportation and handling, and proper documentation. Software will be archived as newer releases become available.

Summary of terms

C	Total system cost
C_A	Average cost of material purchase order ($/order)
C_D	Retirement and Disposal cost
C_{DIS}	Cost of system/equipment disposal
C_{DR}	Cost of system/equipment ultimate retirement
C_H	Cost of maintaining spare item i in the inventory ($/$ value of the inventory)
C_I	Cost of spare item i
C_{IC}	Cost of consumables

C_{II}	Cost of intermediate spare/repair parts
C_{IO}	Cost of organizational spare/repair parts
C_{IS}	Cost of supplier spare/repair parts
C_{M}	Maintenance cost
C_{MFS}	Cost of maintenance facility support ($/site)
C_{ML}	Cost of system/equipment life-cycle management
C_{MS}	Cost of system maintenance
C_{MSH}	Hardware maintenance cost
C_{MSHH}	Hardware maintenance costs
C_{MSHHF}	Field maintenance costs
C_{MSHHM}	Factory maintenance costs
C_{MSHHT}	Test and support equipment
C_{MSHI}	Inventory--spares and material support
C_{MSHM}	System modifications
C_{MSHT}	Technical data costs
C_{MSS}	Software maintenance cost
C_{MSSC}	Configuration management costs
C_{MSSS}	Software maintenance costs
C_{MSSSA}	Adaptive maintenance
C_{MSSSC}	Corrective maintenance
C_{MSSSD}	Debugging and diagnostic equipment
C_{MSSSP}	Perfective maintenance
C_{MT}	Cost of maintenance training ($/student-week)
C_{MTF}	Cost of maintenance facilities
C_{MTT}	Cost of maintenance training
C_{MTTD}	Cost of maintenance training data
C_{MTTF}	Cost of maintenance training facilities
C_{P}	Production cost
C_{PC}	Construction cost
C_{PCI}	Inventory Warehouse acquisition cost
C_{PCM}	Maintenance facilities acquisition cost
C_{PCP}	Production facilities cost
C_{PCT}	Test facilities cost
C_{PD}	System/Product Distribution costs

C_{PDI}	Cost of inventory in warehouses
C_{PDM}	Cost of marketing and sales
C_{PDP}	Cost of packaging
C_{PDT}	Cost of transportation and traffic management
C_{PL}	Cost of initial logistic support
C_{PLD}	Cost of technical data preparation
C_{PLI}	Initial inventory management cost
C_{PLM}	Logistic program management cost
C_{PLP}	Cost of provisioning
C_{PLS}	Initial spare/repair part material cost
C_{PLT}	Cost of initial training and training equipment
C_{PLX}	Acquisition cost of operational test and support equipment
C_{PLY}	Initial transportation and handling cost
C_{PM}	Production/Construction management cost
C_{PP}	System Production costs
C_{PPH}	Cost of Hardware Components
C_{PPHI}	Cost of Hardware Integration
C_{PPHII}	Integration of components
C_{PPHIQ}	Quality control of integration process
C_{PPHIT}	Inspection and test of integrated unit
C_{PPHP}	Make/Buy Partition
C_{PPHPI}	Development costs associated with hardware produced in-house
C_{PPHPII}	Industrial Engineering costs
$C_{PPHPIIC}$	Cost of production control
$C_{PPHPIIE}$	Cost of methods engineering
$C_{PPHPIIM}$	Cost of manufacturing engineering
$C_{PPHPIIP}$	Cost of plant engineering
$C_{PPHPIIS}$	Cost of sustaining engineering
C_{PPHPIP}	Production costs
$C_{PPHPIPN}$	Nonrecurring manufacturing costs
$C_{PPHPIPR}$	Recurring manufacturing costs
C_{PPHPIQ}	Quality Control costs
$C_{PPHPIQA}$	Quality assurance cost
$C_{PPHPIQC_i}$	Cost of qualification test i

$C_{PPHPIQS_i}$	Cost of production sampling test i
C_{PPHPP}	Development costs associated with hardware procured from a vendor or subcontracted out
C_{PPI}	Cost of System Integration
C_{PPII}	Integration of components
C_{PPIT}	Inspection and test of integrated unit
C_{PPS}	Cost of Software Components
C_{PPSI}	Cost of Software Integration
C_{PPSII}	Integration of components
C_{PPSIT}	Inspection and test of integrated unit
C_{PPSP}	Make/Buy Partition
C_{PPSPI}	Costs associated with the duplication of software implemented in-house
C_{PPSPP}	Acquisition costs associated with software procured from a vendor or subcontracted out
C_{PPSPPN}	Networking Capabilities
C_{PPSPPP}	Procurement Costs
C_{PPSPPS}	Site License Agreement
C_{PT}	Printing of system documentation
C_R	Research and Development cost
C_{RD}	Hardware/Software development partition
C_{RDH}	Development costs associated with the hardware portion of the project
C_{RDHC}	Design Completion
C_{RDHI}	Hardware Integration
C_{RDHID}	Documentation costs associated with integrated hardware
C_{RDHIDI}	Costs of compiling in-house documentation
C_{RDHIDL}	Costs of establishing documentation library
C_{RDHIDP}	Costs of compiling vendor documentation
C_{RDHII}	Costs associated with integration of hardware components
C_{RDHP}	Hardware make/buy partition
C_{RDHPI}	Development costs associated with hardware produced in-house
C_{RDHPP}	Development costs associated with hardware procured from a vendor or subcontracted out
C_{RDHPPE}	Evaluation of hardware procured from vendor
$C_{RDHPPEP}$	Product Evaluation of hardware procured from vendor

$C_{RDHPPEV}$	Vendor Evaluation of hardware procured from vendor
C_{RDHPPU}	Unit Costs of hardware procured from vendor
C_{RDHT}	Hardware Test and Evaluation
C_{RDS}	Development costs associated with the software portion of the project
C_{RDSD}	Software Design costs
C_{RDSI}	Software Integration costs
C_{RDSID}	Documentation costs associated with software
C_{RDSIDI}	Costs of compiling in-house documentation
C_{RDSIDL}	Costs of establishing documentation library
C_{RDSIDP}	Costs of compiling vendor documentation
C_{RDSII}	Costs associated with integration of software subsystems
C_{RDSP}	Software make/buy partition
C_{RDSPI}	Development costs associated with software produced in-house
C_{RDSPP}	Development costs associated with software procured from a vendor or subcontracted out
C_{RDSPPE}	Evaluation of software procured from vendor
$C_{RDSPPEP}$	Product Evaluation of software procured from vendor
$C_{RDSPPEV}$	Vendor Evaluation of software procured from vendor
C_{RDSPPT}	Training Costs for software procured from vendor
C_{RDSPPV}	Vendor Fees for software procured from vendor
$C_{RDSPPVN}$	Networking Capabilities for software procured from vendor
$C_{RDSPPVP}$	Procurement Costs for software procured from vendor
$C_{RDSPPVS}$	Site License Agreement for software procured from vendor
C_{RDST}	Software Test and Evaluation costs
C_{REC}	Reclamation value
C_{RF}	Functional Specification of microprocessor system
C_{RI}	Integration and Testing
C_{RID}	Cost of integrating and archiving all documentation
C_{RII}	Cost of Integration of hardware and software
C_{RIT}	Cost of System Test and Evaluation
C_{RM}	System/Product management cost
C_{RP}	Product Planning cost
C_{RT}	Integrated Testing
C_{U}	Cost of utilities ($/site)

F_C	Condemnation factor
Q_A	Quantity of purchase orders
Q_H	Quantity of i items in the inventory
Q_M	Quantity of i items required or demand
Q_{MS}	Quantity of maintenance students
T_T	Duration of training program (weeks)