



Mission-Critical and Mission-Support Software: A Preliminary Maintenance Characterization

Elizabeth K. Bailey Clark, *Software Metrics, Inc.*

James A. Forbes, *Logistics Management Institute*

Emanuel R. Baker, *Software Engineering Consultants, Inc.*

Donald W. Hutcheson, *Weapons Business Development, The Boeing Company*

Within the Department of Defense (DoD), mission-critical software maintenance has been reported to cost between \$700 million and \$20 billion annually. The wide range of estimates results from uncertainty over the definitions of "mission-critical" and "software maintenance" as well as the lack of any catalog of performing activities. The problem is deeper than definitions and level of investment: software maintenance process is poorly characterized in general. The purposes of this study¹ were to undertake an initial characterization of DoD mission-critical software maintenance in terms of its activities and processes, users and stakeholders, amount of resources, and existing formal and informal policy; to identify policy issues; and to outline the scope and major features of potential new or revised policy.

THE PROBLEM, HOWEVER, is deeper than definitions and level of investment.

The software maintenance process is poorly characterized in general. Lacking an adequate characterization of software maintenance, there is no real basis to establish coherent policy. Further, key software maintenance decisions—such as contract choice or organic performance, and whether it should be defined as depot maintenance—are largely ad hoc and reap limited benefits from the results of past decisions.

The terms "software maintenance" and "software support" are both in use, sometimes with modifiers such as "post-production" or "post-deployment." To avoid confusion, we adopted the term software maintenance and defined it to include

- Correction of defects.
- Adaptation to a new host operating environment.
- Incremental functional improvements.

This definition is generally consistent with industry usage. Excluded from this definition are major modifications and upgrades, the purpose of which is major functional improvement.

We found it helpful to distinguish among three categories of mission-related software: mission-critical, embedded; mission-critical, nonembedded; and mission-support (Table 1). Broadly speaking, different organizations may use similar processes within a category; across categories they generally do not.

It also is helpful to characterize software maintenance by application area (Table 2). We gathered data on the first six application areas of Table 2 (shaded in the right column). Given the state of data availability and reasonable limits on study scope, it proved impractical to assure completeness for any category or to achieve a reasonable degree of completeness for other than the first three.

Approach

Our study approach is illustrated in Figure 1.

We separated the research into two segments: quantitative

and qualitative. To establish the "demographics" of software maintenance, e.g., rough order of magnitude estimates of the code base, number of people performing, and annual cost, we started with a database created by the Institute for Defense Analyses for the Commission on Roles and Missions (CORM) of the armed forces. Because it was clear from the beginning that this database (the result of a data call to the services) had some voids, we supplemented it with data we obtained directly from the services. This study does not include software maintenance performed by defense agencies; the decision to exclude defense agencies was driven by the need to establish a reasonable scope of effort for what was envisioned as primarily an exploratory study.

To approach the more qualitative aspects, such as those that have to do with the software maintenance process, we began with a literature review and conducted a series of 15 semistructured interviews at eight service installations. In keeping with the unsettled nature of software maintenance, we focused on developing an understanding of the common norms, meanings, values, and organizational relationships [1]. We were more

Table 1. *Software maintenance categories.*

Type	Cardinal Characteristics	Examples
Embedded	<ul style="list-style-type: none"> • Tightly coupled interfaces • Real-time response requirements • High reliability requirements (life-critical) • Generally severe memory and throughput constraints • Often executes on special-purpose hardware 	B-1 flight software, F-14 flight software
Operational, nonembedded	<ul style="list-style-type: none"> • Multiple interfaces with other systems • Constrained response time requirement • High reliability but not life-critical • Executed generally on commercial off-the-shelf products (COTS) 	C ³ , space systems
Mission-support	<ul style="list-style-type: none"> • Relatively less complex • Self-contained or few interfaces • Less stringent reliability requirement 	automatic test, equipment test, program sets, mission planning, business systems

Application Area	Type	Data Completeness
Weapon systems	Embedded	Essentially complete
Space control	Nonembedded	Essentially complete
Automated test equipment	Support	Essentially complete
C3	Nonembedded	Partial
System integration labs	Nonembedded	Partial
Simulation and training	Nonembedded	Partial
Atmospheric search	Nonembedded	none
War games and mission rehearsal	Nonembedded	none
Intelligence	Nonembedded	none
Business systems	Support	none
Weather	Nonembedded	none
Other	-	

Table 2. Scope of DoD software maintenance.

interested in discerning signposts and perspectives² than trying to determine “facts.” In combination, the demographics research, literature review, and interviews permitted us to do this by characterizing software maintenance in terms of activities and processes, users and stakeholders, amount of effort, and existing formal and informal policy. Policy issues flow from that characterization.

Findings [2]

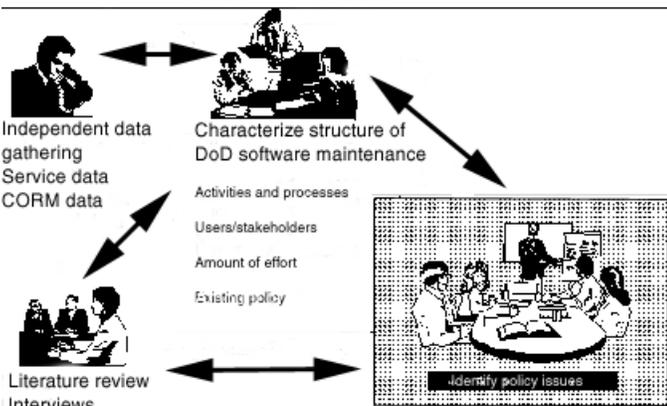
Within the scope of the study, we accounted for an estimated 16,000 government and contract persons performing software maintenance on 278 million source lines of code (SLOC) at a cost of \$1.26 billion annually. We found that approximately 55 percent of these people were government employees, and 45 percent were contractors. Approximately 40 percent focused on software correction, and 60 percent focused on a combination of adaptive and incremental improvements.

Code Base

Figure 2 shows a breakout by the three high-level categories for each service. The Navy and the Air Force have much larger code bases than does the Army.

Although support software is the single largest category in terms of the sheer number of SLOC, it is less costly to maintain than the other two categories. As an indicator of the difference, Table 3 reflects the approximate cost per SLOC per year for

Figure 1. Study approach.



three of the sites in the expanded database.

In interpreting Figure 2, remember that there are significant reliability and validity issues with the underlying data. Although our check of code counts reported in the CORM database against those made available in site visits did not reveal a systematic bias, that is not the same as saying the data are known to be valid. Because only three of the six application areas we examined were reasonably complete, this summary is an underestimate even for the areas we examined. The portrayals shown here are best characterized as approximate representations of the relative sizes of the code bases for the categories we examined. These caveats also apply to the labor force demographics presented and budget impact.

Personnel

Use of operations and maintenance (O&M) funds is almost universal for software maintenance within the application areas studied. The amount of resources is normally determined as a level of effort rather than built up from discrete requirements. In some organizations, the level of effort was fixed in terms of

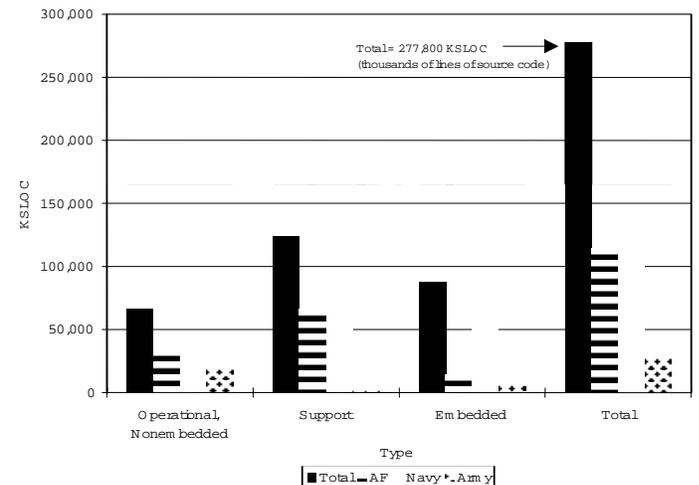


Figure 2. Software code base by service and category.

dollars, in others by the fairly stable size of the labor force. In either case, software maintainers addressed the backlog of requirements to the extent resources permitted. Requirements not satisfied in one planning period, e.g., year, were deferred to the following period. This approach also appears to be consistent with industry software maintenance practice.

Software development and maintenance are labor-intensive. Human effort is generally recognized to be the major cost driver [3, 4]. To estimate the number of people involved in software maintenance, we began with the CORM database personnel counts. Here also, we expanded the CORM database using other data gathered during the study. To determine accuracy, we compared, as we did with the size data, the numbers obtained from the site visits with those in the CORM database.

The CORM database consistently underrepresented the number of people. A comparison between the CORM and the site visits is shown in Figure 3. If the data from the site visits and the CORM data for the same sites were about the same, a

Category	Approximate maintenance cost per line of code per year
Embedded	\$110.00
Nonembedded	\$5.60
Mission-support	\$0.81

Note: The mission-support cost is calculated from North Island ATE TPSs, nonembedded is calculated from CECOM data, and embedded is calculated from B-1B data.

Table 3. Representative maintenance costs by category.

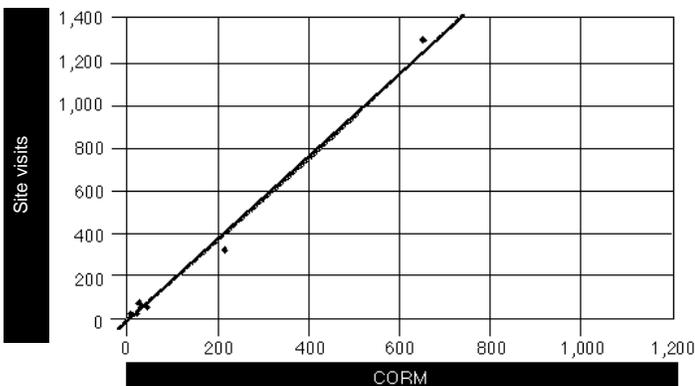
linear plot of the data would have a ratio of 1-to-1 slope. The slope is 1.96, which means that the personnel counts obtained from the site visits were almost twice as large as those from the CORM data call, and this was consistent for all but one of the sites we visited. The one inconsistency was the F/A-18 Hornet aircraft. The CORM data call reflects 30 F/A-18 personnel, all organic, while interviews with F/A-18 software managers indicate the total should be approximately 1,000 (125 organic plus 875 contractors). Since it was such an egregious error, we did not include the F/A-18 in calculating the 1.96-to-1 site-visit-to-CORM data ratio.

Budget Impact

The third measure of magnitude is dollars. We did not use the budget numbers from the CORM data call because it is unclear what these reflect, i.e., labor only or labor and equipment or contract or contract plus organic. As an alternative, we estimated the financial commitment in dollars by multiplying counts of people by average loaded labor rates for organic and contractor personnel. Figure 4 shows the estimated dollars per year for each service.

The rate used for organic personnel was \$67,364, which is a composite rate based on an assumed distribution of 80 percent GS-12 and 20 percent GS-13 (1996 dollars) [5]. The rate used for contractor personnel was \$97,364, which is the median of the rates that were quoted to us during the site visits. The contractor rates ranged from \$55,500 to \$250,000 per year, and this difference generally corresponded with the complexity and

Figure 3. Personnel data from eight site visits compared to CORM data for same sites.



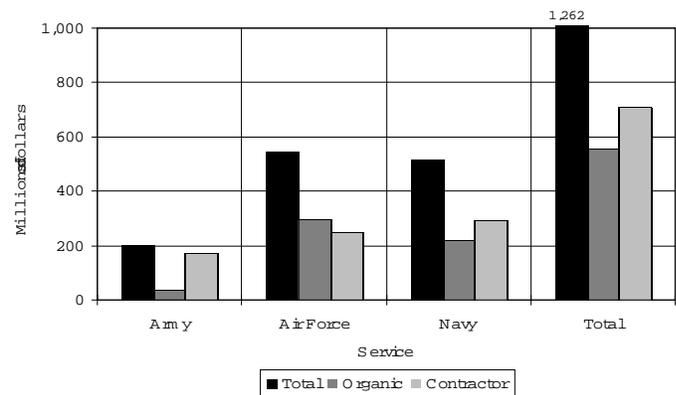
uniqueness of the software being maintained. The difference between organic and contractor rates should not be interpreted to mean that contractors are more expensive. By and large, the contractor labor force was maintaining more complex software that required higher skills. More to the point, we did not attempt to make such a comparison.

The financial commitment that we were able to account for using this procedure is approximately \$1.26 billion annually (\$205 million for the Army, \$543 million for the Air Force, and \$514 million for the Navy).

One of the reasons for characterizing DoD software maintenance was to shed light on the amount of software maintenance that also is depot-level maintenance. It is of interest whether software maintenance is depot level because it affects the department's compliance with the congressional restrictions on how much depot maintenance work can be outsourced [6].

It was not possible to describe what fraction of the \$1.26

Figure 4. Estimated budget impact by service.



billion in software maintenance is depot level. First, it was clear from the interviews that, here also, there is a lack of consensus over definitions. For example, the Air Force would generally classify work on fighter aircraft embedded software as depot maintenance. The Navy did not consider it so. Hence, inclusion or exclusion of software maintenance when reporting compliance with Title 10 U.S.C. limitations on depot maintenance outsourcing was inconsistent. There was a lot of uncertainty in this area, as were differences in counting rules. The Defense Depot Maintenance Council Business Plan for fiscal 1996-2001, which is compiled with service inputs, showed \$275.3 million in contract depot-level software maintenance for fiscal 1996 and an additional 3.2 million depot labor hours of organic support. By contrast, the AP-MP(A)-1397 Depot Maintenance Cost System Report, under which depot-level software maintenance was explicitly required to be reported, reflected \$20.4 million for the same year.

Transition Patterns

Software for the application areas studied normally is developed in the private sector. Although there were many transition pat-

terns from original equipment manufacturer (OEM) to maintainer, three reasonably clear trends emerged:

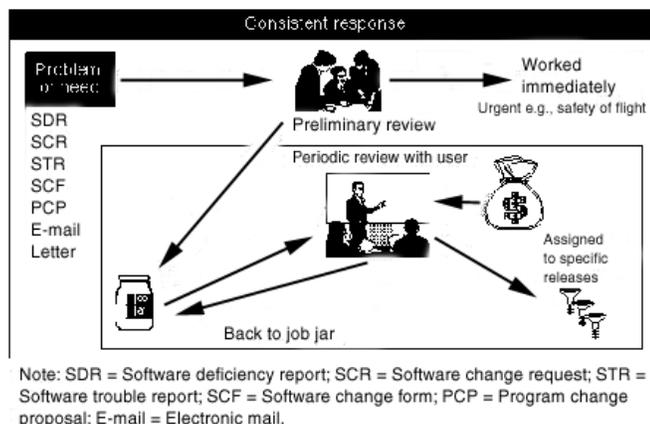
- Pure organic maintenance is the exception and seems limited to mission-support software, such as ATE TPSs. Since the organic and contract sectors have roughly the same skills and would be expected to use the same software environments, we conclude that, except for support software such as ATE TPSs, there is significant difficulty and cost associated with transferring the knowledge of the software necessary for its maintenance. In addition to problems with nondelivery of documentation or computer-aided software engineering environments, this knowledge is probably tacit, i.e., deep knowledge, rather than explicit—that is what makes it hard to transfer. What we have found in practice seems to support this conclusion. Support for this conclusion also is found in the literature on technology management in which David J. Teece [7], examining how companies arrive at make-or-buy decisions, noted that they often choose what is easy to do rather than what is most important to them.
- Organic maintenance of embedded software generally is found only on older models of weapons systems.
- Where attempted, competitive contract support proved both more economical and at least as effective as either sole-source contract support or organic support.
- Based on the empirical evidence, i.e., the established transition patterns, planning for pure organic maintenance or competed maintenance of embedded software is unrealistic. It probably is more realistic to accept OEM involvement in (and initial lead of) embedded software maintenance as an accomplished fact.
- Competed commercial maintenance is viable for mission-critical, nonembedded and for mission-critical, support software.

Communication of Requirements

Communicating requirements clearly is an important part of the software maintenance process. We found uniformity in this process among organizations in the field survey. The typical requirements process (Figure 5) follows these steps:

Figure 5. *Requirements process.*

Question: How are changes and requirements communicated and changes initiated?



- A user initiates it through a problem report or a change request. These reports or requests had almost as many names and acronyms as organizations surveyed. The names included System Deficiency Report, Standard Change Form, Software Trouble Report, and Program Change Proposal, or they could take the form of E-mail or letter input. Interestingly, no one in the Air Force reported using formal Technical Order 00-35-D54 deficiency reports, though this technical order applies to all Air Force agencies and organizations and provides for software deficiency reporting [8].
- The requests typically are screened in a preliminary review to determine the urgency of the problem or change request. Urgent needs, e.g., safety of flight, are worked immediately. The remainder of the requests are accumulated in what the Space and Warning Systems Directorate colloquially termed a *job jar* awaiting a scheduled review [9].
- The requests are periodically reviewed by an established group, e.g., F/A-18 System Change Review Board. Prior to the review, initial estimates of the magnitude of the effort—which changes can be efficiently grouped, etc.—are accomplished by an engineering staff. The reviews often have user participation or input. The group chartered to do the review examines the requests in the job jar, prioritizes them, and selects software changes to be implemented. Selection is based primarily on priority and available funding.
- Requests not selected go back to the job jar for future consideration. Typically, there are more requests than funds.
- Problem reports or change requests selected for implementation are assigned to a software version release.

Neither the size of the backlog of requirements nor the specifics of particular requirements in the backlog drives the budget. Rather, planned support takes the form of a level of effort expressed in dollars or work force. Essentially, the agreed-upon level of effort establishes a “cut line.” On a prioritized list of software maintenance requirements, software changes above the line are implemented; those below it are deferred to the job jar for future funding opportunities. This behavior would indicate that most software maintenance tasks are not of a time-critical nature. It is worth noting that level-of-effort funding is found in commercial software maintenance practices [10]. (There are at least anecdotal indications that it also is found in commercial software development.)

Operable Policy and Military Standards

A primary reason for this study was to understand what is needed in the area of software policy. Consequently, this topic was explored in some detail during the interviews. Policy can be viewed from two different perspectives. First, it can be considered as representing required behavior, i.e., as formal, normative policy, or the common view. Another perspective is to consider policy as providing a framework of consistent expectations regarding how affected parties mutually interact, i.e., as facilitating cooperative action [11,12,13,14]. Given the relative absence of normative software maintenance policy, both perspectives

were potentially important.

The most frequently cited documents were several military standards that prescribed software engineering processes. Almost universally, DOD-STD-2167 or DOD-STD-2167A were mentioned. Several respondents listed MIL-STD-498 as well. Two sites mentioned MIL-STD-1679.³ These military standards describe the documentation to be delivered, formal reviews to be held, and tasks to be addressed in developing or maintaining software. A fairly broad variety of other documents also were listed. These included DoD (especially 5000 series), service, and command regulations and instructions.

It was clear that military standards are the most important source of policy for software maintenance. The single most important reason for this was that the military standards provide a consistent framework of expectations for software developers and software maintainers—two communities that generally have limited interaction during software development. It is on the basis of what is described in the military standards that the software maintenance community knows what to expect in the way of software documentation. A considerable unease was expressed in almost all of the interviews regarding the demise of the military standards. This unease stems from the potential loss of this consistency of expectation. One expectation was the Navy's F/A-18 program, which has successfully eliminated the wall between developer and maintainer through the successful use of integrated product teams (IPTs) [15, 16].

Not surprisingly, given the de facto status of the MIL-STDs as policy, the ongoing elimination of MIL-STDs was an issue for almost all of the organizations we interviewed.

Recommendations

We made two sets of recommendations: one set related to general policy, and a second related to how DoD organizes for software maintenance.

Policy

- Standardize the term *software maintenance* and define it to include correction of defects, adaptation, and incremental improvements. Exclude major modifications.
- Define software maintenance in weapons systems, automatic test equipment, systems integration laboratories, and space control categories as depot maintenance. All four categories are either embedded in or closely tied to mission-essential platforms.
- Make routine the consistent reporting of depot-level software maintenance, as defined above, in the AP-MP(A)-1397 Depot Maintenance Cost System to provide a basis for reporting to Congress and management of depot-level software maintenance generally.
- Invest in process improvement. Consider mandating minimum process capability levels for both organic and contract activities that perform software maintenance.

Organizing for Software Maintenance

- To achieve scale economies, consolidate smaller software maintenance activities into software maintenance centers of

excellence. For each center of excellence, keep or put in place a strong central management structure.

- For embedded software, plan for long-term OEM maintenance. However, it is important to retain enough work organically to maintain smart-buyer capability.
- For mission-critical, nonembedded software, continue consolidation using the government-managed, contractor-performed, centralized-maintenance model employed by the Army Communications Electronics Command and the Air Force Space Systems Support Group.
- For software, such as automated test equipment, test program sets where the software engineering knowledge is relatively easy to transfer. Consider competition to reduce costs. ♦

About the Authors

Elizabeth K. Bailey Clark co-founded Software Metrics, Inc. in 1983. She is a primary contributor to Practical Software Measurement (PSM) and is a qualified PSM instructor. She has worked with numerous clients to implement software measurement. She was a primary contributor to the Software Engineering Institute's core measures. She is currently working with Barry Boehm and Chris Abts from the University of Southern California to develop and calibrate COCOTS, a software COTS integration cost model. She has a bachelor's degree from Stanford University and holds a doctorate from the University of California at Berkeley.

Software Metrics, Inc.
4345 High Ridge Road
Haymarket, VA 20169
Voice: 703-754-0115
E-mail: bkbailey@erols.com

James A. Forbes is a certified professional logistician with more than 25 years of experience. As an Air Force officer, he had extensive experience assessing the impact of technology on organizations and managing technology-driven change. In the private sector, he managed logistics programs for both Technology Applications, Inc. and Analytic Sciences, Inc. before coming to the Logistics Management Institute (LMI). At LMI, he has been a principal researcher in studies of depot maintenance, total ownership cost, and logistics-related information systems.

E-mail: jforbes@lmi.org

Emanuel R. Baker is president of Software Engineering Consultants, Inc., Los Angeles, Calif., and a principal officer of Process Strategies, Inc., a software consulting firm based in Los Angeles and Walpole, Maine. The author or co-author of several articles on software quality, software quality management, software engineering, and software process improvement, he has 40 years of engineering experience, 25 of which has been spent as a software engineer and as a consultant to software development organizations. He held a number of management positions and was manager of the Product Assurance Department of Logi-con's Strategic and Information Systems Division. He has played a

major role in the development of software quality standards for both industry and the Department of Defense. He has a bachelor's degree in engineering (1956) from New York University, New York City, a master's degree in mechanical engineering (1958), and a master's (1974) and a doctorate (1981) in education from the University of Southern California at Los Angeles.

E-mail: erbaker@ix.netcom.com

Donald W. Hutcheson has extensive experience in engineering development, flight testing, program management, senior technology management, and executive management of DoD weapons systems. He joined LMI in 1994 as a logistics analyst. He has a bachelor's degree in mechanical engineering and a master's degree in engineering. He is a member of the DoD Acquisition Corps, certified in program management. While at LMI, he has performed analysis in various areas of logistics reform, co-wrote three reports, and was a recipient of the LMI 1996 President's Award.

Weapons Business Development
The Boeing Company
P.O. Box 516 MC S500-2025
St. Louis, MO 63166-0516
Voice: 314-925-5633
Fax: 314-925-7758
E-mail: donald.hutcheson@mw.boeing.com

References

1. Lyytinen, Kalle J. and K. Klein Heintz, "The Critical Theory of Jurgen Habermas as a Means for a Theory of Information Systems," *Research Methods in Information Systems*, E. Mumford, et al., ed., Holland: Elsevier Science Publishers B.V., 1985, p. 221.
2. Kelle, Udo, "Theory Building in Qualitative Research and Computer Programs for the Management of Textual Data," *Sociological Research Online*, Vol. 2, No. 2, <http://www.socresearchonline.org.uk/socresonline/2/2/1.html>, 3.9.
3. Boehm, Barry W., *Software Engineering Economics*, Prentice-Hall, 1981.
4. Goethert, Wolfhart B., Elizabeth K. Bailey, and Mary B. Busby, *Software Effort and Schedule Measurement: A Framework for Counting Staff-Hours and Reporting Schedule Information*, CMU/SEI-92-TR-21, ESC-TR-92-021, Carnegie Mellon University, Software Engineering Institute, Pittsburgh, Pa., September 1992.
5. The composite organic rate is a weighted average of the rates shown for GS-12s and GS-13s in Table A26-1 of the *Civilian Standard Composite Pay Rates by Grade*, Air Force Instruction 65-503, May 1996.
6. 10 U.S.C. 2466, "Limitations on the Performance of Depot-Level Maintenance of Material," requires that not more than 50 percent of the funds available in a fiscal year to a military department or agency for depot-level maintenance and repair may be used to contract for performance by non-federal-government personnel.
7. Teece, David J., "Technological Change and the Nature of the Firm," *Technological Change and Economic Theory*, 1988, pp. 256-281.
8. U. S. Air Force, TO 00-35D-54, USAF *Deficiency Reporting and Investigation System*, Jan. 15, 1994, pp. 1-1, 1-4, and 1-5.
9. Space and Warning Systems Directorate, Operating Instruction 33-7, *Software Maintenance – Acronyms and Terms*, Vol. 2, Sept. 15, 1995, p. 6. The term *job jar* is used in this operating instruction.
10. Abran, Alain and Hong Nguyenkim, "Analysis of Maintenance Work Categories Through Work Measurement," *Proceedings of the 1991 IEEE Conference on Software Maintenance*, Sorrento, Italy, p. 105.
11. Forester, J., "Selling You the Brooklyn Bridge and Ideology," *Theory in Society*, September 1981, p. 746.
12. Forester, J., "The Policy Analysis-Critical Theory Affair: Wildavsky and Habermas as Bedfellow?" *Journal of Public Policy*, No. 2, 1982, p. 151.
13. Habermas, J., *The Theory of Communicative Action*, 1984, Vol. 1, p. 308.
14. Seidman, S., ed., *Jurgen Habermas on Society and Politics: A Reader*, Beacon Press, Boston, 1989, p. 154.
15. Bailey, Elizabeth K. and Beth Springsteen, *The F/A-18E/F: An Integrated Product Team (IPT) Case Study*, Institute for Defense Analyses, Alexandria, Va., April 9, 1998.
16. Springsteen, Beth, Elizabeth K. Bailey, Sarah H. Nash, and James P. Woolsey, *Integrated Product and Process Development Case Study: Development of the F/A-18E/F (Draft Final)*, Institute for Defense Analyses, Alexandria, Va., April 22, 1999.

Notes

1. The study was performed pursuant to Department of Defense Contract DASW01-95-C-0019.
2. The findings presented are extracted from Logistics Management Institute Report LG518T1, November 1997 and represent the more significant findings of the study. For more detailed findings, please examine the report.
3. MIL-STD-498 replaced both DOD-STD-2167A (for weapons systems and other mission-critical applications) and DOD-STD-7935A (for automated information systems) and brought these two areas together under one standard.