SOFTWARE LIFE-CYCLE COST

DR. GERRY C. WHITE
Naval Training Equipment Center

ABSTRACT

Life-cycle cost estimation can be a means to avoid mistakes in system design that would result in large costs. Successful life-cycle forecasting, however, requires the ability to predict, with reasonable confidence, the total cost (life-cycle cost) associated with the development, acquisition, and ownership of a system. Unfortunately, life-cycle cost analysis has not been satisfactorily applied to computer software systems. Unlike hardware, logistic parameters for software are difficult to predict and measure. Thus, software costs have been difficult to predict. Life-cycle costing of hardware systems has evolved into a systematic approach involving concept formulation, contract considerations, development/production and operations/disposal. Such an approach has been useful to define areas of high-support costs, evaluate alternative support policies, determine impact of operational requirements on support alternatives, and to provide for long-range cost prediction. This paper examines the cost of real-time simulation computers with emphasis on computer software life-cycle costs.

INTRODUCTION

Computers are no longer the mystic entity they once were, but they do present problems to any management which does not understand the complexity of supporting computers that are an integral part of a complex training system. Cost overruns, schedule slippages, and inadequate performance have been commonly encountered in the procurement of such systems. Even after acceptance of these systems, modifications are commonly required to correct faults in the systems.

An improved approach is necessary. It is not enough that hardware and software be developed as a total system, with the usual consideration given to hardware/software tradeoffs. Even an enlightened user with a total-system approach to design and acquisition is likely to experience problems in the support of software for computer-controlled systems, unless software is given special consideration. Future software modifications must be anticipated and planning provided for them.

Software is, of necessity, an item requiring maintenance and must be modified to correct errors, to improve system response, or to reflect changes in operational equipment. There appears to be a shortage of personnel who are capable of making sound decisions regarding the planning for software support of this type.

The development of a maintenance concept is considered one of the most important steps in planning for system support during its life cycle. Unfortunately, maintenance and enhancement of software is viewed as being of lesser importance than the design and development of software, and planning for software support is often inadequate.

Activities which keep systems that meet user needs (maintenance and enhancement) are a necessary expense. It was determined by Swanson (12) that only seventeen percent of such activity was for corrective maintenance. That is, it was necessary in response to the assessment of failures. Adaptive maintenance (performed in anticipation of changes within the data processing environment) or perfective maintenance (elimination of inefficiencies, performance enhancement, and improvement of maintainability) was required in the remaining eighty-three percent of the cases, however.

When one considers that forty to seventy-five percent of total systems engineering and programming resources (2,10) are involved in software maintenance and that this cost is often obscured in other operating expenses (7), the need for effective planning for software maintenance becomes financially clear. Although total expected cost was being used as early as 1968 (3) to evaluate proposals for purchase of computer software systems, software maintenance was not a consideration.

Adequate planning must include all factors that will influence the cost over the full-system life cycle. Anticipated software maintenance is an important part of the life-cycle cost, yet there is a dearth of historical data. Predicting the maintenance costs of computer software is a difficult task, not unlike estimating software development costs.

Support problems limit the operational readiness and availability of any system. Typical is maintenance down time. Much effort has been expended to increase the mean time between failures and to decrease repair time of hardware. Personnel are trained to reduce the adverse effect of failure and malfunction. Logistic support may be used as a constraint in hardware design. Alternative designs are considered if existing designs create support problems. Software has not received this

attention.

Software maintenance costs may possibly be reduced by extensive effort during the development phase. Modifications are relatively cheap during the early software development while those required after the system becomes operational may be twenty times as expensive. The nature of training systems is such that software modifications must be introduced as operational equipment is modified, or even to simulate other similar aircraft. It is possible that in the early planning stages, one could consider the need for future modifications and plan means to easily incorporate these at a later date.

In 1977, Department of Defense (DOD) software costs exceeded $3 billion. Sixty-eight percent of this was for development; only thirty-two percent was allocated to operation and maintenance (1). There has been little planning for software maintenance. Historically, software has included a large number of errors. Although errors are introduced in the system design phases, corrections must often be instituted at the most critical points (during test, evaluation, and acceptance). Consideration must be given to software development using the same concepts and approaches that have successfully controlled hardware costs.

The cost of software is skyrocketing, and results still fail to meet expectations. Applications of engineering techniques could improve this area. Computer technology has reached an era where hardware development costs are declining per unit of capability. This trend is likely to continue. Software development costs (as measured in lines of code per manhour) are rising. The difference is that hardware design is based upon engineering principles and manufactured in highly automated processes. These principles do not exist for software. In addition, managers do not understand software and are unable to generate adequate specifications for its performance, design, or development (9).

Life-cycle costing has required the ability to forecast the amount of the cost with reasonable confidence. This concept must be expanded to include all those factors, including software, which significantly influence the cost of support over the life cycle of the system.

SOFTWARE

Software became an item of significant cost in training devices with the increased use of digital computers for the simulation of weapon systems. In the training environment, extremely complex algorithms must be programmed to simulate the weapon and its environmental characteristics. Such software is costly, and neglect of this software in planning a training device can result in unsatisfactory operation and increased life-cycle cost. With the introduction of new hardware and shrinking budgets, control of software cost is of increasing concern.

Although little data are available on the cost of dedicated computer systems, an examination of automatic data processing (ADP) costs reveals much about software costs in general. It was found (5,9) that forty to forty-five percent of the $6.2 to $8.3 billion cost of 3,460 DOD computer systems in 1973 went for software. The cost of software required in the Naval Training Equipment Center's training devices is sixty-five to seventy-five percent of the total systems cost (6).

SOFTWARE PRODUCTION

Software for military computer systems can cost several million dollars and require several years to develop; yet may result in ineffective hardware incapable of meeting program milestones. The production of this software (systems analysis, design, and programming) consumes approximately twenty-three percent of the ADP costs of DOD computer systems (9).

One of the important requirements for management planning is an accurate estimate of the resources required to complete a project. Estimating the cost of software production is difficult. Considerable design work, good software specifications, and intensive project planning are required for realistic estimates. One common problem has been the poorly estimated cost of computer program development.

Estimates of program development costs are usually based on the number of lines of code to be written. Yet one of the most difficult questions to answer is "how long will it take to program this application?" Programmer productivity varies from 1,000 to 4,000 program statements per year. The average is 2,500 including time to block diagram, code, and test (11). Although there are many significant variables in predicting programming effort, the one most commonly used is delivered lines of code.

Basically, three factors (4) affect the cost of computing: (1) The job to be done (the number of program instructions). This has usually been poorly estimated. Safety factors commonly range from twenty to four hundred percent. (2) The resources with which to do the job; and (3) The environment in which the job is done. Although accurate estimation of computer programming costs is an important prerequisite for effective

programming management, such estimates have been historically unreliable.

SOFTWARE MAINTENANCE

The cost of software maintenance is staggering! Seventy percent of the overall cost of software in the Air Force goes into software maintenance (5). Many ADP installations apply seventy percent of the time of systems analysts and programmers to software maintenance functions (8). Currently, forty percent of overall hardware/software effort is going into software maintenance and this is expected to grow to sixty percent by 1985 (2). It is maintenance of software that consumes a major part of system life-cycle cost. This cost increases as the life cycle is extended.

One item of importance, especially in long-life training system computers, is software maintenance. It has been estimated that program development costs seventy-five dollars per instruction, while maintenance could cost as much as $4,000 per instruction (11). The higher cost has been attributed to poorly designed, poorly structured, and poorly documented older software.

Future software maintenance requirements are difficult to estimate. Extensive data on predicted reliability and maintainability has been required for life-cycle cost analysis of hardware. Little historical data of this type is available for software.

Even in highly reliable systems, malfunctions can be expected and provisions for their correction must be made. Planning for long life-cycle support requires provisions for personnel, money, and other support factors. Some reliable estimate must be made of the effort required and software mainenance must be included.

Kirby (6) found that during the average useful life of eleven years for a major training device, seventy percent of the software costs would be spent on maintenance. In addition, he found that fifty percent of all modifications in Chief Naval Education and Training Support's field organizations were for software.

An outstanding characteristic of every complex military system is long life. Software maintenance for these systems is costly, but failure to plan for this item can result in inflexible hardware, incapable of being modified to fill new system requirements, thus shortening useful life.

HARDWARE DEVELOPMENTS AND SOFTWARE IMPLICATIONS

The DOD has supported the increasing use of large-scale microcircuitry to improve system reliability, reduce life-cycle costs, and to achieve systems with expanded capability. Only recently, however, have large-scale integrated (LSI) microcircuits influenced the design of training devices.

Microcomputers have generally decreased system development time (over hardwired systems) and increased system reliability (due to fewer parts and fewer interconnections and high reliability of the microprocessor itself). Also, these LSI chips are cheaper than minicomputers, as well as being smaller and more flexible.

One disadvantage (or advantage) of the use of microprocessors is that the designer must understand the relation between hardware and software. In addition, modifications to software will require special skills. This is somewhat similar to the early days of computer design when the programmer had a part in building the machine.

It has been predicted (13) that the cost of computer hardware for training devices would decrease as microprocessors assume more functions. Historically, each new generation of computers has had capabilities that far surpassed the previous generation. Microprocessors are continuing this trend, and even direct compilation of code may soon be common.

Such advances will undoubtedly reduce the cost of software, but software costs and especially software maintenance will likely continue to be a major part of systems life-cycle costs. Thus, control of software life-cycle costs continues to be an item of major concern.

CONTROL OF SOFTWARE COSTS

Within the training device community, there has been no "standard" system. Both hardware and software were designed for a specific application. Severe timing constraints and the requirement for real-time interaction with the external environment originally necessitated the use of assembly language in real-time simulation. Work toward standardization has been initiated.

Recently, NAVTRAEQUIPCEN adopted a standard high-level language (real-time FORTRAN) for use in training devices. The use of higher level source language will mean more reliable programs, ability to write structured programs and fewer opportunities for errors. In addition, program maintenance should be simplified and correcting errors and adding enhancements can be better documented and easier to debug.

In order to reduce the time required for software maintenance, software engineers should be provided facilities to simplify testing and

modification of operational programs. This might take the form of immediately accessible time-sharing terminals, capable of accessing programs stored on disc in a large computer institution. This would speed up the process of debugging any errors that might occur, and also permit the training to be increased to the highest fidelity in a reasonable amount of time.

Vendor supplied software is an item NAVTRAEQUIPCEN has no control over. It is expensive and time consuming when operating systems and compilers, and the utility programs do not work as expected. When contractors modify operating systems for their particular application, NAVTRAEQUIPCEN's extensive use of cross assemblers is temporarily rendered ineffective. The acceptance of these systems must be based on more intensive and effective test procedures.

Control over software is essential. Frequent modifications to meet new operational requirements require good documentation. Accurate complete documentation is vital for software configuration control and maintenance throughout the system's life. Modifications without this documentation are expensive, if not impossible.

Training of software personnel is a relatively inexpensive area with good returns. As microprocessors become more widely used, consideration must be given to preparing the software staff for modifications and maintenance of microprocessor software. Personnel costs are high but highly trained software personnel can be an effective cost-control factor.

CONCLUSIONS

Design concepts such as system redundancy, in addition to testing and conditioning (burn-in) of IC's, circuits, and systems, have resulted in hardware systems that meet the reliability requirement of long life and mission-critical aerospace military systems. This reliability became a function of cost and profit. Similar effort must be applied to the software for training device computers. Unfortunately, there is little of the engineering discipline that is characteristic of hardware design in the software area.

Hardware design has involved compromises between many alternatives. Sufficient data are available to optimize specific characteristics which determine maximum efficiency, minimum costs, and minimum weight. This type of data is not available for software.

The life-cycle costs of software have been too long ignored. The problems involved in estimating software production costs must not be allowed to prevent progress in estimating software life-cycle costs. Although there is little historical data available on the cost of software maintenance and enhancement, it is not too late to start the development of a data base in this area.

A large part of training device costs are expended on software. Although the development of software is a high-cost item, the maintenance and enhancement of software is a long-term, high-expense item throughout the system's life cycle.

As with any system, efforts at conservation must be applied to the higher cost areas to be fully effective. In the field of training devices, it is software and software maintenance that is the high-cost item. In this era of limited budgets and increasing inflation, effort must be diverted towards predicting and controlling software costs.

Control of software costs must include prediction of high-cost areas. These are the phases that significantly influence the cost of support over the life cycle of the system. Cost control reduction and prediction is most important. Examination of alternative support concepts and taking advantage of flexibility in design of support systems is a necessity. Cost control can also be affected through reducing mistakes that result in large costs.

Effort expended in determining software life-cycle costs will aid in support planning, identifying high-cost areas, and open the door to more effective software maintenance and enhancement.

BIBLIOGRAPHY

1. AIAA Conference on Software Management, Washington: American Institute of Aeronautics and Astronautics, 1977.

2. Boehm, Barry W. "Software Engineering," IEEE Transactions on Computers, (Dec 1976), pp 1226-1241

3. Department of Defense, Casebook Life-Cycle Costing in Equipment Procurement Publication LCC-2, 1970

4. Farr, L. and B. Nanus., "Factors that Affect the Cost of Computer Programming," Technical Documentary Report No. ESD-TR-64-448. Electronic Systems Command, United States Air Force, 1964.

5. Fisher, David A., Automatic Data Processing Costs in the Defense Department, Arlington, VA: Institute for Defense Analysis, 1974.

6. Kirby, George T., "Digital Computers in Training Devices: Trends and Forecasts"

Tenth NTEC/Industry Conference Proceedings, Orlando, FL, 1977, pp 261-270.

7. Lientz, B. P., E. B. Swanson, and G. E. Tompkins, "Characteristics of Application Software Maintenance," Communications of the ACM, Vol 21, No. 6 (June 1968), pp 466-471

8. Liv, Chester C., "A Look at Software Maintenance," Datamation, 11 (Nov 1976), pp 51-55

9. Naval Postgraduate School. Proceedings: Symposium on the High Cost of Software, Monterey, CA, 1973

10. Riggs, R., "Computer Systems Maintenance," Datamation, 15 (Nov 1969), pp 227-235

11. Rotherg, Brian, Installing and Managing A Computer System, London: Business Books Limited, 1968

12. Swanson, E. B., "The Dimensions of Maintenance," Proceedings 2nd Conference on Software Engineering, (Oct 1976), pp 492-497

13. White, Gerry C., "Impact of Microprocessors on Training Devices," Tenth NTEC/Industry Conference Proceedings, Orlando, FL, 1977, pp 293-296

ABOUT THE AUTHOR

DR. GERRY C. WHITE is a Systems Engineer in the Computer Laboratory at the Naval Training Equipment Center. Previously, he was a Software Engineer in the Software Development Section. While President of John E. Pochie Systems Consultants, he was employed by Memphis State University as an Associate Professor of Computer Systems Technology. He received the B.S. and M.S. degrees in electrical engineering from Christian Brothers College and the University of Tennessee and his Doctorate from Texas A&M University. Dr. White is a member of the Institute of Electrical and Electronics Engineers and is Vice Chairman of the IEEE Computer Society.