

SOFTWARE LOGISTICS SUPPORT ANALYSIS

David B. Wile
Major, United States Air Force

Headquarters Air Force Space Command
Integrated Logistics Support Directorate

ABSTRACT

There is a lack of an effective means to incorporate software support factors into the Logistics Support Analysis (LSA) process. This can result in a suboptimal system design and increased support costs. This paper highlights the deficiencies in the currently applied LSA process, and proposes a course of corrective action.

INTRODUCTION

System supportability concerns are fragmented on some system acquisitions. This is particularly evident when examining the integration of hardware and software support factors. Logistics Support Analysis (LSA) is the logical method to integrate system supportability concerns.

LSA is a subset of the system engineering process in which supportability becomes a design parameter coequal with cost, schedule, and performance. LSA is also the means by which support requirements of a given design are determined and documented.

The LSA military standards (MIL-STD-1388-1A, Logistics Support Analysis,¹ and MIL-STD-1388-2A, Requirements² for a Logistics Support Analysis Record²) apply to the total system. However, the detailed data elements and tasks required to perform LSA on software are lacking. The LSA output reports, generated from the LSA Record (LSAR), currently document only the hardware support requirements. Software support requirements are developed independently under the software military standard, DOD-STD-2167A, Defense System Software Development.³

The identification of software support is frequently done after the system is designed, preventing supportability factors from influencing that design. There are software support analysis techniques that can be made an integral part of

system's LSA process. An approach for synergistically incorporating these techniques is required.

ORGANIZATIONAL ISSUES

Within both government and industry, program management and design teams are often aligned along hardware/software functional areas. Support concerns are also often addressed by a separate logistics staff.

Logistics staffs have traditionally come from a hardware support background. Software designers and managers normally have little experience with hardware support issues. Software support is often perceived as an issue unrelated to hardware support.

The hardware support analysis process is normally performed using logistics support analysis military standards, while software support is determined under the software development DoD standard.

COST IMPACT

Software support is a major consumer of resources, both during the acquisition and operations phases. On some systems, software support is the single largest life cycle cost element. The Quality Assurance Institute estimates that the cost of maintaining software to keep it current, and keep out "bugs", consumes 90% of the data processing budget of some Fortune 500 companies. According to an article in Government Executive, DoD software support costs may reach over \$30 billion per year by 1990.⁴ Dr. Barry Boehm, a recognized expert in software economics, suggests software support cost may reach up to thirteen percent of the entire gross national product by 1990.⁵

MISSION/SUPPORT IMPACT

The LSA standards are intended to apply to the total system. However, the detailed task descriptions, data elements, and

output reports do not provide the necessary tools to conduct an efficient software support analysis. They also do not allow effective integration of hardware and software support factors.

The LSAR "A" sheet lists system level requirements including system availability. The reliability and maintainability requirements that support this availability are allocated to various hardware components and recorded on the "B" and "B1" sheets. Frequently, allocations are not made to software configuration items. This is equivalent to assuming software is 100% reliable.

On communication-electronics systems, software induced failures have a significant impact on system availability, mission reliability, and system restoral. On one recently deployed system, software induced critical failures were occurring daily with hardware induced critical failures occurring monthly. The system recovered from the typical critical software failure by rebooting. Rebooting took, on the average, between 10 and 20 minutes. Critical hardware failures were similarly recovered in 10 - 20 minutes using switchover techniques. For this particular system, software reliability and maintainability factors had 30 times the impact on system availability than hardware. (Note: Based on the results of reliability testing, the contractor subsequently corrected the software, thereby improving its reliability).

Software support also requires the use of program development tools, compilers, text editors, and machine time. Training in the use of these tools may also be required. The support analysis and documentation for software is normally done under DOD-STD-2167A. Due to this lack of a single process, or set of documentation, that treats system support from a system perspective, support suboptimization can result.

TRADE-OFFS

With the segregation of hardware and software design teams, there is little opportunity to perform supportability or performance trade-offs from a system perspective. An example of a system level trade-off is the allocation of machine time.

In addition to the machine time required for mission functions, time is also required for support functions such as operator training, maintenance training, software development, and system testing. Design trade-offs relating to the number of machines, machine sizing, location, user interfaces, and reliability & main-

tainability requirements should be made on the total system mission and support requirements. Optimizing the design a specific function, such as software maintenance) may result in a suboptimization of system capability and/or an increase in total system cost.

Trade-offs can also be performed within the software arena. For example, a faster execution time, or more efficient memory allocation, could be accomplished by writing a specific module in a nonstandard language. This increase in performance must be evaluated against the cost of additional compilers, program development tools, text editors, and training.

SURVEY RESULTS

HQ USAF/LE established an LSA Acquisition Review Group (LSA ARG) to evaluate the LSA process and recommend any required changes. The LSA ARG conducted a survey of over 30 government and industry organizations. A portion of this survey addressed software LSA. The pertinent result are:

- 80% of the respondents stated their contracts required application of LSA during software development.
- 86% stated they felt LSA was necessary for software.
- 80% indicated that LSA, as currently structured, is not an effective software design tool.
- 80% stated they felt the current LSAR format was not adequate to accomplish the required software support analysis.
- 80% reported that they felt the software LSA data base should not be kept separate from the system LSAR data base.
- 89% indicated that they did not include software support cost in their system life cycle cost analysis.

CORRECTIVE ACTION

As shown by the survey, a high percentage of respondents agreed that there are major deficiencies in the area of software support analysis. To correct these deficiencies, several actions should be taken:

- Integrate the LSA standards (MIL-STD-1388-1A, MIL-STD-1388-2A) and software standard DOD-STD-2167A) tasks and associated documentation.
- Establish data elements in MIL-STD-1388-2A that reflect software support parameters.

- Establish output reports that can be digitally inputted into predictive software support models and the resultant trade-offs digitally reininputted into the LSAR.

- Modify existing output reports to include manpower, training, tools, and support equipment required for software support.

PROPOSED DATA ELEMENTS

There are many possible approaches to the software support issue. If both system and software design are to be influenced by supportability factors, a means of predicting software support requirements during the design process is required. Parameters, such as estimated source lines of code, languages used, and estimated code complexity could be used in mathematical estimating relationships to predict software reliability and support requirements.

One model used for predicting software maintenance requirements is the COstructive Cost Model (COCOMO) developed by Dr Barry Boehm. The parameters used in his model, combined with other parameters, could be used as the basis for developing a set of software LSAR data elements.

One possible set of elements is:

LSA Control Number (LSACN) - Reference number in LSAR.

Computer Program Identification Number (CPIN) - Reference number to software program.

Computer Software Configuration Item (CSCI) Number/Title - Reference to specification.

Source Code Language (SCL) - Ada, Jovial, Atlas, Fortran, etc.

Number of Source Lines Of Code (SLOC) - Number of lines before compiling, or parameter such as Elshoff's Estimator of Program Length.

Number of Computer Program Component (CPCs) - Documented in B5/C5 specifications.

Software Type (ST) - Defined in AFR 800-14, Atch 7 (Mission Operational, Mission Support, Mission Development, System Operational, System Support, & System Development).

Software Category (SC) - Commercial Off-The-Shelf (COTS), Modified COTS (MODCOTS), & Newly Developed (DEV).

Software Complexity (COMP) - Estimator such as Halstead's Complexity Numbers, or McCabe's Cyclomatic number.

Software Maturity (MAT) - Estimate of errors remaining at a specified time, calculated using the Goel or other model.

Complexity of Security Requirements (SEC) - Level of certification required.

Software Support Tools (SST) - A listing of program development tools, compilers, etc., needed for software support.

Annual Change Activity (ACT) - Fraction of lines of code that undergoes change by addition and/or modification in a year.

Estimated Machine Time (EMT) - Number of hours of machine time required per month for software development, test, and other software maintenance activities.

ALGORITHMS/OUTPUT REPORTS

Once data elements are established in the LSAR, they must be outputted in a useful format. Algorithms relating the number of lines of code, annual change activity, and number of programmers required for each language would have to be developed and validated. Information on support resources, such as personnel, software tools, compilers, and machine time required must be computed and listed.

A first step in developing algorithms is to collect data on systems in the acquisition and deployment phases, and then conduct regression analysis. Estimating relationships could then be developed and validated. Existing models, such as COCOMO could also be used. Digital output reports from the LSAR, similar to LSA 036 report used for provisioning, could serve as input data for various support models.

Instead of designing all new outputs just for software, some existing reports could be modified to document total system support requirements. For example, it may be feasible to include software manpower requirements in the LSA 002, Personnel and Skill Summary. The LSA-020 might be also be modified to include a section on software support tools required and the LSA 054 modified to include software failure rates.

TASK INTEGRATION

The current system allows the support tasks for software to be performed by software personnel and charged under software Work Breakdown Structure (WBS) items. Related LSA tasks might also be

performed by logistics personnel to document the same effort. This situation may result in the contractor conducting, and the government paying twice, for the same work. Different and conflicting information may exist between these two efforts. Since the results are sent to separate program management and support staffs, the disparities may not be noticed.

To correct this potential duplication of effort, and the resultant suboptimization of support, the tasks of the LSA and software military standards need to be integrated. The documentation of these tasks should be integrated within the LSAR to the maximum extent practical.

CONCLUSION

The current LSA and software standards do not facilitate conducting software LSA. Conducting software LSA from a system perspective could reduce support costs and, at the same time, improve system supportability. There are many possible approaches to conducting software LSA. One such approach was proposed. A concerted effort by software and logistics specialists from government, academia, and industry is required to standardize the methods used. These methods, and their associated data elements, algorithms, and output reports should be institutional in the appropriate military standards.

REFERENCES

1. Mil-Std-1388-1A, Logistics Support Analysis, Department of Defense, Washington, DC, 11 April 1983.
2. Mil-Std-1388-2A, DoD Requirements for a Logistics Support Analysis Record, Washington DC, 14 February 1986.
3. DoD-Std-2167A, Defense System Software Development, Department of Defense, Washington DC, 29 February 1988.

4. "Project Boldstroke", Government Executive, January 1987.

5. Boehm, Dr. Barry, Software Engineering Economics, Englewood Cliffs, NJ, Prentice-Hall, 1981, p18.

6. Logistics Support Analysis Acquisition Review Group Final Report, Vol III, Data Analysis, HQ USAF/LEYM, Pentagon, Washington DC, March 1987.

7. Boehm, Dr. Barry, Software Engineering Economics, Englewood Cliffs, NJ, Prentice-Hall, 1981.

BIOGRAPHY

Major Wile is the Chief, Technical Support Branch, Integrated Logistics Support Directorate, Deputy Chief of Staff, Systems Integration, Logistics, and Support, Headquarters, Air Force Space Command, Peterson AFB, Colorado. Previous assignments include a foreign military sales assignment in Saudi Arabia, duty as Deputy Program Manager for Logistics for space defense systems, and service as a supply officer.

Major Wile is a Senior Member of the Society of Logistics Engineers (SOLE), and has earned the society's Advanced Professional Designation. He has earned a Master of Science Degree in Logistics Management from the Air Force Institute of Technology, and a Bachelor of Science in Business and Management (Summa Cum Laude) from the University of Maryland.

Previously published works include various articles on logistics in the Air Force Journal of Logistics, Proceedings of 1983 International Symposium of the Society of Logistics Engineers, and the Proceedings of the Sixth Joint AFSC/AFLC Reliability & Maintainability Workshop.