ASN(RD&A)

Guidebook for Acquisition of Naval Software Intensive Systems

Office of the Assistant Secretary of the Navy
(Research, Development and Acquisition)

Version 1.0

September 2008

Available at: http://acquisition.navy.mil/organizations/dasns/rda_cheng.

This Page Intentionally Left Blank

Table of Contents

List of Tables and Figures

Executive Summary

Overview

Successful development and acquisition of software is vital for acquiring naval warfighting and business systems. Software intensive systems are inherent in today's complex systems and are often the primary cost, schedule, and performance drivers in naval programs. The Software Process Improvement Initiative (SPII) chartered by the Assistant Secretary of the Navy (Research, Development and Acquisition) (ASN(RD&A)) developed this guidebook to provide a uniform framework for software improvement processes to assist Department of the Navy (DoN) acquisition teams through all phases of software acquisition.

Purpose

The *Guidebook for Acquisition of Naval Software Intensive Systems* is intended to provide support for the entire acquisition team by consolidating in one place background information, enterprise-wide policy, guidelines, proven alternatives, access to additional subject matter expertise, and amplifying detail for key software acquisition activities. It includes:

- General information concerning DoN and Department of Defense (DoD) software acquisition consideration at various stages of software acquisition planning: pre-solicitation, solicitation, source selection, and contract execution;
- Amplifying guidance for ASN(RD&A) policy regarding software process improvement;
- Assistance with implementation of mandated metrics; and
- Assistance with understanding and implementation of Electrical and Electronic Engineers/Electronic Industries Alliance (IEEE/EIA) Standard 12207.

Key Recommendations

This guidebook is a companion document to several ASN(RD&A) policy memos mandating various software acquisition actions, and provides the "how" for those mandates as well as supporting guidance and references for associated activities. Embedded in the "how" descriptions are numerous recommended processes and alternatives. These recommendations include:

- Start early and identify and implement software measures. Work closely with the entire acquisition team during pre-RFP (Request For Proposal) phases to ensure government requirements are captured in the RFP. Post-contract award, work closely with the developers to ensure software measures are efficient, effective, and most importantly applied to minimize program risk.
- Ensure adequate and appropriate training and experience are available on the acquisition team throughout all acquisition phases (adjust as required when transitioning through phases).

- Incorporate both mandated and tailorable contract language to ensure software development and management best practices are identified and implemented.
- Elevate software acquisition activities throughout the acquisition team (program office, Integrated Product Team (IPT), etc.) to a high enough level for visibility, accountability, and effective program management.

1 Introduction to Acquisition of Naval Software Intensive Systems

1.0 Overview

Successful development and acquisition of software is vital for acquiring naval warfighting and business systems. Software intensive systems are inherent in today's complex systems and are often the primary cost, schedule, and performance drivers in naval programs. The development, acquisition, and delivery of software are key to the Navy's ability to successfully conduct its warfighting and business operations. The naval establishment must achieve the ability to develop and acquire software without sacrificing the cost, schedule and performance goals of acquisition programs. The Software Process Improvement Initiative (SPII) chartered by the Assistant Secretary of the Navy (Research, Development and Acquisition) (ASN(RD&A)) developed this guidebook to provide a uniform framework for software improvement processes to assist Department of the Navy (DoN) acquisition teams through all phases of software acquisition.

1.1 Purpose

The *Guidebook for Acquisition of Naval Software Intensive Systems* is intended to provide support for the entire acquisition team by consolidating in one place background information, enterprise-wide policy, guidelines, proven alternatives, access to additional subject matter expertise, and amplifying detail for key software acquisition activities across the acquisition lifecycle. It complements the Department of Defense (DoD) Acquisition guidebook and DoN acquisition instructions and includes:

- General information concerning DoN and DoD software acquisition consideration at various stages of software acquisition planning: pre-solicitation, solicitation, source selection, and contract execution;
- Amplifying guidance for ASN(RD&A) policy regarding software process improvement (see Appendix A, Appendix B, Appendix C, Appendix D, and Appendix E);
- Assistance with implementation of mandated metrics; and
- Assistance with understanding and implementation of Electrical and Electronic Engineers/Electronic Industries Alliance (IEEE/EIA) Standard 12207.

1.2 Using the Guidebook

This guidebook assumes the reader has some understanding and experience with naval acquisition. There are frequent references to DoD acquisition terms and source documents that should be familiar to the reader, with each chapter including reference lists as well as website Uniform Resource Locators (URLs) for additional acquisition guidance. The level of software expertise is assumed to vary across the intended audience, ranging from no exposure through introductory Defense Acquisition University (DAU) courses (e.g., Software Acquisition Management (SAM) 101) to advanced education and experience, so where appropriate basic concepts have been explained in order to introduce more advanced concepts; the basic concepts will serve as general refreshers for more experienced readers.

Users are anticipated to be from a variety of acquisition disciplines (e.g., program management, legal, contracting, etc.). Therefore, while the guidebook as a whole provides useful information, each chapter is also written to stand by itself for focused use by individuals interested in a particular aspect or phase of the acquisition lifecycle. As some appendices are referenced by more than one chapter, all appendices are located at the end of the guidebook. The electronic version has been formatted for ease of printing and maintaining in hard copy should that be preferred.

The Table of Contents provides a useful summary of the topics addressed; Table 1-1 provides a quick reference for high visibility acquisition program office activities and the most relevant guidebook chapters and appendices.

Guidebook Quick Reference	
Focus Area	Guidebook Chapter(s)
Metrics	Chapter 2, Appendix F
Policy	Chapter 1, Appendices A, B, C, D, E
RFP Preparation	Chapters 5, 6, 7
Source Selection	Chapters 5, 8
Staffing	Chapters 3, 4; Appendices G and H

Table 1-1. Guidebook Quick Reference

Questions, comments, and recommendations are invited and encouraged. Input should be provided to the ASN(RD&A) Chief Systems Engineer (CHSENG) office.

1.3 Background

1.3.1 Identifying the Problem

A major contributor to software acquisition problems is the fact that software is often misunderstood and treated like hardware or driven by a hardware schedule, even though there are significant differences between the two. Software isn't manufactured and doesn't wear out; when software fails to perform as expected or required it is primarily because of flaws in the requirements, design, or implementation. Problems with the acquisition of software intensive systems are numerous and well recognized. They have been studied and re-studied, and literally dozens of reports have been issued to educate readers on how to recognize, mitigate, and avoid such problems. As far back as 2000, the Defense Science Board (DSB) Task Force on Defense Software Development and Acquisition Programs highlighted the following findings:

- Requirements were much too complex or rigid;
- Developer lacked software skills and experience;
- Poor software management practices – developer and/or program office;
- Lack of effort up front on system architecture;

- Lack of system engineering trading hardware/software;
- Adherence to policy & directives at expense of system performance & functionality;
- No real financial incentives; and
- Program management did not anticipate or could not fix the problems.[1]

The 2000 DSB Task Force is not the only effort to identify challenges the naval acquisition community faces in procuring software intensive systems within programs of record's established budget, schedule and performance criteria. Other studies and related initiatives which influence software acquisition include the following:

- Documentation of software problems and solutions in reports issued by entities such as the Government Accountability Office (GAO), Software Program Managers Network (SPMN), Software Engineering Institute (SEI), Tri-Service Assessment Initiative (TAI), Naval Research and Advisory Council (NRAC), and the National Defense Industrial Association (NDIA);
- Office of the Chief of Naval Operations (OPNAV) endorsed provisions to adhere to the software development methodologies described in the Open Architecture (OA) initiatives;
- Office of the Secretary of Defense (OSD) sponsored initiatives to address software assurance related threats and vulnerabilities inherent in procuring software from open source industry partners (supplier assurance);
- Initiatives to adopt Model Driven Architecture precepts and development methodologies as part of the naval software procurement efforts; and
- The introduction and adoption of new process driven models such as Lean Six Sigma, Theory of Constraints and First Pass Yield, and tools by some Navy acquisition commands to help improve software procurement success rates.

1.3.2 Public Law

Congressional recognition of serious acquisition issues, in part based on the DSB study findings noted in section 1.3.1 as well as numerous other study findings and individual program breaches and failures, resulted in Section 804 of the Bob Stump National Defense Authorization Act.[2] This mandates that military departments and defense agencies "shall establish" a program to improve software acquisition processes. Called out for inclusion in the program were:

- A documented process for software acquisition planning, requirements development and management, project management and oversight, and risk management;
- Efforts to develop appropriate metrics for performance measurement and continual process improvement;
- A process to ensure that key program personnel have an appropriate level of experience or training in software acquisition; and
- A process to ensure that each military department and defense agency implements and adheres to established processes and requirements relating to the acquisition of software.

[1] Department of Defense. Office of the Under Secretary of Defense For Acquisition and Technology. Report of the Defense Science Board Task Force on Defense Software. Washington, D.C., 2000.
[2] FY03 Defense Authorization Act. Public Law 107-314. 2 Dec. 2002. STAT. 116. 2465. Sec. 804.

1.3.3 Challenges

Software acquisitions span the breadth of naval acquisition and include air, surface, and subsurface platforms; weapons systems; Command, Control, Communications, Computers and Intelligence (C4I) and Information Technology (IT) programs; and many others. The software acquisition challenges those programs face are just as varied and include:

- Personnel issues: The naval establishment has historically not fared well in identifying, developing, and retaining a career work force experienced in software development and/or acquisition. Exacerbating the issue is the complexity of acquisition of Software Intensive Systems (SIS) and Software Intensive Systems of Systems (SISOS). All too often, even when present on staff, the influence of those in the software domain is diminished in the larger acquisition efforts due to the manner in which software acquisition is embedded within the program.
- Legacy Practices: Legacy practices are safest and often the “path of least resistance” for decision makers.[3] Most emphasize practices that are not compatible with SISOS or SIS development, such as the following:
  - Assumption that the system, or systems, is “standalone” or not affected by Systems of Systems (SOS) or changes to environments or interfaces;
  - Sequential versus concurrent engineering;
  - Risk-insensitive versus risk-driven processes;
  - Assumptions and early definition of poorly understood requirements versus better due diligence leading to understanding of needs and opportunities; and
  - Slow, unscalable contractual mechanisms for adapting to rapid change or poorly formed or understood requirements and their allocation among several contracts or contractors.
- Integration of Software Engineering: System Engineering and Software Engineering are frequently not unified in Work Breakdown Structure (WBS), schedule, and management. Cultural differences and differing priorities between System Engineering and Software Engineering remain, despite some efforts to unify the two.[4] Risk management should include all engineering and support staff and contracts in the program office and the contractor(s).
- Process Compliance: In an SOS environment, process selection and compliance is complicated by teams with multiple vendors and multiple schedules. The processes of one vendor are often not compatible with the processes of another vendor even though each company has been appraised at a high maturity level. While individual contractors may be compliant with an agreed upon model or standard for processes, each company’s terminology, tools, analysis methods, and local culture are reflected in their process implementation. Contractors are often reluctant to modify their own processes despite the potential for stronger process interfaces between government and industry team members, hence the expression, “Process Compliance is Global, Process is Local.” Different processes will also produce metrics which each vendor may call the same name as another but use different artifacts or measurement references (e.g., Source Lines of Code (SLOC) vs. Function Point) to produce an incompatible metric. In addition to the engineering aspects of a large procurement, mismatched contracting processes may hinder such things as the contractors’ responsiveness in processing change requests. Contracting and legal issues can prevent program offices and

[3] Boehm, Barry, and Jo Anne Lane. “Using the Incremental Commitment Model to Integrate System Acquisition, Systems Engineering, and Software Engineering.” CrossTalk: The Journal of Defense Software Engineering. October (2007): 4-9.
[4] Boehm, Barry. “Unifying Software Engineering and Systems Engineering.” IEEE Computer, 33 no. 3: 114-116.

contractors from making rapid changes when required.[5] This is exacerbated when software expertise is buried deeply in the Integrated Product Team (IPT) structure or has no interface with other SOS elements.

1.4 Key Recommendations

This guidebook is a companion document to several ASN(RD&A) policy memos mandating various software acquisition actions, and provides the "how" for those mandates as well as supporting guidance and references for associated activities. Embedded in the "how" descriptions are numerous recommended processes and alternatives. These recommendations include:

- Start early and identify and implement software measures. Work closely with the entire acquisition team during pre-RFP (Request For Proposal) phases to ensure government requirements are captured in the RFP. Post-contract award, work closely with the developers to ensure software measures are efficient, effective, and most importantly applied to minimize program risk.
- Ensure adequate and appropriate training and experience are available on the acquisition team throughout all acquisition phases (adjust as required when transitioning through phases).
- Incorporate both mandated and tailorable contract language to ensure software development and management best practices are identified and implemented.
- Elevate software acquisition activities throughout the acquisition team (program office, IPT, etc.) to a high enough level for visibility, accountability, and effective program management.

[5] Hantos, Peter. "Risk Management in System of Systems Development - Are You Ready?" 20th International Forum on COCOMO and Cost Modelling. Center for Software Engineering, USC. University Of Southern California, Los Angeles. 26 Oct. 2005.

References

Boehm, Barry. “Unifying Software Engineering and Systems Engineering.” IEEE Computer, 33 no. 3: 114-116.

Boehm, Barry, and Jo Anne Lane. “Using the Incremental Commitment Model to Integrate System Acquisition, Systems Engineering, and Software Engineering.” CrossTalk: The Journal of Defense Software Engineering. Oct. 2007: 4-9.

Department of Defense. Office of the Under Secretary of Defense For Acquisition and Technology. Report of the Defense Science Board Task Force on Defense Software. Washington, D.C., 2000.

FY03 Defense Authorization Act. Public Law 107-314. 2 Dec. 2002. STAT. 116. 2465. Sec. 804.

Hantos, Peter. “Risk Management in System of Systems Development - Are You Ready?” 20th International Forum on COCOMO and Cost Modeling. Center for Software Engineering, USC. University Of Southern California, Los Angeles. 26 Oct. 2005.

2 Software Metrics

2.0 Overview

Metrics are a critical risk management mechanism for successful acquisition of naval systems and platforms. Metrics provide management visibility into both the software acquisition process and the software development process by providing insight about the progress, quality, and expected completion of a software development effort. Metrics can be considered in categories: lagging metrics (which confirm lessons-learned but do not directly forecast ahead, except to the next build or program); in-process metrics (which may reveal current status but may be hypersensitive to small disturbances) and leading metrics (which attempt to forecast, but can also be hypersensitive unless cross-referenced to other metrics and measurements).

Software metrics contribute either directly or indirectly, singly or in combination, to risk management in the following areas:

- Software acquisition planning;
- Requirements development and management;
- Program office and developer staff planning and resource management; and
- Project management oversight.

Software measurement products provide the opportunity to manage risk and activity up, down, and across organizational lines. The use of common core metrics and analysis and reporting processes (e.g., Probability of Program Success (PoPS)) affords senior management insight into potential problem areas throughout the acquisition timeline, ideally with sufficient warning to recognize early enough to mitigate program impacts. The metrics also facilitate communication using a common, objective, and quantifiable vocabulary across the acquirer, developer, and supply chain organizations, allowing managers in any domain to monitor and affect activities necessary for program and sub-program delivery. Most importantly, regular, well understood metrics and measurement activities contribute to effective management down the chain of command, connecting managers, engineers, and suppliers through well thought out, defined, and clearly understood objectives and accountability.

To be effective, metrics should clearly portray variances between planned and actual performance, present a clear view of trends over time, provide prediction or early detection of situations that require management attention, and support the assessment of the impact of proposed changes on the program. Metrics need to be nearly continuous, readily available, independently verifiable, and entwined in risk management activities. Four mandatory core metrics (described in Sections 2.1 and 2.2) serve as the basis of management display of program risk (including both program office and contractor(s) performance) and will be reported at major and milestone reviews and applicable Systems Engineering Technical Reviews (SETR), Weapons Systems Explosive Safety Review Board (WSESRB), and Technical Warrant reviews.

All Programs of Record (PORs) engaged in the acquisition and development and/or integration of software items, regardless of Acquisition Category (ACAT), must define, develop, collect, and report a set of four core metrics, specific to their program, as directed by Assistant Secretary of the Navy (Research, Development and Acquisition) (ASN(RD&A)) policy (see Appendix D). Current Department of Defense (DoD) and Department of the Navy (DoN) guidance interprets the terms "software intensive system" or "systems of systems" to mean nearly all systems that have any software content; the four core metrics are required by all. These metrics should be described in the Request for Proposal (RFP) and Software Development Plan (SDP), baselined, then collected and analyzed across the full system acquisition lifecycle, and reported by each POR upward through the Program Executive Offices (PEOs) to ASN(RD&A) and/or other designated Milestone Decision Authority (MDA), and WSESRB, as directed by the policy.

This chapter defines the four core software metrics and provides guidelines for the factors to be measured, discusses collection and analysis methods, identifies when the associated measurements are to be taken across the acquisition lifecycle, and provides insight into using the metrics to support program goals. The metrics are applicable across the lifecycle of the program and to both the acquirer (e.g., government) and the developer (most likely a contractor or contractor team).

The centerpiece of developing the core metrics is the definition, preparation and implementation of the artifacts necessary to feed the scoring of the weighted algorithms that will change as the core metrics transition from one phase of acquisition development to the next. Figure 2-1 illustrates how the flow of information from related acquirer products (e.g., a software-focused Work Breakdown Structure (WBS), a software-infused SETR process, and other DoD 5000.2 products but with a software emphasis) provides base artifacts to support the core software metrics program. The discussion of each core metric includes a table of the potential artifacts to be used in mining the data and information necessary to determine program health and status.

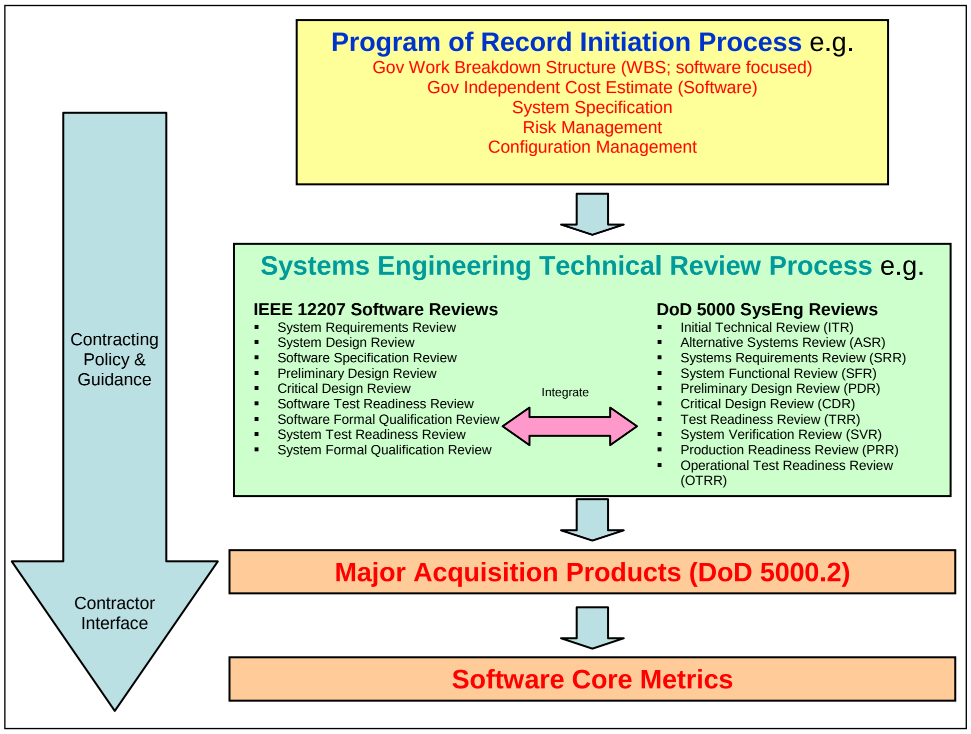

Program of Record Initiation Process e.g.
Gov Work Breakdown Structure (WBS; software focused)
Gov Independent Cost Estimate (Software)
System Specification
Risk Management
Configuration Management
Systems Engineering Technical Review Process e.g.
IEEE 12207 Software Reviews
System Requirements Review
System Design Review
Software Specification Review
Preliminary Design Review
Critical Design Review
Software Test Readiness Review
Software Formal Qualification Review
System Test Readiness Review
System Formal Qualification Review
Integrate
DoD 5000 SysEng Reviews
Initial Technical Review (ITR)
Alternative Systems Review (ASR)
Systems Requirements Review (SRR)
System Functional Review (SFR)
Preliminary Design Review (PDR)
Critical Design Review (CDR)
Test Readiness Review (TRR)
System Verification Review (SVR)
Production Readiness Review (PRR)
Operational Test Readiness Review (OTRR)
Major Acquisition Products (DoD 5000.2)
Software Core Metrics
Contracting Policy & Guidance
Contractor Interface

Figure 2-1. Information Flow to Acquisition Products/Base Artifacts

A key milestone in being able to obtain the fidelity of data necessary to score these metrics lies in a fundamental change of culture in development of the government WBS. Software must be given clear visibility in the WBS and functionally allocated to the detail necessary to be able to manage the risk associated with the software development and integration activities. This should be a part of or match the SDP. That, in turn, will provide the systems engineering visibility for software as early as concept development to determine critical system components and develop risk reduction strategies based on technology maturity issues. The more rigorous treatment of software maturity in the SETR process will in turn provide more accurate and stable information for the various major acquisition products required in the Secretary of the Navy Note (SECNAVNOTE) 5000[1] gate review process leading up to milestone decision points.

2.1 Core Metrics

The four required core metrics are:

- Software Size/Stability;
- Software Cost/Schedule;
- Software Quality; and
- Software Organization.

These metrics are to be provided during key phases of the system acquisition lifecycle, as identified in Table 2-1 below. Column 1 is keyed to additional detail (e.g., artifacts, who collects, etc.) in Sections 2.2.1 through 2.2.4.

ID	Phase	Milestone-Related Period
I	Concept Development	Pre-Concept Decision (CD)
II	Concept Refinement	Post-CD, Leading to Milestone (MS)-A
III	Technology Development	Post MS-A, Leading to MS-B
IV	System Development and Demonstration (SDD) (System Integration)	Post MS-B, Leading to Design Readiness Review (DRR)
V	SDD (System Demonstration)	Post DRR, Leading to MS-C
VI	Production and Deployment	Post MS-C, Leading to Full Rate Production (FRP) Decision
VII	Operations and Support	Post FRP Decision Review

Table 2-1. Key Phases for Metrics Reporting

Each program office should require their software contractors to report metrics in the same manner and format that they will be reported to ASN(RD&A) by the program office, and hold available the data or products measured to support independent verification. Metrics requirements placed on the prime contractor should also be extended to subcontractors performing software development and/or integration. The prime contractor should require their subcontractors to provide them with either the same metrics or the measures necessary to derive them.

[1] Department of the Navy. DASN(RD&A) ALM. SECNAVNOTE 5000, Department of the Navy (DoN) Requirements and Acquisition Process Improvements. 26 February 2008.

In order to be both feasible and meaningful, the metrics are defined to be flexible enough to accommodate different program infrastructures, lifecycles, and products, while at the same time are complete and comprehensive enough to provide objective insight for upper management. In support of these goals, the metrics are artifact based. As noted in ASN(RD&A) policy (see Appendix D), the core metrics should be tailored and implemented consistent with both the program office's and the developer's internal tools and processes. Program offices and developers should agree upon and establish additional metrics or means of insight to address software issues deemed critical or unique to the program, such as software safety requirements. Although flexibility is provided, it is expected that once the metrics are baselined, the program will maintain a rigorous commitment to collecting the same metrics the same way, so that they are statistically stable.

2.2 Specific Guidance for Each of the Core Metrics

The following subsections define a standard set of criteria for the implementation of each of the four core metrics, via a tabular format that spans the seven lifecycle phases identified in Table 2-1. Note that as performance metrics, they can not be collected until performance begins. Therefore, during the pre-development phases, metrics reporting requires an "alternative practice." For example, in the post Milestone (MS)–A period, the metric should be realized via specific RFP language (including proposal evaluation criteria) that requires the contractor to address in their proposal their plan for reporting the four core metrics throughout the lifecycle (or period of performance), together with a preliminary baseline estimate.

2.2.1 Software Size/Stability

Software Size/Stability is a performance metric that covers both software development (primarily new code) and software integration (developed and/or reused/Commercial Off-The-Shelf (COTS) software). Software size is an aspect of the metric that must be baselined with initial measures, followed by continuing and consistent measures of size. Software stability is an aspect of the metric that compares subsequent measures of size to the baseline measures.

The Software Size/Stability metric provides both the POR management and ASN(RD&A) (or other MDA, as designated) with adequate and appropriate levels of visibility into software size estimates, current size status, stability (including planned vs. actual size), and projected growth. The objective of measuring size is to identify potential risk areas affecting staffing, schedule, cost, and resources. Table 2-2 provides the criteria for the implementation of the Software Size/Stability metric.

Phase	I	II	III	IV	V	VI	VII
Baseline/ Basis of Metric	Concept expectation of %-age of system functionality to be delivered by SW (vice, e.g., HW)	Concept expectation of %-age of system functionality to be delivered by SW (vice, e.g., HW)	SW Size Estimates	SW Size Baseline	SW Stability	SW Stability	SW Stability
Who Collects Measure-ments	Program Office	Program Office	Program Office/Bidders	SW developer/ integrator	SW developer/ integrator	SW developer/ integrator	Program Office/ SW developer/ integrator
Who Analyzes	Program Office	Program Office	Program Office	Program Office/ SW developer/ integrator	SW developer/ integrator	SW developer/ integrator	Program Office
Metric	%-age of functionality in SW	%-age of functionality in SW	Estimated SLOC, FP, or Req'ts.	ESLOC, FP, or Req'ts.	ESLOC, FP, or Req'ts.	ESLOC, FP, or Req'ts.	ESLOC, FP, or Req'ts.
Related Artifacts	AoA Plan, FAA, FNA, FSA,ICD, Study Contracts, Integrated Architecture	Acq. Plan, AoA, Draft CDD, Preliminary System Specification, Cost/ Manpower Estimate, Draft RFP, SEP, T&E Strategy	ICE, CARD, CCA, POE, RFP, Proposals, Technology Development Contract, Initial Product Support Strategy, TEMP, PPP, TRA, ISP	SDD Contract, SDP (WBS), Work Packages, Timesheets, Task Mgmt Plans, Product Support Plan	SDD Contract, SDP (WBS), Work Packages, Timesheets, Task Mgmt Plans, Product Support Plan	SDD Contract, SDP (WBS), Work Packages, Timesheets, Task Mgmt Plans, Product Support Plan	SDD Contract, SDP (WBS), Work Packages, Timesheets, Task Mgmt Plans, Product Support Plan
Use of Metrics	Risk, Lessons Learned	Risk, Lessons Learned, Concept Selection	Risk, Lessons Learned, Source Selection	Risk, Lessons Learned, Performance	Risk, Lessons Learned, Performance	Risk, Lessons, Learned, Performance	Risk, Performance, Lessons Learned, Database/ Archival

Table 2-2. Software Size/Stability Metric

Software size estimates should be produced by the program office as early as the Concept Refinement phase, as part of the development of the Analysis of Alternatives (AOA). A point of reference (although not yet a firm baseline) can and should be established in the acquisition plans, prior to MS-A. Although software size estimates at this point will not provide management insight into performance, they will support the generation of adequate contract language and criteria for evaluating cost/schedule proposals and the gates reviews.

As Key Performance Parameters (KPPs) and other requirements are added or derived during Technology Development to support MS-B (from the Initial Capabilities Document (ICD) to the Capability Development Document (CDD)), and as growth and stability are tracked against baselines during System Development and Demonstration (SDD) to support MS-C (from the CDD to the Capability Production Document (CPD)), changes in software size should be measured, analyzed, reported, and acted upon. These phases (Technology Development and SDD) are the most important because they are the performance phases when actual software size is changing (i.e., during software development and integration).

During the Production and Deployment and the Operations and Support phases of the lifecycle, the Software Size/Stability metric should also be monitored, via authorized Software Change Reports (SCRs) and executed Engineering Change Proposals (ECPs), to be used for planning and management activities.

In order to adequately support generation of the selected metrics, the supporting base measures need to be collected on an event-driven basis as well as monthly. Some metrics will require a rate of software growth (necessitating the periodic measurements), while some metrics will require logging the final size of a developed software unit or component (e.g., for earned value and for archival in software size databases/repositories). The event-driven measures are therefore collected whenever a software component or unit has finished its code and unit testing efforts.

All software size and stability measures that are collected and/or delivered during the acquisition, development, upgrade, and/or maintenance of the system should be preserved and archived to strengthen future efforts. This archive should include original estimates, initial planned values, actual values, context data (e.g., type and complexity of the code), reported variances, and corrective action outcomes. The archival mechanism must allow quick and easy access to the size and stability measures and their supporting contextual information.

2.2.1.1 Baseline

Both the initial measures and the expected growth trend should be baselined in order to accurately assess software size and stability and to use the metrics effectively for monitoring and forecasting performance. During the lifecycle, current actual measures of software size should be charted against the baseline values on the control chart at periodic intervals and at event-driven milestones, in order to support performance monitoring and forecasting. The expected growth over time should be used to baseline software stability. For expected linear growth, the slope of the line measuring size against time (including both the mean and the upper and lower control limits on the control chart) should provide a graphical definition of the baseline.

In general terms, the baseline includes:

- Original estimates of software size (which could usefully be plotted as the y-intercept on a timeline control chart);
- Planned values for software size vs. time (for which, if plotted, the slope provides the projected growth curve); this may not be constant (i.e., the expected growth may not be linear);
- Agreed upon bias (the “comfort zone” defined by the upper and lower control limits surrounding the expected growth curve); and
- Expected variances and tolerable thresholds.

Whatever is measured to define the Software Size baseline should continue to be measured and used in the same manner across the lifecycle. Software Stability should likewise be generated and analyzed consistently to provide meaningful comparisons. Therefore, it is important to carefully deliberate over what these baselines will be, and to establish them precisely. As just one example, it is important to consider the percentage of developed code vs. the percentage of COTS. Establish this ratio as part of the estimating process, and preserve it as context information for the baseline and any future rebaseline efforts.

Estimated values for different measures should be updated at significant program milestones when additional information is available upon which the updates may be based, such as completion of software requirements analysis, change/deletion/addition of software requirements or systems requirements impacting software, completion of software design, completion of software coding, completion of a test phase, completion of a software release or other appropriate software development milestones. The Integrated Product Team or other designated agent should determine whether the margin of deviation of actual measurements from estimates is acceptable, considering associated impact on cost and schedule.

It is important to rigorously control configuration of the baselines and the associated metrics. The program office should have an adequately robust configuration management process to ensure such control. Likewise, contract language should require the contractor to execute similar control. Baselines should only be changed when sufficient analysis has been performed and all relevant stakeholder input has been deliberated by an approved configuration control board.

2.2.1.2 Measures

Detailed performance specifications should be generated, keyed to clearly identified requirements from the Joint Capabilities Integration and Development System (JCIDS) products (ICD, CDD and CPD), as well as derived requirements linked to those documents. As technology development progresses and the system requirements are clarified and allocated (e.g., to hardware and software configuration items), a WBS that clearly separates and identifies the software components of system development (traceable to system requirements) will begin to emerge. The WBS should be refined during system development, and it is imperative that it be expanded to encompass measurable units and components of software code. Senior management typically reviews and addresses a WBS to only several levels of indenture. However, effective generation of Software Size/Stability metrics will depend on the exploitation of a WBS that is granular enough to accurately identify and predict software efforts (i.e., a WBS that is functionally decomposed to a measurable software unit level). Ultimately, the SDD contract(s) involving software development and integration should establish and use such a WBS within their Software Development Plan(s). All contract modifications should be keyed to the WBS as they affect software configuration items.[2]

Once a software-separated and traceable WBS is established, it is possible to define specific and meaningful measures that support the Software Size/Stability metric. The metric will be meaningful because the software components to be measured are tied, one-to-one, to WBS elements that are granular enough to be estimated, monitored, and analyzed for earned value. The base measures described below are significant to the Software Size/Stability metric. Some will be required and some will be optional, depending on the selected indicators (see Section 2.2.1.3). Acquirer and developer agreement on all details of the software size/stability metric should be agreed early on in the contracting and development processes.

2.2.1.2.1 Lines of Code Count

Source Lines of Code (SLOC) can be defined in many ways. There must be agreement on the counting methods and rules used to determine total lines of code. The typical measure collected by the software developer is Logical SLOC, measured in units of either SLOC (individual lines of code) or KSLOC

[2] Detailed guidance on generating an adequately decomposed WBS can be found in both Section 2.6 of the NAVAIR EVM Toolkit (DoN NAVAIR. Using Software Metrics and Measurements for Earned Value Toolkit. 2 December 2004.) and Section 4 (specifically Fig. 4.5) of the PSM DoD Implementation Guide ("Measurement for DoD Projects", DoD Implementation Guidance, Practical Software and Systems Measurement, 24 February 2003.)

(thousands of lines of code). Logical SLOC includes executable lines of code and declarations, but does not include comments or blank lines (hereafter the implied word "Logical" will be dropped and SLOC will be used even when KSLOC is the probable unit of measure). It is easy to measure SLOC for any component of developed code; however, a meaningful interpretation of the measure usually requires that all the code is new code, is written in the same language, and is of the same level of complexity.

When there is modified code and/or reused COTS mixed in with new code, then raw SLOC counts can lose meaning. In such situations, a better measure is Equivalent SLOC (ESLOC), where all the various types of SLOC are normalized to the effort of a new SLOC. In this manner, changes in ESLOC will be a better indicator of performance. Alternatively, if the normalization effort is too difficult, then the categories of code (new, modified, and reused) can be tracked and monitored separately, but they can not be compared, nor can they be used in combination for roll-up metrics reporting. In fact, the NAVAIR Earned Value Management (EVM) Toolkit[3] goes so far as to say that the different levels of effort required to implement and integrate each type of software – new, reuse, modified, deleted, automatically generated, ported, and COTS – requires that they all be tracked separately. The Toolkit goes on to provide EVM tips for each type. Not all sources agree. The International Function Point Users Group (IFPUG) states, "The main drawback in the use of ESLOC is that while weighting factors can readily be applied to estimated SLOC, once actual code is produced, it is nearly impossible (without elaborate tagging or bookkeeping methods) to determine the code type composition of the code and therefore where to apply the factors. Many organizations overcome this weakness by continually computing current estimated ESLOC throughout all of the development life cycle."[4]

It is also difficult to employ SLOC as a meaningful measure when various components of code have different levels of complexity (as determined, for example, by applying the McCabe Cyclomatic complexity test). Again, a normalization process must be applied in order to weight the different complexities to yield a meaningful ESLOC metric.

Because SLOC estimates are based on planned functionality as defined by requirements, if all the planned functionality is not implemented, then earned value based on the estimated SLOC might be overstated.[5] That is why it is important to have an adequately granular WBS, which ties functionality to modules of code via precisely and unambiguously defined WBS elements.

2.2.1.2.2 Software Requirements Count

The number of requirements allocated to software is a useful measure only when the refinement of capabilities first to system requirements and then to software allocation has been executed carefully and thoroughly, so that the estimated number of requirements is accurate and the decomposition is clean. For example, the Institute of Electrical & Electronics Engineers (IEEE) Standard 830-1998, *IEEE Recommended Practice for Software Requirements Specifications* provides a list of nine quality attributes for requirements, attention to which should lead to a set of software requirements that is adequately representative of the work breakdown:

- Complete;

[3] Department of the Navy. Naval Air Systems Command (NAVAIR). Using Software Metrics and Measurements for Earned Value Toolkit. 2 December 2004.
[4] International Function Point User Group. Guidelines to Software Measurement, Release 2. International Function Point User Group: Westerville OH, 2004; http://www.ifpug.org.
[5] NAVAIR. Using Software Metrics and Measurements for Earned Value Toolkit.

- Unambiguous;
- Correct;
- Consistent;
- Verifiable;
- Modifiable;
- Traceable;
- Ranked for Importance; and
- Ranked for Stability.

Software requirements provides a very useful measure when applied to new code being developed, but it must somehow be factored/weighted if used to measure modified or reused code, especially when mixed with new code. This is easier said than done, because for new code, requirements typically indicate operations and functions that must be coded, while for modified and reuse code, requirements, to be used as a performance measure, must refer to software porting efforts and interface issue resolutions rather than to functions.

Even if all the code is new, another consideration is that some software requirements may entail a great effort for design, code, test, debug, integration, and documentation while others may require considerably less effort. When using requirements count as a measure, it is assumed that the sizes of the applicable software components to be measured are large enough that these differences will average out in a typical aggregate application of the metric.

While both SLOC and software requirements count can be used, each serves useful and complementary purposes. Factors which may influence selection in specific cases include:

- If the predominant variation between software components is new vs. modified vs. reuse (rather than variations in complexity), then the normalized ESLOC measure may prove more useful than requirements, because methods for defining "equivalent" lines of code may be easier than for defining "equivalent" requirements (as discussed above).
- If the predominant variations between software components are due to complexity, environmental factors, and/or programming language differences, then even if all the code is new code, it may make more sense to measure the software requirements (counts and trends) instead of ESLOC.
- SLOC is generally useful as a measure only during the code and unit test phases, while effective requirements counts have broader coverage. Requirements are more fundamental also, as SLOC estimates and counts are based on planned functionality as defined by requirements. For these reasons, it may be more appropriate to monitor and track the number of requirements allocated to software.
- Requirements are an excellent measure for use in determining earned value measures since they are directly related to evaluating progress in implementing the functionality required by the system. On the other hand, SLOC, as an EVM measure, is poor. For further discussion, see the NAVAIR EVM Toolkit.[6]

[6] NAVAIR. Using Software Metrics and Measurements for Earned Value Toolkit.

2.2.1.2.3 Function Point Count

Function Points (FPs) provide an alternative method of measuring software size, based on what the system does. As the system's functionality increases, the number of FPs increases. FPs measure software size by quantifying functionality from the user's point of view, based solely on logical design and functional specifications. The size of a software item in FPs can therefore be determined early in its life-cycle, and hence can be used to help estimate the effort and time to develop the software.

Function Points are directly derived from software requirements using a rigorously defined set of counting rules. The International Function Point Users Group FP counting rules are the most widely recognized standard.[7] However, there are several other variations and derivatives for counting FPs.

Where different requirements may take different amounts of effort to implement, each FP should take the same amount of effort to implement. This assumes that "adjusted" FPs are used, which take into account the software complexity. This same-effort per FP attribute simplifies determining earned value in comparison to requirements, but it makes the FP analysis more complicated. It is notable that measuring FPs requires expertise in applying the function point counting rules rather than the domain expertise necessary to do an accurate SLOC estimate.

When selecting FPs as a measure, it is important to consider the following:

- At least one team member must be a Certified Function Point Specialist (CFPS) to accurately account for the number of FPs for each task associated with each requirement.
- FP counts are best performed on well defined software requirements specified at the level of detail found in a Software Requirements Specification (SRS).
- FP counts must be continually updated to reflect changes in requirements. While FPs can be applicable to all phases of software development, there may be specific tasks in each phase that are not well suited to earned value allocation.
- Since FPs are applicable to most development phases, they are likely to be more useful than SLOC for earned value purposes. Nevertheless, FPs remain inferior to requirements for tracking earned value.
- Detailed software requirements are required to accurately count either function points or SLOC.[8]

To assist in selecting program-specific measures, a very detailed set of benefits and drawbacks for SLOC count and Function Point has been assembled by the International Function Point Users Group.[9] That document can be downloaded free by IFPUG members from the website. Non-members can order a copy at the website.

2.2.1.3 Indicators

The base measures are combined and analyzed to produce the following indicators:

- Size

[7] IFPUG. Guidelines to Software Measurement, Release 2.
[8] NAVAIR. Using Software Metrics and Measurements for Earned Value Toolkit.
[9] The IFPUG website is at: **http://www.ifpug.org**.

  - ESLOC:
    - Amount and percentage of new, modified, and reuse/COTS code (derived ESLOC)
    - Estimated ESLOC; collected prior to development
    - Planned vs. actual ESLOC; collected during applicable development phases:
      - Designed
      - Coded
      - Tested and complete
  - Requirements:
    - Estimated number of software requirements; collected prior to development
    - Planned vs. actual software requirements; collected during applicable development phases:
      - Designed
      - Coded
      - Tested and complete
  - FP:
    - Estimated FPs; collected prior to development
    - Planned vs. actual FPs; collected during applicable development phases:
      - Designed
      - Coded
      - Tested and complete
- Stability
  - Size estimation accuracy
  - Size stability
    - E.g., a time phased graph of the change in the estimated size of the software over time and at various program milestones
  - Requirements volatility; number and percentage of total requirements for each:
    - New requirements
    - Modified requirements
    - Deleted requirements
  - Requirements change rate
    - The number of software requirements at the beginning of a time period, divided by the number of new, modified or deleted requirements at the end of the time period.

Note that there are fundamental differences between the information needs for COTS and developmental software. For example, for COTS, it is functional performance and not design criteria that must be monitored during development.

The program management team must evaluate the alternative indicators in order to select those that are relevant and most valuable to the program, as follows:

- Establish the program-specific criteria for evaluating alternative indicator choices (e.g., ownership and lifecycle cost of implementing the metric);
- Rank the criteria so that the highest ranked exert the most influence (document the results for stakeholder buy-in);
- Analyze how the possible indicators options would be refined to fit the program (e.g., how well they would apply to the program infrastructure and to the type and extent of software to be developed and/or integrated);
- Analyze how the possible indicators options would be tailored to fit the program (e.g., consider the program environment and various levels of software and their interfaces);
- Evaluate the alternative indicator choices (apply the ranked criteria to the analyzed indicator choices); and
- Select specific indicators to be baselined and used (assess any risks associated with the implementation of selected indicators).[10]

2.2.2 Software Cost/Schedule

It is highly recommended that EVM techniques be used for all software development projects. The benefits of software cost and schedule tracking using Earned Value Management and associated metrics that support it are real and widely acknowledged. EVM can be applied equally to any program, large or small. It provides clear and objective insight into the value produced within a prescribed time and cost constraint. Consistent application of EVM in the software development project(s) will allow the program manager to identify the performance of the software development team and thereby forecast future cost and schedule. Ultimately the purpose of the EVM approach is to measure key performance indicators of the software development process in order to understand and control it. EVM will not guarantee software project success or fix latent problems, it will however identify project status and accurately predict future performance and outcomes. (See Appendix F for additional information on EVM.)

NOTE: There are many ways to implement EVM, but EVM is required: “Agencies must use a performance based acquisition management system, based on American National Standards Institute/Electronic Industries Alliance ANSI/EIA Standard 748, to measure achievement of the cost, schedule, and performance goals.”[11]

Producing software cost and schedule earned value metrics requires collecting estimated and actual cost and schedule data and the earned value associated with the product or component being produced. The following are working definitions for this data:

- Actual Cost (AC). Total costs actually incurred and recorded in accomplishing work performed during a given time period for a schedule activity or work breakdown structure component. Actual cost

[10] Software Engineering Institute. CMMI® for Acquisition, Version 1.2 (Technical Report CMU/SEI-2007-TR-017). Carnegie Mellon University, 2007.
[11] Executive Office of the President. Office of Management and Budget. Circular A-11, Part 7. July 2007.

can sometimes be direct labor hours alone, direct costs alone, or all costs including indirect costs. It is also referred to as the Actual Cost of Work Performed (ACWP).

- Control Account (CA). A management control point where the integration of scope, budget, actual cost, and schedule takes place, and where the measurement of performance will occur. Control accounts are placed at selected management points (specific components at selected levels) of the work breakdown structure. Each control account may include one or more work packages, but each work package may be associated with only one control account. Each control account is associated with a specific single organizational component in the organizational breakdown structure
- Duration. The total number of work periods (not including holidays or other nonworking periods) required to complete a schedule activity or work breakdown structure component. It is usually expressed as workdays or workweeks. Sometimes incorrectly equated with elapsed time.
- Earned Value (EV). The value of completed work expressed in terms of the approved budget assigned to that work for a schedule activity or work breakdown structure component. It is also referred to as the Budgeted Cost of Work Performed (BCWP).
- Earned Value Management (EVM). A management methodology for integrating scope, schedule, and resources, and for objectively measuring program performance and progress. Performance is measured by determining the budgeted cost of work performed (i.e., earned value) and comparing it to the actual cost of work performed (i.e., actual cost). Progress is measured by comparing the earned value to the planned value
- Earned Value Technique (EVT). A specific technique for measuring the performance of work for a work breakdown structure component, control account, or project. It is also referred to as the earning rules and crediting method.
- Effort. The number of labor units required to complete a schedule activity or work breakdown structure component. It is usually expressed as staff hours, staff days, or staff weeks.
- Performance Period. Typically the software data collected for the cost and schedule metric are bounded by discrete beginning and ending period. The data collection performance period may match the status reporting period, but that is up to the program manager to decide. The performance period can be as small as an eight hour shift or as large as monthly; again this is up to the program manager and stakeholders to decide.
- Planned Value (PV). This data may be considered the quantitative assessment or estimated cost of a software product or service identified in the project WBS. It is the authorized budget assigned to the scheduled work to be accomplished for an activity or work breakdown structure component. It may also be referred to as the Budgeted Cost of Work Scheduled (BCWS). It is also the monetary value of a software activity or component that includes the monetary worth of the resources required to perform and complete the activity or component, or to produce the component. A specific cost can be composed of a combination of cost components including direct labor hours, other direct costs, indirect labor hours, other indirect costs, and purchased price. In the earned value management methodology term cost can represent only labor hours without conversion to monetary worth. In most cases where earned value metrics are used to track software development the non-labor related costs are segregated from labor costs. This is done because including material cost in the earned

value calculation distorts the cost variance and index, and the reliability of the Earned Value Estimated At Completion metric.[12]

- Work Package. A deliverable or project work component at the lowest level of each branch of the work breakdown structure. The work package includes the schedule activities and schedule milestones required to complete the work package deliverable or project work component.[13]

Table 2-3 provides the criteria for the implementation of the Software Cost/Schedule metric.

Phase	I	II	III	IV	V	VI	VII
Baseline/ Basis of Metric	SW related IERs, SDXs	SW related IERs, SDXs	Actual SW cost & schedule data	Actual SW cost & schedule data	Actual SW cost & schedule data	Actual SW cost & schedule data	Actual SW cost & schedule data
Who Collects Measure-ments	Sponsors & Advocates	Sponsors & Advocates	Program Office/SW developer/ integrator	Program Office/SW developer/ integrator	Program Office/SW developer/ integrator	Program Office/SW developer/ integrator	Program Office/ SW developer/ integrator
Who Analyzes	Sponsors & Advocates	Sponsors & Advocates	Program Office	Program Office	Program Office	Program Office	Program Office
Metric	# IERs/SDXs produced by SW	# IERs/SDXs produced by SW	Cost/Schedule Variance/ Performance index	Cost/Schedule Variance/ Performance index	Cost/ Schedule Variance/ Performance index	Cost/ Schedule Variance/ Performance index	Cost/ Schedule Variance/ Performance index
Related Artifacts	AoA Plan, FAA, FNA, FSA,ICD, Study Contracts, Integrated Architecture	Acq. Plan, AoA, Draft CDD, Preliminary System Specification, Cost/ Manpower Estimate, Draft RFP, SEP, T&E Strategy	ICE, CARD, CCA, POE, RFP, Proposals, Technology Development Contract, Initial Product Support Strategy, TEMP, PPP, TRA	SDD Contract, SDP (WBS), Work Packages, Timesheets, Task Mgmt Plans, Product Support Plan	SDD Contract, SDP (WBS), Work Packages, Timesheets, Task Mgmt Plans, Product Support Plan	SDD Contract, SDP (WBS), Work Packages, Timesheets, Task Mgmt Plans, Product Support Plan	SDD Contract, SDP (WBS), Work Packages, Timesheets, Task Mgmt Plans, Product Support Plan
Use of Metrics	Risk, Lessons Learned	Risk, Lessons Learned	Risk, Lessons Learned	Risk, Performance, Lessons Learned	Risk, Performance, Lessons Learned	Risk, Performance, Lessons Learned	Risk, Performance Lessons Learned

Table 2-3. Software Cost/Schedule Metric

2.2.2.1 Baseline

The performance measurement baseline for software development projects should be based on a WBS that is compliant with DoD and industry standards and best practices. These include the latest versions of the following:

- DoD Handbook – Work Breakdown Structure, MIL-HDBK-881;
- The Guide to the Project Management Body of Knowledge (PMBOK), ANSI/PMI 99-001;

[12] Practice Standard for Work Breakdown Structures, Second Edition. Sylva: Project Management Institute, October 2006.
[13] PMBOK® Guides. A Guide to the Project Management Body of Knowledge, Third Edition (ANSI/PMI 99-001-2004). Newtown Square, PA: Project Management Institute, 2004.

- Government Extension to the PMBOK third Edition (PMI Global Standard); and
- Practice Standard for Work Breakdown Structures (PMI Global Standard).

2.2.2.2 Measures

Normally, a predefined performance period is used to bound the collection and reporting of software cost and schedule. The performance period can be whatever time span the program manager wants.

- Planned data is available from the project WBS work packages for the performance period. The work package should provide the following data:
  - Work package name; and
  - Planned value of the work to be produced.
- Actual data to be collected includes:
  - Name of task performer;
  - Name of work package;
  - Actual hours effort worked by work package; and
  - Completion status of each work package worked.

The actual data may be collected automatically using a timesheet collection process or system or it may require a separate manual process. The completion status of the work package may be expressed as percent complete. Because of the difficulty in determining the actual percent complete in software development efforts it is recommended that work packages be kept small (approximately 80 hours or less) and that completion status is limited to 0%, 50% or 100%.

2.2.2.3 Indicators

The standard indicators for software cost and schedule using the EVM approach are cost and schedule variance and index using the following algorithms:

- Cost variance (CV). Formula: CV= EV – AC. CV equals earned value (EV) minus actual cost (AC).
- Schedule variance (SV). Formula: SV = EV – PV. SV equals earned value (EV) minus planned value (PV). Schedule variance will ultimately equal zero when the project is completed because all of the planned values will have been earned.
- Cost performance index (CPI). Formula: CPI = EV/AC. CPI equals the ratio of the EV to the AC. A CPI value less than 1.0 indicates a cost overrun of the estimates. A CPI value greater than 1.0 indicates a cost underrun of the estimates. The CPI is the most commonly used cost-efficiency indicator.
- Schedule performance index (SPI). Formula: SPI = EV/PV. SPI equals the ratio of the EV to the PV. The SPI is used in addition to the schedule status to predict the completion date and is sometimes used in conjunction with the CPI to forecast the project completion estimates.

These basic formulas can be applied at any level of interest within the program from tracking individual performance to cumulative performance data for the entire program. The number and levels of EVM reports created is up to the program manager, depending on the level at which the data is collected. From these basic indicators other data can be extrapolated that will aid in program control. Figure 2-2 provides one example of how Earned Value data can be charted to illustrate program status and health.

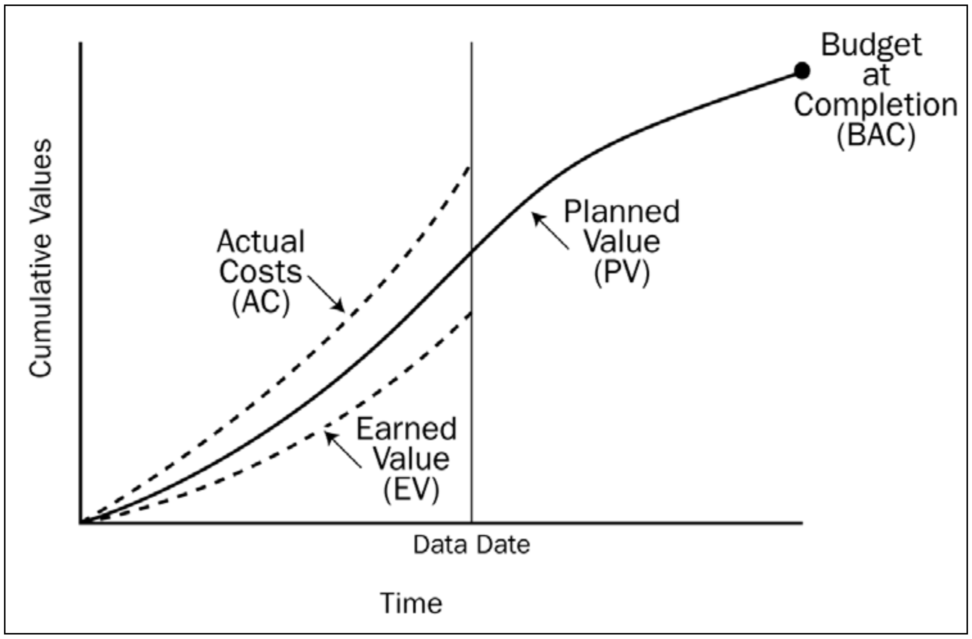


Figure 2-2. Program Status Demonstrated Using EVM

2.2.3 Software Quality

Quality is the degree to which a product or service meets the expectations of a customer. This definition can be interpreted more than one way when it is applied to a DoN acquisition program with a significant software component. Quality metrics are both risk and performance related. Tracking quality metrics can be helpful in revealing the realization of known risks. For instance, increasing scope can be measured by the number of requirement change requests, or the number of software defects can be measured by the number of software trouble reports. Both of these metrics indicate quality problems that also represent program risk. At the same time many of the quality metrics are also indicators of performance. Using quality metrics with earned value techniques can produce performance indicators that are effective in forecasting future performance. Metrics for software development should focus on two distinct aspects of quality: Quality Assurance and Quality Control.

- Quality Assurance (QA) is the application of planned, systematic quality activities to ensure that the program will employ all processes needed to meet requirements. A quality assurance department, or similar organization, often oversees quality assurance activities. QA support may be provided to the program team, the management of the performing organization, the customer or sponsor, as well as other stakeholders not actively involved in the work of the program. The intent of QA is to prevent defects from occurring by enforcing and continuously improving standard processes. This is most often achieved when the processes selected are appropriate for the task and tailored, and when the process standards are known, understood and accepted by the development team. Finally, quality audits should be performed to ensure that the selected processes are in place, operational and being followed.
- Quality Control (QC) involves monitoring software development work products and results to determine whether they comply with relevant requirements and quality standards. It also identifies ways to eliminate causes of unsatisfactory results. QC should be performed throughout the program. QC may be performed by a quality control department or team, and can include taking action to

eliminate causes of unsatisfactory performance. The intent of QC is to capture as many software quality issues as possible before they become part of the software development product set, and most particularly before they reach the customers and end-users. The focus of QC must include all software products, including documentation, training and other non-software items. Finally, identifying and correcting defects must be viewed in a positive light.[14]

The primary purpose of software quality (QA and QC) is to measure and monitor the quality of software development processes and products throughout the project life cycle in order to reduce defects and rework. Quality Assurance and Quality Control provide staff and management with objective insight into processes and associated work products. To achieve this purpose software quality should be included in the project plan, schedule and budget.

The QA and QC team should have sufficient independence and authority to act and take action. They should be able to perform quality audits, peer reviews, on-site reviews and inspections of software development processes and products, and should be able to report findings at all levels of program stakeholders including sponsors, customers and upper management.

Keeping the QA and QC team separate and independent from the software development team is considered more desirable than having inside the software team. Practicing QC may require a working knowledge of statistical quality control, especially sampling and probability, to help evaluate QC outputs. Table 2-4 provides the criteria for the implementation of the Software Quality metric.

[14]PMBOK® Guides. ANSI/PMI 99-001-2004-PMBOK

Phase	I	II	III	IV	V	VI	VII
Baseline/ Basis of Metric	SW related IERS & SDXs	SW related IERS & SDXs	Defects per SLOC	Defects per SLOC, Defects per system interface	Defects per SLOC, Defects per system interface	Defects per SLOC, Defects per system interface	Defects per SLOC, Defects per system interface
Who Collects Measure-ments	Sponsors & Advocates	Sponsors & Advocates	Program Office/ SW developer/ integrator	Program Office/ SW developer/ integrator	Program Office/ SW developer/ integrator	User/Tester	User/Tester
Who Analyzes	Sponsors & Advocates	Sponsors & Advocates	Program Office	Program Office	Program Office	Program Office	Program Office
Metric	% SW generated IERs/SDXs	% SW generated IERs/SDXs	Qty performance index/ variance	Qty performance index/ variance	Qty performance index/ variance	Qty performance index/ variance	Qty performance index/ variance
Related Artifacts	AoA Plan, FAA, FNA, FSA,ICD, Study Contracts, Integrated Architecture	Acq. Plan, AoA, Draft CDD, Preliminary System Specification, Cost/ Manpower Estimate, Draft RFP, SEP, T&E Strategy	ICE, CARD, CCA, POE, RFP, Proposals, Technology Development Contract, Initial Product Support Strategy, TEMP, PPP, TRA	SDD Contract, SDP (WBS), Work Packages, Timesheets, Task Mgmt Plans, Product Support Plan, Defect containment	SDD Contract, SDP (WBS), Work Packages, Timesheets, Task Mgmt Plans, Product Support Plan, T&E, defect containment spreadsheet, Test problem reports	SDD Contract, SDP (WBS), Work Packages, Timesheets, Task Mgmt Plans, Product Support Plan, T&E, test problem reports.	SDD Contract, SDP (WBS), Work Packages, Timesheets, Task Mgmt Plans, Product Support Plan, T&E
Use of Metrics	Risk, Lessons Learned	Risk, Lessons Learned	Risk, Lessons Learned	Risk, Performance, Lessons Learned	Risk, Performance, Lessons Learned	Risk, Performance, Lessons Learned	Risk, Performance, Lessons Learned

Table 2-4. Software Quality Metric

2.2.3.1 Baseline

Defining the baseline for quality software metrics begins early during program initiation with the decisions regarding the software development philosophy and life cycle model. Under the spiral and incremental development methods, it is possible for the System/Subsystem Requirements Review and especially the Software Requirements Review to occur more than once. Such a review could be required for every software spiral, increment or build in the program depending on how the program is structured. Waterfall type development, on the other hand, may occur at the beginning of the development but may also overlap with other phases. Other factors, such as how the program will develop the functional user requirements and how they will be reviewed, approved and maintained will affect the software quality baseline created for the program. These and other factors related to program philosophy, methodology, customer expectations, risks, constraints and assumptions will all influence the quality baseline.

The software project manager must consider what items to put under quality assurance and control and more importantly what baseline values to establish for the comparison of planned and actual quality. The project Work Breakdown Structure should be used as the source for the product quality list. In addition, a defined and quantifiable exit criteria based on the product requirement must be established for both testing and measuring quality. Developing the estimate of the number of defects likely to occur for each quality item must be based upon historical data from previous developments. Also considered is the amount of testing and rework that is expected and acceptable within the context of the project plan and expectations.

The baseline target including upper and lower level ranges established for each category of products must be within acceptable quality levels to meet program and contract requirements. In this instance the baseline for product quality is derived from a combination of historic data and the specific requirements set by the program and contract. Acceptable quality must be defined based upon what level of degradation in the ability of the system to perform mission essential capabilities is acceptable and/or the impact on the development or lifecycle support of the system. The following are guiding principles for determining quality levels for a system:

- No Priority 1 or 2 defects;
- Specify a maximum acceptable sigma level for priority 3 defects for each product under QA/QC. Verify that priority 3 defects are assigned only to those defects which involve human interaction; and
- Specify a maximum acceptable sigma level for priority 4 & 5 defects for each product under QA/QC.

Ultimately the quality baseline must establish the threshold for acceptance of each work product. In other words, what is the exit criterion for declaring that the product meets or exceeds all requirements? Also, it must identify what level of defects during the product development are reasonable compared to previous development efforts and acceptable to the meet the program stakeholder expectations. Finally, the quality baseline must serve as an instrument for evaluating actual performance against baseline standards.

2.2.3.2 Measures

The class of measures associated with software quality is most often defects per product and the number of defects found compared to a predetermined threshold. The majority of software quality defects are found in software requirement specifications and code, although documentation and other related products can also be tested for defects. Depending on the program, non-code products are not as critical to meeting the project requirements and capabilities as is the software code.

Capturing and reporting software product defects should be explicitly included in the project plan and WBS. The type of product tests and reviews and the method for recording and reporting must be defined in the plan. Adhering to the prescribed plan and processes should be under QA control. Defect detection methods may include the following:

- Peer review and minutes showing defects or process changes;
- Structured walk through with defects and corrected processes;
- Formal testing (code, unit, system, interface, interoperability) and tracking of test problem reports, defect categorization, corrected work details; and
- Independent verification and validation with defects and defect densities.

The frequency and the number of iterations required before the software products are expected to be free of defects and of sufficient quality to meet or exceed the exit criteria must be specified by the project plan baseline.

2.2.3.3 Indicators

Quality measurements are not limited to defects found during testing, but also defects found via peer reviews of requirements, design, code, test procedures and other software artifacts. Using the measures identified, the following metrics represent a sample, in order of prevalent use, of what can be produced:

- Priority of defects (industry standard practice, IEEE, International Organization for Standardization (ISO), International Electrotechnical Commission (IEC), etc.);
- Total number of defects;
- Number of defects open over time;
- Number of defects resolved or closed over time;
- Schedule change (additional time) due to test problems;
- Age of defects, or defect containment (that is, not found in next or later phase) and % “leaked” into later phase;
- Defect density (in gross number of errors over total SLOC, better if by configuration item, better still if by use case or module or package or object, but rarer too);
- Rework staff hours;
- Planned versus actual defect count;
- Cost to fix;
- Defect root cause (missed or misunderstood requirement, process violation, tool error, etc.);
- Planned versus actual rework cost and schedule; and
- Planned versus actual defect performance index.

2.2.4 Software Organization

The Software Organization metric is a risk metric based on personnel resources. Personnel measures characterize the amount of effort that is planned versus the amount that is expended by defined products or activities. These measures characterize the number of personnel assigned to a program, the experience and training levels of individuals, and the turnover rate (the rate at which individuals are removed or added to a program). These measures can be used to analyze the allocation of labor and to assess the adequacy of effort planned. Due to the labor-intensive process of a software project, personnel measures are especially critical.

The metric’s fundamental purpose is to indicate whether or not key acquisition personnel billets have been filled and if personnel filling the billets have the necessary knowledge, skills, and abilities (KSAs), via education, training, and experience, to appropriately manage a program. Tracking the planned personnel/staff versus the actual personnel/staff provides insight into future cost and schedule issues as does tracking the KSAs required at each lifecycle phase with the actual KSAs that program personnel possess. Table 2-5 provides the criteria for the implementation of the Software Organization metric.

Phase	I	II	III	IV	V	VI	VII
Baseline/ Basis of Metric	Effort/KSA	Effort/KSA	Effort/KSA/Turnover	Effort/KSA/ Turnover	Effort/KSA/ Turnover	Effort/KSA/ Turnover	Effort/KSA/ Turnover
Who Collects Measure-ments	Program Office	Program Office	Program Office/ Bidders	Program Office/ Contractor	Program Office/ Contractor	Program Office/ Contractor	Program Office/ Contractor
Who Analyzes	Program Office	Program Office	Program Office	Program Office/ SW developer/ integrator	Program Office/ SW developer/ integrator	Program Office/ SW developer/ integrator	Program Office/ SW developer/ integrator
Metric	Planned # of people or planned # of labor hours, KSA	# of people or # of labor hours/actual trng vs required trng	# of people or # of labor hours/actual trng vs required trng/# of people lost & gained	# of people or # of labor hours/actual trng vs required trng/# of people lost & gained	# of people or # of labor hours/actual trng vs required trng/# of people lost & gained	# of people or # of labor hours/actual trng vs required trng/# of people lost & gained	# of people or # of labor hours/actual trng vs required trng/# of people lost & gained
Related Artifacts	AoA Plan, FAA, FNA, FSA,ICD, Study Contracts, Integrated Architecture	Acq. Plan, AoA, Draft CDD, Preliminary System Specifica-tion, Cost/ Manpower Estimate, Draft RFP, SEP, T&E Strategy	ICE, CARD, CCA, POE, RFP, Proposals, Technology Develop. Contract, Initial Product Support Strategy, TEMP, PPP, TRA	SDD Contract, SDP (WBS), Work Packages, Timesheets, Task Mgmt Plans, Product Support Plan	SDD Contract, SDP (WBS), Work Packages, Timesheets, Task Mgmt Plans, Product Support Plan,	SDD Contract, SDP (WBS), Work Packages, Timesheets, Task Mgmt Plans, Product Support Plan,	SDD Contract, SDP (WBS), Work Packages, Timesheets, Task Mgmt Plans, Product Support Plan,
Use of Metrics	Risk, Lessons Learned	Risk, Lessons Learned	Risk, Lessons Learned, Source Selection	Risk, Lessons Learned	Risk, Lessons Learned	Risk, Lessons Learned	Risk, Lessons Learned

Table 2-5. Software Organization Metric

2.2.4.1 Baseline

A baseline must be established according to the staffing plan created by the program and based on historical data of past staffing efforts. This baseline should include identification of critical skills. The main function of the baseline is to indicate the number of billets that need to be filled at each phase of the life-cycle, together with the KSAs required by the personnel filling each billet. The baseline also provides a foundation for evaluation of the availability of appropriate skills for each task in the WBS.

The Software Organization metric can be broken down by the following attributes:

- Labor categories based on the critical roles identified in Table 4-1 in Chapter 4: Program Manager, Chief Systems Engineer, Software Architect, Chief Software Engineer, Process Compliance Manager, Risk Management Manager, Logistics Manager, Contracts, Legal, etc. Other roles that may be included are: coder, tester, configuration management, quality assurance, documentation, etc. Labor costs vary significantly for the various categories, so this information is useful when developing and updating cost estimates.
- Experience factor based on the Career-Long Learning Continuums: Defense Acquisition Workforce Improvement Act (DAWIA) Certification Requirements, Core Plus for Software Acquisition, Experience for Software-intensive ACAT Programs, and Continuing Education Units. Training costs

vary for the various continuums, so this information is also useful when developing and updating cost estimates, as the cost to the program may increase if training courses must be provided to personnel. Additionally, the program schedule may be affected, as more days may need to be added to the schedule if personnel need to take time off from the task to attend training.

- Increment or release, build, or spirals, Computer Software Configuration Items (CSCIs): Many programs have overlapping increments/releases, builds, or spirals occurring at the same time. This breakdown allows it to be determined whether problems for a specific increment/release, build or spiral are related to the Software Organization metric.

These breakdowns allow the analysis of the metrics to determine what the causes of the issues are. Additionally, breakdowns at this level are useful in updating program schedules and costs to account for the current situation.[15]

2.2.4.2 Measures

Three base measures are included in the Software Organization Metric:

- Effort;
- Staff Training and Experience; and
- Staff Turnover.

These base measures are to be collected by the program office and the supplier/contractor. Note, however, that the supplier/contractor measures cannot be collected until after contract award, and only if the requirement for providing these measures is included in the RFP. It would be beneficial to collect these measures monthly so staffing risks and/or issues can be identified earlier in the lifecycle rather than later. For example, if the progress of training staff is not on schedule with the lifecycle phase requirements, then a risk with mitigation steps can be tracked to address the lack of appropriate skills and experience prior to upcoming lifecycle phases (such as Initial Operational Capability (IOC), Full Operational Capability (FOC), etc.).

Specifically, the **Effort** measure counts the number of people or the number of labor hours (months, days, etc. applied to the tasks). This measure can be categorized by product or activity. Generally it directly correlates with cost, but can also correlate with process performance and schedule. This measure helps the program determine if activities are taking more or less effort than anticipated and if resources are being utilized according to the plan.[16] This measure should be collected at all seven of the system acquisition lifecycle phases listed in Table 2-1.

The **Staff Training and Experience** measure compares the actual training and experience of key personnel to the required training and experience for their billet. A target profile can be established by creating a table of key personnel to their required training and experience. See Chapters 3 and 4 for assistance in baselining this measure. This measure determines if current personnel have the experience and training necessary to sufficiently execute the tasks required of them. Additionally, it provides an opportunity for the program to determine what training needs exist and allows them time to prepare a plan

[15] Department of the Navy. Naval Air Systems Command (NAVAIR). Software Metrics Program Handbook, (Rev. 1) (SWDIV-HDBK-7). 1 November 2002.
[16] Department of Defense and US Army. Practical Software and Systems Measurement – A Foundation for Objective Project Management, Version 4.0c., March 2003.

to provide necessary training.[17] As with the Effort measure, this measure should be collected at all seven of the system acquisition lifecycle phases listed in Table 2-1.

The **Staff Turnover** measure represents the number of personnel lost and gained. High turnover within a program impacts cost, schedule, and productivity. This measure helps the program determine the number of staff added or lost from the program, how training and experience levels of the program are being affected by personnel lost and gained, and the specific areas being affected the most by personnel lost and gained.[18] This measure should be collected and reported at the following system acquisition lifecycle phases: Technology Development (Post MS-A, Leading to MS-B), System Development and Demonstration (Post MS-B, Leading to DRR), System Development and Demonstration (Post DRR, Leading to MS-C), Production and Deployment (Post MS-C, Leading to FRP), and Operations and Support (Post FRP).

2.2.4.3 Indicators

The three measures above (**Effort**, **Staff Training and Experience** and **Staff Turnover**) can be collected and used in the following ways.

Effort measures can be collected by tracking planned staff loading versus actual staff loading, as this provides visibility into potential schedule and delivery risks and issues (see Figure 2-3 for an example).[19] Evaluating the availability of appropriate skills for each task in the WBS is also useful but program offices seldom have all the resumes and time sheets to do so.

Resources should be identified in a way that reflects the varying levels of experience of the staff. Staff training and experience can be collected by using the proposed Career-Long Learning Continuums, as discussed in the Software Process Improvement Initiative (SPII) Human Resource report,[20] as this is a recommended framework to classify program staff as DAWIA Level 1, II, or III, where Level I staff have the least amount of training and experience and Level III staff have the most training and work experience.

[17] DoD and US Army. "Practical Software and Systems Measurement – A Foundation for Objective Project Management."
[18] DoD and US Army. "Practical Software and Systems Measurement – A Foundation for Objective Project Management."
[19] Defense Acquisition University. Software Acquisition Management Course, Software Management Metrics Module Teaching Note 6.2, 6.2doc/Version 6, DAU_SAM 762 Metrics.
[20] The report can be accessed at: http://acquisition.navy.mil/organizations/dasns/rda_cheng.

Tracking staff turnover, personnel lost and gained, is also important, especially if the losses are in key critical roles as identified in Chapter 4 such as: the Program Manager, Chief Systems Engineer, Software Architect, or Risk Management Manager. Losing or gaining key personnel can impact the program's cost, schedule, and productivity, as there will be down time in getting new personnel trained and indoctrinated into the program.

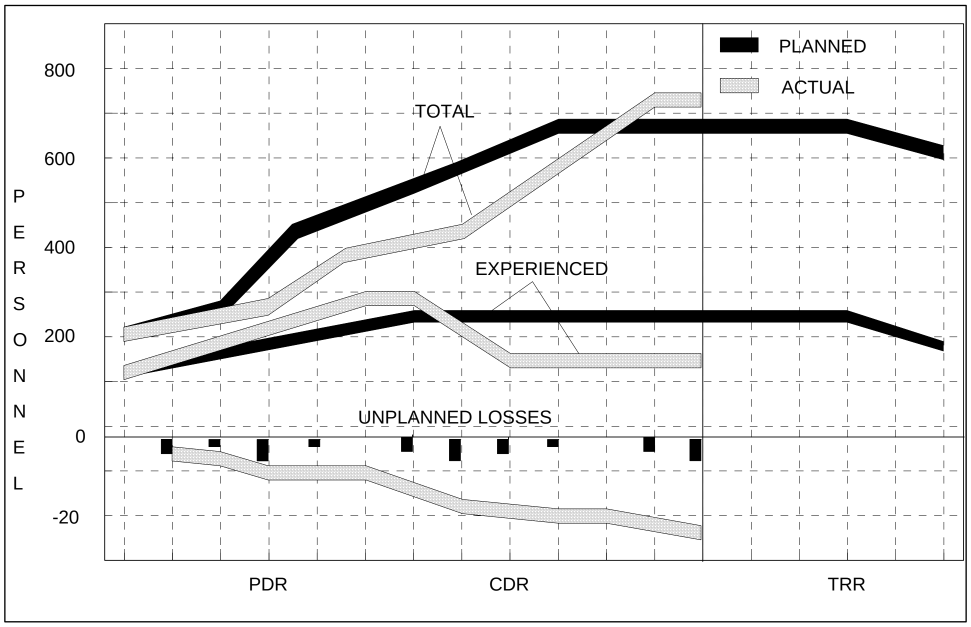


Figure 2-3. Personnel Measure[21]

2.2.4.4 Rules of Thumb for Software Organization Metrics

- The ratio of total personnel to experienced personnel should never exceed 6:1. A ratio of 3:1 is typical.
- Initial staffing should comprise about 25% of the total personnel requirements per the staffing plan.
- The front end should be leveraged with more experienced staff.
- Experience has demonstrated that "late staffing" is high risk when more than 25% of a task has been expended, or is late starting, and when critical billet(s) has not been filled or was filled with a less-than-experienced person.
- The use of part-time staff versus full time staff for critical staff is usually high risk.
- The development schedule depends on the amount of staff-months expended:
  - Understaffing is an early sign of schedule slippage;
  - If behind, catching up cannot always be accomplished by adding more staff. Sometimes adding more staff may further delay the overall schedule;
  - Use of overtime may help over the short term;
  - Additional personnel may work on a short term basis but only for tasks that can be separated or isolated with simple interfaces.

[21] Defense Acquisition University. Software Acquisition Management Course, Software Management Metrics Module Teaching Note 6.2, 6.2doc/Version 6, DAU_SAM 762 Metrics.

- High turnover or loss of critical personnel can be a sign of internal management problems, especially if turnover occurs prior to critical reviews or lifecycle events.

References

Card, David, Elizabeth Clark, Joseph Dean, Fred Hall, Cheryl Jones, Beth Layman, and John Mcgarry. Practical Software Measurement: Objective Information for Decision Makers. New York: Addison-Wesley Professional, 2001.

Chairman of the Joint Chiefs of Staff Instruction (CJCSI) 3170.01F, Joint Capabilities Integration and Development System, 1 May 2007.

Chairman of the Joint Chiefs of Staff Manual (CJCSM) 3170.01C, Operation of the Joint Capabilities Integration and Development System, 1 May 2007.

Defense Acquisition University. Defense Acquisition Guidebook. <https://akss.dau.mil/dag/>.

Defense Acquisition University. Software Acquisition Management Course, Software Management Metrics Module Teaching Note 6.2, 6.2doc/Version 6, DAU_SAM 762 Metrics.

Department of Defense. Department Of Defense Handbook (HDBK 881) – Work Breakdown Structure. July 2005. <http://www.acq.osd.mil/pm/currentpolicy/wbs/MIL_HDBK-881A/MILHDBK881A/WebHelp1/MILHDBK881A.htm>.

Department of Defense. DoD 5000.4-M-2 Software Resources Data Report (SRDR) instructions. <http://dcarc.pae.osd.mil/Policy/srdr/index.aspx>.

Department of Defense. USD(AT&L). DoD Instruction 5000.2, Operation of the Defense Acquisition System. 12 May 2003.

Department of Defense and US Army. Practical Software and Systems Measurement – A Foundation for Objective Project Management, Version 4.0c., March 2003.

Department of the Navy. DASN(RD&A) ALM. SECNAVNOTE 5000, Department of the Navy (DoN) Requirements and Acquisition Process Improvements. 26 February 2008.

Department of the Navy. Naval Air Systems Command (NAVAIR). Software Metrics Program Handbook, (Rev. 1) (SWDIV-HDBK-7). 1 November 2002.

Department of the Navy. Naval Air Systems Command (NAVAIR). Using Software Metrics and Measurements for Earned Value Toolkit. 2 December 2004.

Department of the Navy. Naval Air Systems Command (NAVAIR). Measures for Software Intensive Programs. 20 July 2005.

Department of the Navy. ASN(RD&A) Software Process Improvement Initiative (SPII) Human Resource Report, 06 November 2007; report can be accessed at: <http://acquisition.navy.mil/organizations/dasns/rda_cheng>.

Executive Office of the President. Office of Management and Budget. Circular A-11, Part 7. July 2007.

FY03 Defense Authorization Act. Public Law. 107-314. 2 Dec. 2002. STAT. 116. 2465. Sec. 804.

Government Extension to the PMBOK Guide, Third Edition. Sylva: Project Management Institute, 2006.

Hantos, Peter. “Defense Acquisition Performance Assessment - The Life Cycle Perspective of Selected Recommendations.” CrossTalk: The Journal of Defense Software Engineering. May 2007: 25-29.

IEEE/EIA 12207; specifically IEEE/EIA 12207.0-1996, Software Life Cycle Process; Industry Implementation of International Standard ISO/IEC 12207: 1995 – Standard for Information Technology; March 1998. (This contains ISO/IEC 12207 in its original form and six additional annexes)

IEEE Standard 1061™-1998 (R2004) — IEEE Standard for a Software Quality Metrics Methodology

ISO/IEC CD 15939 - Information Technology — Software Measurement Process

International Function Point User Group. Guidelines to Software Measurement, Release 2. International Function Point User Group: Westerville OH, 2004.

“Measurement for DoD Projects”, DoD Implementation Guidance, Practical Software and Systems Measurement, 24 February 2003.

PMBOK® Guides. A Guide to the Project Management Body of Knowledge, Third Edition (ANSI/PMI 99-001-2004). Newtown Square, PA: Project Management Institute, 2004.

Practice Standard for Work Breakdown Structures, Second Edition. Sylva: Project Management Institute, October 2006.

Pyzdek, Thomas. The Six Sigma Handbook: The Complete Guide for Greenbelts, Blackbelts, and Managers at All Levels, Revised and Expanded Edition. New York: McGraw-Hill, 2003.

Software Engineering Institute. CMMI® for Acquisition, Version 1.2. Carnegie Mellon University, 2007.

U.S. Air Force. Probability of Program Success (PoPS) Spreadsheet Operations Guide, Version 3.1. July 2006.

This Page Intentionally Left Blank

3 Role Based/Right Fit Training

3.0 Overview

Software complexity is growing exponentially in Department of Defense (DoD) projects. Software intensive systems are often the primary cost, schedule, and performance drivers in naval programs. Unfortunately, Navy acquisition personnel are frequently not sufficiently prepared to perform the tasks required by their positions when acquiring software systems. Assistant Secretary of the Navy (Research, Development and Acquisition) (ASN(RD&A))'s Software Process Improvement Initiative highlighted several issues:

- Software acquisition professionals fall into three broad categories: software generalists ("the masses"), software experts, and Green Team members (a subset of the experts);
- Core software competencies have not been fully identified;
- Elements necessary to determine "Role Based/Right Fit (RB/RF)" training include acquisition disciplines, competencies, and available training (existing and/or to be modified);
- Identification of naval software acquisition management and engineering RB/RF training opportunities, experience, and continuing education units should leverage the Defense Acquisition Workforce Improvement Act (DAWIA) Certification construct and incorporate the Defense Acquisition University (DAU) Core Plus Framework; and
- Naval solutions must be developed in conjunction with similar, ongoing DoD efforts in order to leverage expertise and training opportunities.

Research report findings, proposed process for determining training and education solutions, and recommendations cross the Department of the Navy (DoN) enterprise, apply beyond program offices, and form the foundation for Naval participation in the Office of the Secretary of Defense (OSD)-led DAWIA Software Acquisition Training and Education Working Group (SATEWG); this chapter excerpts discussion relevant to program offices. The complete report (hereinafter referred to as the "SPII HR report") can be accessed at: http://acquisition.navy.mil/organizations/dasns/rda_cheng.

3.1 Functional Disciplines

Six acquisition disciplines provide the foundation for software acquisition management. A description of the responsibilities of software acquisition generalists in each of the disciplines addressed is given below. The descriptions were adapted from the Acquisition Technology and Logistics (AT&L) Workforce Resources Position Category Descriptions (PCDs) and form the basis for the roles and responsibilities identified in Chapter 4: Software Acquisition Project Staffing Considerations (note that for certain functional assignment purposes, "Systems and Software Engineering" are divided into separate disciplines for more focused functional analysis).

- Program Management – responsible for Navy acquisitions including weapon systems, command and control systems, information management systems, etc. All these systems include software as part of the acquisition process. Program Managers need a broad understanding of software acquisition and systems engineering principles to translate information from support personnel (logisticians, contract specialists, systems and software engineers, legal specialists, and test & evaluation specialists) into program decisions.
- Systems Planning, Research, Development, and Engineering (SPRDE) – Systems and Software Engineering – plan, organize, and conduct engineering activities relating to the design, development, fabrication, installation, modification, sustainment, and/or analysis of systems or systems components across the entire life cycle. This discipline includes SPRDE Software/Information Technology (IT) Engineers who plan, organize, and conduct engineering activities relating to the design, development, and/or analysis of software and information technology systems or system components.
- Test & Evaluation Engineering – plan, organize, manage, or conduct tests and/or evaluations associated with concepts, emerging technologies, and experiments as well as prototypes, new-, fielded-, or modified-C4ISR (Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance) systems, weapons or automated information systems, equipment or materiel throughout all acquisition phases to include developmental tests, and support to in-service tests and operational tests.
- Acquisition Logistics – plan, develop, implement and manage effective and affordable support strategies throughout the life cycle for weapons, materiel, or information systems. Logisticians perform a principal joint and/or component logistics supportability role during the acquisition and sustainment phases of the system and software life cycle. Logisticians also develop and implement performance-based approaches for logistics systems support. Products and services delivered by logisticians sustain system operational readiness.
- Contracting – develop alternatives to produce best value supplies and services, as well as manage all aspects of the life cycle of a contract or other vehicle. Apply statutory and policy procurement related requirements; support attainment of government socio-economic objectives; conduct market research; acquisition planning; cost and price analysis; solicitation and selection of sources; preparation, negotiation, and award of contracts through various methods to include negotiation; perform all phases of contract administration; and terminate or close out contracts.
- Legal (Intellectual Property Attorneys) – perform contract advisory service to the DoD and other government agencies in negotiation, administration, settlement of contracts, and subcontracts for software-intensive products/services. Legal specialists function as consultants to various organizations under the Defense Acquisition Regulatory Council to develop innovative legal solutions to the business and other challenges facing the Navy and Marine Corps to enhance warfighting capabilities of the naval service.

3.2 Competencies (a.k.a. Knowledge, Skills, and Abilities)

Accurate identification of required competencies are critical to support the curriculum review and development effort needed to ensure the best and most relevant training is provided to software acquisition management and engineering personnel. The SPII HR report captures the initial naval review of competencies required by the six core disciplines described in Section 3.1; see Table 3-1 for an excerpt.

This set of competencies (see Appendix G) has been provided to the SATEWG for the basis of their Knowledge, Skills, and Abilities (KSA) review and, when jointly approved, will reflect those KSAs necessary for program office personnel involved with software acquisition and management.

Program Management Level I Competencies
Software Acquisition Management Regulatory/Technical Framework Application & Analysis
Give examples of best system strategies for SW intensive systems
Explain the effect of current system strategies on SW Acquisition Mgmt
Summarize the strengths and weaknesses of current strategies
Explain the impact of acquisition strategy on SW project planning and SW engineering methods
Explain the impact of Acquisition Reform

Table 3-1. Competencies for Level I Program Managers (partial)

3.3 Recommended Training

In May 2006, ASN(RD&A) identified a need for key government program office personnel to have a minimum level of knowledge of software acquisition and engineering management practices (see Appendix A). Two courses were identified as requirements for all government Program Managers, Deputy Program Managers, and Technical Directors/Chief Engineers assigned to an acquisition category (ACAT) I or II program. Subsequent to this direction, ASN(RD&A) expanded upon the guidance for software acquisition and management training by emphasizing that "Necessary courses will be as defined by Defense Acquisition Workforce Improvement Act (DAWIA) Level I/II/II certification for appropriate career fields. Program managers will ensure their functional organization includes personnel meeting DAWIA certification for the six functional disciplines." [1]

3.4 SATEWG

In August 2007, a Tri-Service Education Training meeting was held at Air Force Institute of Technology (AFIT) with participants from OSD, US Air Force, US Army and the Navy. At this meeting the SATEWG was established to review competencies within the six acquisition disciplines based on the Navy's SPII HR report's RB/RF training process and recommendations as a baseline. The group was formally tasked by Deputy Under Secretary of Defense (Acquisition Technology and Logistics) (DUSD(AT&L)) to "affirm required software competencies and gaps in DAWIA curricula, develop a plan for resolving the gaps, and initiate course updates, beginning with Program Manager and SPRDE career fields."[2] This group continues to identify required competencies needed to ensure the best and most relevant training is provided to software acquisition management and engineering personnel. Ongoing naval participation is led by ASN(RD&A) Chief Systems Engineer (CHSENG), coordinating closely with Program Executive Offices (PEOs) and Systems Commands (SYSCOMs) to capture ongoing competency and training perspectives.

[1] Department of the Navy. ASN(RD&A) memo. Policy for Acquisition of Naval Software Intensive Systems. 16 September 2008.
[2] Department of Defense. DUSD(AT&L). memo. Establishment of Software Acquisition Training and Education Working Group. 19 February 2008

SPII HR report recommendations noted the value of “front line” (program office) participation in providing feedback from the acquisition workforce to ensure appropriate and adequate software acquisition training.

3.5 Summary

The value of appropriate competencies throughout the program office cannot be understated; they contribute significantly to program success and are integral to one of the four core metrics (see Chapter 2 for more information on the metric “Software Organization”). Initial naval identification of competencies to support acquisition disciplines has been completed, and is being used by OSD as the baseline for a joint review and determination of associated training and experience requirements. This effort is currently ongoing and it is expected that changes to recommended training will be identified periodically. Program managers should ensure their functional organization includes personnel meeting DAWIA certification for the six functional disciplines.[3]

[3] Current DAWIA certification information can be accessed at: http://www.dau.mil/workforce/index_sub5_CareerFildCertStandardsFY08.asp.

References

2004 AT&L Competency Workbooks can be accessed at: <https://acc.dau.mil/CommunityBrowser.aspx?id=39310&lang=en-US>.

2006 Clinger-Cohen Core Competencies can be accessed at: <https://acc.dau.mil/CommunityBrowser.aspx?id=156918>.

2007 AT&L Human Capitol Strategic Plan v.3.0 can be accessed at: <https://acc.dau.mil/CommunityBrowser.aspx?id=118508&lang=en-US>.

AT&L Workforce Resources Position Category Descriptions (PCDs) can be accessed at: <http://www.dau.mil/workforce/PCDs.asp>.

DAWIA Certification Requirements (DAU 2008 Catalog) can be accessed at: <http://www.dau.mil/workforce/index_sub5_CareerFildCertStandardsFY08.asp>.

Defense Acquisition University Core Plus Framework (DAU 2008 Catalog) can be accessed at: <http://www.dau.mil/workforce/index_sub1_CorePlus.asp?eventid=1583>.

Defense Acquisition University. Defense Acquisition Guidebook (Chapter 4: Systems Engineering). <https://akss.dau.mil/dag/GuideBook/PDFs/Chapter_4.pdf>.

Department of Defense Directive 500.52 (Jan. 12, 2005) Defense Acquisition Technology, and Logistics Workforce Education and Career Development Program can be accessed at: <http://www.dtic.mil/whs/directives/corres/pdf/500052p.pdf>.

Department of Defense. DUSD(AT&L). memo. Establishment of Software Acquisition Training and Education Working Group. 19 February 2008.

Department of the Navy. ASN(RD&A) memo. Policy for Acquisition of Naval Software Intensive Systems. 16 September 2008.

Department of the Navy. ASN(RD&A) Software Process Improvement Initiative (SPII) Human Resource Report, 6 November 2007 can be accessed at: <http://acquisition.navy.mil/organizations/dasns/rda_cheng>.

This Page Intentionally Left Blank

4 Software Acquisition Project Staffing Considerations

4.0 Overview

Software has become pervasive in most of the weapon and business systems under development by the U.S. Navy, and many programs have experienced cost overruns and schedule slippages due to software development problems. Often, the root cause of these problems has been related to inadequate and inexperienced staff at both the acquiring program office and at the supplier. Appendix H provides detail regarding some of the difficulties faced when trying to populate projects with experienced system and software engineering personnel. This chapter provides recommended project staffing considerations and a tailorable organization structure for large software acquisition projects. Key concepts that are unique (or more critical) when staffing a large software intensive system acquisition are also discussed. Staffing and organizational issues were examined in the context of anticipating even larger software intensive and system of systems acquisitions, since these large acquisitions are more demanding and presumably more dependent upon experienced software acquisition staff for their success. Software staffing challenges and issues, requisite staff disciplines, software domain experience, and training needs were factors considered in developing the recommendations. Organizational constructs and practices that were successful in the past but that may not work on large software intensive acquisitions are also identified.

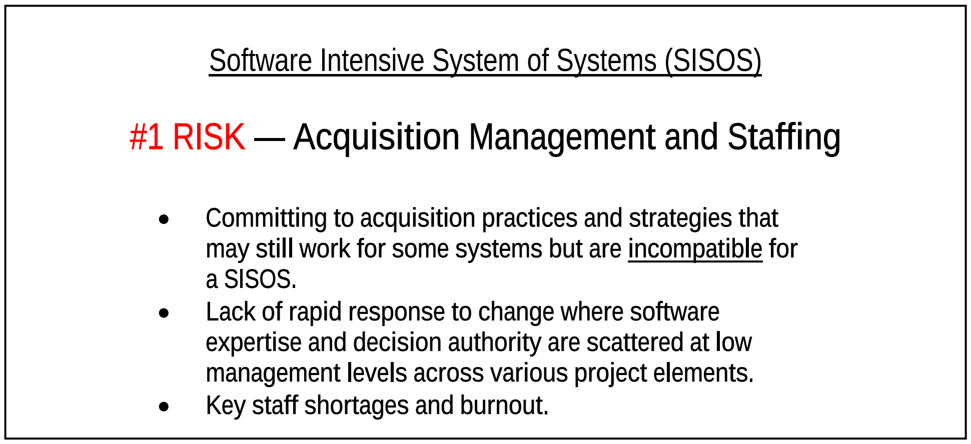
This chapter focuses on Department of the Navy (DoN) acquisition program offices vice product developers. It is written from the perspectives of acquisition programs focused on Software Intensive Systems (SIS) and Software Intensive Systems of Systems (SISOS). Thus, software staffing recommendations in this guidebook include expansive staffing recommendations to accommodate the most complex software environments. Program offices will determine the specific billet/personnel mix appropriate to their environment; programs with less intensive software components should balance risk versus talent accordingly (e.g., program offices may elect to combine positions or realign responsibilities as appropriate). Regardless of combinations or re-alignments, anything below the minimum set increases risk.

4.1 Software Staffing Challenges

The advent of software intensive systems established the need to push software engineering considerations into the system engineering process, thereby producing a more unified approach that is better suited to developing large and complex systems.[1] While this approach seemed sound, the staffing and actual execution of the acquisition plan were often poorly accomplished for a variety of reasons, and program failures still resulted. Studies have indicated that inadequate software staffing (in numbers, experience, and timing (too late on-the-job)) is a major contributing factor in some of these program failures. [2] While this problem lingers, the movement toward even larger software intensive systems of

[1] Boehm, Barry. "Unifying Software Engineering and Systems Engineering." IEEE Computer, 33 no. 3: 114-116.
[2] National Defense Industrial Association. (2006). Systems Engineering Division Task Group Report: Top Software Engineering Issues Within Department Of Defense And Defense Industry and Boehm, Barry. "Software Engineering: What Have We Learned? Where Are We Going?" NGC Tutorial 16 July 2007.

systems is introducing additional staffing and organizational considerations. Staffing, planning, and architectural issues are top-level acquisition risks as indicated in Figure 4-1 and Figure 4-2 below. Furthermore, those with this software expertise need not only to understand the architectural and risk issues associated with SIS and SISOS developments but somehow need to have an influence on or in higher-level decisions and earlier phases than is historically the case.

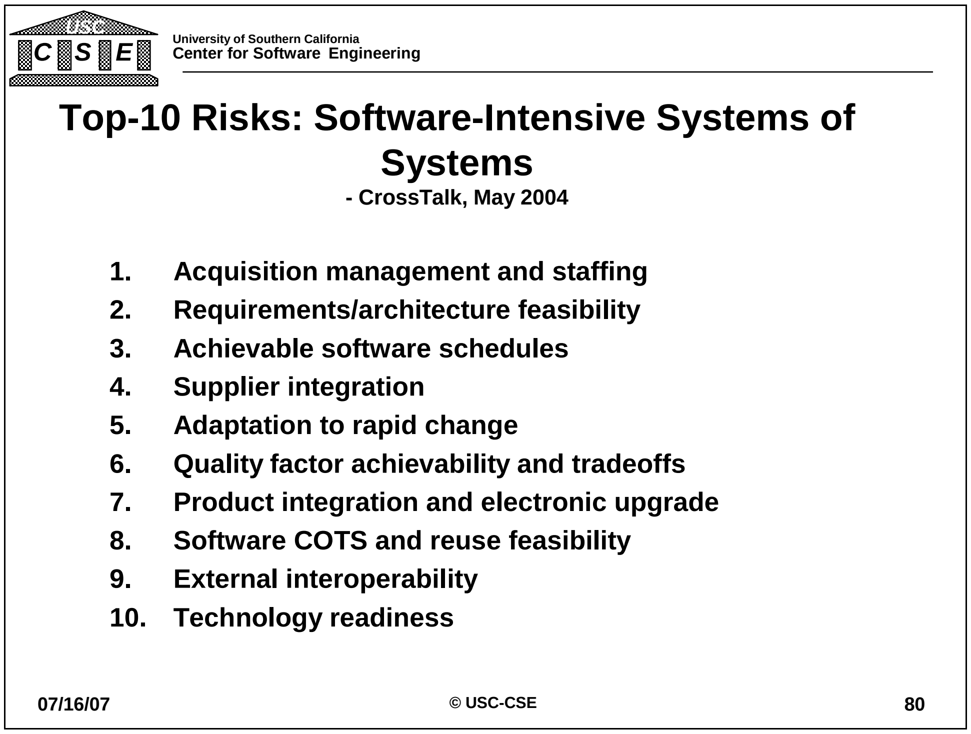


Software Intensive System of Systems (SISOS)

#1 RISK — Acquisition Management and Staffing

- Committing to acquisition practices and strategies that may still work for some systems but are incompatible for a SISOS.
- Lack of rapid response to change where software expertise and decision authority are scattered at low management levels across various project elements.
- Key staff shortages and burnout.

Figure 4-1. SISOS Top Risk[3]



University of Southern California
Center for Software Engineering

Top-10 Risks: Software-Intensive Systems of Systems

- CrossTalk, May 2004

1. **Acquisition management and staffing**
2. **Requirements/architecture feasibility**
3. **Achievable software schedules**
4. **Supplier integration**
5. **Adaptation to rapid change**
6. **Quality factor achievability and tradeoffs**
7. **Product integration and electronic upgrade**
8. **Software COTS and reuse feasibility**
9. **External interoperability**
10. **Technology readiness**

07/16/07 © USC-CSE 80

Figure 4-2. SISOS Risk Areas[4]

[3] Boehm, Barry, A. Windsor Brown, Victor Basili, and Richard Turner. "Spiral Acquisition of Software-Intensive Systems of Systems." CrossTalk:The Journal of Defense Software Engineering. May (2004).
[4] Boehm, Barry. "Software Engineering."

4.2 Key Concepts in SIS and SISOS Staffing

Software Intensive Systems involve many developers, sometimes under one contract. Systems of systems involve multiple developers and vendors (legacy, existing, and future) and rarely are they under the same or even related contracts. Both situations require coordination and continuous communication, and in general require more up-front planning and staffing considerations. Discussed below are some of the key areas that significantly influence software roles and responsibilities within a Navy program office.

4.2.1 Domain Experience

For SIS and SISOS programs, it is critical that the Navy increase the influence of the software domain in acquisition by increasing the number of personnel qualified to perform SIS and SISOS acquisition functions. Adding more checklists, review sessions, or cross-training is not efficient; increasing the influence of the software domain requires that:

- Individuals assigned to these functions must act and be viewed as full-fledged members of the core acquisition team, and must be assigned and fully accepted as a critical billet to the Program Manager (PM).
- Generalists must also be firmly grounded in the issues germane to development and support of software intensive systems, while experts for critical billets should be recruited from competencies heavily involved in software support, so that they have the required hands-on experience in the software domain.
- Expertise is often phase-specific, so planned turnover is required to assure right-fit is maintained. Additionally, the fast-paced nature of change in the software domain forces management to find a process to ensure this workforce is refreshed with personnel having recent and relevant experience and training. It is important for competency managers within a Competency Aligned Organization/Integrated Product Team (CAO/IPT) organization to forecast the necessary skills and the associated training required to ensure personnel are capable of supporting current and future naval acquisition programs.

4.2.2 Risk Management

The multi-organization multi-vendor environment of large acquisitions requires more intensive risk management as new risks are incurred. Risk Management must be a fully integrated, combined effort spanning different organizations that use different tools and have different risk management boards. It is a key process area that should be addressed (i.e., staffed) up front and early in a software intensive acquisition program. Experience has demonstrated that “late staffing” is high risk when more than 25% of a task has been expended, or is late starting, and when a critical billet(s) has not been filled or was filled with a less-than-experienced person. The use of part-time staff versus full time for critical billets is usually high risk except on the smallest subsystems.

4.2.3 Software Architecture

Software architecture is vitally important, and is referred to as "*the set of design decisions which, if made incorrectly, may cause your project to be cancelled.*"[5] An architecture trade-off analysis is needed to validate key requirements and define interfaces between system components. This requires a considerable up-front expenditure of both program office and contractor resources. SIS and SISOS programs must have a Software Architect (critical billet) to ensure that overall technical feasibility and rapid change capabilities are considered as much as system performance. This software engineer's role in overall system architecture analysis is also critical to properly meet the Navy's emphasis on Service Oriented Architectures (SOA).[6]

4.2.4 Acquisition Life Cycle Planning

Secretary of the Navy Instruction (SECNAVINST) 5000.2D and Department of Defense Instruction (DoDI) 5000.2 contain the statutory, regulatory, and contract reporting information and milestone requirements for acquisition category (ACAT) programs, but these plans and reports do not encompass the entire scope of issues pertinent to software intensive systems. Many planning documents do seem to address items such as "supportability" but coverage and accountability have been weak; for example, SECNAVINST 5000.2D states that the PM "shall document the product support strategy in the acquisition strategy." For the most part, the "support" mentioned here falls within the domain of Performance Based Logistics (PBL) and may be covered in the Acquisition Logistics Support Plan (ALSP) and can therefore be met without ever addressing software design, architecture, or maintaining warranties for Commercial Off-The-Shelf (COTS), for example. This scattered approach requires that the software engineering and software acquisition community be fully engaged in developing multiple and disparate planning documents spending a great deal of time on small parts of many documents changing regularly during each phase. Otherwise, key aspects of software transition and support are sometimes neglected, or parts are not supported, and costly change orders or test failures ensue. All software life-cycle management issues must be addressed "as one," beginning at program initiation through sustainment. Staffs are required to address the programmatic diversity, complexity, communication, and coordination requirements of SIS and SISOS acquisitions and require additional lead-time and planning details for the proper phasing of personnel resources.

To address these issues adequately, program office staff must have software experience. When critical billet Software Leads report directly to IPT Leads/Chief Engineer, and have direct access to them or their decisions, it enables better software acquisition life cycle planning.

4.2.5 Cost Estimation

Program budgets are often established based on initial cost estimates using poorly defined requirements and incomplete technical baseline definitions. It is important for an initial estimate to establish adequate risk margins to accommodate the seemingly inevitable growth in software requirements, which translate into increased software size, cost, and delay. It is equally important that historical Department of Defense (DoD) data on similar software developments be used to generate the cost estimate and as cross checks for size, schedule, and effort. Programs must resist pressure to fit their program into an artificially determined budget or constrained schedule when the technical baseline does not support the challenge. That is,

[5] Rozanski, Nick, and Eóin Woods. Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives. New York: Addison-Wesley Professional, 2005.
[6] Department of the Navy. Information Technology Applications and Data Management. SECNAV 5000.36A. 19 December 2005.

systems engineering must address software adequately and consistently in the early phases (with updates as requirements are better defined). Software procurement comparisons will be difficult as Earned Value Management System (EVMS) and even Work Breakdown Structure (WBS) for software acquisitions vary widely in coverage and depth. Without truly valid cost artifacts to compare to, having properly experienced or trained persons on-site at the beginning of estimating becomes the single most important factor. There exists no policy, procedure, or template to prevent re-using previously incorrect data to produce yet another erroneous cost estimate. The adoption, use, and updating of formal estimation methods and tools by software experienced estimators is key to improving software acquisition estimates.

4.2.6 Metrics and Measurement of Staff Experience and Training

Chapter 2 identifies and describes a core foundation of four metrics that all Programs of Record (POR), regardless of ACAT category, must define, develop, measure, and report specific to their program. Included in the core metrics is Software Organization. The purpose of the Software Organization metric is to measure compliance that key program personnel are identified, on-the-job, and have an appropriate level of experience or training. The adequacy and timeliness of staffing levels, with appropriate experience or recently completed training, is to be reported through Probability of Program Success (PoPS) and in compliance with SECNAVNOTE 5000 dated February 26, 2008, subj: Department of the Navy (DoN) Requirements and Acquisition Process Improvements.

4.3 Core Disciplines

The Software Process Improvement Initiative (SPII) HR report noted in Chapter 3 identified six acquisition disciplines: Program Management, Contracting, Acquisition Logistics, Systems & Software Engineering, Test & Evaluation Engineering, and Legal. The report serves as the initial naval input to ongoing Office of the Secretary of Defense (OSD) sponsored reviews of software acquisition management competencies, and can be accessed at http://acquisition.navy.mil/organizations/dasns/rda_cheng. This chapter builds upon that report and separates Software Engineering and Systems Engineering. The distinction is important when examining the technical "hands-on" experiences of the individuals responsible for critical insight/oversight of large or complex acquisitions. The seven core disciplines in which program personnel must be proficient are shown in Figure 4-3 below.

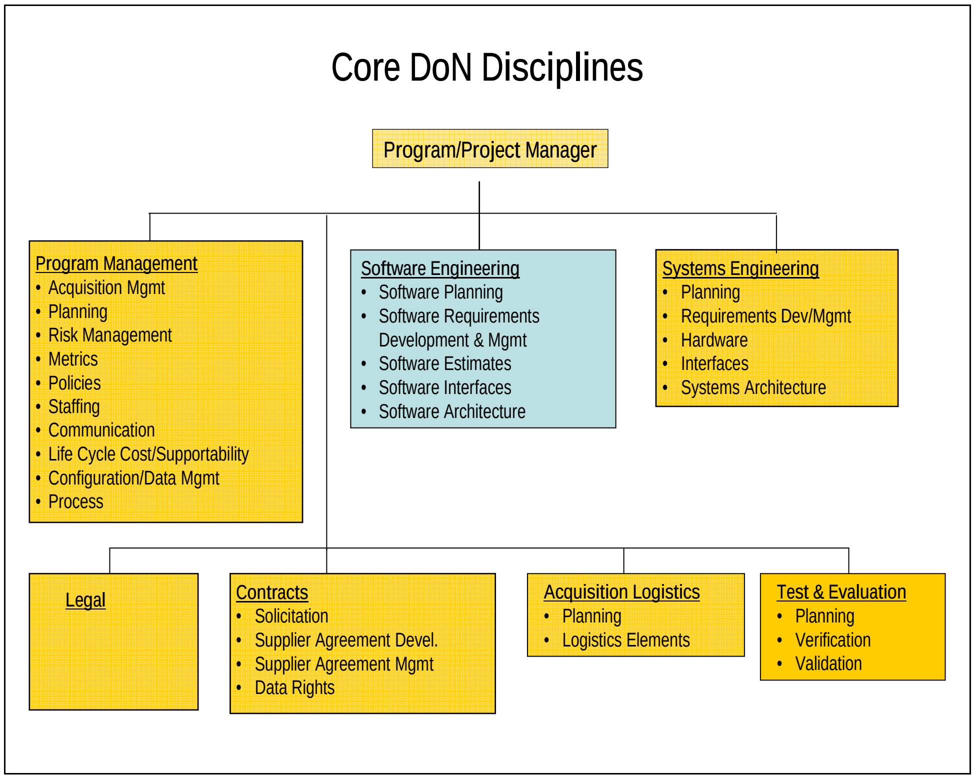


Figure 4-3. Core Disciplines

Knowledge, Skills, and Abilities (KSAs) associated with these disciplines, a measure in the core metrics and reported through PoPS, are discussed in Chapter 3.

4.4 Elevating Software Expertise Within the IPT Structure

For SIS and SISOS, the familiar IPT structure has produced some unintentional consequences. From a software perspective, the typical IPT:

- Fragments the technical effort;
- Does not provide a software-trained management structure; and
- Does not place software at a level commensurate with its complexity and risk, as depicted in Figure 4-4.[7]

[7] Hantos, Peter. "Risk Management in System of Systems Development - Are You Ready?" 20th International Forum on COCOMO and Cost Modelling. Center for Software Engineering, USC. University Of Southern California, Los Angeles. 26 Oct. 2005.

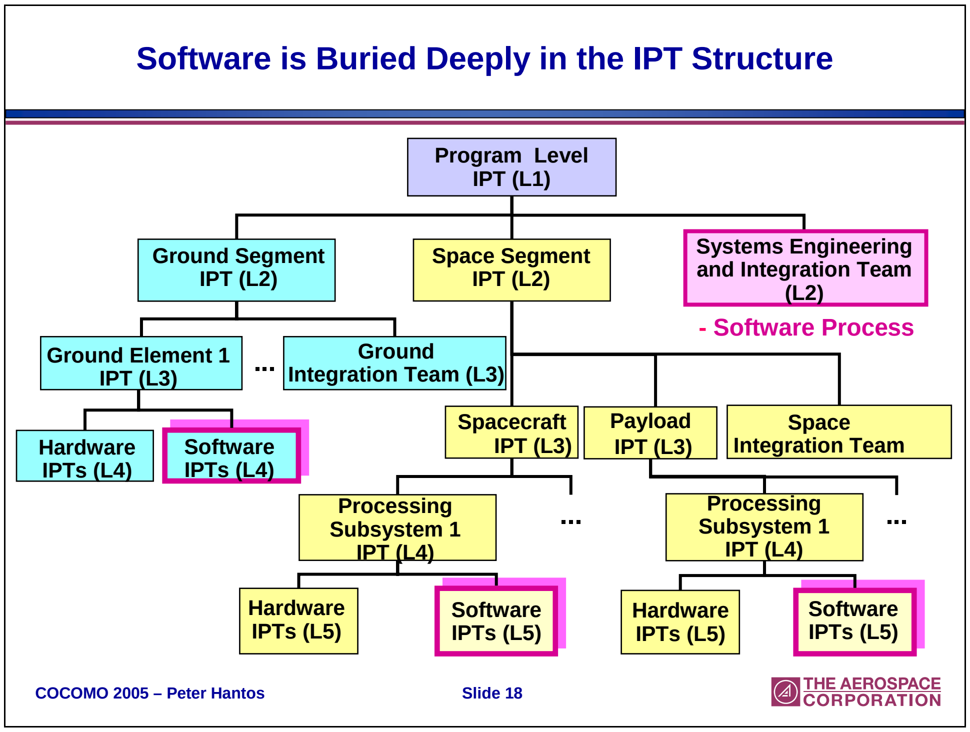


Figure 4-4. Program IPT Structure[8]

If software is pervasive, then software acquisition expertise should be pervasive in the acquirer's organization. Software is important and decisions made for or about the software throughout the acquisition process directly affect the success of the program. Yet software managers and technical leads are often found deep in organizational structures, which inherently forces them to arrive late in the program. Software knowledge and experience must be embedded in program leadership to be effective and responsive, and staffed earlier than has been traditional, but this applies mainly to critical billets. Software managers and software technical leads need to report directly to the IPT leader, and the overall Software Lead should report directly to the Program Manager or Chief Systems Engineer. Software Leads should hold positions whose importance is commensurate with the importance or risk of the software product.

To bring the right solutions to bear on these issues, decision makers should have software training and experience. Elevating software expertise within the IPT and program structure provides more knowledgeable decision makers and thereby provides overall risk reduction. An alternative program IPT structure that may help to accomplish this is shown in Figure 4-5. Note that the *Program-wide Staff* positions shown in this figure identify critical billets with *Program-wide Responsibilities* where software experience is required.

[8] Hantos, Peter.

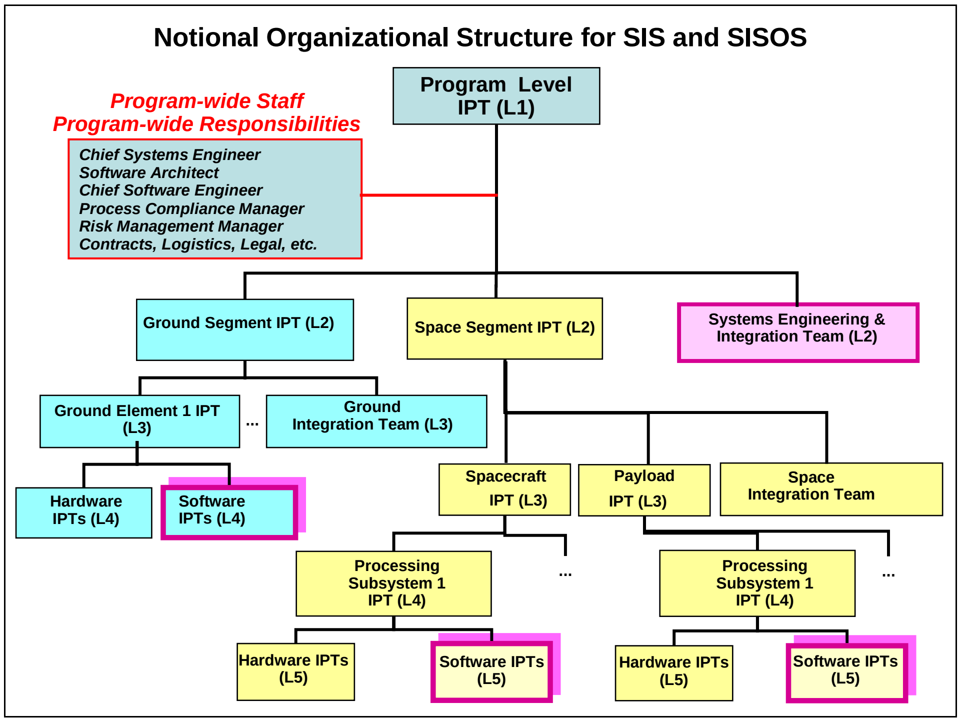


Figure 4-5. Alternative Program IPT Structure

4.5 Roles and Responsibilities

For any program, roles and responsibilities must be established and clearly documented by the Program Manager; they act as a contract between all members of the program. Roles and responsibilities should be documented in early planning documents, and carried forward and expanded in the Software Development Plan (SDP). Roles and responsibilities are necessary in order to:

- Establish the responsibilities and expectations of each member;
- Establish working relationships between team members; and
- Establish authority to commit resources.

The critical roles and responsibilities of positions requiring a level of software expertise for SIS and SISOS program acquisition are identified in Table 4-1.

SIS and SOS Acquisition Programs Critical Positions Requiring Software Experience*	
CRITICAL ROLE	OVERALL SOFTWARE RESPONSIBILITY
Program Manager	Single point of accountability for accomplishing program objectives for total life-cycle systems management, including software acquisition and software sustainment. Responsible for appropriate staffing according to schedule (organization metric).
Chief Systems Engineer	Defines system concept, high-level design, and key interfaces. Responsible for software/hardware/architecture trade-offs, trade studies, engineering analyses, and for liaison with software teams. Responsible for the requirements engineering process, including the integration of software and hardware requirements internally and within the SIS and SOS (size/stability metric). Defines roles and stakeholders. Confirms traceability of contract technical modifications to WBS (affects cost/schedule and quality metrics). Merges metrics with Chief Software Engineer's data and provides to Program Risk Manager.
Software Architect	Overall architecture, technical feasibility, and performance of the integrated software system. Makes visible or traces requirements, complex hardware (Field Programmable Gate Array (FPGA)), COTS, safety, security risks into architecture. (rolls into size/stability metric, affects cost/schedule metric).
Chief Software Engineer or Software Lead	Monitor and control the software acquisition project's progress and compliance (size/stability metric); initiate corrective actions when the project's performance deviates significantly from the supplier's software development plan(s) (cost/schedule metric). This includes evaluating initial cost and plans, and tracking actual cost and technical progress against development plans (EVMS/WBS). Overall technical responsibility for software quality, safety, security, and performance (quality metric). Confirms traceability of contract modifications to WBS (affects cost/schedule and quality metrics). Provides metrics to Chief Engineer(s) and Program Risk Manager.
Process Compliance Manager	Develops and tailors software acquisition process; monitors and measures developers' software process compliance in accordance with the Software Development Plan (SDP) quality metric; CMMI® and IEEE/EIA Standard 12207 knowledgeable. Provides (software) quality/defect metrics to Chief Engineer(s).
Risk Management Manager	Defines integrated program-wide software risk management plan and ensures risk management procedures and mitigation plans are in place across all organizations and IPTs; highlights high and serious safety risks, consolidates, incorporates and interprets metrics and reports overall software risk.
Logistics Manager	Ensure that the computer resources planning effort is developed in conjunction with an acquisition logistics support plan and that software supportability and life cycle management are properly addressed during development (keeps others accountable for changes in metrics which affect whole life cycle costs).
Contracts, Legal, etc.	Software data rights, software metrics (contractor staff, size, cost, quality), EVM and WBS, IEEE/EIA Standard 12207 and CMMI compliance, RFP language, and SOW.

** may be combined based on ACAT or SIS/SOS role*

Table 4-1. Critical Roles and Responsibilities

4.6 Organizational Structure and Staffing Levels

Inadequate staffing, insufficient planning, little software experience, or poor communications within the ranks of critical billets have been identified as leading indicators of impending failure in the past, but with

SIS and SISOS the impact of these problems is magnified considerably and felt sooner. To help alleviate these problems, an organizational structure is recommended for an SIS or SISOS program office that reflects the following:

- Fulfillment of the critical roles and responsibilities table above;
- The need for a program/project software architect; (Hint: can also fulfill overarching integration leader.)
- The need for a cross-IPT Software Leader reporting directly to the Program Manager or Chief Engineer;
- The use of organic, or at least co-located, staff whenever possible; (Hint: require in contract that lead vendor house Program Manager's Office (PMO), critical vendor and subcontractor representatives in one location at their cost or have virtual teleconference meetings daily to compensate. Travel budget will otherwise be very large.)
- The need for dedicated risk management of software, *within* the system, and *within* the SIS or SOS; (Hint: assign an overarching SOS lead or integrator.)
- The need for a program training budget to keep personnel current in the emerging management and technical aspects of SIS and SISOS development.*

*Note that the DoN CAO, while responsible for providing requested resources to a program, needs specific guidance on KSAs and experience required but cannot, under current resource and budget constraints and changeover to CAO, be assumed to have an asset specifically trained for each application or meet staff churn rates in some phase transitions.

4.6.1 Functional Assignment Matrix

Programs are usually funded at levels that inhibit full staffing of all the disciplines shown in Figure 4-3, even with an experienced and knowledgeable Program Executive Office (PEO). This contributes to program risk and transfers software risk, causal factors, and responsibilities to others who may not necessarily be appropriately trained and experienced for specific critical software management roles. Particularly, they may be deficient in knowledge of CMMI® or equivalent maturity models or lack expertise in implementing Institute of Electrical & Electronics Engineers/Electronic Industries Alliance (IEEE/EIA) Standard 12207 or equivalent standards. Without the necessary understanding of the principles, it is hard to identify a risk when a person has inadequate knowledge of that area (often referred to as "you don't know what you don't know").

A Functional Assignment Matrix is contained in Appendix I, and a sample portion of this matrix is shown below in Figure 4-6. This matrix lists each of the core DoN acquisition disciplines, the critical billet/positions most often associated with each of these disciplines, and the primary functions performed by each position. The *Program-wide staff positions* shown in Figure 4-5 and in Table 4-1 are included in this matrix. This matrix may be used to assess staff coverage, especially in areas that are critical to program success, and to help in assessing overall organization risk exposure at program initiation and thereafter. Tailoring of this matrix to accommodate a particular acquisition strategy and to best allocate functional responsibilities across available personnel should be performed by each program prior to use and recorded in the Systems Engineering Plan (SEP), integrated schedules, and SDP and assessed at Systems Engineering Technical Reviews (SETR) and milestone reviews. While the roles may be tailored or realigned, the "x"'s represent the minimum critical functions to be accomplished by the program office.

DISCIPLINE ➔	PROGRAM MANAGEMENT				SYSTEMS ENGINEERING				SOFTWARE ENGINEERING				LOGISTICS		TEST & EVALUATION			CONTRACTS			LEGAL	
NOMINAL POSITION TITLE ➔ / FUNCTION 🡇	Program Manager	Risk Management Manager	Project (Sub-system) Manager	Process Compliance Manager	Requirements Manager	Chief Systems Engineer	Systems Process Manager	Sub-System Lead Engineer	Software Architect	Chief Software Engineer	Software Process Manager	Sub-System Software Engineer	Logistics Manager		Lead T&E Engineer	DT Lead Engineer	OT Lead Engineer	Procurement Contracting Officer	Administrative Contracting Officer	In-Plant Program Rep or DCMAO	Counsel	
Planning																						
Task Planning (WBS, SMP/PP)	X		X																			
Acquisition Planning	X												X					X				
Progress/Earned Value: Planning			X			X		X				X						X				
Acquisition Plan	X																					
All Test Planning															X							
Communications						X																
Cost estimating			X																			
Develop Software Development Plan												X										
Develop the SEP						X																
Ensure tech reviews scheduled and executed				X																		
Establish Data Repository				X																		
File Nomenclature				X																		
IBR lead			X																			
Identify and obtain in-house and CSS support	X																					
Identify required metrics				X																		
IPT Organizational Structure								X														
Master schedule	X																					
Monitoring and Controlling the Plan				X																		
Overall Cost	X																					
Overall Schedule	X																					

Figure 4-6. Functional Assignment Matrix (partial)

4.6.2 Tailoring the Organization's Staffing Levels

Organizational tailoring is based on adapting the recommendations from the preceding sections to particular programs. The types of positions and responsibilities cited are consistent with the development of a major platform, such as an aircraft, after Milestone B. Clearly, other types of programs at other stages of acquisition must adjust for the specific needs of their programs, including some of the following program unique considerations.

4.6.2.1 Program Unique Considerations

Each program is different, and program managers must make many critical decisions about what is most important to program success. Inadequate or delayed staffing increases risk and directly affects the program manager's ability to execute the acquisition program. Decisions regarding program office staffing, as well as vendor staffing, are driven by the acquisition strategy, funding, expertise of in-house resources, availability of support contractors, the level of insight/oversight required, etc. Other factors to consider when tailoring staffing levels and identifying critical positions are discussed below. Initial tailoring decisions should be clearly documented and disseminated in the Acquisition Plan or the SEP. A roadmap identifying staffing needs for each development phase should be produced to assure staff is in place when required. This roadmap should be consistent between program office and developer.

- Funding issues: As an acquisition progresses, risks arise, funding profiles often change, priorities are re-evaluated, and precious staffing resources may need to be changed or moved to critical areas. These staffing-level choices should be made based on a pre-determined methodology by a PM or consistently assigned person. That is, the program should have a plan (based on risk mitigation or other criteria) on how, using a consistent method, it will evaluate and decide to change staffing levels during the acquisition process.

- Software Experience: Key billets must be staffed with individuals having software experience (and training) as indicated by the *Program-wide staff positions* that are shown in Figure 4-5 and described in Table 4-1. For smaller programs, a single individual may be able to perform multiple functions if the increased risk is documented and approved by the PM.
- IEEE/EIA Standard 12207 Considerations: Compliance with IEEE/EIA Standard 12207 is required for DoN software acquisitions.[9] It provides a common software lifecycle framework for both acquiring and supplying software products and services, and was developed with the acquisition of software in mind. The processes described in IEEE/EIA Standard 12207.0 and the life cycle data described in IEEE/EIA Standard 12207.1 should be tailored by the acquisition office and/or the supplier in determining staffing levels to comply with these processes.
- Program Phase: Software development has its greatest activity after hardware/software allocation and prior to production. Afterward, software requirements and maintenance may have significant activity if there are either problems or requirements changes/upgrades. When a program is either in concept definition or in production, the software activities tend to ebb and the roles associated with software management are reduced.
- Role of Government: Congress has recently promoted the concept of the government program as the Lead Systems Integrator (LSI) of contractor-produced products. The LSI responsibility adds roles and requires stricter and expanded contracts to the list of activities shown in Figure 4-3.

4.7 Software Acquisition Training

Training is often under-utilized for a variety of reasons. For SIS or SISOS, critical training requirements tailored to the roles and responsibilities of team members and focused on software issues relevant to large acquisitions are essential. Some program personnel need to be experts in specific areas, whereas others, depending on their role, may only need cursory knowledge on a particular subject. The SPII HR report contains an extensive list of existing software knowledge and skills training sources that are presently available, and notes that personnel "experience" is a prerequisite for SIS ACAT level programs. The report can be accessed at: http://acquisition.navy.mil/organizations/dasns/rda_cheng.

With the rapid advances in technology and methods, it is a fact that any training plan will soon become out of date within as little as two years and additional training needs will continue to be added rather than deleted. The training plan and course content both need to be kept up to date. Program Managers should ensure staff compliance with Defense Acquisition Workforce Improvement Act (DAWIA) certification levels as directed by ASN(RD&A).[10] Program and Competency Managers need to plan for and set aside sufficient funds to meet training needs, and refresh these periodically to keep the technical workforce current.

[9] Department of the Navy. ASN(RD&A) memo. Software Process Improvement Initiative Contract Language. 17 November 2006.
[10] Department of the Navy. ASN(RD&A) memo. Policy for Acquisition of Naval Software Intensive Systems. 16 September 2008.

4.8 Recommendations

From a staffing and training perspective, key recommendations for Program Managers are as follows:

- Recognize that SIS and SISOS are different from legacy systems and will stretch even sufficiently experienced critical staff to address software domain issues. Plan, add, or train accordingly.
- Develop staffing plans early, synchronized to the program's life cycle phases, to ensure that funding and staff are available when they are needed. This includes resources for development and training of program office and support personnel.
- Staff the critical *Program-wide staff positions* (see Figure 4-5) that have *Program-wide responsibilities* (see Table 4-1) identified in this report with individuals having relevant software domain experience. If relevant experience is not available, train the individual *before* placing them in the position. Staffing late or staffing poorly have the same detrimental effect.
- Recognize that undercutting software staffing needs, for whatever reason, is a false economy and is done at the risk of the program.
- Use the Functional Assignment Matrix (see Appendix I) as a guide in assessing organizational structure, software expertise, and staffing requirements for SIS and SISOS programs and "tailor-down" accordingly for less demanding acquisitions.

References

Boehm, Barry. "Software Engineering: What Have We Learned? Where Are We Going?" NGC Tutorial. 16 July 2007.

Boehm, Barry. "Unifying Software Engineering and Systems Engineering." IEEE Computer, 33 no. 3: 114-116.

Boehm, Barry, A. Windsor Brown, Victor Basili, and Richard Turner. "Spiral Acquisition of Software-Intensive Systems of Systems." CrossTalk:The Journal of Defense Software Engineering. May 2004.

Boehm, Barry, and Jo Anne Lane. "Using the Incremental Commitment Model to Integrate System Acquisition, Systems Engineering, and Software Engineering." CrossTalk: The Journal of Defense Software Engineering. October 2007:4-9.

Department of Defense. Risk Management Guide for DoD Acquisition, Sixth Edition (Version 1.0). August 2006.

Department of the Navy. Information Technology Applications and Data Management. SECNAV 5000.36A. 19 December 2005.

Department of the Navy. ASN(RD&A) memo. Policy for Acquisition of Naval Software Intensive Systems. 16 September 2008.

Department of the Navy. ASN(RD&A) memo. Software Process Improvement Initiative. 15 May 2006.

Department of the Navy. ASN(RD&A) memo. Software Process Improvement Initiative Contract Language. 17 November 2006.

Department of the Navy. ASN(RD&A) Software Process Improvement Initiative (SPII) Human Resource Report, 6 November 2007; report can be accessed at: http://acquisition.navy.mil/organizations/dasns/rda_cheng.

Department of the Navy. ASN(RD&A) Software Process Improvement Initiative. Software Acquisition Management Focus Team. "As Is State" Report. 17 May 2007.

Department of the Navy. ASN(RD&A) Software Process Improvement Initiative. Software Acquisition Management Focus Team. "To Be State" Report.

Department of the Navy. Naval Air Systems Command. Memo from Director, Software Engineering. Increasing Software Domain Influence During Acquisition of Naval Aviation Systems. (undated).

Ferguson, Jack. "New CMM Math." 2nd Annual CMMI Technology Conference and User Group. DTIC. 14 Nov. 2002.

Hantos, Peter. “Risk Management in System of Systems Development - Are You Ready?” 20th International Forum on COCOMO and Cost Modelling. Center for Software Engineering, USC. University Of Southern California, Los Angeles. 26 Oct. 2005.

“Information Systems Division (ISD) Software Training Plan.” NASA. 15 Apr. 2005, Version 1.1.

Myers, Charles, Jr.. “The Human Element in Technology Adoption.” CrossTalk: The Journal of Defense Software Engineering. December (1997).

National Defense Industrial Association. (2006). Systems Engineering Division Task Group Report: Top Software Engineering Issues Within Department Of Defense And Defense Industry.

Rozanski, Nick, and Eóin Woods. Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives. New York: Addison-Wesley Professional, 2005.

This Page Intentionally Left Blank

5 Software Development Technique Selection

5.0 Overview

Software development techniques can have considerable impact on program schedule and product quality. Sufficient guidance exists that describes and compares the waterfall and evolutionary approaches or development processes,[1] but little is available for assessing utility of software development methods and tools. These are addressed in this chapter of the guidebook and its related appendices, with guidance intended to assist acquisition professionals and the developer community during both pre-solicitation and source selection acquisition phases.

The definition for "development techniques" was derived from Barry Boehm.[2] He describes processes as defining the order and exit criteria of the stages of development while methods and tools are the development techniques, which are applied within those stages, and often produce specific products. This definition highlights the fact that software development techniques differ from concept refinement to maintenance and a wide array of techniques (methods and tools) are available.

The improper, or unskilled, application of tools and methods can lead to schedule slips, cost overruns, poor quality, and customer dissatisfaction. Some tools produce non-standard outputs and thus limit the ability to distribute products for review by Subject Matter Experts (SMEs) who may be dispersed across organizations and may not have licenses for the tool. Methods and tools may incorporate automation, increasing speed to code, but also increasing the time to conduct fault isolation and implement corrections, affecting both schedule and quality. This is why the mandatory Software Development Plan (SDP) must include both the processes and the software development techniques, realizing that processes can be a subset of the overall development technique but semantics amongst practitioners is not always so precise.

5.1 Understanding Existing Software Development Techniques

Techniques include processes, methods, and tools. Many methods and tools that support different phases of development can be reorganized into different sequences. Tools are summarized in Appendix J, noting strengths and weaknesses. Methods and techniques evolve, some rapidly, so the appendix captures just a snapshot of today's market and practices, and is most useful in comparing suggested utility. Some techniques involve considerable cost and learning curve, and thus are most appropriate for larger programs with longer schedules. Note, however, that the longer program will also suffer random, sometimes significant, upgrades to techniques and toolsets which will directly affect productivity. Others improve speed and accuracy but significantly increase man-hours. The benefits of these techniques must be evaluated against the program risks, constraints, cost, and schedule.

[1] Pfleeger, Shari Lawrence. Software Engineering: Theory and Practice. Upper Saddle River, NJ: Prentice Hall, 1998.
[2] Boehm, Barry. "A Spiral Model of Software Development and Enhancement." IEEE Computer. Vol. 21, May 1988: 61-72.

Software development techniques that are currently popular were assessed at the time of this publication. The range of these techniques is illustrated in Tables J-1 and J-2, in Appendix J. Tables include the name of the technique, a definition, basic features, advantages and disadvantages. Table J-1 contains predictive software development techniques appropriate where requirements are understood and Table J-2 contains adaptive software development techniques appropriate when requirements are not initially well-understood.

For some programs, multiple techniques may be effectively combined. When multiple candidate development techniques are identified, other discriminatory features can be added to the tables such as performance, schedule, cost, supportability, security, and safety. These features can be used as evaluation criterion as described in the best practices section of the Data Analysis Center for Software.[3] Programs should employ software development SMEs in the assessment of proposals. Those SMEs should use Appendix K or similar checklists to assess proposed development techniques and process sequence, including combinations.

5.1.1 Predictive Software Techniques

Predictive techniques, also known as plan driven or disciplined methodologies, focus on planning the future in detail. A predictive team can report exactly what features and tasks are planned for the entire length of the development process. Predictive software development techniques are usually the most appropriate/logical choice when requirements are mature and stable. Predictive software techniques are not recommended when requirements are poorly understood, as the software estimates that are used to establish the Acquisition Program Baseline schedule and costs will be invalid, and will ultimately cause a baseline breach.

Predictive teams have difficulty changing direction. The plan is typically optimized for the original objectives. Predictive teams will attempt to limit change by modifying the charter of their software configuration change boards to strongly filter out any but “must-have” changes.

Predictive techniques are appropriately applied in waterfall development efforts wherein the phases of requirements, design, development, test, and integration are sequential and the outcome of each phase is deterministic. Predictive techniques are not suitable for programs where requirements must be discovered or refined, including incremental development (where requirements for the next increment are not yet known), and spiral development, or any hybrid of those. Table J-1 in Appendix J describes a few popular predictive techniques.

5.1.2 Adaptive Software Development Techniques

Adaptive methods are employed where requirements are not well understood and are designed to accommodate change. These techniques are characterized by iteration and are applied in spiral, incremental, or hybrid spiral/incremental development efforts. Adaptation can occur between the iteration cycles, and the customer is often involved in determining exit criteria and objectives for the next iteration.

Incremental and spiral or hybrid approaches are also used when technology is still maturing, when funding is staggered, and/or to manage complexity. Table J-2 in Appendix J describes a few popular adaptive techniques.

[3] The Data Analysis Center for Software. http://www.dacs.dtic.mil.

5.2 Framework for Evaluating Emerging Software Techniques

It is essential that program managers use a methodology for assessing suitability of emerging techniques, specifically software development methods and supporting tools. While the following is highly recommended in guiding this assessment, programs may select another similar methodology, but one methodology should be selected and used throughout the life cycle to first select and thereafter assess updates and changes.

5.2.1 Evaluating Software Development Methods

New software development methods are frequently promoted in software and management journals. Evaluation of the effectiveness of these methods can be difficult given the lack of substantial data. A process is provided here to rate the individual features of a candidate method for purposes of developing a weighted score reflecting a single rating for use in comparing relative merits of different methods, given program and organization objectives and constraints.

The process is as follows:

1. Use an Electrical and Electronic Engineers/Electronic Industries Alliance (IEEE/EIA) Standard 12207 framework to list the salient features useful in guiding the development process to where it needs to be, and then provide an accountability mapping to the elements of the framework. The features are selected to define principal components. An example is illustrated in Table 5-1.
2. Attach a weighted variable to each selected feature corresponding to relative importance. During evaluation of the feature a number or a color scheme (i.e. green, amber, red) are equally effective. The identification of important features and their relative importance can be accomplished by the Integrated Product Team (IPT) consisting of stakeholders in the life-cycle processes. In accordance with acquisition policy the relative importance of each category is described in the Request For Proposal (RFP) and evaluation criteria.
3. In a working group or integrated product team environment, synthesize detailed questions about the emergent software development technique under assessment.
4. Attach a scoring scheme to each question, then devise a method to total the score under each salient feature.
5. Set the relative values of the weight variables using input from the working group. Some normalization factor may be required. Document this in the Source Selection criteria.
6. Sum all the output scores from each branch in the process as depicted in Figure 5-1. The ‘score(s)’ can be used for relative comparison of different techniques. Where color scheme only is used at evaluation, comparison occurs across similar features. Otherwise the linear model shown accommodates an output score.

Key Features For Emergent Software Development Technique Assessment Process		
FEATURE	MAPPING TO IEEE/EIA STANDARD 12207	WEIGHTED VARIABLE
Business Practices	Primary	A
Requirements Management	Primary	B
Assessment and Oversight	Organizational	C
Training	Organizational	D
Configuration Management	Supporting	E
Quality Assurance	Supporting	F

Table 5-1. Key Features for Emergent Software Development Technique Assessment Process

Questions developed by the working group or IPT will be tailored for the specific project, and may cover some or all of the areas in Table 5-1. Example questions for each of the areas are provided here:

- Business practices (Business Implications)
  - Does the software development method support existing business practices (programmatic performance tracking, deliverables, and reviews)?
  - Is the new method supported with existing or mature cost models for estimation?
  - Has the new method been used in a similar application for a similar Program Executive Office/Program Manager (PEO/PM)?
  - If the software will be active on a network, do business joint ventures and partnerships, especially contracts, support cyber requirements, supplier assurance, information assurance, anti-tamper, and safety where applicable?
- Requirements Management (Software Engineering)
  - Does the method support open architecture objectives including modularity and interface standards?
  - Is the method appropriate given the level of requirements stability?
  - Is the method supported with existing or affordable tools?
  - Does the method provide standard, portable products?
  - Does the method have a technique to highlight or tag certain requirements for mission critical, safety-critical, security/information assurance, or anti-tamper priorities?
- Assessment and Oversight (Acquisition Management)
  - Does the method support objectives for the current acquisition phase of the program?
  - Is the learning curve for the new technique and supporting tools acceptable?
  - Does the method support program metrics?
- Training (Human Resources)
  - Are developers experienced in similar methods?
  - Is training available and affordable?
  - Will training be useful in other programs or projects within the organization?
  - Will training contribute to employee retention?

- Configuration Management (Development)
  - Is the method compatible with configuration management tools already in place at the vendor and/or acquirer?
  - Are associated tools and documents available and adequately configuration managed?
- Quality Assurance (Development)
  - Does the method support program review and verification strategies?
  - Does the method support measurable quality goals (i.e. requirements volatility, software trouble report distribution, cost and schedule variances, defect tracking, etc.)? Note that naval policy requires that programs regularly provide metric data on defects up the chain of command.
  - Does the method support defect prevention?

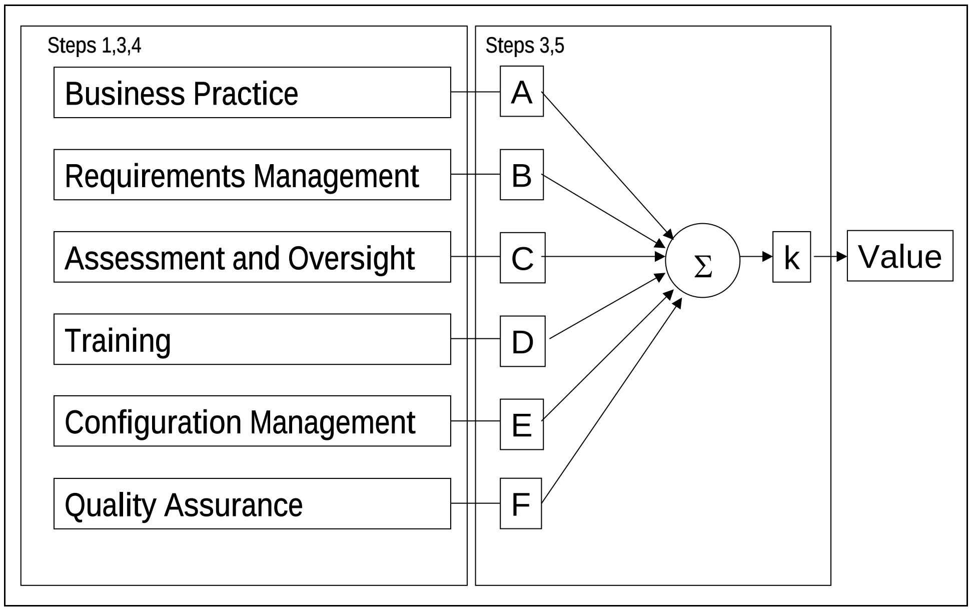


Figure 5-1. Process Model Output

5.2.2 Evaluating Software Development Tools

System/software development tools can be effectively used to support each phase of the Software Development Lifecycle (SDLC). Tools can enhance productivity, improve configuration control, reduce errors and rework, and support common “situational awareness” of program products and progress. Tools can be applied throughout the SDLC for requirements refinement, requirements tracking, design development, design walk-through, user assessment, version control, change control, shared development environments, coding, code automation, testing and test automation, training, maintenance, and data collection for progress metric and quality control analyses.

Tools should be complete and consistent with the software development technique. That is, they should span the life cycle without requiring re-formatting or re-entry of data phase-to-phase, outputs must be open (readable by all IPTs) or have affordable licenses, etc. The obsolescence of Commercial-off-the-Shelf (COTS) tool sets or the vendors themselves should also be addressed as these have historically greatly diminished the sustainment of naval products and led to great loss.

5.2.2.1 Acquirer Responsibility in New Programs

Acquisition reform has placed the emphasis on performance-based contracting and incentives. Contractors are generally not directed to employ specific tools. Still, the acquirer will be interested in minimizing the risk that will be applied by the program with respect to inconsistent or inappropriate tools. When program staffs write RFPs and evaluate proposals, the following suggestions can minimize risk:

- Evaluate whether the provider has achieved or demonstrated mature software processes. Capability Maturity Model Integration (CMMI®) Maturity Level 3 (or higher) processes, or an equivalent standardized software process assessment methodology, indicate a mature development process and are generally sufficient to indicate provider capability in tool selection and application.
- If specific tools are to be proposed*, evaluation criteria could include:
  - Provider experience with the tools, for learning curve and schedule considerations.
  - Whether tools produce standard output formats (to minimize dependency on specific tools and to support automated data transfer between tools).
  - Is the license cost reasonable and supportable?
  - Will maintenance costs be reasonable?
  - Provider skills and impact of learning curve on schedule.

**Note: To facilitate this assessment the proposed SDP included in the response should address whatever tools the offeror intends to use, as described in the data item description example in Appendix L.*

5.2.2.2 Acquirer Responsibility in Existing Programs

Even when a program is well underway, the acquirer may be involved in decisions to implement new or different tools, particularly if the new tool(s) will need to interoperate or exchange data with other tools in use by the developer or within the program office. For example, if the developer decides to implement new configuration management tools, it would benefit both the program office and the developer if the configuration tool could import data from the requirements management tool (which may be operated by another contractor for the program office). Appendix K provides a detailed process for evaluating tools. Each tool type is described in terms of what, who, and why, and a short list of questions is provided under each to assist in determining suitability. The following tool types are addressed in Appendix K:

- Configuration Management;
- Requirements Definition and Management;
- Systems Analysis and Modeling;
- Development;
- Defect Tracking;
- Source Code (Security) Analysis; and
- Testing.

References

Boehm, Barry. "A Spiral Model of Software Development and Enhancement." IEEE Computer. Vol. 21, May 1988: 61-72.

The Data Analysis Center for Software. <http://www.dacs.dtic.mil>.

Davis, Noopur, and Julia Mullaney. The Team Software Process^{SM} (TSP^{SM}) in Practice: A Summary of Recent Results. CMU/SEI-2003-TR-014. Pittsburgh: Carnegie Mellon University, Software Engineering Institute, 2003.

IEEE/EIA 12207, "IEEE standard for Industry Implementation of International Standard ISO/IEC 12207 Standard for Information Technology"

MIL-STD-1521B, "Technical Reviews and Audits for Systems, Equipments, and Computer Software", June 1985.

Net-centric Enterprise Solutions for Interoperability (NESI) web site. <http://nesipublic.spawar.navy.mil>.

Pfleeger, Shari Lawrence. Software Engineering: Theory and Practice. Upper Saddle River, NJ: Prentice Hall, 1998.

Pressman, Roger S., Software Engineering: A Practitioner's Approach, Sixth Edition. McGraw-Hill International Edition.

This Page Intentionally Left Blank

6 Acquisition Planning

6.0 Overview

The Department of Defense (DoD) acquisition process is controlled by a set of key documents, including DoD Directive 5000.1, “The Defense Acquisition System,” May 12, 2003, and DoD Instruction 5000.2, “Operation of the Defense Acquisition System,” May 12, 2003. These documents cover the overall flow of activities culminating in the acquisition of a system. The process by which the government performs “source selection” is covered in Federal Acquisition Regulations (FAR) Subpart 15.3 and Department of Defense and Defense Logistics Agency (DLA) supplements. This process covers informal pre-solicitation activities, formal solicitation of proposals, and evaluation of proposals. The Source Selection Authority (SSA) is responsible for defining the specific source selection process to be used for an acquisition.

The process involves a series of steps leading up to Milestone B and subsequently contract award after Milestone B. These steps include defining an Acquisition Strategy, an Acquisition Plan (AP), and a Source Selection Plan (SSP). This chapter does not to replace or explain these overarching directives. Rather, it highlights areas of acquisition planning with unique or significant software acquisition implications.

6.1 Acquisition Strategy

The Acquisition Strategy serves as the roadmap for program execution from program initiation through post-production support. The Defense Acquisition Guidebook (DAG)[1] describes several important considerations to apply when creating a strategy. These are listed in Table 6-1.

▪ Acquisition Approach	▪ Environment, Safety, Occupational Health
▪ Systems Engineering	▪ Relief, Exemption, and Waiver
▪ Best Practices	▪ Human Systems Integration
▪ Modular Open Systems Approach	▪ Research and Technology Protection
▪ Information Assurance	▪ Information Technology
▪ Capability Needs Summary	▪ Resource Management
▪ Business Considerations	▪ Integrated Test and Evaluation
▪ Product Support	▪ Risk Management
▪ Program Structure	▪ Interoperability

Table 6-1. DAG Acquisition Strategy Considerations

[1] Access to the Defense Acquisition Guidebook is at: https://akss.dau.mil/dag/.

All of the considerations in Table 6-1 have implications for defining software strategy and planning. For each specific acquisition, it is important to evaluate these considerations for applicability, and to ensure that appropriate plans are defined that address any risks and issues. During strategy formulation, the software Subject Matter Experts (SMEs) associated with an acquisition program need to engage the acquisition staff very early in the process to ensure that software development considerations are appropriately articulated and included in the strategy and associated plans.

6.2 Handling of Requirements

Development of a software intensive system should be driven by the needed capabilities and characteristics of the system. Depending on the strategy chosen by the program office, a system specification can be written by the government based on the Capability Development Document (CDD) (and Naval System Design Specifications (SDS) per SECNAVNOTE 5000.2) and provided to prospective contractors as a part of the Request For Proposal (RFP) package, or the responsibility of writing the system specification can be assigned to the contractor post-award. This latter approach is often associated with a Statement of Objectives (SOO) instead of an RFP. Regardless of the selected strategy, software SMEs need to be engaged from the initial steps in this process throughout the development life cycle to ensure that software concerns are appropriately addressed as the requirements are finalized.

In IEEE Standard 830-1998, *IEEE Recommended Practice for Software Requirements Specifications,* there is a list of nine quality attributes for requirements:

- Complete;
- Unambiguous;
- Correct;
- Consistent;
- Verifiable;
- Modifiable;
- Traceable;
- Ranked for importance; and
- Ranked for stability.

Paying attention to these is important to ensure that the requirements management process is conducted efficiently and effectively. Lessons-learned have shown that each facet above which is left out of the requirement can result in a cost change order later. The software SMEs assigned to render the requirements should fulfill these attributes for the program.

6.3 Implementation of IEEE/EIA Standard 12207

Assistant Secretary of the Navy (Research, Development and Acquisition) (ASN(RD&A)) policy mandates the use of Institute of Electrical & Electronics Engineers/Electronic Industries Alliance (IEEE/EIA) Standard 12207 (see Appendix B, Appendix C, and Appendix E). The purpose of applying this industry standard is to implement disciplined software development processes and provide a standard enterprise-wide framework within which both program offices and developers can determine and follow commonly understood processes throughout the acquisition life cycle. This standard also facilitates consistent measurement across all naval programs to allow ASN(RD&A) and Program Executive Offices (PEOs) to make programmatic comparisons. The purpose of this section is to assist Department of the Navy (DoN) program offices in their interpretation or use of IEEE/EIA Standard 12207 to conduct acquisitions that involve the development and/or maintenance of software, including the composing of COTS-based systems. Appendix M provides a more in-depth summary of the IEEE/EIA Standard 12207 processes and objectives.

6.3.1 Overview of IEEE/EIA Standard 12207

IEEE/EIA Standard 12207 is an international standard that "applies to the acquisition of systems and software products and services, to the supply, development, operation, and maintenance of software products, and to the software portion of firmware, whether performed internally or externally to an organization." The standard is organized into three volumes:

- IEEE/EIA 12207.0 - 1996. (International Organization for Standardization/International Electrotechnical Commission (ISO/IEC 12207)) Standard for Information Technology—Software life cycle processes;
- IEEE/EIA 12207.1 - 1997. (ISO/IEC 12207) Standard for Information Technology—Software life cycle processes—Life cycle data; and
- IEEE/EIA 12207.2 - 1997. (ISO/IEC 12207) Standard for Information Technology—Software life cycle processes—Implementation Considerations.

Volume 1 describes the core concepts, including the processes, activities, and tasks that span the system development life cycle. Volume 2 describes the different types of data items (documentation) that can result from applying the processes. Volume 3 provides some guidance of different approaches for applying the standard.

IEEE adopted the ISO/IEC Standard 12207 in 1995 for industrial use. The Department of Defense adopted the IEEE/EIA Standard 12207 in 1998.

This standard establishes a common framework for software life cycle processes, with well-defined terminology, that can be referenced by the software industry and acquirers of software intensive systems.

6.3.2 Definitions

- *Life Cycle Model*: In the context of the development, operation, and maintenance of a software product, a life cycle model is a defined set of processes, activities, and tasks, and their sequencing and interrelationships, spanning the life of the system from its definition to the termination of its use.
- *Process*: A set of interrelated activities designed to accomplish a specified goal. Table 6-2 lists all IEEE/EIA Standard 12207 processes and their associated activities. For example *Development* is a process. Within *Development* there are thirteen activities as shown in Table 6-2. One of these activities is *Software Coding and Testing* which has five tasks.
- *Activity*: A set of actions which, taken as a whole, transform inputs into outputs.

- *Tasks*: Specific actions performed to accomplish an activity. The way that each task is performed, such as testing, is called the *technique* or *method*.
- *Method/Technique*: The approach used to accomplish the task.

6.3.3 IEEE/EIA Standard 12207 Processes and Activities

IEEE/EIA Standard 12207.0 defines a set of processes, activities, and tasks that cover the system development life cycle. The DoN has adopted this standard because it provides a useful and flexible framework within which software acquisitions can be defined. Also, by employing this standard, terminology can be common across programs and organizations (government and contractor).

IEEE/EIA Standard 12207 divides the processes into three categories: primary, supporting, and organizational. These are listed in Tables 6-2, 6-3, and 6-4.

IEEE/EIA Standard 12207.0 Primary Processes and Activities	
Process	*Activities*
Acquisition	▪ Initiation ▪ Request-for-proposal [-tender] preparation ▪ Contract preparation and update ▪ Supplier monitoring ▪ Acceptance and completion
Supply	▪ Initiation ▪ Preparation of response ▪ Contract ▪ Planning ▪ Execution and control ▪ Review and evaluation ▪ Delivery and completion
Development	▪ Process implementation ▪ System requirements analysis ▪ System architectural design ▪ Software requirements analysis ▪ Software architectural design ▪ Software detailed design ▪ Software coding and testing ▪ Software integration ▪ Software qualification testing ▪ System integration ▪ System qualification testing ▪ Software installation ▪ Software acceptance support
Operation	▪ Process implementation ▪ Operational testing ▪ System operation ▪ User support
Maintenance	▪ Process implementation ▪ Problem and modification analysis ▪ Modification implementation ▪ Maintenance review/acceptance ▪ Migration ▪ Software retirement

Table 6-2. IEEE/EIA Standard 12207.0 Primary Processes and Activities

IEEE/EIA Standard 12207.0 Supporting Processes and Activities	
Process	*Activities*
Documentation process	▪ Process implementation ▪ Design and development ▪ Production ▪ Maintenance
Configuration management process	▪ Process implementation ▪ Configuration identification ▪ Configuration control ▪ Configuration status accounting ▪ Configuration evaluation ▪ Release management and delivery
Quality assurance process	▪ Process implementation ▪ Product assurance ▪ Process assurance ▪ Assurance of quality systems
Verification process	▪ Process implementation ▪ Verification - Contract verification - Process verification - Requirements verification - Design verification - Code verification - Integration verification - Documentation verification
Validation process	▪ Process implementation ▪ Validation
Joint review process	▪ Process implementation ▪ Project management reviews ▪ Technical reviews
Audit process	▪ Process implementation ▪ Audit
Problem resolution process	▪ Process implementation ▪ Problem resolution

Table 6-3. IEEE/EIA Standard 12207.0 Supporting Processes and Activities

IEEE/EIA Standard 12207.0 Organizational Life Cycle Processes and Activities	
Process	**Activities**
Management Process	▪ Initiation and scope definition ▪ Planning ▪ Execution and control ▪ Review and evaluation ▪ Closure
Infrastructure Process	▪ Process implementation ▪ Establishment of the infrastructure ▪ Maintenance of the infrastructure
Improvement Process	▪ Process establishment ▪ Process assessment ▪ Process improvement
Training Process	▪ Process implementation ▪ Training material development ▪ Training plan implementation

Table 6-4. IEEE/EIA Standard 12207.0 Organizational Life Cycle Processes and Activities

6.3.4 General Strategy for Applying 12207

The Navy's overall strategy is to use IEEE/EIA Standard 12207 as the framework for defining, characterizing, and evaluating the approach to be followed for managing software activities. According to the November 06 contract language policy memo (see Appendix B), offerors are required to submit a draft version of their Software Development Plan (SDP) as part of their proposal package. This SDP is to follow the framework defined by IEEE/EIA Standard 12207, including its processes, activities, and tasks, or provide and maintain a cross-reference matrix showing coverage of IEEE/EIA Standard 12207 by their SDP.

Modern software development efforts generally follow an incremental approach. In incremental development, the system is developed in a series of builds, with each build being enhanced from the previous one and taking advantage of lessons-learned during the previous increments. This stepwise approach allows the developers to break down the development into a series of manageable steps, and thereby reduces the overall level of risk. For systems using an incremental development approach, some of the intermediate increments may be deployable, and can be provided to the user community for early utilization, thereby gaining significant utility and user feedback.

After contract award, the winning contractor(s) will complete the SDP and submit it as a Contract Data Requirements List (CDRL) for government approval. The content and format for the SDP may be specified in the Statement of Work (SOW) or SOO, or may be specified in an SDP Data Item Description (DID). A sample SDP DID and implementations guidelines are provided in Appendix L. The SDP will be updated whenever the development processes, tools, or organization change or are improved, or as mutually agreed with the government.

Government program offices also need to follow the IEEE/EIA Standard 12207 framework as they define their acquisition plans. The relevant IEEE/EIA Standard 12207 process is the Acquisition Process, described in section 5.1 of the standard (IEEE/EIA Standard 12207.0). This process includes five activities:

- Initiation;
- RFP preparation;
- Contract preparation and update;
- Supplier monitoring; and
- Acceptance and completion.

During the acquisition planning stage, activities two and three need to be addressed with the resulting plans documented.

Because not all projects need to apply all of the processes, activities, and tasks that are defined by IEEE/EIA Standard 12207.0, projects must evaluate the expected work effort and select those that are relevant. For example, if a program involves only development, with deployment covered under a different contractual vehicle, then the activities relating to the Operation Process may be tailored out of the required SDP content. Likewise, if the government has a particular interest in requiring that certain activities be performed in a specific way, using preferred techniques and/or tools, then these may be defined in the RFP to tailor the SDP as a part of this language. Program offices are responsible for specifying the minimum set of required processes, activities, and tasks that constitute the contractor's compliance. Programs may selectively add unique processes, activities, and tasks as appropriate based on program needs and unique characteristics as long as they are provided in the RFP, or by change order later. Regardless of the acquisition, program offices should not tailor out the IEEE/EIA Standard 12207 Improvement Process, as it is a core element of program success. As a part of gaining approval for the Acquisition Plan/Strategy, any processes that are tailored out of the SDP need to be justified and approved.

Compliance with IEEE/EIA Standard 12207 is defined as the performance of all the processes, activities, and tasks selected from IEEE/EIA Standard 12207 by the RFP or the Tailoring Process for the software project. The performance of a process or an activity is complete when all its required tasks are performed in accordance with the pre-established criteria and the requirements specified in the contract as applicable.

There is no intention to change the internal processes used by the offerors for their software policies and practices. However, for the proposal, the offerors will map their approach to IEEE/EIA Standard 12207 to facilitate the government's ability to understand, evaluate, and monitor development activities. In effect, this will normalize each approach, and will separate the actual plans from any unique descriptions applied to each process. For example, should two offerors propose to follow an "agile" development process, having them map their activities to the IEEE/EIA Standard 12207 framework will allow the government to identify and objectively evaluate any actual differences in their plans.

Again, this tailoring is to be accomplished by describing, in the RFP package, those activities and processes that are relevant to the work effort and that must be addressed in the offeror's SDP. This tailoring can be based on Tables 6-2, 6-3 and 6-4. All requirements relating to selected processes should be outlined in the Acquisition Plan and the Source Selection Plan, to ensure that the Source Selection process appropriately applies them to the proposal evaluation process.

6.4 SETR Guidance

Acquisition excellence has changed the way the DoD designs, develops, manufactures, and supports systems. DoD technical, business, and management approaches for acquiring and operating systems have, and continue to, evolve. For example, DoD no longer relies on military specifications and standards to define and control how developers design, build, test, and support new systems. Today DoD uses commercial hardware and software, promotes open systems architecture, and encourages streamlining processes, just to name a few of the initiatives that affect the way DoD does business. At the same time, the Office of the Secretary of Defense (OSD) has reduced the level of oversight and review of programs and manufacturers' plants, while requiring these commercial products safely operate weapons systems and remain secure even when connected to the global internet.

While the new acquisition model gives government program managers and their contractors broader control and more options than they have enjoyed in the past, it also exposes them to new risks. To aid in the management of this risk, the naval enterprise is harmonizing a universal System Engineering Technical Review (SETR) process (see Figure 6-1) that offers a full spectrum of reviews throughout the lifecycle of a product. Full details on the SETR Process can be found at http://www.navair.navy.mil/serc/index.cfm. Until this new process is completed and established as policy, PMOs should rely on their own review process as defined by their SYSCOM.

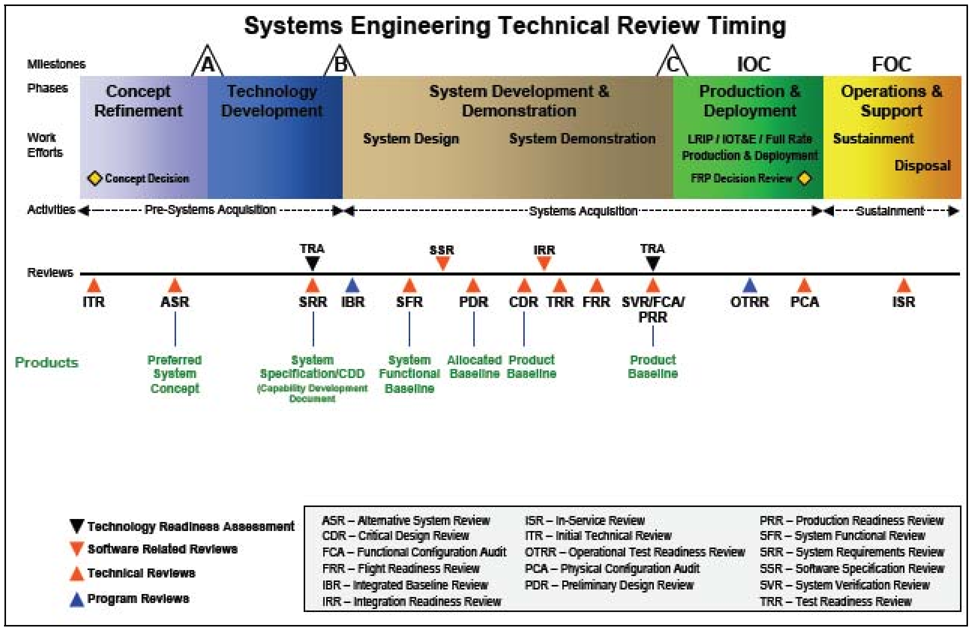

Systems Engineering Technical Review Timing
Milestones
A
B
C
IOC
FOC
Phases
Concept Refinement
Technology Development
System Development & Demonstration
Production & Deployment
Operations & Support
Work Efforts
System Design
System Demonstration
LRIP / IOT&E / Full Rate Production & Deployment
Sustainment
Disposal
Concept Decision
FRP Decision Review
Activities
Pre-Systems Acquisition
Systems Acquisition
Sustainment
Reviews
TRA
SSR
IRR
TRA
ITR
ASR
SRR
IBR
SFR
PDR
CDR
TRR
FRR
SVR/FCA/ PRR
OTRR
PCA
ISR
Products
Preferred System Concept
System Specification/CDD (Capability Development Document
System Functional Baseline
Allocated Baseline
Product Baseline
Product Baseline
Technology Readiness Assessment
Software Related Reviews
Technical Reviews
Program Reviews
ASR – Alternative System Review
CDR – Critical Design Review
FCA – Functional Configuration Audit
FRR – Flight Readiness Review
IBR – Integrated Baseline Review
IRR – Integration Readiness Review
ISR – In-Service Review
ITR – Initial Technical Review
OTRR – Operational Test Readiness Review
PCA – Physical Configuration Audit
PDR – Preliminary Design Review
PRR – Production Readiness Review
SFR – System Functional Review
SRR – System Requirements Review
SSR – Software Specification Review
SVR – System Verification Review
TRR – Test Readiness Review

Figure 6-1. Systems Engineering Technical Review Timeline

The SETR process is comprised of best practices collected over many years of program execution and was acquired from various sources. While the SETR process addresses the full spectrum of systems engineering disciplines, the electronic version of the checklists provides, for each review, a sort capability (see the website referenced in the preceding paragraph). This capability allows the user to easily focus on those entry and exit criteria associated with a specific technical discipline (e.g., logistics, software, training, etc.). In particular, the software manager can use this tool to support the program manager in selecting the appropriate questions that must be answered to review software development and integration.

ASN(RD&A) recognizes that there is programmatic risk (addressed via SECNAVNOTE 5000.2 and the two pass/six gate process) and technical risk (addressed by the SETR). It is always true that risk is inherent in any acquisition program and ASN(RD&A) considers it essential that program managers take appropriate steps to manage and control all risks. The SETR is the risk assessment from the technical community. The SETR process is a useful risk mitigation tool because it can be used to verify the following:

- Requirements for the acquired system are defined (including granularity to address software-specific requirements);
- Requirements are transformed into an effective system (including effective software components and subsystems);
- The processes are consistent and repeatable for the entire lifecycle, where necessary;
- The provider/supplier uses a systematic approach in providing the required products and services;
- The test resources (personnel, facilities, test assets, test plans, ranges, etc.) are available and the product is ready for test;
- The product is ready for Operational Evaluation (OPEVAL) (validation environment, Operational Test & Evaluation Force (OPTEVFOR) personnel, ranges, etc.);
- Services/products are sustainable throughout the lifecycle (including an adequate software support activity); and
- Systems are properly disposed of when they are retired from service.

SETR reviews are meant to be tailored to the schedules, goals, and objectives of individual programs. Each program manager, working with their in-house acquisition and technical management team and the developer/supplier, should review the SETR process against their own program plan and determine the applicability of each review and each element within each review to their own program. Completed reviews should become program of record milestones and amended to an updated program baseline schedule. Execution of the SETR process and a disciplined approach in the correction of deficiencies will lower the overall program risk.

The SETR assessments are only a snapshot; the software metrics defined in this guidebook as well as additional metrics appropriate to each program should be required of the developer and integrated with the SETR process. Metrics are collected and used between assessments to continuously detect and mitigate risk. Again the program manager with their management team should tailor the metrics to the specific information needs of their program. Design agents should use metrics to manage software development. During the review process, these metrics will be used to validate the status of the program when utilizing the SETR process.

The SETR process is under review for modification and adoption enterprise-wide; it is offered here as a model for a successful risk management process.

References

Defense Acquisition University. Defense Acquisition Guidebook. <https://akss.dau.mil/dag/>.

This Page Intentionally Left Blank

7 Contract Solicitation

7.0 Overview

This chapter provides guidance for organizations planning to conduct a competitive procurement for a software system or systems with major software components. Current Department of Defense (DoD) and Department of the Navy (DoN) guidance interprets the terms “software intensive system” or “systems of systems” to mean nearly all systems that have any software content. This chapter describes the steps needed to prepare for the solicitation and the procurement materials. The assumption is that the steps described in Chapter 6, Acquisition Planning, have been implemented, including the preparation of an Acquisition Strategy. Recognizing that overall there is ample guidance and policy for acquisition planning, this chapter will not duplicate that guidance but rather emphasize those areas of importance for acquisition of software systems.

During solicitation planning, it is particularly important that software Subject Matter Experts (SMEs) work closely with the Program Manager (PM) and the rest of the acquisition team to ensure that the appropriate considerations for software are included in the planning for acquisition and source selection. The types of critical SME knowledge and experience are discussed in Chapter 4. During this preparation for solicitation period, two important documents to be developed (among others) that are influential on software acquisition success are: first, the Source Selection Plan (SSP), and then the Request for Proposal (RFP).

Chapter 5 discussion of software development techniques also includes solicitation guidance information specific to software development techniques and should be reviewed in concert with this chapter at this time.

7.1 ASN(RD&A) Software Acquisition Contract Language Policy

To facilitate effective acquisition, Assistant Secretary of the Navy (Research, Development, and Acquisition (ASN(RD&A)) has defined policy regarding the contract language to be used for software acquisition. This policy (see Appendix B) will help in formulating the solicitation approach, performing source selection, and, later, in monitoring contract execution. Within this policy, ASN(RD&A) defines language that must be included in solicitations or contracts under which contractor(s) are required to perform “software development.” The language covers desired work content and practice (e.g., Statement of Work/Statement of Objectives (SOW/SOO)), RFP Section L content (information to be provided by offerors regarding their software development approach and their related experience), and RFP Section M (evaluation factors for software). The memorandum provides specific criteria for the type of contracts and work efforts to which this might apply. This information is intended to provide more visibility to the acquisition program concerning the software development approach to be taken by the offerors, thereby assisting in the source selection decision. The policy applies to software development performed by government contractors and subcontractors, regardless of tier, and it is the responsibility of the prime contractor to ensure that computer software developed or delivered by subcontractors under an affected DoN contract follows the policy. Requiring this in the RFP will curtail more expensive change orders in the future and allow the prime to flow-down these requirements to their team for consistent metrics.

On 13 July 2007, this policy was amplified with a second policy memo (see Appendix C) which provides additional guidance on the language to be incorporated into RFP packages.

7.2 Source Selection Plans

The Source Selection Plan is described by Defense Federal Acquisition Regulation Supplement (DFARS) PGI 215.3, and can be accessed at **http://www.acq.osd.mil/dpap/dars/dfarspgi/current/index.html**. This plan is to be prepared at the direction of the PM, in coordination with the contracting organization. The SSP describes the approach to be taken for source selection and is necessary to proceed since the source selection team will rely on the SSP for direction when evaluating the proposals. The source selection plan is the written guide for the source selection process (this plan is sometimes referred to as the Technical Evaluation Master Plan). It presents the source selection organization and responsibilities; the proposed evaluation factors, any significant sub-factors, and their relative importance; the evaluation process, including specific procedures and techniques; and a schedule of significant events in the source selection process before public announcement. The SSP forms the basis for Sections L and M of the RFP and must be formally approved by the Source Selection Authority (SSA) before issuing any RFP.

DFARS 215.303 designates the following as a minimum set of information to be included in the SSP:

- *The organization, membership, and responsibilities of the source selection team.* For software acquisitions, it is important to ensure that the source selection team contains software SMEs. These experts should, to the maximum extent practical, be experienced in the specific disciplines associated with the planned work effort. Such expertise is necessary to ensure that the software aspects of the offerors' proposals are evaluated effectively and efficiently.
- *A statement of the proposed evaluation factors and any significant subfactors and their relative importance.* These factors should include those associated with the software effort expected for the acquisition, and will be placed into Section M of the RFP when finalized. Note that the role of the offerors' proposed Software Development Plans should be included as a key factor.
- A description of the evaluation process, including specific procedures and techniques to be used in evaluating proposals. For software, there are some procedures and techniques that need to be used.
- A schedule of significant events in the source selection process, including the release of the RFP, documentation of the source selection decision, and announcement of the source selection decision. If there are any software-specific events (such as a bidders' conference to discuss considerations regarding the government's views on approach to be followed), these should be included in the schedule.[1]

SSPs must adhere to the acquisition strategy and methods documented in the approved Acquisition Plan. The size and detail of each SSP will be tailored for the specific program or requirement. The SSP needs to

[1] Defense Acquisition University Continuous Learning Module. *CLC 007 Contract Source Selection.*

be approved before solicitation release, and forms the basis for the development of Sections L (Instructions, Conditions, and Notices to Offerors) and M (Evaluation Factors For Award) of the solicitation. In some cases, sections L & M are prepared in advance and are attached to the SSP.

For software acquisitions, it is critical to ensure that acquisition staff with sufficient software expertise contribute to the development of the SSP. This expertise is to support the definition of the software-specific evaluation factors and the evaluation process.

7.2.1 Source Selection Team Membership

It is important to ensure that the SSP identifies source selection team members to include experts in software engineering and architecture as well as other necessary technologies. These experts should, to the maximum extent practical, be experienced in the specific disciplines associated with the planned work effort to ensure that the best value offeror is selected. While general software engineering expertise is required, it is important that this expertise include, as much as practicable, experience with technologies that are relevant to the system to be produced. For example, database experience might not be as useful for embedded real-time software systems. Likewise, experience in web-technologies is not as useful as software experience in radar systems for systems that involve development of radar software. It is important that the source selection team include someone familiar with the Capability Maturity Model Integration® to ensure that the offeror has provided proof of processes consistent with a Level 3 organization, which will be reflected in the Software Development Plan as well. Hence, source selection for software systems needs to be a multi-disciplined team effort from the earliest planning stages. The size and composition of the team should be tailored specifically to the acquisition. The teams, whether large or small, should be established to ensure continuity and ongoing involvement of appropriate contracting, technical, logistics, legal, user, contract administrators, and other experts to ensure a comprehensive evaluation of each offer.

7.2.2 Evaluation Process Planning

The evaluation process should be defined as a part of the SSP to ensure that the source selection team follows a pre-defined, consistent approach applied across all proposals received from offerors.

The evaluation will be based on the information provided within the proposal. For software this includes the items listed in RFP Section L. The evaluation will be performed against the specific evaluation factors listed in RFP Section M. These evaluation factors are defined based on general software engineering expectations as well as specific characteristics of the planned work effort as influenced by the software to be acquired.

The overall evaluation process will be defined by the acquisition team. In general, for the software portion of the source selection, the following activities will be performed for each proposal:

- Inventory the information submitted, and ensure that the information is complete and responsive to the RFP Section L requirements;
- Review the evaluation factors and ensure the Source Selection Team has a consistent interpretation of their meaning and resultant scores;
- Apply each of the evaluation factors to the submitted materials and determine a score;

- Apply ranking criteria to each factor; and
- Determine overall software score for each offeror.

When evaluating the software development aspects of the proposals, the five factors discussed in Section 7.6 should be assessed and appropriately balanced.

7.3 Requests for Proposals

The RFP is the formal document provided to prospective vendors that defines (among other items) the system to be acquired, the work content of the effort, the materials to be included with the proposals, and the criteria to be used to evaluate the proposals. The ASN(RD&A) policy of 17 November 2006 provides language to be included in all RFPs and contracts that contain software development, acquisition and life cycle support (see Appendix B for sample Section L&M Material). This chapter discusses five key sections within the RFP that require software SME attention:

- Statement of Work or Statement of Objectives;
- Section L - Instructions, Conditions and Notices to Offerors;
- Section M - Evaluation Factors for Award;
- System Requirements; and
- Deliverables (Contract Data Requirements Lists (CDRLs)).

7.4 SOW/SOO

The work content of the effort must be described within an RFP to provide a basis for the offerors to create and submit proposals. This work content is often described in a Statement of Work or alternately a Statement of Objectives. Different organizations may have different practices and locations for this information. According to MIL-HDBK-245D Handbook For Preparation Of Statement Of Work (SOW), 3 April 1996:

> "The SOW defines (either directly or by reference to other documents) all work (non-specification) performance requirements for contractor effort. Qualitative and quantitative design and performance requirements are contained in specifications developed according to MIL-STD-961."

The handbook additionally states:

> "After contractor selection and contract award, the contract SOW becomes a standard for measuring contractor performance."

For software efforts, there are certain important aspects to the work effort that need to be specifically addressed in the SOW/SOO. Sections 7.4.1 through 7.4.6 address six of these aspects:

- Software Development Plan;
- Open Architecture/Open Systems;
- Intellectual Property;
- Systems Assurance, Software Assurance, and Information Assurance;

- Shared Development Environment; and
- Software Integrated Process Teams.

7.4.1 Software Development Plan

The language required by ASN(RD&A) policy addresses the need for offerors to define their software development approach in a Software Development Plan (SDP), structured in accordance with the framework defined in Institute of Electrical and Electronic Engineers/Electronic Industries Alliance (IEEE/EIA) Standard 12207. The intent of the policy language is not to change the internal standards and practices for contractors, but rather to communicate those standards and practices to DoN acquisition managers and contracts personnel in a standardized manner — specifically, the IEEE/EIA Standard 12207 framework. The content and format for the SDP may be specified in the SOW or SOO, or may be specified in an SDP Data Item Description (DID). A sample DID for an SDP is provided at Appendix L. The policy also requires a draft SDP to be provided with each proposal to be considered during source selection. The proposed SDP provided as part of an offeror's proposal shall form the basis for a completed SDP to be delivered after contract award as a CDRL, subject to government review and approval. The SDP is to cover the prime contractor and any and all subcontractors who will be contributing to the software work effort. The SDP can be delivered in one or more volumes, if there are different approaches used either by the different organizations or for different parts of the system. The information content of the SDPs shall follow the framework of IEEE/EIA Standard 12207 regarding subject content, level of detail, and completeness. The RFP should require that the offeror describe how the SDP will be periodically evaluated and updated, as part of the continuous process improvement subject to government review and approval. While there is no requirement that the specific IEEE/EIA Standard 12207.1 documents need to be created, their information content must be provided in some format, as appropriate, for the proposed work effort. If software documentation will be included in the list of deliverables, it is recommended that the titles and content be consistent with IEEE/EIA Standard 12207.1. In addition, all information relating to the software development processes, activities, tasks, techniques, and tools to be used on an effort must be described in their response to proposal.

7.4.2 Open Architecture/Open Systems

The Naval Open Architecture (NOA) initiative is the Department of Navy's application of the DoD's Modular Open System Approach (MOSA) initiative. Existing naval policy requires programs to implement open architecture principles as an integral part of their acquisition. This policy is contained in OPNAV letter "Requirement for Open Architecture (OA) Implementation" dated 23 Dec 2005.[2] A program's successful incorporation of open architecture involves both technical and business models. When planning for acquisition of software, programs are required to conform to Open Architecture/Open System (OS) principles in order to:

- Encourage competition and collaboration through the development of alternative solutions and sources.
- Build modular designs and disclose design data to facilitate evolutionary designs, technology insertion, competitive innovation, and alternative competitive approaches from multiple qualified

[2] Department of the Navy. Deputy Chief of Naval Operations (Warfare Requirements and Programs (N6/N7) memo. Ser N6N7 / 5U916276. "Requirement for Naval Open Architecture." 23 Dec 2005.

sources. This requires offerors to provide at least Government Purpose Rights (GPR) license rights for the products.

- Build interoperable joint warfighting applications and ensure secure information exchange using common services (e.g., common time reference), common warfighting applications (e.g., track manager) and information assurance as intrinsic design elements.
- Perform market research, identify, and/or develop reusable application software selected through open competition of "best of breed" candidates, reviewed by subject matter expert peers and based on data-driven analysis and experimentation to meet operational requirements. Both the Software Hardware Asset Reuse Enterprise (SHARE) and the Net-Centric Enterprise Solutions for Interoperability (NESI) are examples of naval information environments that provide a means for discovering potential artifacts.[3]
- Ensure life cycle affordability including system design, development, delivery, and support while mitigating Commercial-off-the-Shelf (COTS) obsolescence by exploiting the Rapid Capability Insertion Process/Advanced Processor Build methodology.

To aid in compliance, the DoN has released a contract guidebook to help programs conform to the policy ("Naval Open Architecture Contract Guidebook for Program Managers"). The guidebook has been released in two versions – one for industrial distribution, the other for program managers (government only). The guidebook contains suggestions for RFP contract language (Sections C, H, L, and M) that facilitates achieving open architecture goals, and can be accessed at https://acc.dau.mil/CommunityBrowser.aspx?id=105662.

Of particular importance in the evaluation of OA proposals is the systems engineering and software engineering balance of openness, warranty management, security or Information Assurance (IA), and safety.

7.4.3 Intellectual Property

The following Intellectual Property (IP) discussion is usually applied to non-commercial software. Acquiring more than Standard rights for commercial software is often cost-prohibitive, but the cost must be carefully weighed against any potential benefits to the government. Program offices need to exercise care to ensure that the context into which COTS items are placed is defined with sufficient rights so that the government can pursue alternative solutions for future upgrades. That is, the interfaces to the COTS products must be available to the government. Also, while many offerors will appear to be providing all COTS, or make grand OA claims, this section still applies to whatever software is required to configure and integrate these COTS items into a system that operates as required. This software includes so-called "glue" code that enables integration, scripts that configure the COTS and the operating systems, database (e.g., Structured Query

[3] Software Hardware Asset Reuse Enterprise (SHARE) – https://viewnet.nswc.navy.mil (requires CAC or digital certificate for access) – SHARE provides a central repository to which Navy program offices and contractors can contribute assets for which the Navy has at least government purpose rights. Additionally, program offices and organizations that have contracts with the Navy can withdraw assets from SHARE to improve them, learn from them, or build additional capabilities that can easily interface with them. This capability helps share information among contractors as well as increases competition because additional contractors are able to build capability for systems to which they previously could not gain access.
Net-Centric Enterprise Solutions for Interoperability (NESI) – http://nesipublic.spawar.navy.mil/ – NESI is a joint effort between the United States Navy's Program Executive Office for C4I & Space and the United States Air Force's Electronic Systems Center. It provides implementation guidance which facilitates the design, development, maintenance, evolution, and use of information systems for the Net-Centric Operations and Warfare (NCOW) environment.

Language (SQL)) code that drives the COTS, and whatever else that is needed to make it all work. It is recommended that program managers use this section to better understand the requirements going into RFPs and to assure completeness. Table 7-1 describes the technical data rights associated with commercial data items, and Table 7-2 defines those rights for non-commercial items.

7.4.3.1 Overview

Intellectual property deals with the rights associated with the products produced by contractors, including the various software products. The establishment of IP terms and conditions is a critical aspect of any software acquisition activity. Without the proper data rights, programs will not be able to legally use their deliverables the way they want or need, regardless of what other portions of a contract appear to say.

It is critical, legally speaking, that the RFP and the offeror's response distinguish between commercial and noncommercial software. Commercial software is set forth in DFARS 252.227-7014(a) as software developed or regularly used for non-governmental purposes and either 1) sold, leased, or licensed to the public; 2) offered for sale, lease, or license to the public; 3) doesn't meet the two prior conditions but will be available for commercial sale, lease, or license in time to satisfy the delivery requirements of this contract; or 4) meets any of the prior three conditions and would require only minor modification to meet the requirements of the contract. Commercial computer software should be acquired under the licenses customarily provided to the public unless such licenses are inconsistent with federal procurement law or do not otherwise satisfy user needs. For example, a commercial computer software license may be modified to refer to federal law instead of a particular state law or modified to request source code in order to support a program requirement to integrate the software into an existing system. Noncommercial software is any software that does not meet the description of commercial software.

For noncommercial software the DFARS includes a standard set of license rights that delineate what the government can expect, but if these are either 1) not cited, 2) not exercised, or 3) not appropriate for the needs of the government, then the ability of the government to take full advantage of the products being acquired will be compromised. It is important to understand that, according to law, the contractor typically owns whatever they develop, such as computer software, computer software documentation, or technical data unless a special works clause is provided in the contract. The government only receives license rights to use these items. It is therefore crucial that the government negotiates license rights that are needed for any specific acquisition. The DFARS standard license language provides rights only if the DFARS clauses are placed into the contract. Even if cited however, it is possible that the rights might not meet the needs of any specific acquisition. Further, the government may have difficulty exercising its rights in software it does not possess. Appropriate Contract Data Requirements Lists or other contract deliverables should be prepared for any software that the government program intends to use, modify or distribute to other contractors. One effective strategy is to include in the RFP a statement based on DFARS 252.227-7017 that requires offerors to provide unlimited rights for all products except for those that they explicitly list.

Beware of software tools that the offeror will use in producing their software. A specific CDRL item should call out tools and the IP or warranty on using them with specific settings to produce the deliverable software product. These details must be called out in the contract for any warranty or future modification or distribution to other government contractors.

7.4.3.2 Assessment of Planned Work — Data Rights Requirements Analysis

It is the responsibility of the contracting officer to put the proper data rights clauses into the contract, but it is the responsibility of the program office to provide the contracting officer with a complete assessment of the planned work effort. This assessment should include a determination of the contemplated present uses of the software or other deliverables as well as an assessment of any future uses of the software products or tools used in their production. This assessment is called a “Data Rights Requirements Analysis” (DRRA) and should be conducted prior to contract award using the offeror’s response, taking into consideration such factors as multiple site or shared use requirements, and whether the government’s software maintenance philosophy will require the rights to modify or have third parties modify the software or the tools used to modify it.

Programs should work within their Program Executive Offices (PEOs), sustainment or in-service maintainers, and across their Communities of Interest (COI) in considering future needs for data and other intellectual property rights in a structured, focused manner. The naval portfolio manager may also be able to help, but these contacts must be noted in the SSP or they may not be allowed during deliberations. The goal of this assessment is to identify opportunities or requirements for information and information product sharing and then to structure contracts accordingly. Such an assessment should include both a cross-domain and enterprise-wide review of the component “marketplace” – both supply and demand. The results of this analysis should guide the program office in determining the intellectual property and intellectual property rights that it requires the contractor to deliver. If the DRRA determines that the standard data rights clauses do not provide sufficient rights to meet the program’s needs and the future needs of the federal government, additional rights may be obtained later through negotiations with the contractor, usually at an additional cost. It is important to perform a trade-off analysis between the additional cost and the benefits realized from obtaining the rights. Tables 7-1 and 7-2 summarize the different characteristics of each rights category including Small Business Innovation Research (SBIR) rights, along with criteria for their application.

Commercial Technical Data (TD) and Computer Software (CS) Data Rights Assertion Categories

Rights Category	TD or CS?	Criteria for Applying Rights Category	Permitted Uses Within Government	Permitted Uses Outside Government
Standard DFARS "7015" Rights	TD only	Default category for all commercial TD (TD pertaining to commercial items) except those qualifying for Unlimited Rights.	Unlimited; except may not be used for manufacture.	Only with contractor's written permission or for emergency repair/ overhaul.
Unlimited Rights (UR)	TD only	Commercial TD that: 1) has previously been provided to government or is already publicly available without restrictions; 2) is "form, fit and function"; 3) is a correction to TD previously delivered to the government; 4) has been provided to the government with UR from a prior contract; or, 5) is necessary for operation, maintenance, installation or training.	Unlimited; no restrictions	
Standard Commercial License	CS only	Default rights category for all commercial CS.	As specified in the license customarily offered to the public. DoD must negotiate for any specialized needs, or if any of the license terms are unacceptable to the government.	
Specifically Negotiated License Rights	Both TD and CS	Mutual agreement of the parties; should be used whenever the standard categories do not meet both parties' needs.	As negotiated by the parties; however, by statute, the government cannot accept less than the minimum standard 7015 rights in commercial TD.	

Table 7-1. Commercial TD and CS Data Rights Assertion Categories[4]

[4] Department of the Navy.PEO IWS 7.0. *Naval Open Architecture Contract Guidebook*. Version 1.1. 25 October 2007.

Non-Commercial Technical Data (TD) and Computer Software (CS) Data Rights Assertion Categories				
Rights Category	TD or CS?	Criteria for Applying Rights Category	Permitted Uses Within Government	Permitted Uses Outside Government
Unlimited Rights (UR)	TD and CS	Applies to: 1) TD/CS that is developed exclusively at government expense; 2) TD that is test data; 3) TD that is form, fit and function data; 4) TD that is necessary for operation, maintenance or training; 5) Corrections or changes to TD/CS previously delivered to the government; 6) TD/CS otherwise publicly available; 7) CS documentation deliverables; and, 8) TD/CS whose GPR have expired.	Unlimited; no restrictions. Note: If a third party copyright is asserted in TD/CS that is delivered with UR, under DFARS 227.7203-9 the delivering contractor must grant or obtain for the government license rights that permit the government to reproduce, perform or display the software or documentation; distribute copies; and, through the right to modify data, prepare derivative works. If the contractor does not obtain an appropriate license for the government, then the contractor should not incorporate the unlicensed copyrighted material into the deliverable TD/CS without the Contracting Officer's written approval	
Government Purpose Rights (GPR)	TD and CS	Development with mixed funding.	Unlimited; no restrictions.	For "Government Purposes"; no commercial use. Must have recipient sign a Non-Disclosure Agreement (NDA).
Limited Rights (LR)	TD only	Development exclusively at private expense.	Unlimited; except may not be used for manufacture.	Emergency repair/overhaul; evaluation by foreign government; may also disclose subject to a prohibition on any further disclosure after notifying the asserting contractor.
Restricted Rights (RR)	CS only	Development exclusively at private expense	Government may: 1) Use on one computer at a time; 2) Transfer to another government entity (transferor must destroy all copies); 3) Make minimum backup copies; and 4) Modify, provided there is no release or disclosure outside government.	Emergency repair/overhaul (w/NDA). Support contractors may use (w/NDA).
Prior Government Rights (DFARS 252.227-7028)	Both TD and CS	Whenever government has previously acquired rights in the deliverable TD/CS.	Same as under previous contract.	
Specifically Negotiated License Rights (SNLR)	Both TD and CS	Mutual agreement of the parties; use whenever the standard categories do not meet both parties' needs.	As negotiated by the parties; however, must not be less than LR in TD and must not be less than RR in CS.	
SBIR Data Rights	Both TD and CS	Whenever TD/CS is generated under a SBIR contract, **regardless of funding**. SBIR Data Rights expire five years after completion of the SBIR project from which such TD/CS were generated.	Within government, use and disclosure is unlimited.	Cannot release or disclose SBIR data outside of government, other than support services contractors, except: 1) As expressly permitted by the contractor; 2) For evaluation purposes; or, 3) For emergency repair or overhaul. When disclosed outside government, an NDA is required.

Table 7-2. Non-Commercial TD and CS Data Rights Assertion Categories[5]

[5] PEO IWS 7.0. *Naval Open Architecture Contract Guidebook*.

7.4.3.3 Principles of DRRAs

There are some principles to consider when performing a data rights assessment:

- Data rights issues are complex and require careful examination of the program's requirements and overall "fit" within the enterprise. Establishing the data rights strategy for a program requires expert guidance from government attorneys and the contracting officer to determine the best strategy.
- Proper experts should be used to review program data rights requirements – strategy development should involve software and architecture experts, an intellectual property lawyer, a contracting officer and the Program Manager.
- It is typically very expensive to acquire the broader data rights or to create additional options for software maintenance after the initial contract is in place. Inadequate data rights typically result in paying large sums of money to acquire the required rights or having only one option for software maintenance: sole source procurement to the creator of the software. Sole sources have little incentive to offer lowest cost.
- Insufficient data rights prevent the government from using deliverables in the most optimal way.
- Data rights will impact maintenance over 30 or more years of a system's life.
- Programs should perform a Business Case Analysis (BCA) as a part of assessing the IP needs to determine whether obtaining the desired rights is the correct business decision.

7.4.3.4 DRRA Considerations

A DRRA should address the following issues:

- Is this a new or existing procurement?
- What type of procurement or assistance vehicle is/will be involved (Federal Acquisition Regulations (FAR)/DFARS contract, grant or cooperative agreement).
- Does the government already have data rights in existing software or other deliverables that permit the government to leverage (i.e., modify and/or enhance) that existing software for this new contracting effort (including necessary architecture/design/interface documentation)?
- What clauses already exist regarding data rights?
- What are the benefits of broader data rights clauses? For example, will acquiring more than restricted/limited rights impact procurement cost without providing value?
- Will one of the standard DFARS levels of data rights ("unlimited," "government purpose" or "restricted/limited") be acceptable, or do the data rights need to be specifically tailored/negotiated for this procurement?
- Does the number of anticipated changes to the software and the required response time for those changes warrant the possible additional cost or fewer bidders on the procurement?
- Will the government obtain at least Government Purpose Rights? If not, is the asset isolated at the lowest component level? If not, is it non-critical? If not, what is the justification for less than GPR?
- Has the Program identified potential components and artifacts that can be provided to the offerors as Government Furnished Information (GFI)?

- Does the government have the right to provide the information to third parties? If not, should the government negotiate a license for this right?
- What is the likelihood that the government will perform the software maintenance (i.e., error corrections and enhancements) in-house?
- What is the likelihood that the software maintenance will be competed and awarded to a third party?
- Might there be any situations that would require licensing outside the federal government (e.g., Foreign Military Sales (FMS) or commercial)?
- Does the government require the rights to modify the deliverables now or in the future (modifications include updates, corrections and enhancements)?
- Will the government need special tools to be able to modify the deliverables?
- Do the components to be acquired fit within an existing, approved government architecture, or can they be easily modified to fit into an approved architecture? Does the government have sufficient rights to perform this modification?
- Does the government need to maintain configuration control over the deliverables? If so, the government needs to obtain sufficient license terms to perform this maintenance.

When performing the DRRA, it is important to address both the long-term as well as the short-term needs, since software could be in use for 30 or more years.

After the DRRA has been conducted, the contracting officer will determine if the standard data rights clauses provide the rights that the contractor and the government need to accomplish the stated objectives. If additional rights are required, the contracting officer can enter into negotiations with the contractor to acquire such rights.

Other Sources of Information about Intellectual Property Rights: The Federal Acquisition Regulations (FAR) and Defense Federal Acquisition Regulations (DFARS) are the primary sources of information regarding data rights. Applicable FAR/DFARS intellectual property/technical data/software provisions include:

- FAR 52.227-11, Patent Rights – Retention by the Contractor (Short Form);
- FAR 52.227-12, Patent Rights – Retention by the Contractor (Long Form);
- DFARS 252.227-7013, Rights in Technical Data – Noncommercial Items;
- DFARS 252.227-7014, Rights in Noncommercial Computer Software and Noncommercial Computer Software Documentation;
- DFARS 252.227-7015, Technical Data – Commercial Items;
- DFARS 252.227-7016, Rights in Bid or Proposal Information;
- DFARS 252.227-7017, Identification and Assertion of Use, Release, or Disclosure Restrictions;
- DFARS 252.227-7018, Rights in Non-commercial Technical Data and Computer Software – Small Business Innovation Research (SBIR) Program;
- DFARS 252.227-7019, Validation of Asserted Restrictions – Computer Software;
- DFARS 252.227-7020, Rights in Special Works;

- DFARS 252.227-7025, Limitations on the Use or Disclosure of Government- Furnished Information Marked with Restrictive Legends;
- DFARS 252.227-7027, Deferred Ordering of Technical Data or Computer Software;
- DFARS 252.227-7028, Technical Data or Computer Software Previously Delivered to Government;
- DFARS 252.227-7030, Technical Data – Withholding of Payment; and
- DFARS 252.227-7037, Validation of Restrictive Markings on Technical Data.

FAR/DFARS materials can be accessed at http://www.acq.osd.mil/dpap/sitemap.html

7.4.3.5 Commercial-Off-The-Shelf and Open Source Software

Currently, the DFARS require the identification of non-commercial software to which the government has less than unlimited rights. It is also necessary for acquisitions to require identification of any commercial software (such as COTS, open source, “Freeware”, etc.) that are planned to be, or actually are, a part of the software items to be delivered in fulfillment of the contract requirements. In particular, some open source license terms may prove to be too restrictive for Government use. Contractors also need to identify the license terms of any software tools or libraries used to build the software products, to allow the Government an opportunity to plan for system sustainment. The Navy Open Architecture Guidebook has suggested Section L proposal language related to the use of proprietary or vendor unique elements, as well as suggested language relating to the use of Open Source Software.

7.4.4 Systems Assurance, Software Assurance, and Information Assurance

DoD systems must protect their information systems and the information processed by these systems. The presence of hostile agents, whose goal is to disrupt the proper operation of naval systems, presents challenges to system development. To overcome these challenges, DoD has mandated procedures designed to protect systems and the information they handle. There are numerous terms used to refer to these considerations, many of which have multiple meanings across industry and government, including the DoD, National Aeronautics and Space Administration (NASA), National Institutes of Standards and Technology (NIST), and others. The terms used in this guidebook are distillations of these other definitions and have been chosen to more directly address software engineering concerns.

The term *systems assurance* generally refers to activities that focus on ensuring that the system functions as intended, is free of exploitable vulnerabilities, and protects critical program information. These activities require the use of systems engineering practices that minimize the introduction of vulnerabilities across the entire supply chain. In systems engineering terms, vulnerabilities are not just malicious code but include requirements gaps, architecture and design flaws, dead code, and open interfaces among the risks to be addressed.

The term *software assurance* generally refers to those activities that focus on ensuring that the software portions of systems achieve the same goals as systems assurance. Key practices are applied to ensure that the software:

- Conforms to all allocated requirements and standards;
- Achieves predictable execution under benign, degraded, and hostile environments; and
- Avoids vulnerabilities that could be exploited by hostile agents.

One important aspect of systems and software assurance is that of system safety. Overall requirements for this area are covered in standards such as MIL-STD-882, MIL-HDBK-245, and MIL-STD-961.

Of particular concern for software developers is the potential threats presented by the use of software obtained from external sources. Such software includes COTS, open source, "freeware," among others. All such software packages need to be analyzed to ensure that latent vulnerabilities do not exist hidden in the code. For that software for which the source code is available, developers need to analyze the code to ensure the absence of risk. For software for which the source code is not visible, other scanning techniques must be applied.

Special note: DO-178B,[6] the commercial standard used to assess software, may not handle software assurance to the extent that is needed to ensure the software is safe to use and there is no malicious code. Care should be taken when using this standard.

Information assurance (IA)is more specific in that it refers to those measures that protect and defend information and information systems against unauthorized access, control, and disruption. IA assumes that the traditional passwords, guns, gates, and guards approach will keep hostile agents out, or if they are inside, then they can be detected and neutralized. IA has three goals:

- *Confidentiality* – to prevent unauthorized disclosure of information;
- *Information integrity* – to prevent unauthorized modification or destruction of information;
- *Service integrity* – to prevent unauthorized disruption of service.

This protection also requires providing for restoration of the information systems by incorporating protection, detection, and reaction capabilities. *Anti-tamper (AT)* is another design practice that refers to the prevention or delay of exploitation of critical technologies.

As a part of the RFP, program offices need to carefully define the extent of IA that is required for the system to be acquired and for their own networks while developing the product(s). Offerors need to assess these needs and respond in their proposals, describing their approach to achieving the necessary IA. An important consideration is determining the type and extent of IA required, determined at least partially by the nature of the system under development and the nature of the offeror's own development network.

Another area, and of particular cost and importance, is AT features to protect against the unauthorized alteration of software and data. Nearly all new software products are required to have some hardware or software AT feature and a separate certification by the Anti-Tamper Executive Authority. Anti-Tamper assumes that the hostile agent has gotten hold of or is inside the system. At the systems engineering level this AT feature can be very difficult to integrate with other security and safety designs.

Programs must be aware that the offeror's development environment, the product's interoperating and integration features, the Critical Program Information (CPI), and the Threat Assessment will together determine the cost associated with all of these assurances and Anti-Tamper for software. Early discussion with systems and software architects, software SMEs, Counter Intelligence, National Security Agency (NSA), Naval Criminal Investigative Service (NCIS), and the Anti-Tamper Executive Authority (or service

[6] RTCA/DO-178B, "Software Considerations in Airborne Systems and Equipment Certification," December 1, 1992.

representative) will help estimate these costs and proper responses to proposals by offerors. Note that these factors may also negatively impact Open Architecture and Open Source goals.

For software engineering, the impact of software assurance and IA will be visible in the requirements that are imposed on the software systems. In a benign, friendly or “firewalled by others” environment, systems will be allocated requirements that are related to the intended mission of the system and the capabilities required, except Anti-Tamper will probably still be required. Software assurance and IA will augment and modify these requirements for those aspects specific to the hostile environment defined in the Threat Assessment provided by Counter Intelligence and NCIS. As such, in preparing for solicitation, the specific threats that may impact naval systems need to be identified so that offerors can propose effective and efficient solutions. Note that Threat Assessments are usually classified and associated carry costs should be considered.

7.4.5 Shared Development Environment

One important element of planning for acquisition is to ensure that there will be adequate visibility into the various software artifacts being developed by the contractors. This visibility is important to be able to effectively observe and measure progress, product quality, and adherence to process. Recognizing that achieving such visibility may increase acquisition cost, the program office needs to trade off the benefits against any increased cost. As a part of this trade off, however, there are significant cost avoidance benefits that must be appropriately considered. Any cost impact will be based on the resources required to provide the government access into the developer’s development environment, since in any case, the developers will need to create an Information Development Environment (IDE) for their development effort. Other items potentially affecting cost will be security considerations and the need to appropriately segregate access based on IP licenses provided to users of the development environment. The metrics (see Chapter 2) provided by the contractor will provide “snapshot” visibility but the advantage to a shared environment is being able to observe inchstones in real-time. Still, the preferred approach is to establish an information environment to which both the government and the contractors have simultaneous access and visibility. There are three key aspects to providing this visibility.

The first aspect is to require that the information environment (Figure 7-1) is created. This environment is to contain all development artifacts (intermediate and final) as well as CDRLs (both under development and released). In addition, all of the contents of this environment are to be visible to the government real-time, including those on Navy/Marine Corps Intranet (NMCI) computers. The analogy is a disk farm with shared access to which to the government and the contractors (prime and sub) have equal access. Any artifacts created should be visible to the government in “real-time” (as the items are being created, modified, and evolved) and should not require an explicit transfer of any items for the government to be able to see them. This includes all project artifacts, such as software development folders, briefings, reports, hazard analyses, CDRLs, notes, review records, test results, etc.

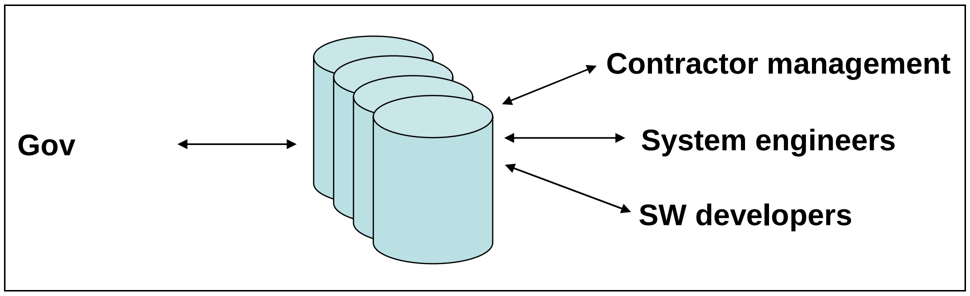


Figure 7-1. Information Environment

The second aspect is to provide to the government continuous, real-time, remote access (Figure 7-2) to the information environment. The government needs to require that the information environment be available to the government remotely, via secure communications lines provided as a part of their funded effort. This ensures that the government will not need to travel to examine the various software artifacts. Since the access still costs the government, balance the periodic travel costs the program might have to spend against the cost of remote but continuous access.

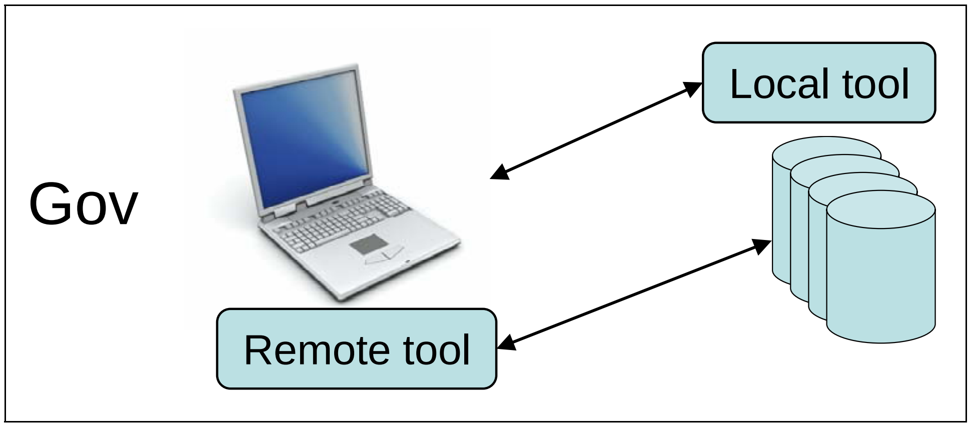


Figure 7-2. Continuous, Real-Time, Remote Access

The third aspect is to ensure that the government has access to the tools necessary to gain access to the items within the information environment. As an example, if the contractor decides to use the requirements management tool, Dynamic Object Oriented Requirements Systems (DOORS®), government will require copies of the tool along with necessary licenses. Whether the contractor is to provide the tools and licenses or whether the government will acquire the tools and licenses is subject to negotiation and program planning. Remember maintenance and support costs as well with this option. It is important to ensure that the tools can be used remotely with minimal impact to performance. That is, if remote use slows down the analysis process, then a solution needs to be defined that provides a copy local to the government staff. The government also needs access to whatever training may be needed to properly configure the tools and efficiently use the tools. If the government does not need full access to the tools (such as would be required for the actual developers), this limited use needs to be included as a part of the planning for the information environment.

Government access to the information environment can be controlled relative to read-only or read/write, depending on the purpose and contents of the information. In any case, there should be areas within the environment dedicated for government purposes, to which read/write access is granted.

The goal is to enable the government to monitor the real-time progress of the development team, and to be able to examine in depth the artifacts at will, including the ability to confirm metrics. This ability will make the process of monitoring progress and quality much easier and immediate. This will also facilitate the review process since advance analysis can be conducted. If necessary, and it generally is, access can be controlled in the case of subcontractor, (to financial information and the like) through the use of privileges.

The following language can be included in the RFP to address the shared environment:

Upon contract award, the contractor shall establish a program-wide integrated IDE that:

- Contains all program development information, including intermediate and final artifacts, both in-work and completed, developed and used as a part of the development activity, to include, at a minimum, all design data items, review materials (briefings and reports), technical reports and briefings, all software measures and metric reports, peer review reports, program schedules, and all Software Development Folders (SDF) (including all requirements, design documents, code, test cases, test results, and other items).
- Provides a web-enabled interface allowing continuous, real-time access (remote and local) to all items contained in the IDE.
- For information maintained in special formats (such as within database tools used to manage requirements), access will be provided either via web-enabled tool interfaces or via remote tool invocation, or both, ensuring minimal user access delays.
- Provides continuous, real-time access for all stakeholders, including government staff, contractor staff, subcontractors, Independent Verification and Validation (IV&V) activities, and others as necessary and appropriate.
- Segregates information according to need-to-know and security levels, protecting the enclaves using access controls and, if necessary, separate networks. For example, restricted and proprietary information can be kept private from subcontractors.
- Upon contract completion, the IDE shall be delivered to the government as a CDRL.

It should be noted that the above description for a shared environment provides the program office exceptional visibility and insight into the contractor's software activities and needs to be planned with full consideration of security risks and additional expense. As the security environment tightens with respect to both classified and unclassified information, shared electronic access must be carefully considered, and if implemented, carefully controlled. The cost burden of a shared environment will likely be reflected either explicitly or implicitly in an offeror's price, although this can be reduced if the government directly funds the mechanisms needed to provide such access (e.g., secure communication lines). Program managers should carefully consider the tradeoffs between value and cost, and form their requirements for a shared development environment accordingly.

7.4.6 Software Integrated Process Teams

Programs are encouraged to require offerors to establish a Software Integrated Process Team (IPT) pre-award in the RFP/SDP, or post-award via Memorandum of Understanding (MOU), to provide a forum in which the government and the contractor can coordinate on all aspects of the software effort. Such a group can assist the government in monitoring contractor progress and adherence to process, and also provide a mechanism by which the government can influence and encourage process improvement. This should be done regardless of whether the program has a shared software IDE. These are sometimes called Software Engineering Process Groups (SEPGs), Engineering Process Groups (EPGs), or Software Councils.

It is expected that the proposed SDP will likely not be complete and comprehensive, since it is developed in a limited time period in response to the RFP. It is also expected that there may be changes identified by the government that need to be made to more completely meet government expectations. Because the winning offeror will be selected based at least partially on the contents of the proposed SDP, such changes are likely to be minor and limited. As such, one role of the Software IPT is to complete the SDP and prepare it for its initial submission to the government for approval after award.

The following language can be included in the RFP to address the Software IPT:

> Upon contract award, the Contractor and the government shall jointly establish a Software Integrated Process Team (IPT). This team shall consist of contractor and government representatives, and shall be co-chaired by the program office Chief Software Engineer and the contractor Chief Software Engineer (to include subcontractor Chief Software Engineers as appropriate).
>
> The Software IPT should be tasked to define, document, monitor, and improve the software development approach being used for the software effort. Specifically, the Software IPT shall:

- Define and document the software development approach to be used for the work effort. The approach is to be documented in the contractors' SDP, which is to be based on the proposed SDP submitted with the offeror's proposal.
- Secure government approval for the SDP. Approval is facilitated by having government representatives serving on the Software IPT.
- Identify and make process improvements to the software approach, and document these in the SDP. These improvements are to be based on lessons-learned, suggestions from staff, industrial advancements, and other sources.
- Control all changes to the SDP.
- Monitor development progress, assess effectiveness of the development approach, and monitor adherence to the defined process. One key mechanism for monitoring is attendance at technical reviews conducted in accordance with the SDP and the Navy Technical Review process. Another is the use of a separately-scheduled process assessment review (Independent Technical Assessment (ITA)), conducted specifically to determine degree of adherence to the SDP process and to assess the effectiveness of the SDP as it is being applied.
- Monitor industry-wide lessons-learned, evolution of standards, advances in relevant technology, tool utility and availability, and other information that may prove to be valuable for the software work effort.

- Advise program management in areas relating to the software effort.

  The Software IPT is not responsible for management of the software effort, for performing software quality assurance, or for acting as an IV&V agent. The IPT however shall rely on existing program management and on the QA/IV&V function to provide sufficient information to facilitate their monitoring of progress and adherence to plan.

Any meeting required by an RFP will incur an additional cost. Although the IPT addresses software process vice requirements, government involvement in any IPT has a high potential to cause requirement creep and additional costs via change orders. Decisions made regarding changing process and requirements may result in additional costs and risks, but may also result in cost avoidance later in the development. The costs and risks should be considered by the Program Manager.

The role of the government in the Software IPT can vary depending on the specific needs of the program. If the program decides to assume a proactive role and becomes involved in forging the software process, they will also be assuming more of the responsibility and risk. Likewise, if the program assumes a passive role, allowing the developers to make process decisions, they will be deferring the risk to the developing organization but have less of an influence in defining the process to be followed. The program needs to explicitly decide the optimal level of responsibility and risk, and the associated costs of participation, and define their role accordingly.

7.5 Section L - Instructions, Conditions, and Notices to Offerors

Section L of the RFP contains, among other items, a list of the information that offerors must submit as a part of their proposals. This list is defined to be compatible with the evaluation factors (RFP Section M) to ensure that the information is available during source selection. ASN(RD&A) policy (see Appendix B and Appendix C) addresses the information required from offerors to support the source selection process. For software acquisitions, there are four categories of information required:

- Software development approach to be taken, including:
  - A *proposed SDP*. This proposed SDP will form the basis for a completed SDP to be delivered after contract award as a CDRL deliverable, subject to government review and approval. The information content of the SDPs shall follow the framework of IEEE/EIA Standard 12207 regarding subject content, level of detail, and completeness. Regardless of whether a software Earned Value Management System (EVMS) is used, a software Work Breakdown Structure (WBS) is encouraged. Necessary information includes an organization chart with critical software billets and core metrics.
  - An *SDP Rationale* which describes why the specific approach defined in the SDP is appropriate for the system to be procured. Note that this should include OA, assurances (systems, software, information), and prospective CPI with current protection rationale.
- Related systems experience, including:
  - A description of previous experience in developing software of the same nature as this solicitation.
  - A description of the extent to which personnel who contributed to these previous efforts will be supporting this solicitation.

- Related process experience, including:
  - A description of previous experience in developing similar software using the same or similar processes and approaches as proposed for this solicitation.
  - A description of the extent to which personnel who contributed to these previous efforts using these processes will be supporting this solicitation.
- Process maturity, including:
  - A description of how their proposed processes are equivalent to those processes articulated by Capability Maturity Model Integration (CMMI®) Maturity Level 3, or equivalent process model.
  - A description of any previous CMMI® or equivalent model-based process maturity appraisals performed, including an identification of the organizational entity and location where the appraisal was performed, the type of evaluation, the date of the award, and the level earned.

This information will provide the basis for determining the level of confidence the Navy can place in the offeror's ability to successfully complete the software development effort.

7.6 Section M - Evaluation Factors for Award

The primary basis for evaluating the software aspects of an offeror's proposed work effort is their proposed SDP, submitted as a part of their proposal. The SSP provides a list of proposed factors – when completing RFP Section M these must be finalized before releasing the RFP.

ASN(RD&A) policy (see Appendix B and Appendix C) provides a set of factors that fall into five categories:

- Offeror's proposed software development approach - evaluate the offeror's proposed software development approach to ensure it is appropriate for the system to be developed and meets standard levels of completeness and process quality. For this evaluation, the government should rely primarily on the draft SDP and the SDP rationale. A critical part of this evaluation is to assess the proposed development approach against the specific needs of the system to be acquired. A well documented process will be ineffective if it is not suitable to the system characteristics. For example, a real-time embedded system will require different approaches and techniques than will a web-based system used for information dissemination – not only are the tools and libraries different, so are the planning and design processes. The SOW/SOO describes the system characteristics and needs, and software SME(s) should use it as a baseline against which to assess the offeror's development approach.
- Offeror's experience with developing software of the same nature as that being acquired with this solicitation. Experience gained from developing systems with different characteristics than the system to be acquired will not be as useful as that obtained by developing similar systems. Software SME(s) should compare the experience, the SDP, and rationale using core metrics: size/stability, staff, quality, and cost/schedule.
- Offeror's experience with developing software using the same or similar approach as proposed. If an offeror proposes to apply an approach that is different than what the organization has used previously, there is an increased risk due to the inexperience with that approach. These differences may be drastic or simple. It is important to select SMEs that are able to identify any differences based on Section L requirements.

- Maturity of offeror's processes – evaluate the results of previous standard process maturity appraisals, if provided. Those appraisals performed within 24 months prior to proposal submission will be of most value. It is important that software SME(s) evaluate the proposed staff and their roles associated with the previous development projects that were part of the appraisals. Organizations that apply processes that are ingrained in the corporate culture, that have been applied with demonstrated discipline, and that will be followed by staff familiar with those processes, may be able to reduce risk associated with the development effort.
- Qualifications of proposed staff – evaluate the number of proposed staff experienced in developing software of the same nature as that being acquired with this solicitation and in using the processes proposed

These factors should be described in Section M and should directly correlate to the proposal items required in Section L. Section M should also provide weighing for evaluation factors based on the total score for software source evaluation. No solicitations should include evaluation factors that could provide an unfair advantage for or inherently benefit a prospective provider(s).[7]

7.7 System Requirements

RFPs need to contain a description of the requirements of the system to be acquired, such as in a System Specification. In some cases, these requirements (based on Capability Description Documents and other information sources) may be general, with the expectation that the offerors will refine these based on their experience and engineering judgment. In other cases, these requirements will be specific, based on clearly-defined capabilities and expectations. The ultimate goal is to ensure that the system requirements specifications meet the nine qualities described in IEEE Standard 830-1998, *IEEE Recommended Practice for Software Requirements Specifications*.

- Complete – All external behaviors are defined;
- Unambiguous – Every requirement has one and only one interpretation;
- Correct – Every requirement stated is one that software shall meet;
- Consistent – No subset of requirements conflict with each other;
- Verifiable – A cost-effective finite process exists to show that each requirement has been successfully implemented;
- Modifiable – Software Requirements Specifications structure and style are such that any changes to requirements can be made easily, completely, and consistently while retaining structure and style;
- Traceable – Origin of each requirement is clear, and structure facilitates referencing each requirement within lower-level documentation;
- Ranked for importance – Each requirement rated for criticality to system, based on negative impact should requirement not be implemented; and
- Ranked for stability – Each requirement rated for likelihood to change, based on changing expectations or level of uncertainty in its description.

[7] Executive Office of the President. Office of Management and Budget. Circular A-76. 29 May 2003.

At the time the RFP is prepared, it is not expected that the system specification will meet all of these qualities. As development proceeds, the requirements will be refined. However, at RFP time, it is important that several of these be observed to the maximum extent possible. These include correctness, consistency, verifiability, traceability, and ranking for importance and stability. These latter two are particularly important for the developers to assess the impacts of refining the requirements and of making design tradeoff decisions.

In particular, for cost control, it will be important to identify requirements which are necessary to fulfill the warfighter's need, what are Threshold values, and what are Objective values. This action will help to avoid any "gold plating" which would artificially increase the costs of development.

In addition, it is important to reduce the level of ambiguity as much as feasible, to ensure that no contractually ambiguous terms exist (e.g., TBD, best, most, customarily, obvious, etc.)

Depending on the acquisition approach taken by the program office and the developers, not all of the requirements might be defined up front (this approach is sometimes called evolutionary development). In the case of some systems of systems there are requirements that may always be ambiguous because the other program interfaces have yet to exist, but the Joint Interoperability Test Center (JITC) will still need to test that functionality. If there is a need to explore the requirements space and to allow the requirements to evolve as the system matures, then it is critical that the RFP be very specific about this expectation to ensure that those requirements defined later in the development effort do not disrupt the emerging system design, the baseline, or cost and schedule without a planned process response. This approach recognizes the risk of defining certain requirements too early, with the potential impact being an additional cost of rework when the initial requirements need to be updated.

7.8 Deliverables

Information items that are to be delivered to the government as a result of contract performance are generally referred to as Contract Data Requirements List. A CDRL is a list of authorized data requirements for a specific procurement that forms a part of the contract. The CDRL is the standard format for identifying potential data requirements in a solicitation, and deliverable data requirements in a contract. Subpart 215.470 of the DFARS requires the use of the CDRL in solicitations when the contract will require delivery of data.

The purpose of the CDRL is to provide a standardized method of clearly and unambiguously delineating the government's minimum essential data needs. The CDRL groups all of the data requirements in a single place rather than have them scattered throughout the solicitation or contract. However, it is still good practice to label data throughout the RFP sections that program stakeholders and SME(s) identify as a CDRL item and then sweep that reference into the final CDRL before publishing the RFP (word processing makes this simple and ensures completeness). Adding missed CDRLs later is likely to add significant cost to the acquisition.

CDRLs are specified by one or more DD Form 1423s (see Appendix N), each containing data requirements and delivery information. CDRLs are linked directly to specific SOW tasks and managed by the program office. Data requirements can also be identified in the contract via Special Contract Clauses (e.g., DFARS), which define special data provisions (e.g., Rights in Data, Warranty, etc.).

Depending on the work content of the planned effort, there are multiple CDRLs generally required for software development activities. These are correlated with specific activities as defined by IEEE/EIA Standard 12207.0 and with the specific document described in IEEE/EIA Standard 12207.1. As a part of planning for the work effort, the program office must identify those CDRLs which are important to the acquisition and describe them in the CDRLs list.

Important CDRLs for software acquisitions include (but are not limited to):

- Software Development Plan;
- Software Architecture Description;
- Software Requirements Description;
- Software Design Description;
- Software Interface Design Description;
- Software Test Plan;
- Test procedures, scripts, cases, and results;
- Source code (to include code written in C++, Java, SQL, OS scripts, VHDL, and others); and
- Development tools (to include compilers, assemblers, analysis tools, CM tools, configuration build tools, and others).

For all deliverable items, the license terms shall be specifically stated. Each deliverable shall be labeled with the license terms and conditions. Source code, for example, shall contain appropriate headers within the code segments (see Appendix N for a sample). For those items that are commercial items, the acquiring agency can determine the optimal strategy for obtaining any tools or data. For those items that are not provided with unlimited or GPR rights to the Government, the acquiring agency can decide the optimal approach to acquiring license rights, if necessary and cost-effective.

For reasons of efficiency, requiring multiple formal CDRLs can be expensive in terms of contractor effort. When planning for deliverables, program offices should assess how to streamline the delivery process, both in terms of reducing the sheer number of CDRLs as well as facilitating their delivery. For example, electronic delivery is strongly recommended. One approach may be to exploit the shared development environment structure described in section 7.4.5 and define the entire development environment as a single, continuous deliverable. With this approach however, it is important to ensure that the important information items are clearly and discretely identified in accordance with the data item requirements described in IEEE/EIA Standard 12207.1 associated with performed processes. A piece-wise approach, but more government labor-intensive (even if the program office has sufficient software experience and ability) is to require snapshots of development and configuration databases that SME(s) can then use to form whatever report or metric is required.

7.9 Summary

The source selection process for Software Intensive Systems (SIS) and Software Intensive System of Systems (SISOS) builds on the same core preparations and checklists as were used for hardware based systems. However the “invisible” nature of software development requires making the processes and the products more visible, and earlier, in order to manage requirements, risk, and deliverables. These issues cost money and are therefore highly regarded in proposals and change orders. The guidelines in this chapter were provided to highlight those that most benefit the program, the government, and good requirements and risk management.

DFARS, DoD, and ASN(RD&A) have provided policy that is highlighted herein to guide the program office towards processes and artifacts that lessons-learned have taught can maximize visibility and provide the evidence that the program office applied due diligence. Aside from mandated contract language (see Appendix B), the above guidelines are not absolutely required but program offices will be expected to address these guidelines and the use of SME(s) where experience has shown that such application benefits the government, including at various gate and Probability of Program Success (PoPS) reviews throughout the lifecycle.

References

Defense Acquisition University Continuous Learning Module. *CLC 007, Contract Source Selection*.

Defense Acquisition University. Defense Acquisition Guidebook. <https://akss.dau.mil/dag/>.

Defense Acquisition University. Planning for Source Selection. 18 November 2005.

Department of the Army. AMC Pamphlet No. 715-3, Contracting For Best Value—A Best Practices Guide To Source Selection. 1 January 1998.

Department of Defense. USD(AT&L) memo. Risk in the Source Selection Process Working Group. 31 May 2006.

Department of the Navy. ASN(RD&A) memo. Software Process Improvement Initiative Contract Language. 17 November 2006.

Department of the Navy. ASN(RD&A) memo. Software Process Improvement Initiative (SPII) Guidance for Use of Software Process Improvement Contract Language. 13 July 2007.

Department of the Navy. Deputy Chief of Naval Operations (Warfare Requirements and Programs (N6/N7)) memo Ser N6N7 / 5U916276. Requirement for Naval Open Architecture. 23 December 2005.

Department of the Navy. Naval Air Systems Command. NAVAIR Acquisition Guide. 20 September 2006.

Department of the Navy. Naval Air Systems Command. NAVAIR Instruction 4200.39B, Principles And Procedures For Competitive Source Selection. 5 September 2003.

Department of the Navy. Naval Sea Systems Command. Acquisition Planning Guide. September 2006.

Department of the Navy. PEO IWS 7.0. Naval Open Architecture Contract Guidebook for Program Managers (Current Version). <https://acc.dau.mil/CommunityBrowser.aspx?id=105662>.

Executive Office of the President. Office of Management and Budget. Circular A-76. 29 May 2003.

IEEE/EIA 12207.0-1996. Industry Implementation of International Standard ISO/IEC 12207: 1995 (ISO/IEC 12207) Standard for Information Technology—Software life cycle processes. March 1998.

IEEE/EIA 12207.1-1996. Industry Implementation of International Standard ISO/IEC 12207: 1995 (ISO/IEC 12207) Standard for Information Technology—Software life cycle processes—Life cycle data. April 1998.

IEEE/EIA 12207.2-1996. Industry Implementation of International Standard ISO/IEC 12207: 1995 (ISO/IEC 12207) Standard for Information Technology—Software life cycle processes—Implementation considerations. April 1998.

John, William A., *Service Contracting—A desk guide to best practices (1998 edition)*. Prepared for the Navy Acquisition Reform Office. http://acquisition.navy.mil/acquisition_one_source/program_assistance_and_tools/handbooks_guides_reports/service_contracting_guide.

Mickaligher, Michael J., CPCM, SAS. *Best Value Contracting: Selection by Perception*. APMP, Spring 2001.

PBSA Course, developed through a partnership between the Institute for Supply Management (ISM), the National Contract Management Association (NCMA), and the Department of Defense. Copyright permission has been granted for use in USACCE's Center for Excellence for Service Contracting.

RTCA/DO-178B, "Software Considerations in Airborne Systems and Equipment Certification," December 1, 1992.

Software Engineering Institute, Understanding And Leveraging A Suppliers CMMI® Efforts: A Guidebook For Acquirers. Carnegie Mellon University. March 2007.

Software Engineering Institute. CMMI® for Development, Version 1.2. Carnegie Mellon University. August 2006.

U.S. Department of Energy. Office of Procurement and Assistance Management. Source Selection for the Source Selection Official. 29 September 2005.

8 Contract Source Selection

8.0 Overview

The overall objective of source selection is to competitively select a source that meets program objectives and requirements. The results of the software source assessment will support the decision process used in competitive contracting to select the proposal that offers the best value to the government. "Best value" refers to the expected outcome of an acquisition that, in the government's estimation, provides the greatest overall benefit to the government in response to the government's requirements.[1] The results of this evaluation will be combined with other proposal evaluations, including evaluations of the cost proposal.

Every source selection begins with a basic planning stage (see Chapters 5 and 6). Initially, a capability must be identified and funding must be obtained. Requirements may include aircraft, missile, training systems, components, software, technology advancement projects, maintenance and logistics, management training services, other service contracts, etc. A strategy for fulfilling the capability that includes schedule development must be created and the type of source selection to be conducted must be determined. Key source selection personnel need to be identified. Documentation that justifies and plans the acquisition must be developed and approved by the responsible acquisition authorities, such as the Program Manager (PM), Milestone Decision Authority, and/or others as appropriate. Market research needs to be conducted to assess the feasibility of the desired capabilities and system concept. This may require release of Requests for Information (RFIs) to secure the advice of relevant industrial sources. For software, this will involve market research to assess the viability of current software engineering practices to achieve the desired capabilities. A Source Selection Plan (SSP) is developed along with the solicitation package that will be included in the Request for Proposal (RFP). Ultimately, the RFP is developed and released, proposals are received, the evaluation is conducted, the source is selected, and contract is awarded.

The source selection activity is guided by the SSP, which contains, among other items, the process to be used for the evaluation. Source selection is also guided by the RFP, including the description of the work effort (e.g., Statement of Work (SOW), Statement of Objectives (SOO)) and the specific evaluation factors contained in RFP Section M. See Chapter 7 for a discussion of what offerors must include in their proposals. The source selection team relies on the SSP to provide specific guidance on how to evaluate offerors' proposals.

Software source selection is an integral part of overall system acquisition efforts and therefore, the focus remains on increasing warfighting capabilities in a shorter time at a reduced cost. It is essential to consider the overall set of proposed business and technical practices to achieve this goal. An important aspect of such an evaluation is the offeror's approach to deliver systems that have maximum use of pre-existing

[1] Department of the Navy. Naval Air Systems Command. NAVAIR Instruction 4200.39B, Principles And Procedures For Competitive Source Selection. 5 September 2003.

naval assets, as achieved through the definition of modular, interoperable systems that adhere to open standards. The proposed software development approach needs to incorporate an open approach that exploits and increases opportunities for innovation and competition, enables reuse of components, facilitates rapid technology insertion, and reduces maintenance constraints.[2] Extensive reuse of existing components must however be balanced against any potential Information Assurance (IA) threats that may exist through vulnerabilities within these components. In addition, openly publishing interfaces to critical systems may provide avenues of attack if these systems are not protected from intrusion, so this practice must be balanced against these persistent threats. Hence, in assessing the software development approach, the mechanisms and controls to be employed must be assessed to ensure that sufficient protection is provided to the systems and their operation. This will include adherence to software and information assurance policies and standards.

Chapter 5 discussion of software development techniques also includes source selection evaluation information specific to software development techniques and should be reviewed in concert with this chapter when preparing for and conducting a software source selection.

8.1 Source Selection Software Evaluation Guidance

ASN(RD&A) policy, as defined in the 17 Nov 2006 memorandum "Software Process Improvement Initiative Contract Language" (see Appendix B), defines the software development-related information to be provided by offerors in their proposals regarding their technical approach and their background (RFP Section L). The policy also identifies the factors to be used when evaluating these proposals (RFP Section M). To the extent practicable, any additional evaluation factors should be limited to commonly used factors (e.g., a demonstrated understanding of the government's requirements, sufficient maturity of the critical technologies (as determined by a Technology Readiness Assessment (TRA)), technical approach, management capabilities, technical experience, personnel qualifications, manufacturing plan, facilities and equipment). The source selection activity is guided by these two RFP sections.

The principal purpose of any software evaluation process is to:

- Determine which proposals are acceptable and/or within the competitive range.
- Provide a sound basis for the Source Selection Authority (SSA) to make an informed and reasoned selection by:
  - Presenting a clear picture of the issues considered during evaluation by identifying areas of uncertainty as well as those which provide substantial assurance of a successful outcome.
  - Listing the strengths, weaknesses, and risks of the proposed approaches.[3]

For software, the offeror's software development approach is a key evaluation factor in the overall source selection decision. Additional factors include items such as previous related experience, related process experience, process maturity, cost (or price), historical productivity rates for similar types of software, and schedule. For software development, human capital is also an underlying factor. Determining the balance between all available factors is a key part of creating an effective evaluation methodology. This section discusses evaluation of key software factors.

[2] Navy Office of Information, *RhumbLines,* 12 December 2006.
[3] Department of the Army. AMC Pamphlet No. 715-3, Contracting For Best Value—A Best Practices Guide To Source Selection. 1 January 1998. http://www.amc.army.mil/amc/rda/rda-ap/ssrc/ssp_toc.htm

8.1.1 Software Development Approach

As part of the source selection process for software acquisitions, the government needs to determine if the proposed software development approach is the best one for the system to be developed. This should be the key factor. If a proposal describes a well-documented, frequently-used approach that does not match the needs of the system to be developed, then such an approach should not be selected since its implementation will not match the needs of the government. For example, if technology is not mature, a spiral approach would be preferred over the waterfall approach.

The proposed Software Development Plan (SDP) and the SDP rationale, submitted as a part of the offerors' proposals, are key documents that help the government make this determination. The assessment of the "best approach" should be based on the technical aspects of the planned work effort, and should be made based on the expert judgment of the source selection subject matter experts, including the software engineers on the team as well as the associated technical domain experts. The risk associated with each proposed approach needs to be assessed and compared across all submitted offers. This risk addresses the suitability and executability of the proposed approach in the context of the expected work effort, and involves multiple aspects of the planned effort. A partial list of the contributors to the risk include:

- Completeness of the approach regarding activities and tasks to what will be needed by the development effort;
- Selection of techniques that are suitable for the system to be developed;
- Compatibility and interoperability of the tools to be used; and
- Suitability of the process model selected to systems needs.

Because the SDP will govern software development, it is important that it contain detail sufficient to provide an objective basis during source selection for assessing the proposed approach, and during development for governing developer activity and monitoring adherence to the plan. Although it is not necessary that the proposed SDPs are complete, they need to contain sufficient information to be able to determine the quality of the planned development approach and its appropriateness to the system to be acquired. Vague and high level SDPs should be deemed less acceptable as they suggest a lack of a standard corporate process and an uncertainty regarding the appropriate activities, tasks, and techniques to be applied.

ASN(RD&A) policy requires that Electrical and Electronic Engineers/Electronic Industries Alliance (IEEE/EIA) Standard 12207 be used as the framework for the SDP (see Appendix B).[4] Although IEEE/EIA Standard 12207 covers the entire software life cycle, contractors are required to use only those parts that apply to the software development efforts they are proposing to perform under the applicable contract. In particular, for software development efforts that are of small value compared to the total value of the contract, contractors can satisfy the intent of the policy language with minimal effort by developing tailored SDPs to address the specific work to be performed (see Appendix C).[5]

[4] Department of the Navy. ASN(RD&A) memo. Software Process Improvement Initiative Contract Language. 17 November 2006.
[5] Department of the Navy. ASN(RD&A) memo. Software Process Improvement Initiative (SPII) Guidance for Use of Software Process Improvement Contract Language. 13 July 2007.

When evaluating an SDP, the government should ensure that the SDP has all required information as defined in the SDP Data Item Description (DID) (see Appendix L) and tailored by the RFP, and describes how the development will be conducted including (but not limited to):

- The life cycle process model (the "strategy");
- Software project planning and estimation;
- Risk engineering;
- Configuration management and change control of process and products;
- How the activities will be performed;
- How the tasks for each activity will be performed;
- The specific techniques and tools to be used for each activity;
- The criteria and constraints under which the development will take place;
- The organization chart of who will perform the work and which are the critical billets or functional roles; and
- The processes and measurements that will be provided for the core software metrics required in the RFP.

As noted by ASN(RD&A) policy,[6] evaluation should include an assessment of the suitability of the offeror's proposed approach to the expected work content of the software development effort. Comparing the offeror's proposed software size, schedule, budget, and staff based on their interpretation of the RFP to similar Navy projects will help in understanding the risks to the software elements of the program. The government can also assess the risk associated with applying the proposed approach to the planned work effort, based on the offeror's proposal approach to performing a project with a similar process. An evaluation could include the assessment of the credibility of the offeror's proposed approach to items such as labor hours, list of materials and project schedule, as long as the software Earned Value Management System (EVMS) or Work Breakdown Structure (WBS) to that level was in the RFP. The evaluation could appraise work records to determine the relevant work processes and procedures and the nature, difficulties, uncertainties and risks associated with performing the work required under the prospective contract.

The government should refer to the SDP rationale to understand the basis for choices made in the SDP. In particular, if any activities, tasks, and/or techniques are omitted from the SDP, the rationale must explain the reason why they are omitted. It will be important to evaluate the SDP to determine if the offeror's SDP meets the requirements of the RFP or if they just submitted a "boiler-plate" SDP.

Based on the Navy's evaluation of the proposed SDP and other aspects of the proposal, prior to contract award the Navy may request all offerors to clarify and resubmit their SDPs. As stated above, the proposed SDP should not be expected to contain the same level of detail as the SDP submitted post-award in accordance with the Contract Data Requirements List (CDRL). The same level of detail and completeness is not expected to exist at the proposal stage. Rather, the offerors will describe what they judge to be the most important aspects of their software. This judgment will provide an indication of the offerors' approach

[6] ASN(RD&A) memo. Software Process Improvement Initiative (SPII) Guidance for Use of Software Process Improvement Contract Language.

and is to be used as a part of the selection process. After contract award, the SDP becomes a CDRL item subject to government approval as well as an enforceable contract requirement. Specific RFP clauses should delineate the detail for each key factor and the frequency of updates to support this management. The remaining details will be added per the RFP and SOW or SOO, government comments and suggestions will be incorporated, missing items will be included, and the initial operating version of the SDP will be delivered as a program CDRL. This delivery of the SDP should take place as soon after award as feasible, but no later than 30 days prior to commencement of software development activity to allow for review and approval beforehand.

8.1.2 Related Systems Experience

Corporate experience and past performance are two additional evaluation factors to use in competitive source selection. Corporate experience is a measure of the extent to which the offeror has recently performed the same or similar work, while past performance is feedback regarding the quality of the offeror's work and the satisfaction of their customers. These are separate but equally important factors. The Contractor Performance Assessment Reporting System (CPARS)[7] is a tool used to help evaluate corporate experience and past performance by documenting contractor performance on systems and non-systems contracts.

8.1.2.1 Corporate Experience

It is important for the offerors to have direct experience in performing software work on systems of a nature similar to the one being procured. Knowledge of the application domain is crucial to being able to successfully develop systems that will operate within that domain. Without such experience and knowledge, much of the development effort will potentially be spent on staff coming up to speed. For example, extensive, successful experience in developing database-oriented systems does not directly assist in developing embedded real-time systems. Associated with this factor is determining the extent of experience that the key staff have in related systems. Likewise the credibility of the cost or price is questionable if the corporate experience provided no basis for cost because they've never done this before. In today's larger merged corporations, the fact that one division did such work does not mean that another division has any of that experience; in fact the opposite may more often be true. If there is concern in this area, it will be important to request the rationale behind the software estimate to include the equation for size estimation, productivity rates, and experience levels assumed for the software staff. In general, contractor corporate experience assessments or evaluations provide a record, both positive and negative, for a given contract during a specified period of time.

8.1.2.2 Past Performance

Past performance evaluation examines an offeror's past performance on other contracts to determine its ability to perform as proposed. Emphasizing past performance in source selection has become a standard practice in the last few years and can help ensure that the winning teams (prime contractors and major subcontractors) are likely to meet performance expectations. In fact, past performance must be evaluated on all negotiated competitive acquisitions expected to exceed $100,000, unless the Contracting Officer documents why past performance is not a relevant factor for the particular acquisition. When evaluating performance data, consideration should be given to items such as:

[7] To gain access to CPARS, visit: http://www.cpars.csd.disa.mil.

- The relevancy, complexity and ultimate mission success of the contract.[8]
- Data presented by the offeror, data in existing government databases, data from cognizant procuring and contract administration offices, data from on-site surveys, and data from other customers or the offeror.[9]

Evaluating past performance requires gathering and evaluating additional information not typically found in proposals. To access past performance data often requires making inquiries of third parties about contractor performance on other contracts and evaluating responses. Again, ensure that the data relates to the company's division in the proposal doing the work and not some other division that once did similar work within a recently acquired partner or another profit-center (profit centers handle the estimating of work and are generally separated even within the same "company" so that both experience and price-bases can vary widely despite a common "name"). It would be important to know if there was an independent assessment of the offeror's software team to verify that the team was performing at the Capability Maturity Model Integration (CMMI®) level the team claimed they had achieved to give credence to the past performance. Some offerors claim to be at a higher CMMI® level than they are actually performing, which affects efficiency and quality. See section 8.1.4 for more detail.

8.1.3 Related Process Experience

Offerors need to provide a description of previous experience in developing software using the same or similar processes and approaches as they have defined in the SDP. This is important because, historically, developers who apply processes in which they have little or no experience encounter higher levels of risk as they learn how to apply the processes and techniques. As a part of this description, offerors should describe the extent to which key personnel who contributed to these previous efforts will be supporting this effort.[10] The organization chart delivered with the proposal (as a part of the SDP) should specify where these personnel are assigned. The offerors need to plan to report any changes to their assignments to the government.

During source selection, evaluation could be based on the extent, depth, and quality of recent corporate experience in performing the same or similar work. Particular weighting factors can be placed on the degree to which the offeror's management can demonstrate a concise relationship between its past process data and systemic improvement efforts and relating those improvement efforts to the systemic improvement management approach to be used during execution of the proposed contract.[11]

8.1.4 Process Maturity

During source selection, it is important for the government to assess the maturity of the offerors' processes as well as their suitability to the proposed work effort. In this context, a mature process is one that is defined, documented, and implemented, as evidenced by audit data or analyses, and which has been previously used by the offeror. It is a risk if an offeror is proposing to use a development approach that they have not previously used, that the government is not familiar with, or that is not adequately documented. In such a situation, the offeror's score should potentially be reduced for this factor. Providing information

[8] Defense Acquisition University Continuous Learning Module. *CLC 007, Contract Source Selection*.
[9] PBSA Course, developed through a partnership between the Institute for Supply Management (ISM), the National Contract Management Association (NCMA), and the Department of Defense.
[10] ASN(RD&A) memo. Software Process Improvement Initiative Contract Language.
[11] PBSA Course.

relating to the maturity of the process and the offerors' experience levels with the proposed process can help in raising the confidence that the offeror is capable of successfully implementing the proposed development approach. In this situation, the offeror's score should be increased for this factor. Note, however, that simply proposing a new process or technique should not exclude an offeror – it is hoped that industry will provide innovation, and by its nature, innovation results in a change from the previous way of doing business. The overall context of the approach must be evaluated, with all associated factors weighed, including expected benefits as well as the level of risk that the new process may present or old risk that it may mitigate. In many situations, a new approach may represent the best value for the government.

It is also crucial that the offeror have a viable and effective process improvement activity which guides their proactive pursuit of opportunities for process improvements based on lessons-learned. One tool for aiding in this assessment is the Software Engineering Institute's CMMI® model which defines levels of process maturity and capability (note that other equivalent maturity models exist that can be cited by offerors). A valuable source of information on how to apply CMMI® ratings is the Software Engineering Institute guidance on understanding and leveraging a supplier's CMMI® efforts, found in *Understanding and Leveraging a Supplier's CMMI® Efforts: A Guidebook for Acquirers*.

Offerors are required to submit information relating to their use of process maturity models as a part of their proposals. Specifically, offerors are required to provide:

- A description of how their proposed processes are equivalent to those processes articulated by CMMI® Maturity Level 3, or equivalent, model-based maturity model;
- A description of any previous CMMI® or equivalent model-based process maturity appraisals performed; and
- An identification of the organizational entity and location when and where the appraisal was performed, the type of evaluation, the organization performing the evaluation, and the level earned.

It is important to note that source selection is not required to perform any formal appraisals of the offerors' CMMI® capability/maturity levels. Rather, CMMI® can serve as a common reference "yardstick" for any process maturity evaluation. The program office should assess the first item listed above by ensuring that the offeror's SDP contains formal documentation of those process areas described by CMMI® for Development, v1.2, as key practices essential to levels 2 and 3 of process maturity/capability. CMMI® level 3 process areas are listed in Table 8-1 and must be fully documented in the offeror's SDP for the offer to be considered to be equivalent to CMMI® 3. As described previously, the SDP documentation shall be in the framework defined by 12207. Note that these CMMI® process areas are equivalent to the processes and activities defined within 12207.

CMMI® Process Area	12207 Process	12207 Activity	SDP Section
Requirements Management	Development	System Requirements Analysis Software Requirements Analysis	7.7, 7.9
Project Monitoring and Control	Management	Execution and Control	6
Project Planning	Management	Planning	10.2
Supplier Agreement Management	Acquisition	–	6.7
Configuration Management	Configuration management	–	10.2
Measurement and Analysis	Management	–	6.6
Process and Product Quality Assurance	Quality assurance	–	10.3
Product Integration	Development	Software integration System integration	7.13 7.15
Requirements Development	Development	System Requirements Analysis Software Requirements Analysis	7.7 7.9
Technical Solution	Development	–	7
Validation	Validation	–	7.6
Verification	Verification	–	7.5
Organizational Process Definition	(see note)	–	–
Organizational Process Focus	Improvement	–	10.6
Organizational Training	Training	–	6.5
Integrated Project Management	Management	–	6
Risk Management	Supply Management	Planning –	6.8
Decision Analysis and Resolution	Management	Planning, Execution and Control	6.1
Note – *Organizational process definition* is assessed by evaluating the offeror's previous experience in applying the proposed process			

Table 8-1. CMMI® Level 3 Process Areas

In assessing an offeror's description of previous CMMI® or equivalent appraisals including when and where the appraisals were conducted, source selection will rely primarily on the Appraisal Disclosure Statement (ADS) (if one is available) which is to be submitted as a part of the offeror's proposal. Source Selection will also rely on other information provided by the offerors in their proposal as noted in Section 8.1.2. It is important to ensure that any process maturity appraisal earned by an offeror is associated with the organization which will perform the work, and involve key personnel who were part of the work efforts appraised whenever possible.

It is not necessary for an offeror to have been appraised at any CMMI® level to be eligible for an award. There is no requirement that an offeror have any formal CMMI® appraisal in order to win an award. However, having such an appraisal may help to raise confidence in any specific offeror when evaluated as a part of other criteria, and can be used to differentiate between two offerors with comparable proposals (tie-breaker).

8.1.5 Human Capital

An additional aspect of source selection is the assessment of the quality and extent of the qualifications of the offeror's proposed key personnel and their formal and informal education qualifications. The qualifications of key personnel can be evaluated based upon the extent to which the education and experience meet the minimum qualification requirements of the labor category and then upon the extent to which the education and experience are relevant to the proposed tasks. In instances when resumes are submitted, cited references may be verified. Typically, the government will review the resumes, when submitted, against the portions of the Statement of Work pertaining to the minimum requirements for personnel in various labor categories. The resume review helps the government to assess the quality and extent of qualifications of the offeror's proposed key personnel. Education may also be rated as either satisfactory or unsatisfactory.[12] It is highly recommended that key personnel supporting software development have formal education in software engineering as a part of their degree programs.

8.1.6 Additional Guidance

8.1.6.1 Open Architecture

There are certain aspects of system development that need to be evaluated as a matter of policy.[13] Specifically, software source selections need to consider the following key Naval Open Architecture (OA) principles[14] and ensure that offerors adhere to these principles:

- Encourage competition and collaboration through the development of alternative solutions and sources.
- Build modular designs and disclose design data to facilitate evolutionary designs, technology insertion, competitive innovation, and alternative competitive approaches from multiple qualified sources. This requires offerors to provide at least Government Purpose Rights (GPR) license rights for the products.
- Build interoperable joint warfighting applications and ensure secure information exchange using common services (e.g., common time reference), common warfighting applications (e.g., track manager) and information assurance as intrinsic design elements. System and subsystem interfaces should be identified and standardized.
- Perform market research, identify, and/or develop reusable application software selected through open competition of "best of breed" candidates, reviewed by subject matter expert peers and based on data-driven analysis and experimentation to meet operational requirements.
- Ensure life cycle affordability including system design, development, delivery, and support while mitigating Commercial-off-the-Shelf (COTS) obsolescence. One suggested tactic is to exploit the Rapid Capability Insertion Process/Advanced Processor Build methodology.

Note that these principles do not exclude the possibility of offerors proposing the use of proprietary and/or commercial components, as long as the interfaces to these components are made available so that the government could contract for an equivalent component to be developed at a later time. The main goal is to

[12] PBSA Course.
[13] Department of the Navy. Deputy Chief of Naval Operations (Warfare Requirements and Programs (N6/N7)) memo Ser N6N7 / 5U916276. Requirement for Naval Open Architecture. 23 December 2005.
[14] Naval OA Strategy. 26 October 2005.

maximize the value of the acquired system(s) and to minimize life cycle costs over the lifetime of the system. Adhering to an open system/architecture approach will assist in achieving this goal.

In addition, while there is an intention to make maximum use of existing assets, such as legacy systems or other systems already deployed or in development, at times the reuse of these assets may compromise the overall performance goals of the system to be acquired. The government needs to weigh these effects against their overall plans and missions, and needs to recognize that by requiring such reuse, the government needs to accept whatever risk this reuse represents.

8.1.6.2 Technical Evaluation

The less definitive the requirement, the more development work required, or the greater the performance risk, the more technical or past performance considerations may play a dominant role in source selection.[15] Technical evaluation in particular involves the government assessment of the technical differences and the offeror's ability to perform the prospective contract successfully.

In technical evaluations, the government will consider an offeror's knowledge of the work statement in terms of the concept, purpose, importance, salient operational procedures, inherent problems and ideas for solutions. The evaluation will also include an appraisal of the contractor's knowledge of the Statement of Work's relationship to other significant information necessary to present an understanding of the contractor's ability to accomplish services (e.g. logistics, maintenance programs and databases, management approach, including approach to Open Architecture (OA), and task staffing).[16] In tradeoffs, for example, there is an assessment of each offeror's ability to accomplish the technical requirements. Often a summary, matrix, or quantitative ranking of each technical proposal using evaluation factors is included/considered.

Evaluators must examine each proposal individually in detail to measure it against the evaluation factors and subfactors in the solicitation. Evaluators ask questions, such as How much? or How well? to help assign a rating and document the basis for the rating. Normally, technical evaluations should be conducted independent of the cost/price evaluations so that technical findings and conclusions will not be influenced by knowledge of the offered costs. There may be certain instances, however, in which it may be appropriate to give the entire evaluation team access to price/cost information to ensure the best possible overall evaluation and enhance the evaluation of cost realism. Such a review can help verify perceived technical strengths, weaknesses or risks and/or ensure consistency between the cost/price and technical segments of the proposals. All evaluators must have the required functional expertise and training to evaluate the particular area of the proposal to which they are assigned. They should also be thoroughly familiar with the solicitation and the Source Selection Plan.[17]

8.2 Finalizing the Source Selection

At the conclusion of the software source selection evaluation process, a decision is made regarding whether to conduct discussions or to act on initial offers. For example, if the government desire is to have offerors revise SDPs, then it should conduct open discussions so that all offerors have the opportunity to

[15] Federal Acquisition Regulation (FAR) Part 15.101, paragraph 15.101-1.
[16] PBSA Course.
[17] AMC Pamphlet No. 715-3, Contracting For Best Value—A Best Practices Guide To Source Selection.

make improvements. If such discussions are to be conducted, a competitive range consisting of the most highly rated offers is established. The report becomes the official record documenting the logic and rationale used to arrive at the evaluation and ratings.

The SSA decision will be based on a competitive assessment of proposals against all source selection criteria in the solicitation. The SSA's decision will be documented and the documentation shall include the rationale for any business judgments and tradeoffs, including costs, made or relied on by the SSA. In fact, FAR 15.305 (proposal evaluation) and FAR 15.308 (source selection decision) require that the evaluation of the relative strengths, deficiencies, and significant weaknesses of a proposal be documented in a contract file. The report should reflect the government's evaluation consistent with the evaluation criteria stated in the solicitation. FAR 15.308, for example, states that the source selection statement should:

- Succinctly and accurately provide rational for the selection decision;
- Explain how the successful offeror compares to the other offerors based on the solicitation evaluation criteria;
- Provide the rationale for any business judgments and technical/cost tradeoffs, i.e., the benefits associated with additional cost; and
- Address any significant differences between the Source Selection Official's (SSO's) judgment and the Source Evaluation Board's (SEB's) evaluation. [18]

Appendix O provides a sample best value checklist that can be used to select the proposal that offers the "best value" to the government.[19] Even still, many government agencies will obtain assistance from outside sources during major acquisitions to ensure that the agency's best value award meets the requirements of the solicitation, standards of the legal decisions, and can withstand any protest proceeding after contract award.[20]

[18] The SSO is the official responsible for the overall management of the source selection process. The SEB is an alternate name for the source selection team, the group who evaluates and compares proposals submitted by offerors.
[19] NAVAIR Training Systems Division, *Best Value Process—Guiding Principles*.
[20] Mickaligher, Michael J., CPCM, SAS. *Best Value Contracting: Selection by Perception*. APMP, Spring 2001.

References

Defense Acquisition University. Defense Acquisition Guidebook. <https://akss.dau.mil/dag/>.

Defense Acquisition University Continuous Learning Module. *CLC 007, Contract Source Selection*.

Defense Acquisition University. Planning for Source Selection. 18 November 2005.

Department of the Army. AMC Pamphlet No. 715-3, Contracting For Best Value—A Best Practices Guide To Source Selection. 1 January 1998.

Department of Defense. USD(AT&L) memo. Risk in the Source Selection Process Working Group. 31 May 2006.

Department of the Navy. ASN(RD&A) memo. Software Process Improvement Initiative Contract Language. 17 November 2006.

Department of the Navy. ASN(RD&A) memo. Software Process Improvement Initiative (SPII) Guidance for Use of Software Process Improvement Contract Language. 13 July 2007.

Department of the Navy. Deputy Chief of Naval Operations (Warfare Requirements and Programs (N6/N7)) memo Ser N6N7 / 5U916276. Requirement for Naval Open Architecture. 23 December 2005.

Department of the Navy. Naval Air Systems Command. NAVAIR Instruction 4200.39B, Principles And Procedures For Competitive Source Selection. 5 September 2003.

Department of the Navy. Naval Air Systems Command. NAVAIR Acquisition Guide. 20 September 2006.

Department of the Navy. Naval Sea Systems Command. Acquisition Planning Guide. September 2006.

Department of the Navy. PEO IWS 7.0. Naval Open Architecture Contract Guidebook for Program Managers (Current Version). <https://acc.dau.mil/CommunityBrowser.aspx?id=105662>.

Executive Office of the President. Office of Management and Budget. Circular A-76. 29 May 2003.

IEEE/EIA 12207.0-1996. Industry Implementation of International Standard ISO/IEC 12207: 1995 (ISO/IEC 12207) Standard for Information Technology—Software life cycle processes. March 1998.

IEEE/EIA 12207.1-1996. Industry Implementation of International Standard ISO/IEC 12207: 1995 (ISO/IEC 12207) Standard for Information Technology—Software life cycle processes—Life cycle data. April 1998.

IEEE/EIA 12207.2-1996. Industry Implementation of International Standard ISO/IEC 12207: 1995 (ISO/IEC 12207) Standard for Information Technology—Software life cycle processes—Implementation considerations. April 1998.

John, William A., *Service Contracting—A desk guide to best practices (1998 edition)*. Prepared for the Navy Acquisition Reform Office. http://acquisition.navy.mil/content/view/full/3505.

Mickaligher, Michael J., CPCM, SAS. *Best Value Contracting: Selection by Perception*. APMP, Spring 2001.

PBSA Course, developed through a partnership between the Institute for Supply Management (ISM), the National Contract Management Association (NCMA), and the Department of Defense. Copyright permission has been granted for use in USACCE's Center for Excellence for Service Contracting.

Software Engineering Institute. CMMI® for Development, Version 1.2. Carnegie Mellon University. August 2006.

U.S. Department of Energy. Office of Procurement and Assistance Management. Source Selection for the Source Selection Official. 29 September 2005.

This Page Intentionally Left Blank

9 Contract Execution

9.0 Overview

This section provides guidance for some key activities to be performed by the program office subsequent to contract award for the purpose of monitoring and tracking the development effort. These activities depend on the integrated information environment established to facilitate government access to development artifacts, the program's Software Process IPT established at contract inception, and the use of the Software Development Plan (SDP) that defines the activities and tasks to be performed (see Chapter 7:Contract Solicitation and Chapter 8:Contract Source Selection for more information). A key avenue for facilitating the monitoring of contractor progress is the technical review process defined for the program. The technical review processes are established by Program Executive Offices (PEOs) and Systems Commands (SYSCOMs) as appropriate for each acquisition effort; this guidebook is not intended to replace those technical reviews. Rather, this chapter discusses some important aspects that need to be monitored and some key activities to be performed by the program office and which extend throughout specific technical review requirements for the purpose of monitoring and tracking the development effort.

9.1 Naval Technical Review Team

Upon contractor award, the program office and other government stakeholders should identify a set of staff who are software Subject Matter Experts (SMEs) and who will serve as reviewers and monitors of the software work effort. This staff can be selected from program office personnel, lab staff, independent contractors, and consultants. Some will be assigned to full-time responsibilities, while others will become involved at specific points during the development effort, such as major reviews or events. The Technical Review Team (TRT) will be a major resource for the program office during the software work effort. The individual SMEs may also be changed out or in depending on the phase and their expertise.

9.2 Software Process IPT

Chapter 7 of this guidebook recommends the establishment of a program Software Integrated Product Team (IPT) to assist in monitoring and improving the software processes and practices. Upon contract award, the contractor and the government jointly establish this IPT, consisting of both contractor and government representatives, co-chaired by the program office's Chief Software Engineer and the contractor's Chief Software Engineer (to include subcontractor Chief Software Engineers, or equivalent critical billets, as appropriate).

The Software IPT may be tasked to define, document, monitor, and improve the software development approach being used for the software effort. See Chapter 7 (Section 7.4.6) for a more detailed discussion of IPT activities.

9.3 Shared Development Environment

Chapter 7 (Section 7.4.5) of this guidebook recommends a requirement for the contractor to establish a shared development environment that provides continuous, real-time access to all software development products and artifacts, in-work, intermediate, and final. This environment can serve as a primary source of information to support the on-going monitoring of contractor progress, metrics or measurement evidence, and the quality of the emerging products. Since it will likely contain all of the associated Software Development Folders (SDFs) used by the developers, it will be important for the government to examine only those areas that are of particular interest, depending on the status of the work effort. This will ensure an efficient approach to monitoring. This will also support a “statistical” approach to artifact analysis, allowing government TRT members to examine various products whenever the need arises. Note that there will never be the need for the developers to “release” any items to the shared development environment. Rather, all artifacts are already in location, available for government inspection. While this very open shared environment is recommended, it can entail additional costs. If the government decides to require a facility with less government access, it will result from a trade-off between the visibility and the additional costs. Environments with less visibility, such as government access to only released items, provide some benefits but do not provide the same level of insight for program offices.

9.4 The Role of the Software Development Plan in Monitoring the Developer

According to Navy policy, a developer’s Software Development Plan (SDP) is to provide complete documentation for the developer’s software approach. The SDP is required to be a government-approval contract deliverable item to ensure that the government understands and concurs with the approach to be taken, and is updated whenever there is a change to people, process, schedules, or tool sets. Developers may decide to document their approach in multiple plans, such as Software Configuration Management Plans and Software Quality Assurance Plans. In such a situation, these plans are considered to be part of the SDP by extension.

Policy requires that the SDP shall follow the overall framework defined in Electrical and Electronics Engineers/Electronic Industries Alliance (IEEE/EIA) Standard 12207.0 and 12207.1, and include a description of all planned processes, activities, and tasks, as well as the specific techniques and tools to be used. Also to be included is a description of the specific products resulting from these tasks, and the schedule for when these are planned to be produced:

> “Work breakdown structure of the life cycle processes and activities, including the software products, software services and nondeliverable items to be performed, budgets, staffing, physical resources, software size, and schedules associated with the tasks;…”[1]

(In the context of the SDP, the budgets are to be specified in terms of planned labor (staff-months or staff-years) rather than dollar costs. By using labor, monitoring progress is simplified and can more easily be correlated to traditional cost models.)

[1] IEEE/EIA 12207.1-1996. Industry Implementation of International Standard ISO/IEC 12207: 1995 (ISO/IEC 12207) Standard for Information Technology—Software life cycle processes—Life cycle data. April 1998. Section 6.11.

Because of the comprehensive nature of the information to be documented by the SDP, it is not expected that it will be delivered as a single volume. Instead, information relating to software development may be documented in multiple volumes, the collection of which shall form the SDP. Examples of separate volumes include (but are not limited to):

- Software configuration management;
- Software quality assurance;
- Software schedule, Work Breakdown Structure (WBS) or Earned Value Management Schedule (EVMS);
- Software processes and tools;
- Software build plan; and
- Program Protection Plan (PPP) with Critical Program Information (CPI) list.

Additionally, separate SDP volumes may be developed to document different development approaches due to:

- The participation of subcontractors in the software effort, and/or
- The need to adapt to different technical aspects of subsystems/components within the system to be developed.

The SDP will serve as the primary standard or baseline against which contractor progress, contractor adherence to process, and emerging product quality will be assessed. To facilitate this assessment, continual monitoring of contractor activity is facilitated through the use of the shared and integrated Information Development Environment, described in Chapter 7. Appearance of various artifacts resulting from the execution of the planned activities and tasks will assist in tracking progress and adherence to plan. The program office should consider any variation from the plan defined in the SDP as a potential risk to program success. Specifically the government should review all deviations and waivers to the approved SDP processes, not in real time however as that cost is prohibitive. At all program technical reviews, defined and scheduled in accordance with the Navy technical review process, the program office and program stakeholders should use the SDP as the benchmark against which the actual software activity performed (or not) by the contractor team is assessed.

9.5 Naval Open Architecture

The development of open systems is a core goal of the Navy. The potential for cost savings, performance improvements, and schedule risk reduction through multiple source availability is key to the Navy's future. The Naval Open Architecture (NOA) initiative has defined several key attributes of NOA-compliant development efforts, including both programmatic and technical characteristics. As a part of monitoring the software effort, the program office should ensure that the contractor's original plans for achieving NOA-compliant open systems are being followed, and that the objectives are being realized. This includes any stated goals for the reuse of existing assets, the maximum exploitation of smaller vendors who provide specialized and competitive skills, and the achievement of systems designs that are highly modular, supporting the ability to replace components at low levels and enhancing maintainability. Such system architectures need to exploit open and published standards to ensure that optimal levels of reuse can be achieved.

9.6 Intellectual Property

As described in Chapter 7, the application of Intellectual Property (IP) terms and conditions needs to be consistent with the Defense Federal Acquisition Regulation Supplement (DFARS), providing to the government the maximum license terms possible. Such terms support the general open architecture goals by minimizing proprietary information within the system. During the conduct of the software effort, the government needs to ensure that the appropriate IP (e.g., Proprietary) markings are applied to all development artifacts, including presentation materials, software code, and other items. In general, no such materials should be marked as Proprietary without explicit government approval. The program office should be aware that once marked by a contractor, it generally takes money to remove the mark and unless the government requests that the markings are removed within a specific time period (usually three years), the material reverts to the license levels specified by the markings.

9.7 Software Risk

Management of a software effort is essentially the management of risk. Due to its nature, software development always carries a level of risk that is proportional to its level of precedence. Systems that implement significant advances in technology and performance over previous systems naturally carry more risk than systems that slowly evolve. Developing software however is always unprecedented and hence always presents risk that must be carefully monitored and mitigated. As such, it is an indication of potential pitfalls if a program reports that its software risk is low.

For the purposes of this guidebook, a risk is a potential change of some aspect of a project or of its environment that, should it occur, will adversely affect the project's likelihood of being successfully completed. The change, if it occurs, is initiated by an event that is often called the risk trigger. The adverse effect to the project that is caused by the change is called the risk impact or consequence. The probability that such an event will occur is called the risk likelihood. If the change occurs, the risk is no longer potential. At that point, it is called a problem and the program office will need to deal with the consequence(s). Once a risk becomes a problem, a new risk arises associated with the likelihood of the measures applied to mitigate the effects of the problem. Hence, problems need to be tracked the same way as risks.

For any risk that is identified, programs need to define a mitigation plan that describes the actions to be taken should the risk occur. Initially, at program inception, known risks need to be defined in the program plan. Along with these risks, metrics that are to be used to track the risk need to be defined, as well as mitigation actions necessary to minimize or eliminate the impact to the program's success. Risk triggers lead a change so identifying risk trigger events is key to avoiding a risk. When risks are identified during development, the mitigation activities generally must be in addition to those activities already in place for the program. Risk mitigation strategies can take many forms. Some can focus on minimizing the likelihood that the risk trigger will occur. Others can focus on minimizing the effects of the impact, or on shifting the time when the impact will be felt to a point when the effects are minimized.

Programs that involve software effort should define and follow an active and viable risk management activity that operates continuously until the retirement of the system. This risk management process should be completely integrated into the program management process to ensure that risks are identified and mitigated in a timely manner.

For software, the SDP will define the activities, intermediate products, schedule, and all other aspects of the expected work effort. SDPs must conform to the framework defined by IEEE/EIA Standard 12207. Part of the required information is a description of the activities and tasks to be performed, the processes to be used, the specific products resulting from these tasks, and the schedule for when these are planned to be produced. Incorporated into the SDP are the descriptions and mitigations of known risks. During development, additional situations may arise that would also present risk to the effort. These need to be identified, tracked, and mitigated as appropriate. Correlating the details in the plans with the likelihood of being able to successfully follow the plan is a key part of risk management. Government acceptance of any plan requires that the inherent risk is fully understood and accepted, and that adequate mitigation plans are fully integrated into the plan. A government approval of the SDP, or the constituent volumes, acknowledges the stated risk and accepts the proposed mitigations as adequate.

During development, the government should continuously compare the elements defined in the SDP against the actual work effort. Variations from the plan should be specifically tagged for analysis and risk assessment. Some variations might be benign while others might represent an emerging risk item which must be tracked and, if necessary, mitigated.

9.7.1 Software Risk Management Focus Areas

Often programs that are in compliance with their SYSCOM's risk management policy still lack the robustness provided by industry standard practices, especially when it is scaled up to enterprise-wide systems of systems and the failure to expand a SYSCOM risk management process to other stakeholders becomes very obvious. The guidance in the following paragraphs is intended to provide Program Managers (PMs) focus areas for early risk identification and mitigation efforts. PMs should carefully examine the following topics, deliberate on their impact to the overall program requirements, make a commitment to successful software risk management, and ensure the allocation of associated budget, staffing, and schedule to implement the guidance. After taking those initial steps, program managers should exploit industry lessons learned and best practices knowledge bases (as recommended by Naval Research and Advisory Council (NRAC) 2006) through a comparison of their existing risk management process and issues against industry standard processes as described, for example, in the Project Management Body of Knowledge (PMBOK®) or the CMMI®. As recommended by Defense Science Board (DSB) 2000, the comparison should take the form of an Independent Expert Review to ensure total objectivity.

Individuals with experience or training in a given source of risk should be given responsibility for developing elements of risk from a risk source taxonomy and presenting their findings to the entire program team for adjudication at risk assessment meetings. The product of the meeting will be a comprehensive list of risks from all sources and potential trigger events. Follow-on activities would include qualitative and quantitative analysis of each element, decisions about which elements can/cannot be mitigated, mitigation planning, and so on. The first step, however, is to minimize "gotchas" by considering all potential sources and triggers of risk. These program-wide risk assessments should be updated in the same way as the initial assessment, that is, annually, preceding major technical reviews, or prior to milestone decisions.

9.7.1.1 Acquirer and Supplier

The government program office and the contractor/supplier will each have elements of risk that remain in-house, but to the greatest extent possible, the acquirer and the supplier should integrate their risk management processes. Integration might include: standardized criteria for ranking and prioritization; a single prioritized risk list; both agencies participating together in mitigation activities; standard reporting formats; a common glossary of terms; standard methods of analysis; joint Risk Review Boards; and a shared risk information database.

9.7.1.2 Software Risk Strategy

Program offices should develop a comprehensive risk management strategy that includes software-related risks to a level that can be adequately monitored and measured. Specifically, individual software-related risks should be decomposed, tracked, and addressed to a level that is granular enough to provide adequate management insight, realistically scheduled, and are conveyed in budgeted mitigation planning. Risk plans should be documented and vetted for approval with experienced SMEs. Controlled updates to the risk management best practices and procedures should be maintained.

9.7.1.3 Risk Identification

Documented risk plans should include the detailed process for how risks will continue to be identified and managed. All relevant stakeholders, both by organization and by functional discipline, including software stakeholders, should also be identified in the Risk Management Plan. These stakeholders should be engaged in clarifying the software risks and mitigating them.

9.7.1.4 Risk Handling

Program Managers are encouraged to establish Risk Management Manager as a critical staff role and assign a specific person to that role (see Chapter 4, Table 4-1). The Risk Management Manager's authority and responsibilities should be specified in a charter. A Risk Review Board with a defined membership that is cross-functional and adequately trained in the risk process, and that is directed to meet on a periodic or event-driven basis is also valuable in handling risk. The government team, the supplier team, and other program stakeholders should be included in the membership, as appropriate. The Risk Review Board should categorize, evaluate, quantify, and prioritize the risks, generate and monitor timely trigger conditions and mitigation plans, assign personnel and due dates for risk mitigation, track implementation in a structured manner, and report risk management activities directly to the PM.

9.7.1.5 Risk Database

A risk database should be defined and established that documents and tracks all risks, including a methodology for providing a segregated view of all the software risks and a methodology for communicating risks, mitigation plans, and associated metrics to all team members from the working level to executive level stakeholders. The database can be used in generating risk metrics for periodic status reporting to senior management. Staff-appropriate training for using the risk database tool(s) should be provided. Further, efficiency should be sought so that information entered once is compatible with outputting many different and disparate reports such as those feeding forward to milestone reviews, safety and security risk reports, and Secretary of the Navy Note (SECNAVNOTE) 5000[2] pass and gates reviews and Probability of Program Success (PoPS) measures.

9.7.1.6 Risk Environment

An environment for openness in identifying realistic consequences to foreseeable risk triggers and risk issues is critical for effective risk management. All stakeholders should be encouraged to identify risks and

[2] Department of the Navy. DASN(RD&A) ALM. SECNAVNOTE 5000, Department of the Navy (DoN) Requirements and Acquisition Process Improvements. 26 February 2008.

program offices may want to determine and implement incentives for risk identification, particularly when the risks are associated with critical program goals and objectives.

9.8 Requirements Development and Management

A leading cause of acquisition program breaches is requirements growth. Acquisition programs progressively elaborate, that is, new information appears over time that justifies changes to baseline configurations and estimates. A certain amount of requirements growth will occur, but rigorous processes for initial requirements development and the management of requirements is necessary to ensure that any requirements defined later in the development effort do not disrupt the emerging system design.
The guidance in the following paragraphs is intended to assist PMs in ensuring that all customer requirements, product and product component requirements, and interface requirements are clearly defined and understood by program stakeholders and requirements are managed. Risk worsens with failure to capture all requirements, failure to clearly define requirements, failure to engage stakeholders in review and approval of requirements, and failure to manage existing requirements or derived requirements. Any program work being performed should be directed toward meeting those requirements – no more, no less, and requirements growth should be contained.

9.8.1 Acquirer and Developer/Integrator

The government acquiring program office should be an equal partner with their software developer(s) and/or integrator(s) in the requirements development and management processes. The government should negotiate an integrated teaming arrangement whereby they have complete insight into the vendors' processes and tools for defining and refining requirements and managing the changes to them. They should be included as a key player within requirements definition and maturation processes. The acquirer/developer/integrator team should identify all relevant program stakeholders and involve them in these processes.

9.8.2 Plan for Management, Then Plan for Development

Requirements management is a critical aspect of any software acquisition. Comprehensive, up-front planning for overall requirements management, including software considerations, at the system level must include specific management considerations for software-specific requirements. Systems level planning should be immediately followed with detailed plans for software requirements development, including the allocation of system and interface requirements to software, with backwards comparisons into the system and systems of systems architectures. Requirements management and development plans should be established and maintained, including integration of controlled updates to requirements-related best practices and procedures.

9.8.3 Lifecycle Considerations

Systematic execution of requirements definition and system requirements flow-down, including end user needs, should be defined in detailed plans. These should be matched to the WBS and documented versus labor hours. The entire system lifecycle model and its decomposition, including software subsystem development and acquisitions through various spirals, increments, and block upgrades, should be clearly mapped out and addressed, including dependencies, linkages and interfaces.

9.8.4 System Requirements

All relevant stakeholders should be identified and engaged early in the planning. This includes all related technical stakeholders (e.g., software and hardware experts, Technical Warrant holders, etc.). Clearly defining the methods for eliciting the operational needs, expectations, system capability requirements, systems of systems interface requirements, safety and security requirements, and constraints of stakeholders will support identification, consensus, and commitment to achieving system requirements.

9.8.5 Capabilities Versus Requirements

The purpose of the Joint Capabilities Integration and Development System (JCIDS) process is to govern the overall identification of system needs and implementation. Accordingly, the mechanisms used to define capabilities, and to refine these into system and subsystem requirements, need to be fully understood and carefully applied. The Initial Capabilities Document (ICD) or Joint Capabilities Document (JCD) will provide the most general capabilities of the system. The Capabilities Development Document (CDD) specifies the system technical performance criteria of the system that will deliver the capability that meets operational performance criteria specified in the JCD or ICD. The Capabilities Production Document will outline the capabilities to be tested at the system level, usually by Operational Test & Evaluation (OT&E).

While the operational force thinks in capabilities, the system and software engineers think in terms of requirements and functions. There is a risk that as capabilities are translated and derived into requirements and requirements are allocated to functions, the initial capability can easily be misinterpreted and misdefined as they are refined into allocated requirements. Therefore the review process validating the allocated requirements to lower level software components and functions requires experienced SMEs to ensure that the transformations preserve the original intent as expressed in the capabilities. Experience has demonstrated that many expensive changes to systems could have been avoided had this process been better managed.

9.8.6 Software Requirements

Establishing an agreed-upon allocated software baseline requires engaging all stakeholders, particularly software-savvy ones. Program offices should ensure that:

- All software requirements are unambiguous, implementable, and verifiable (by test, analyses, or inspection);
- Each requirement specifies a single, clear requirement for tracking purposes;
- Requirements groupings are documented to support the development and integration of configuration components, subsystems, and interfaces; and
- The full set of requirements, taken as a whole, is complete.

Note that through the development process, many requirements which start out as being not fully defined due to lack of knowledge will mature into clearly stated precise specifications. This level of knowledge must be explicitly recognized to ensure that the initial requirements sets are complete.

Models and simulators must be designed, configuration controlled, and also, under Secretary of the Navy Instruction (SECNAVINST) 5200.40, certified. Requirements and architectures that cannot be modeled will result in delayed risk mitigations. Additional efficiencies are realized if these models, simulators, and stimulators are also open source or non-proprietary.

9.8.7 Refinement and Traceability of Requirements

Concepts of Operations (CONOPS) and operational use-case scenarios should be rigorously applied in order to refine requirements and provide realistic estimates of the cost, schedule, and size for software implementation and even maintenance or obsolescence of COTS components later. Including nominal, off-nominal, and unexpected scenarios in a thorough analysis ensures a comprehensive set of repeatable estimates. Definition of measurable software WBS elements which support meaningful monitoring of software performance and acquisition will contribute significantly to traceability of requirements. Also of value in refinement and traceability of requirements is inclusion in the WBS of interconnected software work packages across the full life cycle to a level that supports bi-directional traceability of software requirements from initial system requirements analysis down to system acceptance testing and back.

9.8.8 Requirements Database

Program offices are encouraged to define and establish a requirements database that documents all software code and interface requirements and their bi-directional traceability, including a methodology for providing a segregated view of all the software requirements (including metrics for percent implemented/coded and percent tested/completed/passed/failed). The ability to assign special importance tags or criteria to individual or groups of requirements will add efficiency for mission-critical, safety-critical, and security status. For programs that are relatively small-scale and/or programs with a relatively short acquisition life cycle, a requirements list, vice database, may be sufficient. Appropriate training for using the requirements database tool(s) should be provided.

9.8.9 Knowledge, Skills, and Abilities

Software risk management, requirements development and requirements management requires domain expertise and should be adequately reflected in the staffing plan. Appropriate staff personnel should be assigned for allocating system requirements to software in an implementable manner that will achieve operational intent. See Chapters 3 and 4 for further discussion of staff personnel knowledge, skills, and abilities (KSA) as well as related training and assignment timing issues.

9.8.10 Requirements Changes

Growth in cost and schedule due to requirements changes arise from multiple sources, including: the need to update the documentation and the software associated with the requirement; changes required to modify the software surrounding the code changes made to incorporate the changes; and the testing required to either repeat any previously performed verification and validation or to develop new test/verification procedures. It is also common, and very difficult to separate, that requirements creep in costs results from other repair(s) not actually directly related to the specific change or addition the government contracts for, but have accumulated as a by-product of defect repair and misunderstanding. That is, it is common that defects due to the initial "try" are to be fixed when this new change or addition goes in and so are priced with the change order. After all, if other surrounding software is not also repaired with this change then it is not ready to receive the change. This is a source of significant growth in cost and size and cannot be seen or measured unless the other risk processes in this chapter are embedded in the program office and contractor.

Collecting associated software metrics and implementing a robust EVMS that clearly indicates when variances in performance against baselines can be traced to requirements management will assist in monitoring and managing the impact from requirements changes. See Chapter 2 for a more detailed discussion of metrics and reporting of program health.

Program offices should also provide a clear mechanism for tracking and dealing with requirements changes, including scope creep, ensuring connectivity to the software change control process. Note that requirements will always evolve over the development activity. Sometimes they are simply refined or enhanced with more detail, and this is simply part of the development process. Other times, they are changed to actually add new functions. In order to control and allow these changes, a rigorous approval process needs to be defined that includes concurrence of all stakeholders, approved adjustments to cost and schedule baselines, and associated funding. Because some requirements growth is considered a healthy program attribute, it should be anticipated and built into the program budget, usually as a management reserve. The status of this funding line should be a standard metric reported to higher level management, with reserve-remaining a key risk issue.

9.8.11 Review Board

An organizational entity (e.g., a software Requirements Review Board (RRB)) with a defined membership that is cross-functional, meets on a periodic or event-driven basis, and reports directly to senior management, can facilitate managing requirements development and implementation. The RRB should systematically address all potential changes to software requirements (and may be subordinate to a systems requirements board). Board members should be adequately trained to quantitatively assess the impacts to architecture, warranties for COTS, implementation progress, cost, schedule, and operational intent caused by requirements changes. The results of these assessments should be incorporated into the risk management process.

References

Department of the Navy. DASN(RD&A) ALM. SECNAVNOTE 5000, Department of the Navy (DoN) Requirements and Acquisition Process Improvements. 26 February 2008.

IEEE/EIA 12207.1-1996. Industry Implementation of International Standard ISO/IEC 12207: 1995 (ISO/IEC 12207) Standard for Information Technology—Software life cycle processes—Life cycle data. April 1998.

This Page Intentionally Left Blank

Acronyms

AC	Actual Cost
ACAT	Acquisition Category
ACWP	Actual Cost of Work Performed
ADS	Appraisal Disclosure Statement
AFIT	Air Force Institute of Technology
ALM	Acquisition and Logistics Management
ALSP	Acquisition Logistics Support Plan
AOA	Analysis of Alternatives
AP	Acquisition Plan
ASI	Automatic Software Inspection
ASN(RD&A)	Assistant Secretary of the Navy (Research, Development and Acquisition)
ASR	Alternative Systems Review
AT	Anti-Tamper
AT&L	Acquisition Technology and Logistics
BAC	Budget At Completion
BCA	Business Case Analysis
BCWP	Budgeted Cost of Work Performed
BCWS	Budgeted Cost of Work Scheduled
C4I	Command, Control, Communications, Computers and Intelligence
C4ISR	Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance
CA	Control Account
CAO	Competency Aligned Organization
CARD	Cost Analysis Requirements Description
CASE	Computer Aided Software Engineering
CCA	Clinger-Cohen Act
CD	Concept Decision
CDD	Capability Development Document
CDR	Critical Design Review
CDRL	Contract Data Requirements List
CFPS	Certified Function Point Specialist
CHSENG	Chief Systems Engineer
CI	Configuration Item
CJCSI	Chairman of the Joint Chiefs of Staff Instruction
CJCSM	Chairman of the Joint Chiefs of Staff Manual
CM	Configuration Management
CMMI®	Capability Maturity Model Integration
COI	Communities of Interest
COTS	Commercial Off-The-Shelf
CPARS	Contractor Performance Assessment Reporting System
CPD	Capability Production Document

CPI	Cost Performance Index
CPI	Critical Program Information
CS	Computer Software
CSCI	Computer Software Configuration Item
CSRS	Civil Service Retirement System
CV	Cost Variance
DAG	Defense Acquisition Guidebook
DASN	Deputy Assistant Secretary of the Navy
DAWIA	Defense Acquisition Workforce Improvement Act
DAU	Defense Acquisition University
DFARS	Defense Federal Acquisition Regulation Supplement
DID	Data Item Description
DLA	Defense Logistics Agency
DoN	Department of the Navy
DoD	Department of Defense
DoDI	Department Of Defense Instruction
DOORS®	Dynamic Object Oriented Requirements Systems
DRR	Design Readiness Review
DRRA	Data Rights Requirements Analysis
DSB	Defense Science Board
DT	Development Test
DUSD	Deputy Under Secretary of Defense
EAC	Estimate At Completion
ECP	Engineering Change Proposal
EIA	Electronic Industries Alliance
ESLOC	Equivalent SLOC
EV	Earned Value
EVM	Earned Value Management
EVMS	Earned Value Management System
EVT	Earned Value Technique
FAA	Functional Area Analysis
FAR	Federal Acquisition Regulations
FDD	Feature Driven Development
FERS	Federal Employees Retirement System
FMS	Foreign Military Sales
FNA	Functional Needs Analysis
FOC	Full Operational Capability
FP	Function Point
FPGA	Field Programmable Gate Array
FRP	Full Rate Production
FSA	Functional Solutions Analysis
FY	Fiscal Year
GAO	Government Accountability Office
GFI	Government Furnished Information
GPR	Government Purpose Rights
HR	Human Rights
HW	Hardware

IA	Information Assurance
IBR	Integrated Baseline Review
ICD	Initial Capabilities Document
ICE	Independent Cost Estimate
IDE	Information Development Environment
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical & Electronics Engineers
IEPR	Independent Expert Program Reviews
IER	Information Exchange Requirement
IFPUG	International Function Point Users Group
IMS	Information Management System
IOC	Initial Operational Capability
IP	Intellectual Property
IPT	Integrated Product Team
ISM	Institute for Supply Management
ISO	International Organization for Standardization
ISP	Information Support Plan
IT	Information Technology
ITA	Independent Technical Assessment
ITR	Initial Technical Review
IV&V	Independent Verification and Validation
JCIDS	Joint Capabilities Integration and Development System
JITC	Joint Interoperability Test Center
KPP	Key Performance Parameter
KSA	Knowledge, Skills, and Abilities
KSLOC	Kilo-SLOC
LR	Limited Rights
LSI	Lead Systems Integrator
MDA	Milestone Decision Authority
MOSA	Modular Open System Approach
MOU	Memorandum of Understanding
MS	Milestone
NASA	National Aeronautics and Space Administration
NCIS	Naval Criminal Investigative Service
NCMA	National Contract Management Association
NCOW	Net-Centric Operations and Warfare
NDA	Non-Disclosure Agreement
NDIA	National Defense Industrial Association
NESI	Net-Centric Enterprise Solutions for Interoperability
NIST	National Institutes of Standards and Technology
NMCI	Navy/Marine Corps Intranet
NOA	Naval Open Architecture
NRAC	Naval Research and Advisory Council
NSA	National Security Agency
OA	Open Architecture
OOP	Object Oriented Programming
OPEVAL	Operational Evaluation

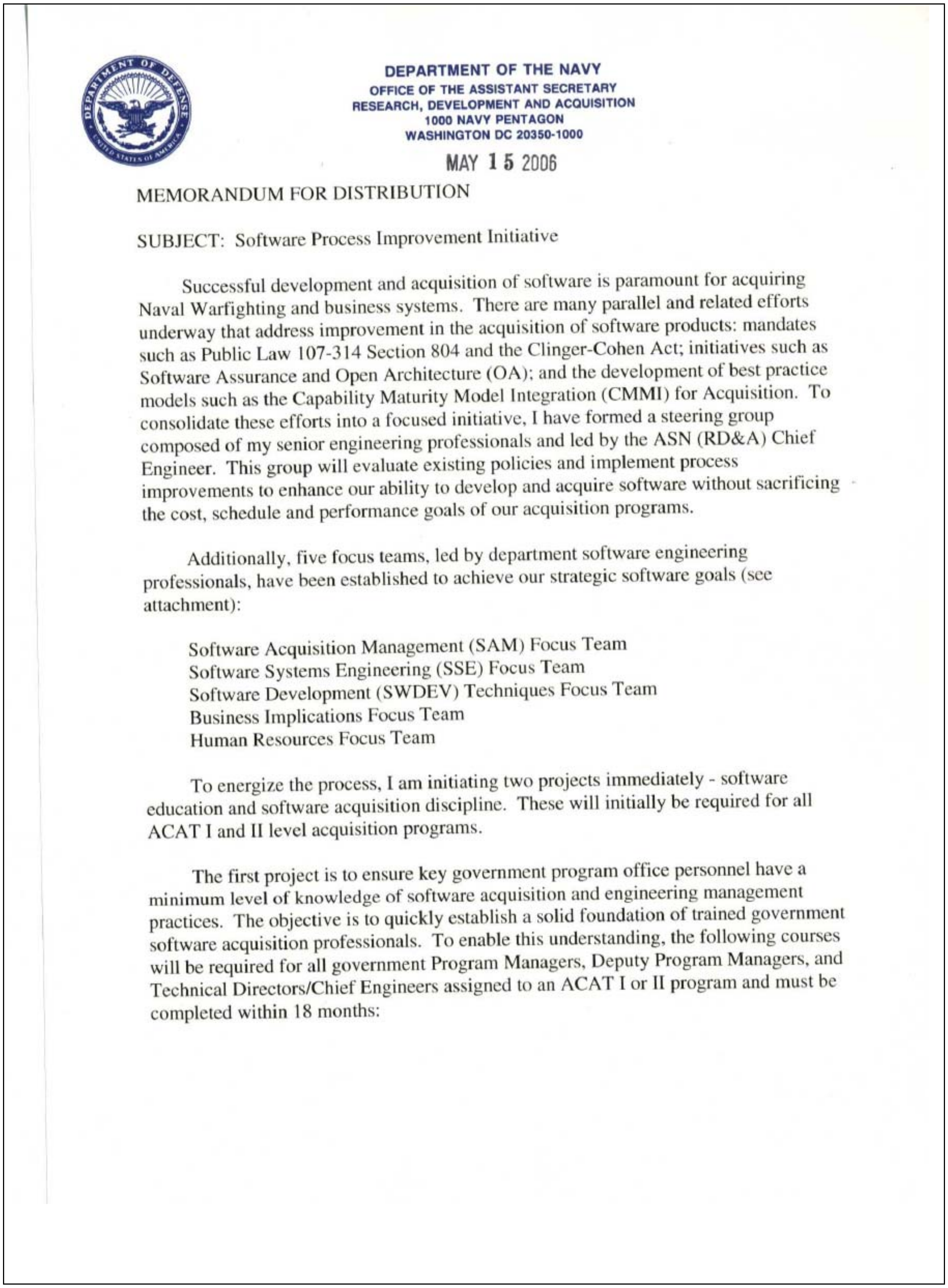
OPNAV	Office of the Chief of Naval Operations
OPTEVFOR	Operational Test & Evaluation Force
OS	Open System
OSD	Office of the Secretary of Defense
OT	Operational Test
OT&E	Operational Test & Evaluation
OTRR	Operational Test Readiness Review
PBL	Performance Based Logistics
PCD	Position Category Description
PDR	Preliminary Design Review
PEO	Program Executive Office
PM	Program Manager
PMBOK®	Project Management Body of Knowledge
PMI	Project Management Institute
POE	Program Office Estimate
PoPS	Probability of Program Success
POR	Program of Record
PP	Product Plan
PPP	Program Protection Plan
PRR	Production Readiness Review
PV	Planned Value
PWBS	Program Work Breakdown Structure
QA	Quality Assurance
QC	Quality Control
RB/RF	Role Based/Right Fit
RFI	Request for Information
RFP	Request For Proposal
RR	Restricted Rights
RRB	Requirements Review Board
SAM	Software Acquisition Module
SATEWG	Software Acquisition Training and Education Working Group
SBIR	Small Business Innovation Research
SCA	Source Code Analysis
SCM	Software Configuration Management
SCR	Software Change Report
SDD	System Development and Demonstration
SDLC	Software Development Life Cycle
SDF	Software Development Folder
SDP	Software Development Plan
SDS	System Design Specifications
SDX	System Data Exchange
SEB	Source Evaluation Board
SECNAVINST	Secretary of the Navy Instruction
SECNAVNOTE	Secretary of the Navy Notice
SEI	Software Engineering Institute
SEP	Systems Engineering Plan
SEPG	Software Engineering Process Group

SERC	Systems Engineering Resource Center
SETR	Systems Engineering Technical Review
SFR	System Functional Review
SHARE	Software Hardware Asset Reuse Enterprise
SIS	Software Intensive Systems
SISOS	Software Intensive Systems of Systems
SLOC	Source Lines of Code
SME	Subject Matter Expert
SMP	Software Management Plan
SNLR	Significantly Negotiated License Rights
SOA	Service Oriented Architecture
SOO	Statement of Objectives
SOS	Systems of Systems
SOW	Statement of Work
SPI	Schedule Performance Index
SPII	Software Process Improvement Initiative
SQL	Structured Query Language
SPMN	Software Program Managers Network
SPRDE	Systems Planning, Research, Development, and Engineering
SRDR	Software Resources Data Report
SRR	Systems Requirements Review
SRS	Software Requirements Specification
SSA	Source Selection Authority
SSO	Source Selection Official
SSP	Source Selection Plan
SV	Schedule Variance
SVR	System Verification Review
SW	Software
SYSCOM	Systems Command
T&E	Test & Evaluation
TAI	Tri-Service Assessment Initiative
TD	Technical Data
TEMP	Technical Evaluation Master Plan
TRA	Technology Readiness Assessment
TRR	Test Readiness Review
TRT	Technical Review Team
UR	Unlimited Rights
URL	Uniform Resource Locator
USD	Under Secretary of Defense
WBS	Work Breakdown Structure
WSESRB	Weapons Systems Explosive Safety Review Board

This Page Intentionally Left Blank

Appendix A

May 06 Software Process Improvement Initiative Policy Memo



DEPARTMENT OF THE NAVY
OFFICE OF THE ASSISTANT SECRETARY
RESEARCH, DEVELOPMENT AND ACQUISITION
1000 NAVY PENTAGON
WASHINGTON DC 20350-1000

MAY 15 2006

MEMORANDUM FOR DISTRIBUTION

SUBJECT: Software Process Improvement Initiative

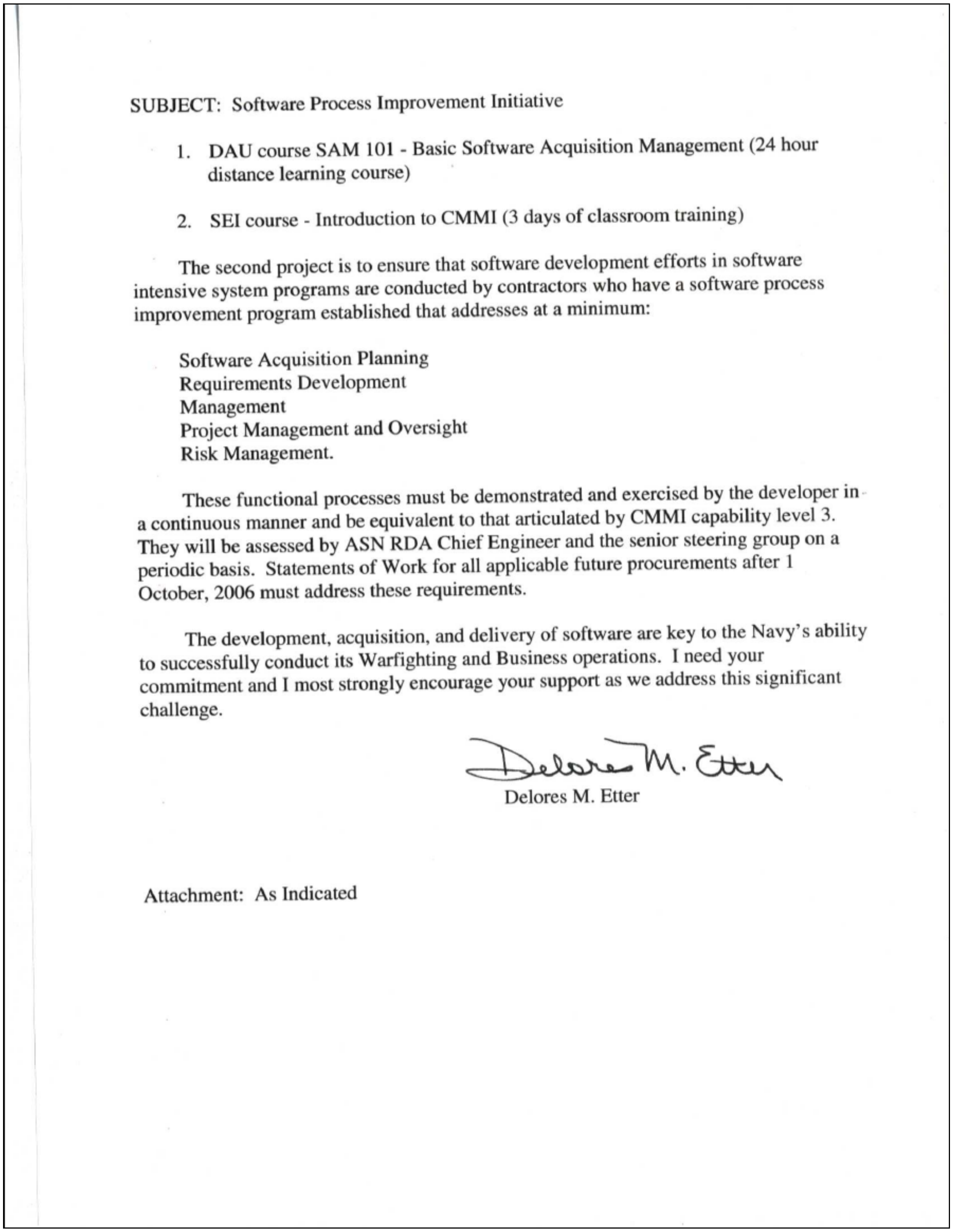
Successful development and acquisition of software is paramount for acquiring Naval Warfighting and business systems. There are many parallel and related efforts underway that address improvement in the acquisition of software products: mandates such as Public Law 107-314 Section 804 and the Clinger-Cohen Act; initiatives such as Software Assurance and Open Architecture (OA); and the development of best practice models such as the Capability Maturity Model Integration (CMMI) for Acquisition. To consolidate these efforts into a focused initiative, I have formed a steering group composed of my senior engineering professionals and led by the ASN (RD&A) Chief Engineer. This group will evaluate existing policies and implement process improvements to enhance our ability to develop and acquire software without sacrificing the cost, schedule and performance goals of our acquisition programs.

Additionally, five focus teams, led by department software engineering professionals, have been established to achieve our strategic software goals (see attachment):

Software Acquisition Management (SAM) Focus Team
Software Systems Engineering (SSE) Focus Team
Software Development (SWDEV) Techniques Focus Team
Business Implications Focus Team
Human Resources Focus Team

To energize the process, I am initiating two projects immediately - software education and software acquisition discipline. These will initially be required for all ACAT I and II level acquisition programs.

The first project is to ensure key government program office personnel have a minimum level of knowledge of software acquisition and engineering management practices. The objective is to quickly establish a solid foundation of trained government software acquisition professionals. To enable this understanding, the following courses will be required for all government Program Managers, Deputy Program Managers, and Technical Directors/Chief Engineers assigned to an ACAT I or II program and must be completed within 18 months:



SUBJECT: Software Process Improvement Initiative

1. DAU course SAM 101 - Basic Software Acquisition Management (24 hour distance learning course)

2. SEI course - Introduction to CMMI (3 days of classroom training)

The second project is to ensure that software development efforts in software intensive system programs are conducted by contractors who have a software process improvement program established that addresses at a minimum:

Software Acquisition Planning
Requirements Development
Management
Project Management and Oversight
Risk Management.

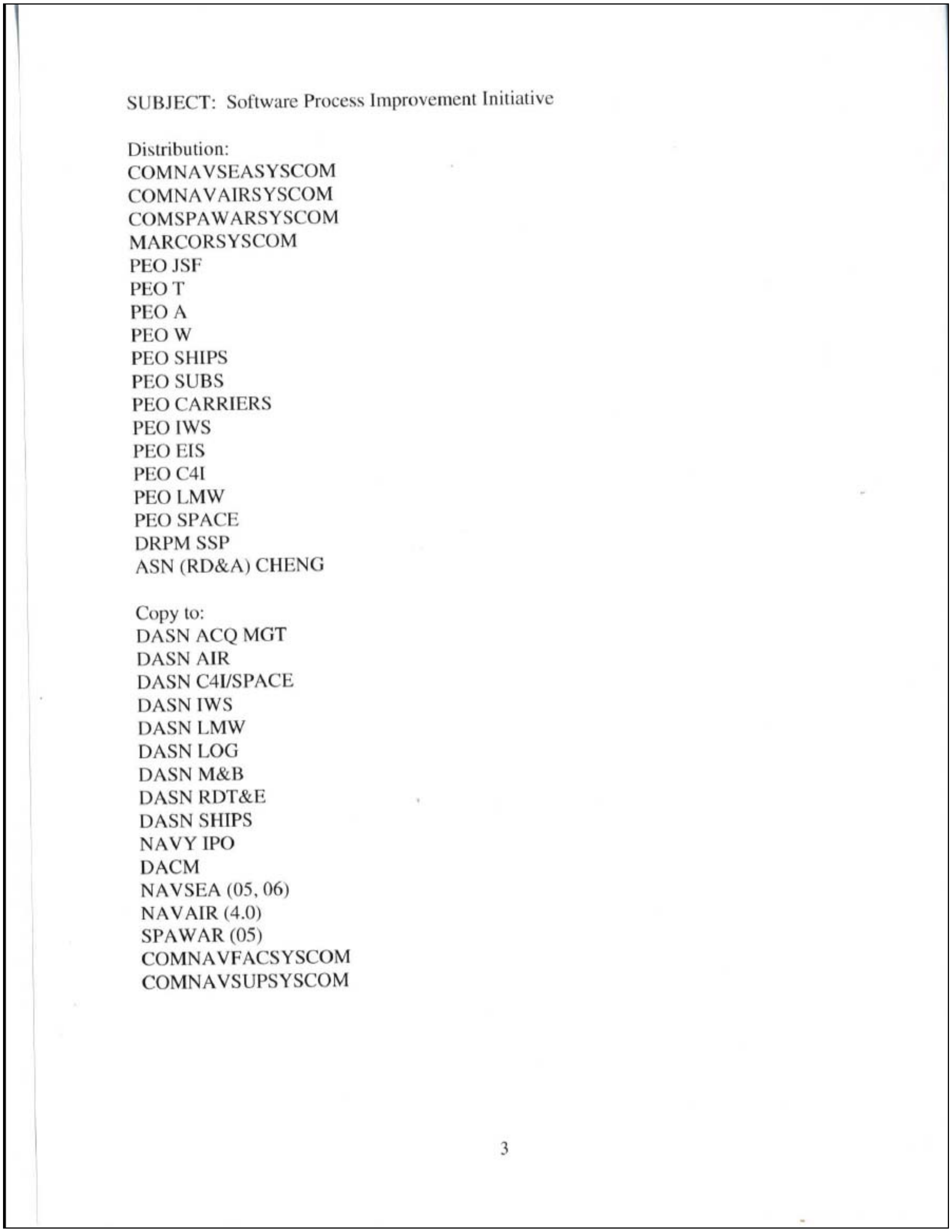
These functional processes must be demonstrated and exercised by the developer in a continuous manner and be equivalent to that articulated by CMMI capability level 3. They will be assessed by ASN RDA Chief Engineer and the senior steering group on a periodic basis. Statements of Work for all applicable future procurements after 1 October, 2006 must address these requirements.

The development, acquisition, and delivery of software are key to the Navy's ability to successfully conduct its Warfighting and Business operations. I need your commitment and I most strongly encourage your support as we address this significant challenge.

Delores M. Etter

Delores M. Etter

Attachment: As Indicated

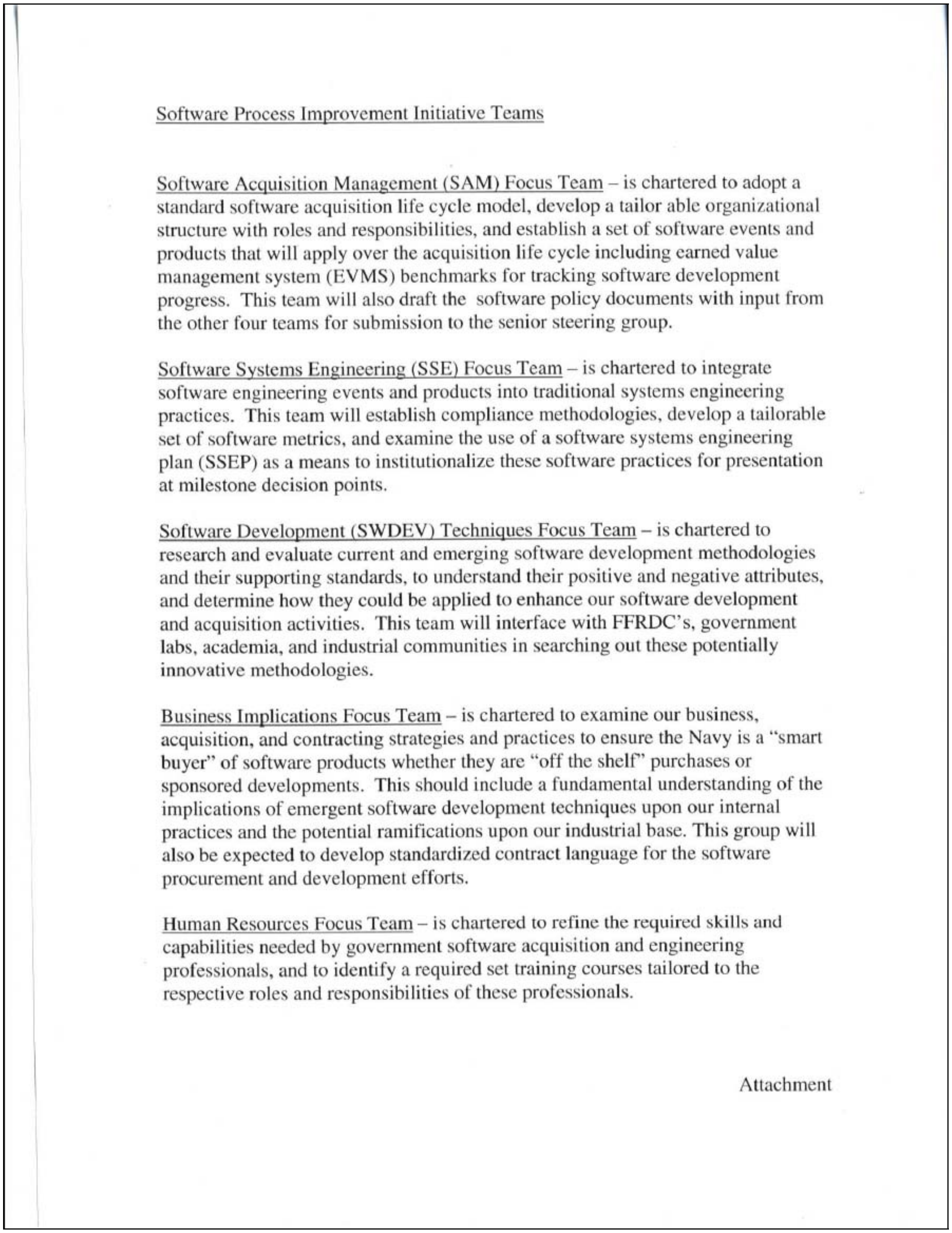


SUBJECT: Software Process Improvement Initiative

Distribution:
COMNAVSEASYSCOM
COMNAVAIRSYSCOM
COMSPAWARSYSCOM
MARCORSYSCOM
PEO JSF
PEO T
PEO A
PEO W
PEO SHIPS
PEO SUBS
PEO CARRIERS
PEO IWS
PEO EIS
PEO C4I
PEO LMW
PEO SPACE
DRPM SSP
ASN (RD&A) CHENG

Copy to:
DASN ACQ MGT
DASN AIR
DASN C4I/SPACE
DASN IWS
DASN LMW
DASN LOG
DASN M&B
DASN RDT&E
DASN SHIPS
NAVY IPO
DACM
NAVSEA (05, 06)
NAVAIR (4.0)
SPAWAR (05)
COMNAVFACSYSCOM
COMNAVSUPSYSCOM

3



Software Process Improvement Initiative Teams

Software Acquisition Management (SAM) Focus Team – is chartered to adopt a standard software acquisition life cycle model, develop a tailor able organizational structure with roles and responsibilities, and establish a set of software events and products that will apply over the acquisition life cycle including earned value management system (EVMS) benchmarks for tracking software development progress. This team will also draft the software policy documents with input from the other four teams for submission to the senior steering group.

Software Systems Engineering (SSE) Focus Team – is chartered to integrate software engineering events and products into traditional systems engineering practices. This team will establish compliance methodologies, develop a tailorable set of software metrics, and examine the use of a software systems engineering plan (SSEP) as a means to institutionalize these software practices for presentation at milestone decision points.

Software Development (SWDEV) Techniques Focus Team – is chartered to research and evaluate current and emerging software development methodologies and their supporting standards, to understand their positive and negative attributes, and determine how they could be applied to enhance our software development and acquisition activities. This team will interface with FFRDC's, government labs, academia, and industrial communities in searching out these potentially innovative methodologies.

Business Implications Focus Team – is chartered to examine our business, acquisition, and contracting strategies and practices to ensure the Navy is a "smart buyer" of software products whether they are "off the shelf" purchases or sponsored developments. This should include a fundamental understanding of the implications of emergent software development techniques upon our internal practices and the potential ramifications upon our industrial base. This group will also be expected to develop standardized contract language for the software procurement and development efforts.

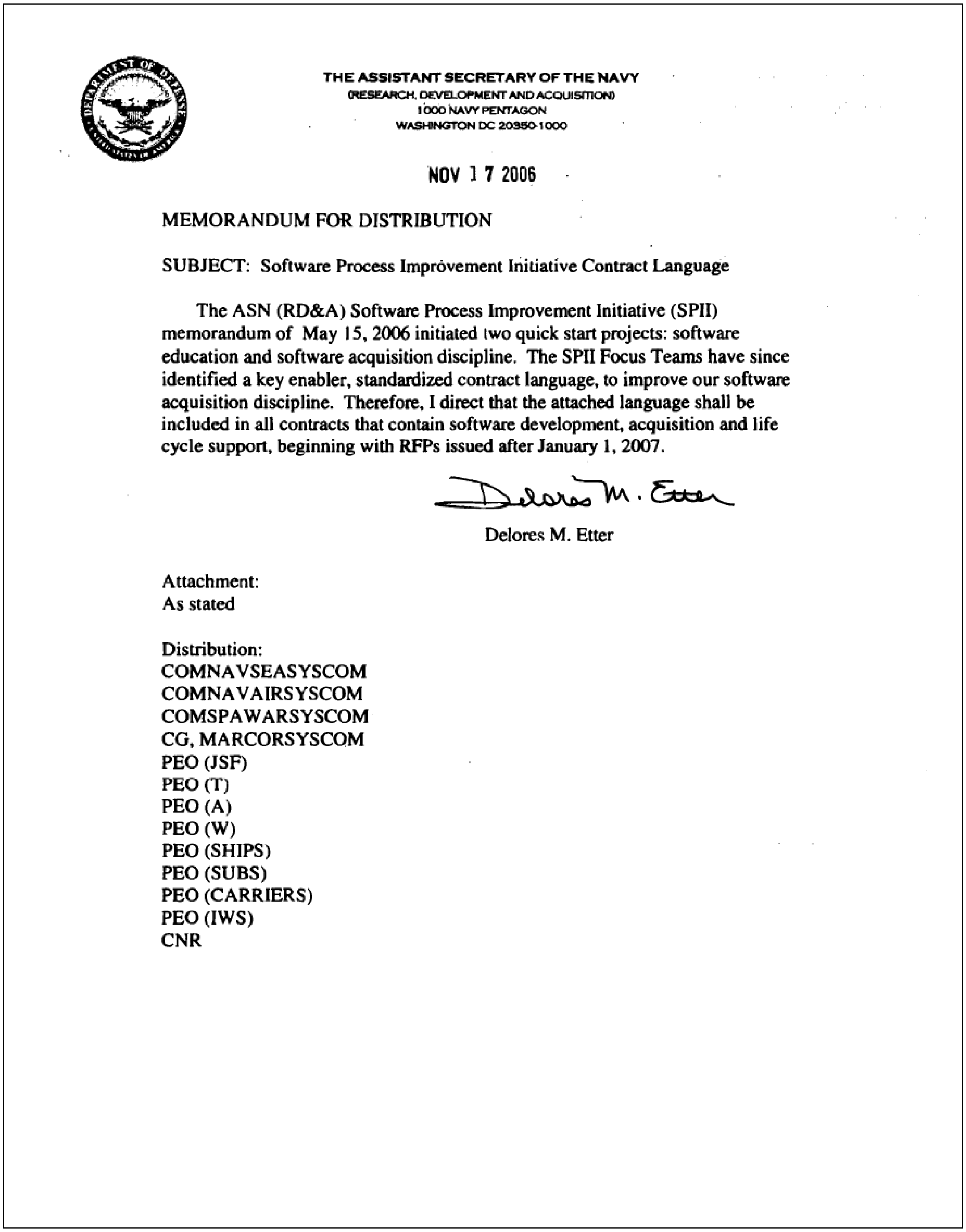
Human Resources Focus Team – is chartered to refine the required skills and capabilities needed by government software acquisition and engineering professionals, and to identify a required set training courses tailored to the respective roles and responsibilities of these professionals.

Attachment

This Page Intentionally Left Blank

Appendix B

November 06 Software Process Improvement Initative Contract Language Memo



THE ASSISTANT SECRETARY OF THE NAVY
(RESEARCH, DEVELOPMENT AND ACQUISITION)
1000 NAVY PENTAGON
WASHINGTON DC 20350-1000

NOV 17 2006

MEMORANDUM FOR DISTRIBUTION

SUBJECT: Software Process Improvement Initiative Contract Language

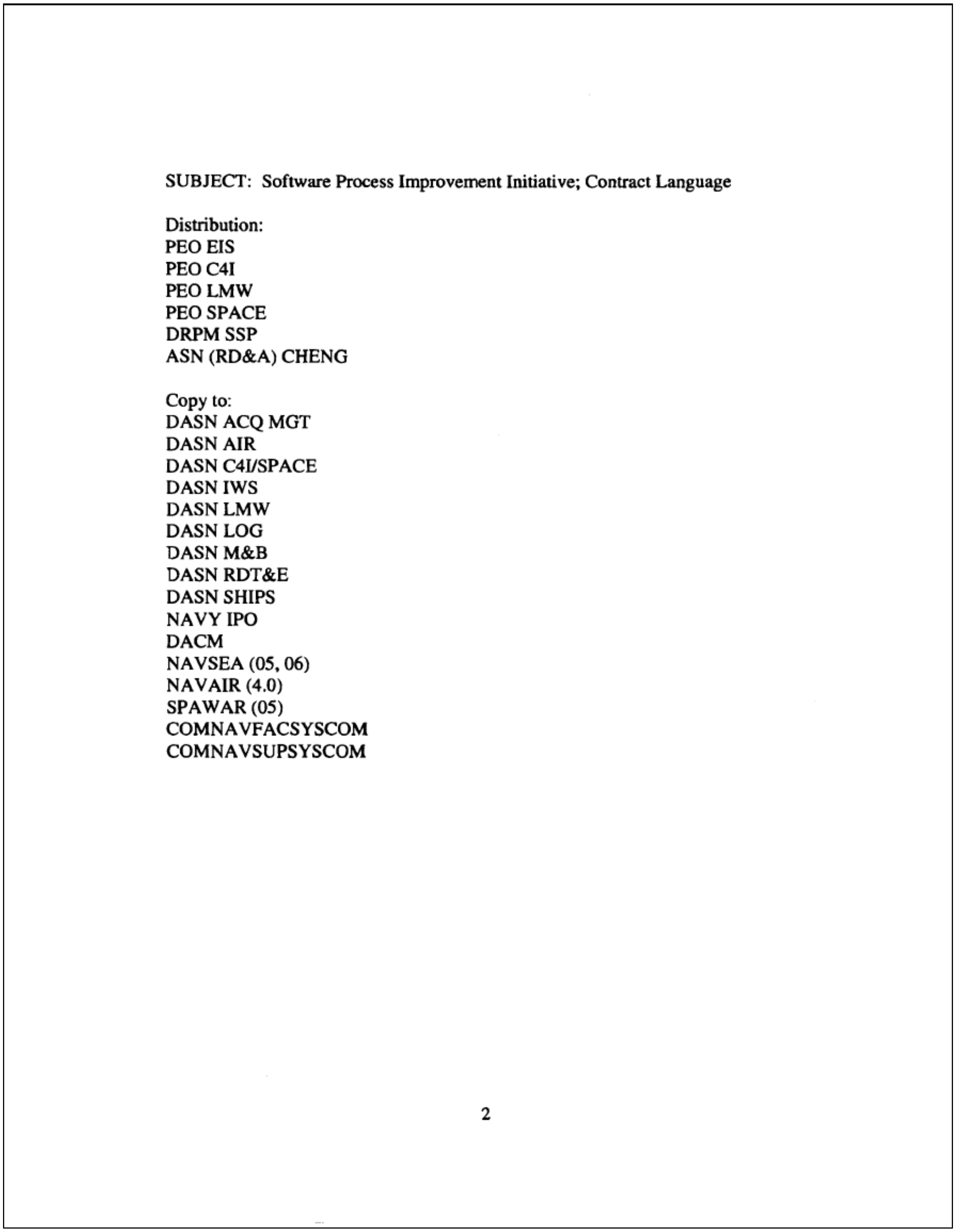
The ASN (RD&A) Software Process Improvement Initiative (SPII) memorandum of May 15, 2006 initiated two quick start projects: software education and software acquisition discipline. The SPII Focus Teams have since identified a key enabler, standardized contract language, to improve our software acquisition discipline. Therefore, I direct that the attached language shall be included in all contracts that contain software development, acquisition and life cycle support, beginning with RFPs issued after January 1, 2007.

Delores M. Etter

Delores M. Etter

Attachment:
As stated

Distribution:
COMNAVSEASYSCOM
COMNAVAIRSYSCOM
COMSPAWARSYSCOM
CG, MARCORSYSCOM
PEO (JSF)
PEO (T)
PEO (A)
PEO (W)
PEO (SHIPS)
PEO (SUBS)
PEO (CARRIERS)
PEO (IWS)
CNR

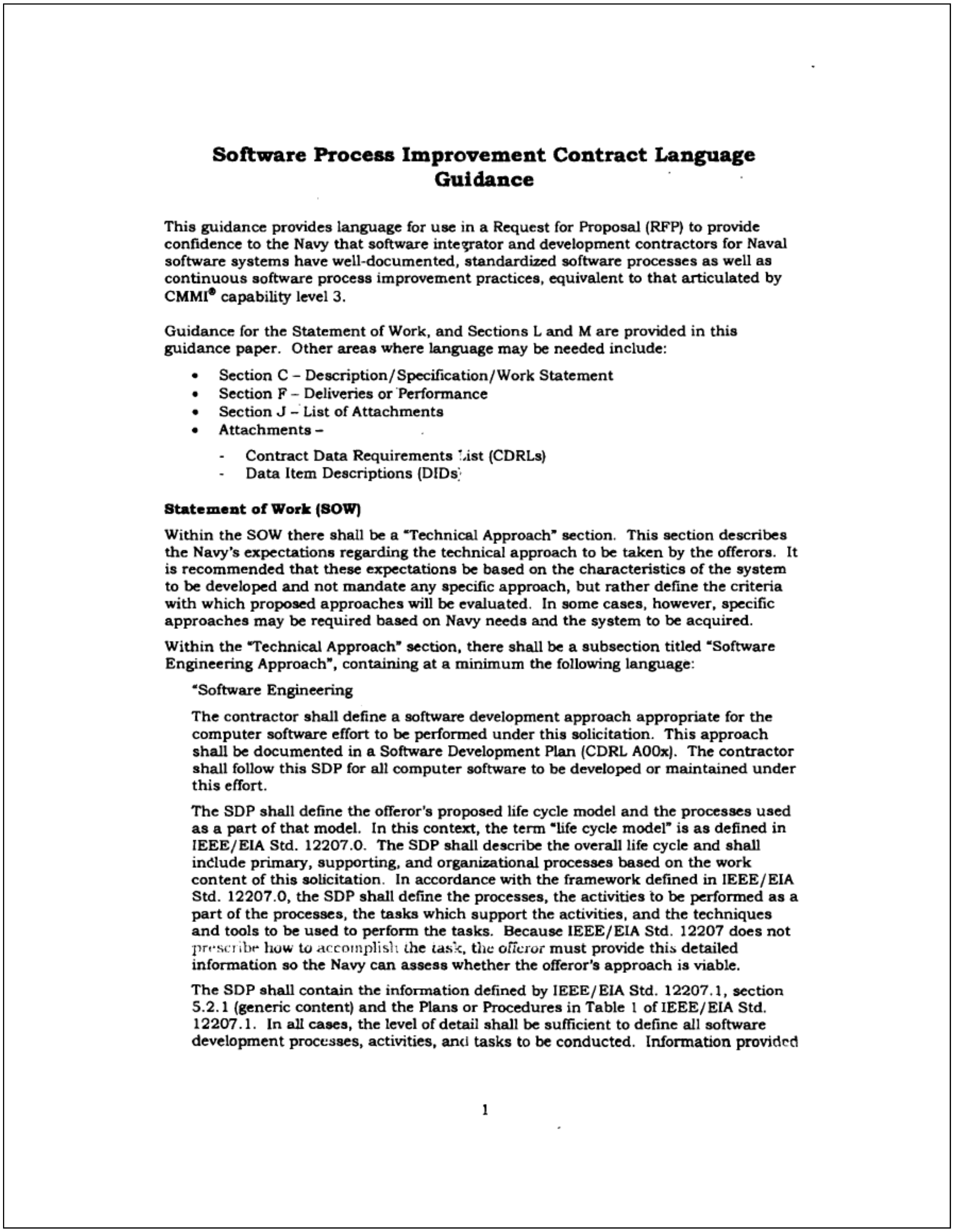


SUBJECT: Software Process Improvement Initiative; Contract Language

Distribution:
PEO EIS
PEO C4I
PEO LMW
PEO SPACE
DRPM SSP
ASN (RD&A) CHENG

Copy to:
DASN ACQ MGT
DASN AIR
DASN C4I/SPACE
DASN IWS
DASN LMW
DASN LOG
DASN M&B
DASN RDT&E
DASN SHIPS
NAVY IPO
DACM
NAVSEA (05, 06)
NAVAIR (4.0)
SPAWAR (05)
COMNAVFACSYSCOM
COMNAVSUPSYSCOM

2



Software Process Improvement Contract Language Guidance

This guidance provides language for use in a Request for Proposal (RFP) to provide confidence to the Navy that software integrator and development contractors for Naval software systems have well-documented, standardized software processes as well as continuous software process improvement practices, equivalent to that articulated by CMMI® capability level 3.

Guidance for the Statement of Work, and Sections L and M are provided in this guidance paper. Other areas where language may be needed include:

- Section C – Description/Specification/Work Statement
- Section F – Deliveries or Performance
- Section J – List of Attachments
- Attachments –
  - Contract Data Requirements List (CDRLs)
  - Data Item Descriptions (DIDs)

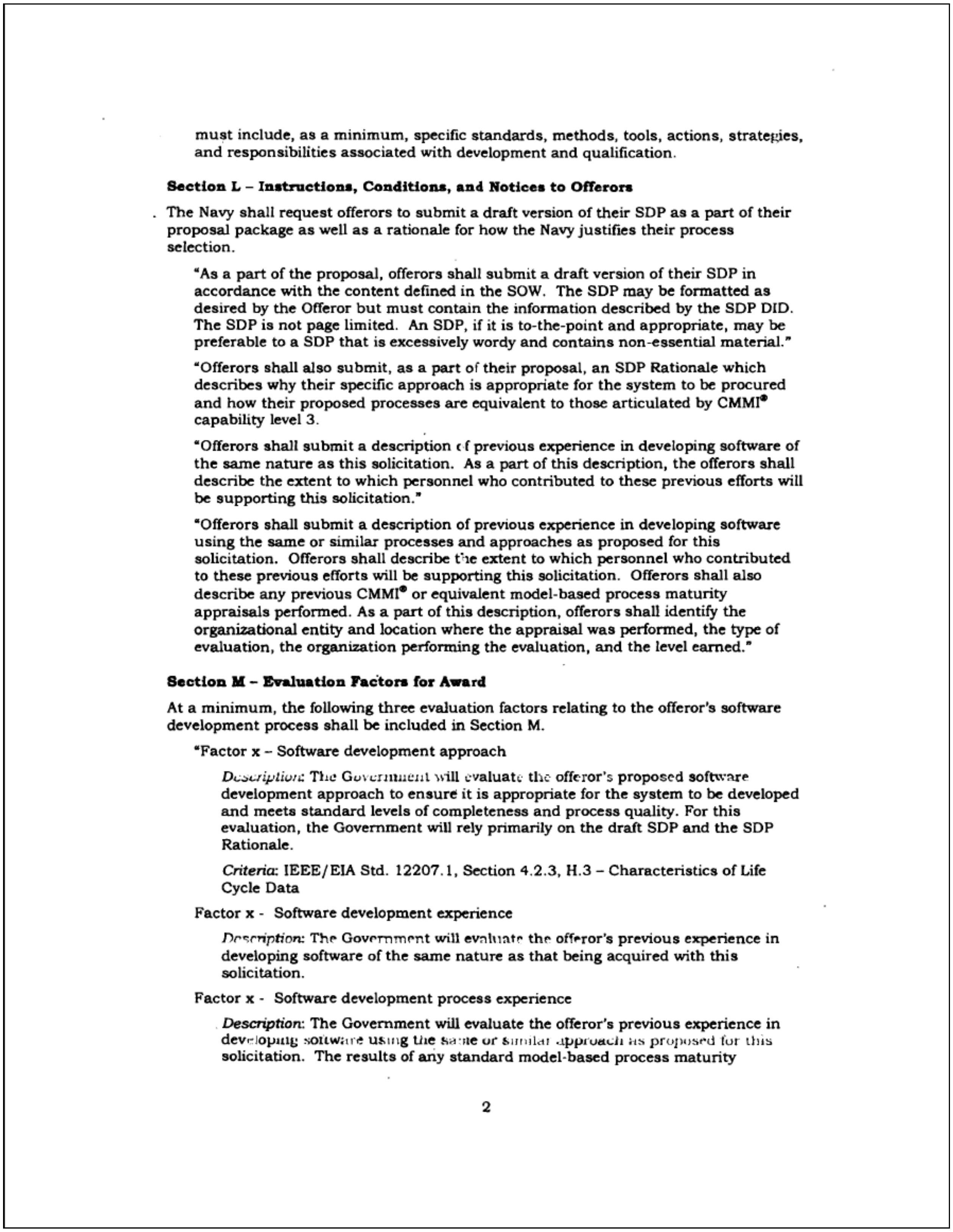
Statement of Work (SOW)

Within the SOW there shall be a "Technical Approach" section. This section describes the Navy's expectations regarding the technical approach to be taken by the offerors. It is recommended that these expectations be based on the characteristics of the system to be developed and not mandate any specific approach, but rather define the criteria with which proposed approaches will be evaluated. In some cases, however, specific approaches may be required based on Navy needs and the system to be acquired.

Within the "Technical Approach" section, there shall be a subsection titled "Software Engineering Approach", containing at a minimum the following language:

> "Software Engineering
>
> The contractor shall define a software development approach appropriate for the computer software effort to be performed under this solicitation. This approach shall be documented in a Software Development Plan (CDRL A00x). The contractor shall follow this SDP for all computer software to be developed or maintained under this effort.
>
> The SDP shall define the offeror's proposed life cycle model and the processes used as a part of that model. In this context, the term "life cycle model" is as defined in IEEE/EIA Std. 12207.0. The SDP shall describe the overall life cycle and shall include primary, supporting, and organizational processes based on the work content of this solicitation. In accordance with the framework defined in IEEE/EIA Std. 12207.0, the SDP shall define the processes, the activities to be performed as a part of the processes, the tasks which support the activities, and the techniques and tools to be used to perform the tasks. Because IEEE/EIA Std. 12207 does not prescribe how to accomplish the task, the offeror must provide this detailed information so the Navy can assess whether the offeror's approach is viable.
>
> The SDP shall contain the information defined by IEEE/EIA Std. 12207.1, section 5.2.1 (generic content) and the Plans or Procedures in Table 1 of IEEE/EIA Std. 12207.1. In all cases, the level of detail shall be sufficient to define all software development processes, activities, and tasks to be conducted. Information provided

1



must include, as a minimum, specific standards, methods, tools, actions, strategies, and responsibilities associated with development and qualification.

Section L – Instructions, Conditions, and Notices to Offerors

. The Navy shall request offerors to submit a draft version of their SDP as a part of their proposal package as well as a rationale for how the Navy justifies their process selection.

"As a part of the proposal, offerors shall submit a draft version of their SDP in accordance with the content defined in the SOW. The SDP may be formatted as desired by the Offeror but must contain the information described by the SDP DID. The SDP is not page limited. An SDP, if it is to-the-point and appropriate, may be preferable to a SDP that is excessively wordy and contains non-essential material."

"Offerors shall also submit, as a part of their proposal, an SDP Rationale which describes why their specific approach is appropriate for the system to be procured and how their proposed processes are equivalent to those articulated by CMMI® capability level 3.

"Offerors shall submit a description of previous experience in developing software of the same nature as this solicitation. As a part of this description, the offerors shall describe the extent to which personnel who contributed to these previous efforts will be supporting this solicitation."

"Offerors shall submit a description of previous experience in developing software using the same or similar processes and approaches as proposed for this solicitation. Offerors shall describe the extent to which personnel who contributed to these previous efforts will be supporting this solicitation. Offerors shall also describe any previous CMMI® or equivalent model-based process maturity appraisals performed. As a part of this description, offerors shall identify the organizational entity and location where the appraisal was performed, the type of evaluation, the organization performing the evaluation, and the level earned."

Section M – Evaluation Factors for Award

At a minimum, the following three evaluation factors relating to the offeror's software development process shall be included in Section M.

"Factor x – Software development approach

Description: The Government will evaluate the offeror's proposed software development approach to ensure it is appropriate for the system to be developed and meets standard levels of completeness and process quality. For this evaluation, the Government will rely primarily on the draft SDP and the SDP Rationale.

Criteria: IEEE/EIA Std. 12207.1, Section 4.2.3, H.3 – Characteristics of Life Cycle Data

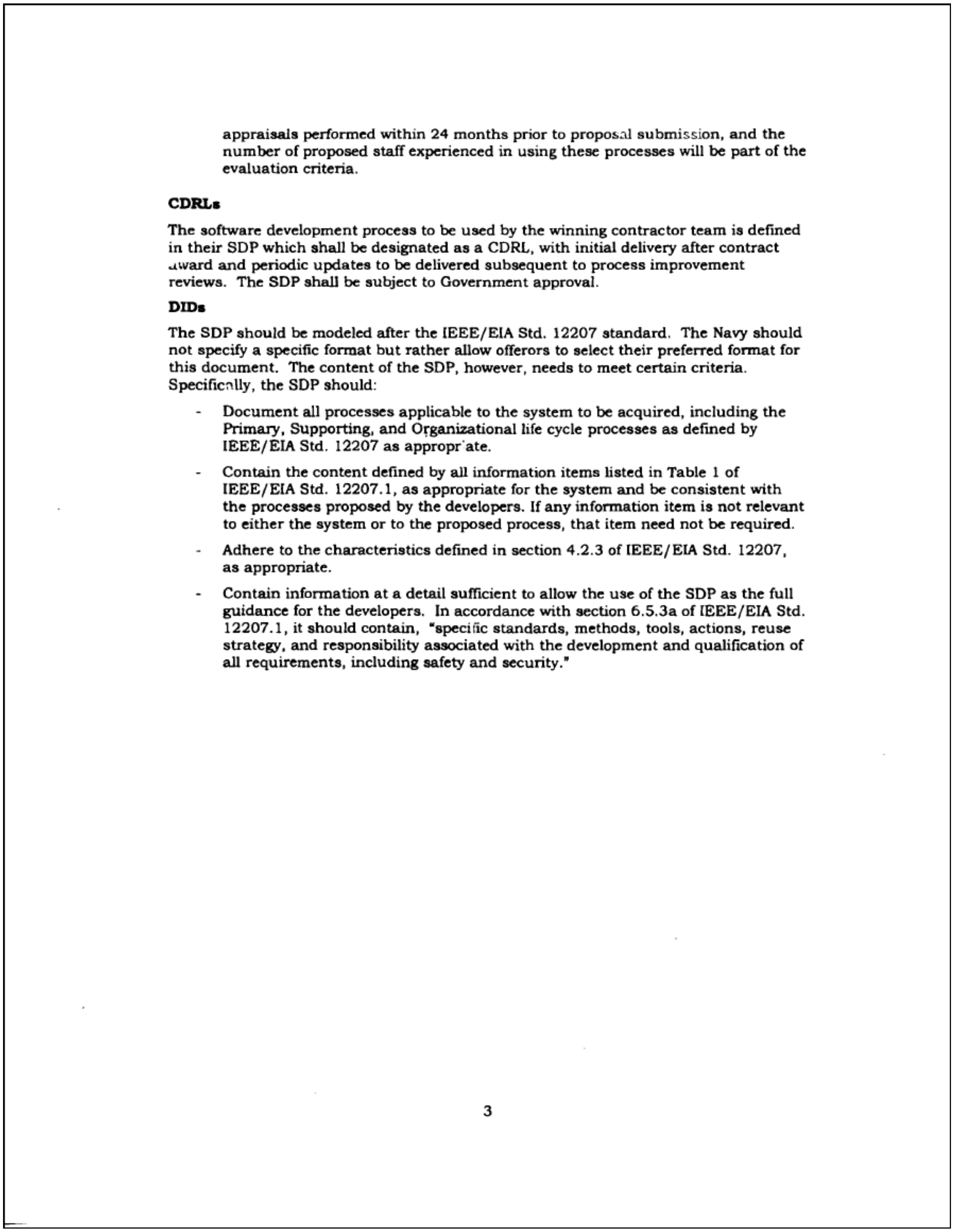
Factor x - Software development experience

Description: The Government will evaluate the offeror's previous experience in developing software of the same nature as that being acquired with this solicitation.

Factor x - Software development process experience

Description: The Government will evaluate the offeror's previous experience in developing software using the same or similar approach as proposed for this solicitation. The results of any standard model-based process maturity

2



appraisals performed within 24 months prior to proposal submission, and the number of proposed staff experienced in using these processes will be part of the evaluation criteria.

CDRLs

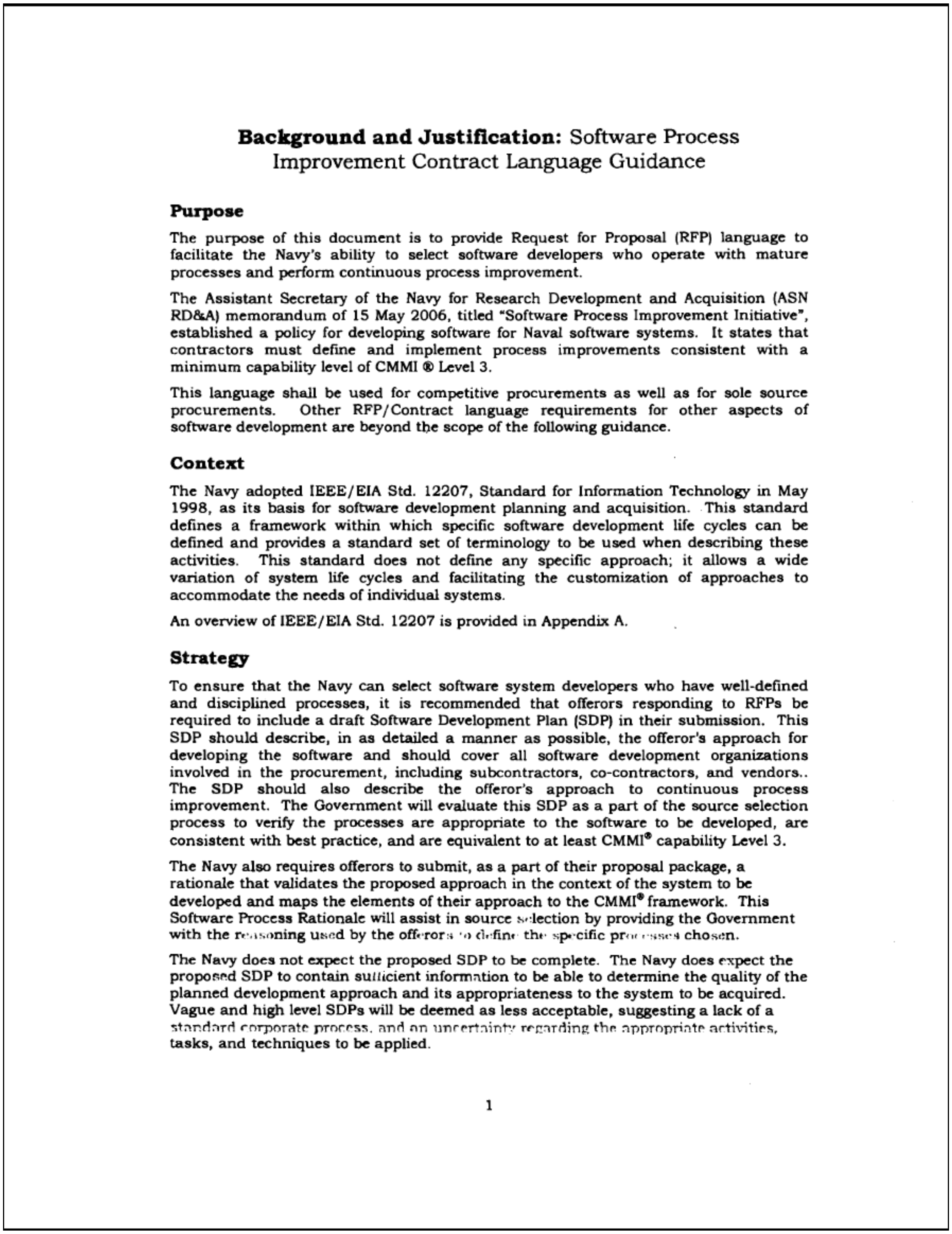
The software development process to be used by the winning contractor team is defined in their SDP which shall be designated as a CDRL, with initial delivery after contract award and periodic updates to be delivered subsequent to process improvement reviews. The SDP shall be subject to Government approval.

DIDs

The SDP should be modeled after the IEEE/EIA Std. 12207 standard. The Navy should not specify a specific format but rather allow offerors to select their preferred format for this document. The content of the SDP, however, needs to meet certain criteria. Specifically, the SDP should:

- Document all processes applicable to the system to be acquired, including the Primary, Supporting, and Organizational life cycle processes as defined by IEEE/EIA Std. 12207 as appropriate.
- Contain the content defined by all information items listed in Table 1 of IEEE/EIA Std. 12207.1, as appropriate for the system and be consistent with the processes proposed by the developers. If any information item is not relevant to either the system or to the proposed process, that item need not be required.
- Adhere to the characteristics defined in section 4.2.3 of IEEE/EIA Std. 12207, as appropriate.
- Contain information at a detail sufficient to allow the use of the SDP as the full guidance for the developers. In accordance with section 6.5.3a of IEEE/EIA Std. 12207.1, it should contain, "specific standards, methods, tools, actions, reuse strategy, and responsibility associated with the development and qualification of all requirements, including safety and security."

3



Background and Justification: Software Process Improvement Contract Language Guidance

Purpose

The purpose of this document is to provide Request for Proposal (RFP) language to facilitate the Navy's ability to select software developers who operate with mature processes and perform continuous process improvement.

The Assistant Secretary of the Navy for Research Development and Acquisition (ASN RD&A) memorandum of 15 May 2006, titled "Software Process Improvement Initiative", established a policy for developing software for Naval software systems. It states that contractors must define and implement process improvements consistent with a minimum capability level of CMMI ® Level 3.

This language shall be used for competitive procurements as well as for sole source procurements. Other RFP/Contract language requirements for other aspects of software development are beyond the scope of the following guidance.

Context

The Navy adopted IEEE/EIA Std. 12207, Standard for Information Technology in May 1998, as its basis for software development planning and acquisition. This standard defines a framework within which specific software development life cycles can be defined and provides a standard set of terminology to be used when describing these activities. This standard does not define any specific approach; it allows a wide variation of system life cycles and facilitating the customization of approaches to accommodate the needs of individual systems.

An overview of IEEE/EIA Std. 12207 is provided in Appendix A.

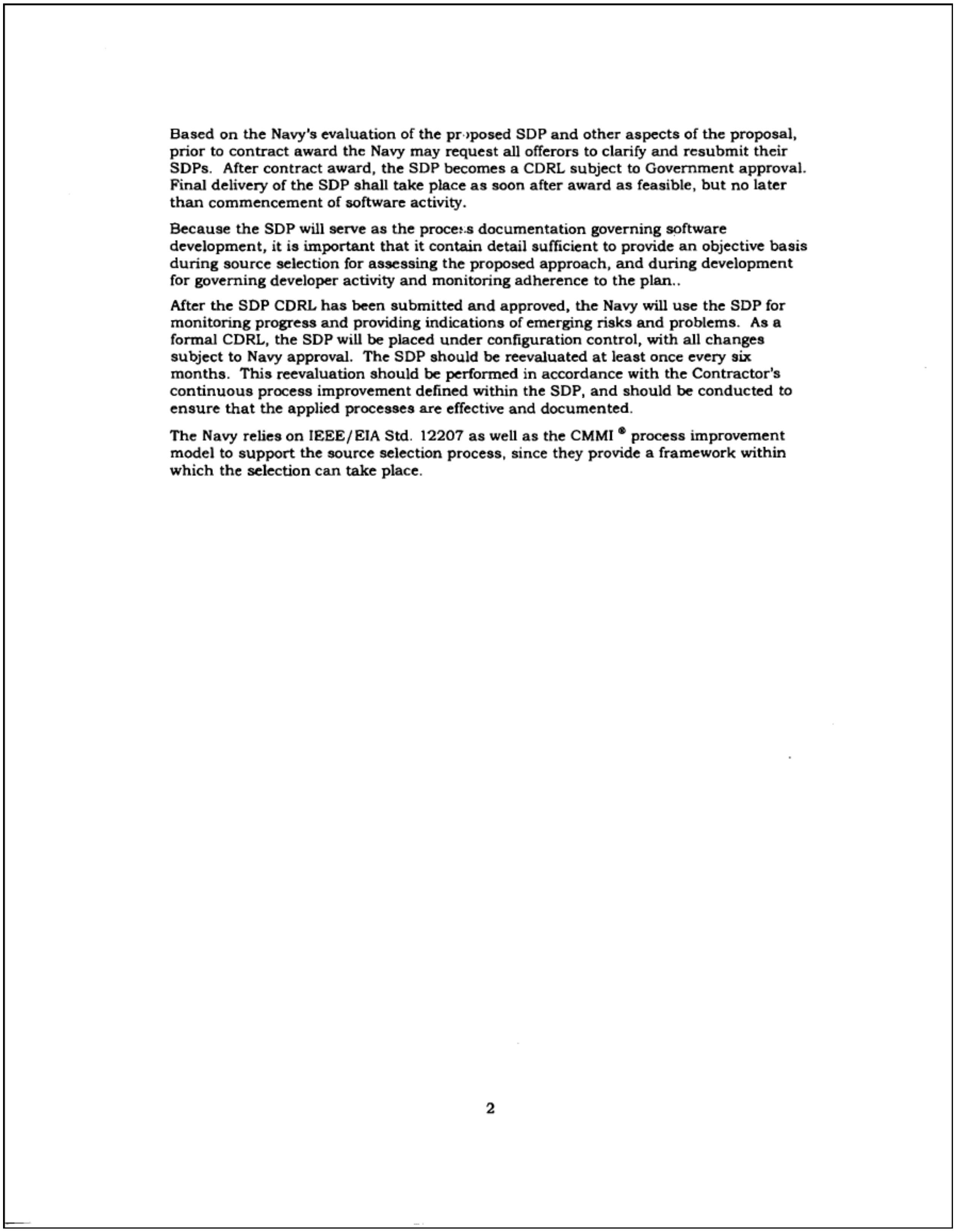
Strategy

To ensure that the Navy can select software system developers who have well-defined and disciplined processes, it is recommended that offerors responding to RFPs be required to include a draft Software Development Plan (SDP) in their submission. This SDP should describe, in as detailed a manner as possible, the offeror's approach for developing the software and should cover all software development organizations involved in the procurement, including subcontractors, co-contractors, and vendors.. The SDP should also describe the offeror's approach to continuous process improvement. The Government will evaluate this SDP as a part of the source selection process to verify the processes are appropriate to the software to be developed, are consistent with best practice, and are equivalent to at least CMMI® capability Level 3.

The Navy also requires offerors to submit, as a part of their proposal package, a rationale that validates the proposed approach in the context of the system to be developed and maps the elements of their approach to the CMMI® framework. This Software Process Rationale will assist in source selection by providing the Government with the reasoning used by the offerors to define the specific processes chosen.

The Navy does not expect the proposed SDP to be complete. The Navy does expect the proposed SDP to contain sufficient information to be able to determine the quality of the planned development approach and its appropriateness to the system to be acquired. Vague and high level SDPs will be deemed as less acceptable, suggesting a lack of a standard corporate process, and an uncertainty regarding the appropriate activities, tasks, and techniques to be applied.

1



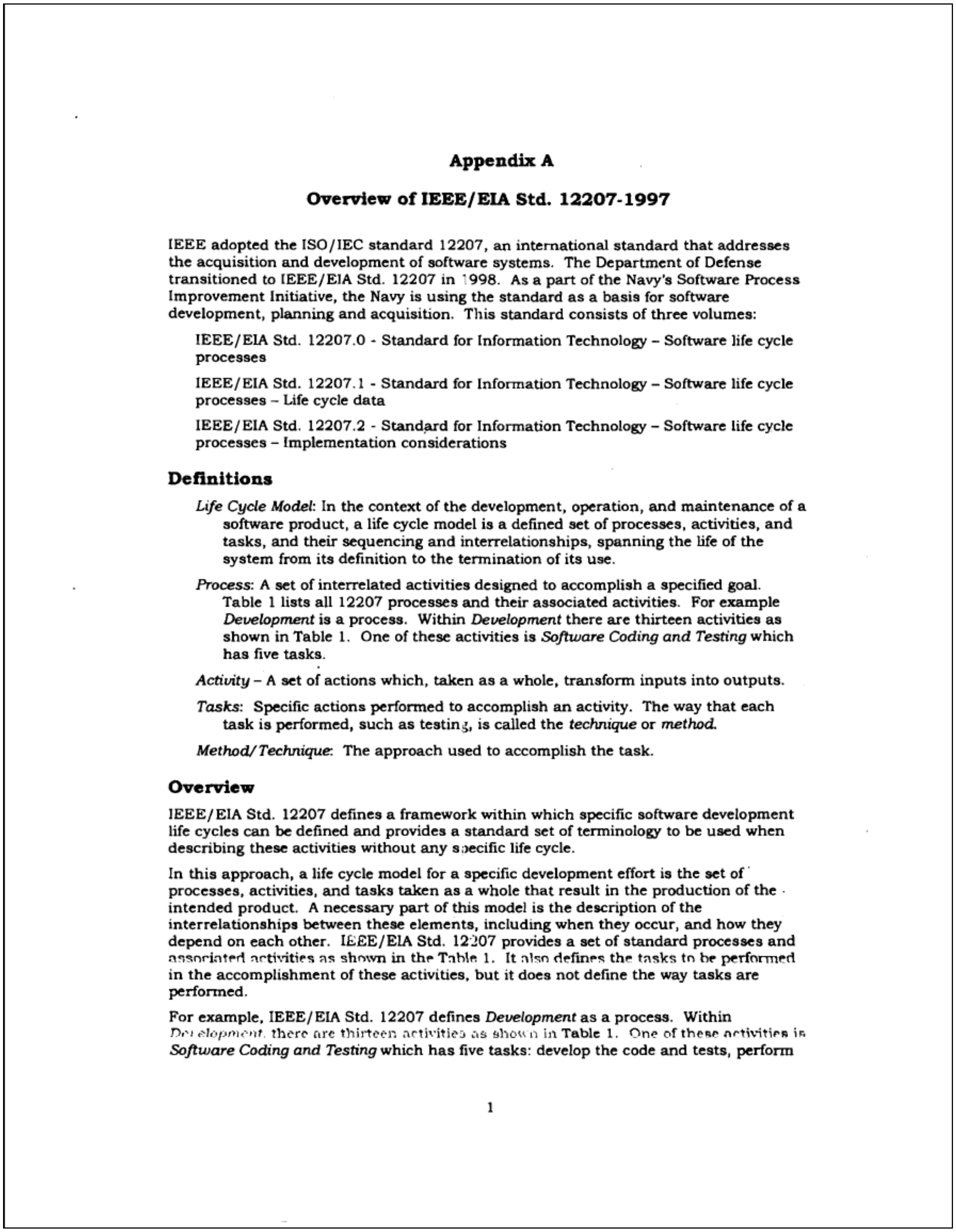
Based on the Navy's evaluation of the pr›posed SDP and other aspects of the proposal, prior to contract award thc Navy may request all offerors to clarify and resubmit their SDPs. After contract award, the SDP becomes a CDRL subject to Government approval. Final delivery of the SDP shall take place as soon after award as feasible, but no later than commencement of software activity.

Because the SDP will serve as the proce:.s documentation governing software development, it is important that it contain detail sufficient to provide an objective basis during source selection for assessing the proposed approach, and during development for governing developer activity and monitoring adherence to the plan..

After the SDP CDRL has been submitted and approved, the Navy will use the SDP for monitoring progress and providing indications of emerging risks and problems. As a formal CDRL, the SDP will be placed under configuration control, with all changes subject to Navy approval. The SDP should be reevaluated at least once every six months. This reevaluation should be performed in accordance with the Contractor's continuous process improvement defined within the SDP, and should be conducted to ensure that the applied processes are effective and documented.

The Navy relies on IEEE/EIA Std. 12207 as well as the CMMI ® process improvement model to support the source selection process, since they provide a framework within which the selection can take place.

2



Appendix A

Overview of IEEE/EIA Std. 12207-1997

IEEE adopted the ISO/IEC standard 12207, an international standard that addresses the acquisition and development of software systems. The Department of Defense transitioned to IEEE/EIA Std. 12207 in 1998. As a part of the Navy's Software Process Improvement Initiative, the Navy is using the standard as a basis for software development, planning and acquisition. This standard consists of three volumes:

IEEE/EIA Std. 12207.0 - Standard for Information Technology – Software life cycle processes

IEEE/EIA Std. 12207.1 - Standard for Information Technology – Software life cycle processes – Life cycle data

IEEE/EIA Std. 12207.2 - Standard for Information Technology – Software life cycle processes – Implementation considerations

Definitions

Life Cycle Model: In the context of the development, operation, and maintenance of a software product, a life cycle model is a defined set of processes, activities, and tasks, and their sequencing and interrelationships, spanning the life of the system from its definition to the termination of its use.

Process: A set of interrelated activities designed to accomplish a specified goal. Table 1 lists all 12207 processes and their associated activities. For example *Development* is a process. Within *Development* there are thirteen activities as shown in Table 1. One of these activities is *Software Coding and Testing* which has five tasks.

Activity – A set of actions which, taken as a whole, transform inputs into outputs.

Tasks: Specific actions performed to accomplish an activity. The way that each task is performed, such as testing, is called the *technique* or *method.*

Method/Technique: The approach used to accomplish the task.

Overview

IEEE/EIA Std. 12207 defines a framework within which specific software development life cycles can be defined and provides a standard set of terminology to be used when describing these activities without any specific life cycle.

In this approach, a life cycle model for a specific development effort is the set of processes, activities, and tasks taken as a whole that result in the production of the intended product. A necessary part of this model is the description of the interrelationships between these elements, including when they occur, and how they depend on each other. IEEE/EIA Std. 12207 provides a set of standard processes and associated activities as shown in the Table 1. It also defines the tasks to be performed in the accomplishment of these activities, but it does not define the way tasks are performed.

For example, IEEE/EIA Std. 12207 defines *Development* as a process. Within *Development*, there are thirteen activities as shown in Table 1. One of these activities is *Software Coding and Testing* which has five tasks: develop the code and tests, perform

1

testing on the unit, update user documentation, update test requirements and schedule, and evaluate the test results and code according to a set of six criteria defined by the standard.

Example

As an example, consider a project to develop software for a weapons control system. Suppose that the offeror proposes to develop the system in a series of builds. The life cycle model for this project would consist of a description of the number of builds, when they will be developed, and the specific processes, activities, and tasks to be performed to create the builds. Figure 1 illustrates what this model may look like.

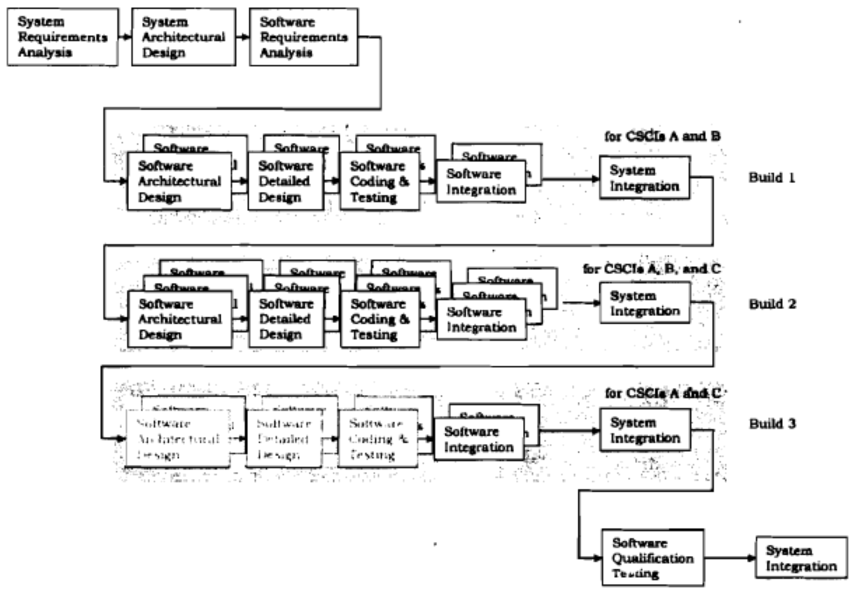

System Requirements Analysis
System Architectural Design
Software Requirements Analysis
Software Architectural Design
Software Detailed Design
Software Coding & Testing
Software Integration
System Integration
for CSCIs A and B
Build 1
for CSCIs A, B, and C
Build 2
for CSCIs A and C
Build 3
Software Qualification Testing
System Integration

Figure 1. Sample Life Cycle Model

A Software Development Plan (SDP) would consist of a description of the life cycle model chosen, the activities planned, their constituent tasks, and the approach to be used in performing each task. This description would include a schedule for each activity and task to be performed, as well as the criteria to be used to determine whether the task was successful. IEEE/EIA Std. 12207 defines the generic content of such plans, but does not define a specific format.

2

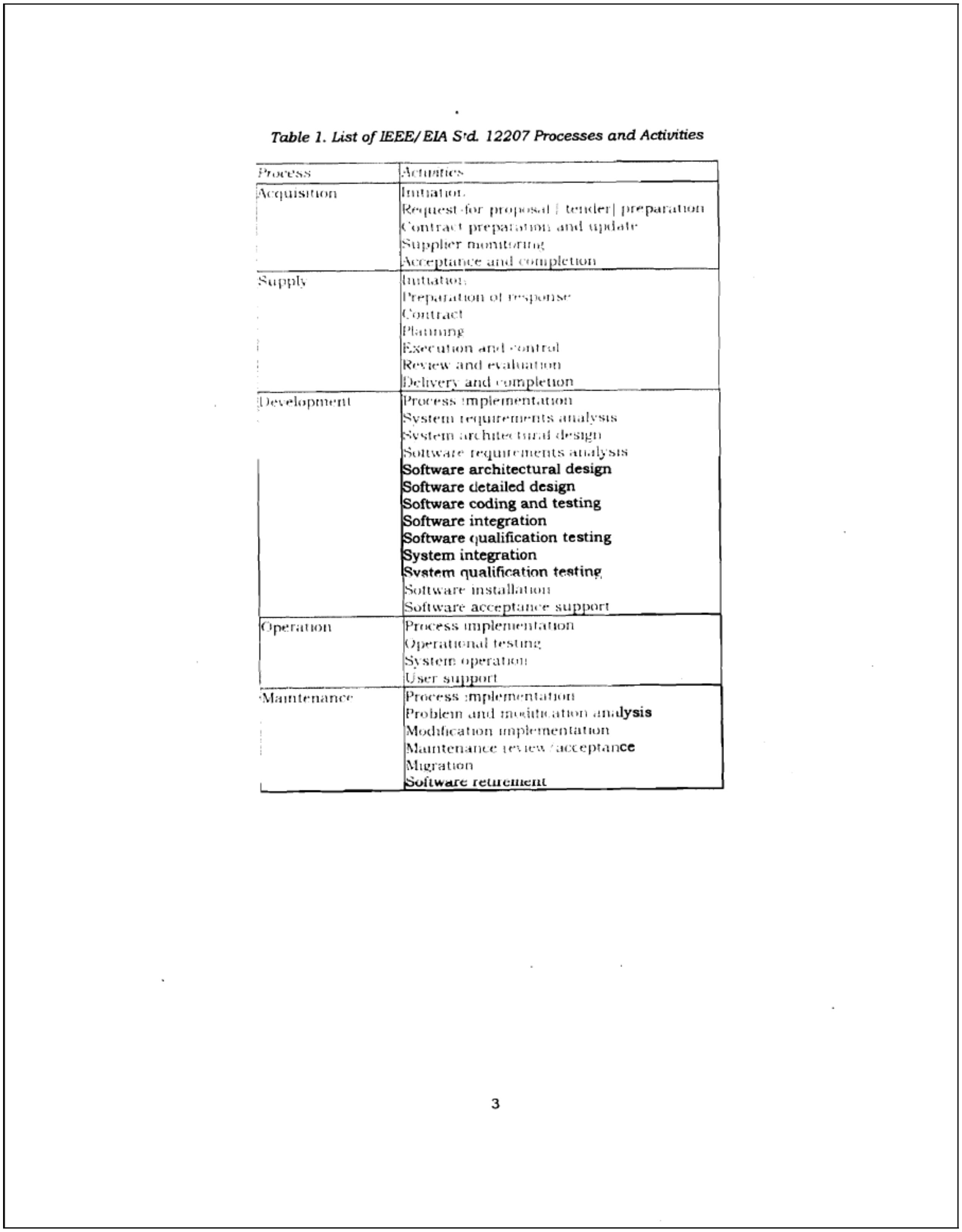


Table 1. List of IEEE/EIA Std. 12207 Processes and Activities

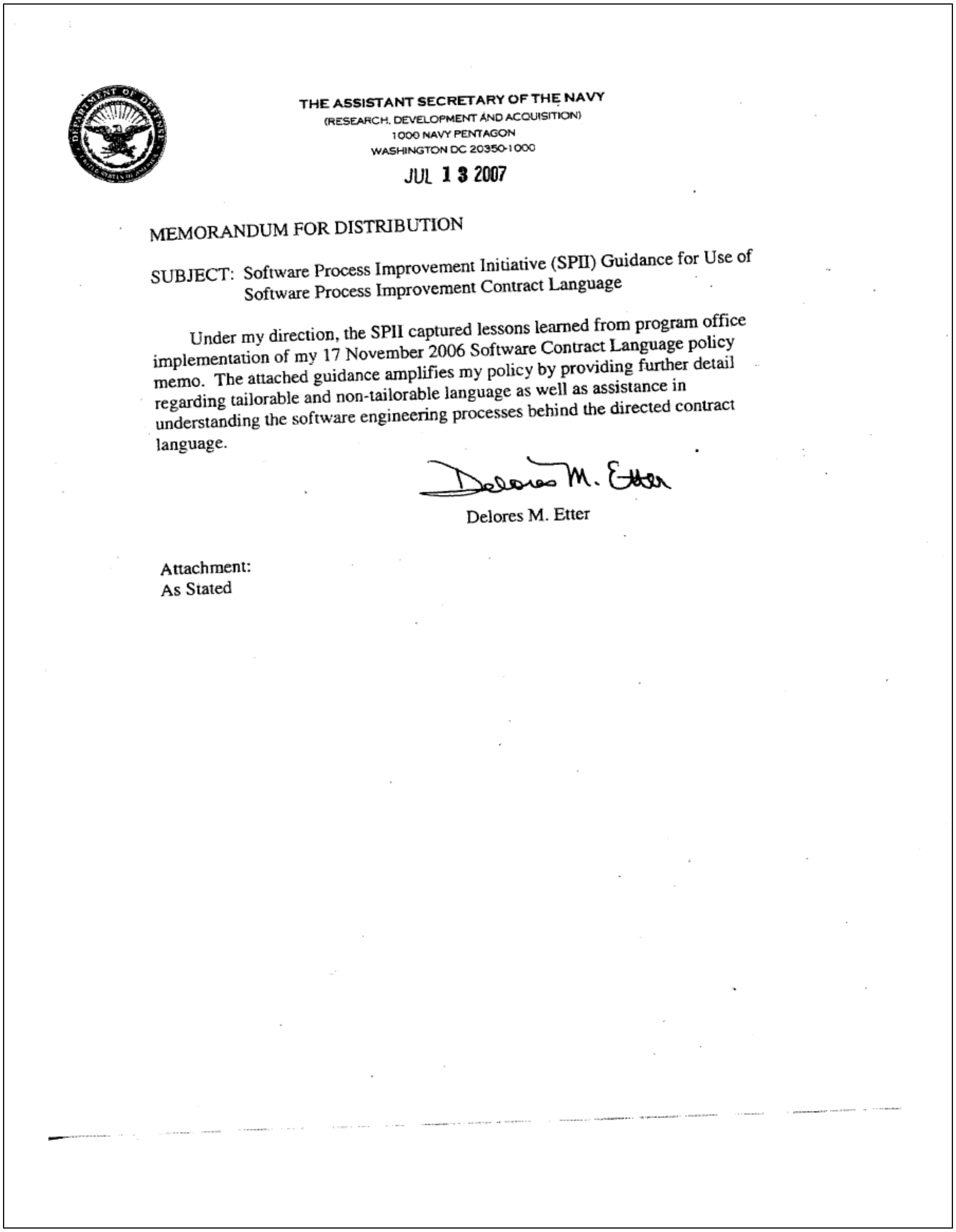
Process	Activities
Acquisition	Initiation Request-for-proposal [tender] preparation Contract preparation and update Supplier monitoring Acceptance and completion
Supply	Initiation Preparation of response Contract Planning Execution and control Review and evaluation Delivery and completion
Development	Process implementation System requirements analysis System architectural design Software requirements analysis Software architectural design Software detailed design Software coding and testing Software integration Software qualification testing System integration System qualification testing Software installation Software acceptance support
Operation	Process implementation Operational testing System operation User support
Maintenance	Process implementation Problem and modification analysis Modification implementation Maintenance review/acceptance Migration Software retirement

3

This Page Intentionally Left Blank

Appendix C

July 07 Software Process Improvement Initiative (SPII) Guidance for Use of Software Process Improvement Contract Language Memo



THE ASSISTANT SECRETARY OF THE NAVY
(RESEARCH, DEVELOPMENT AND ACQUISITION)
1000 NAVY PENTAGON
WASHINGTON DC 20350-1000

JUL 13 2007

MEMORANDUM FOR DISTRIBUTION

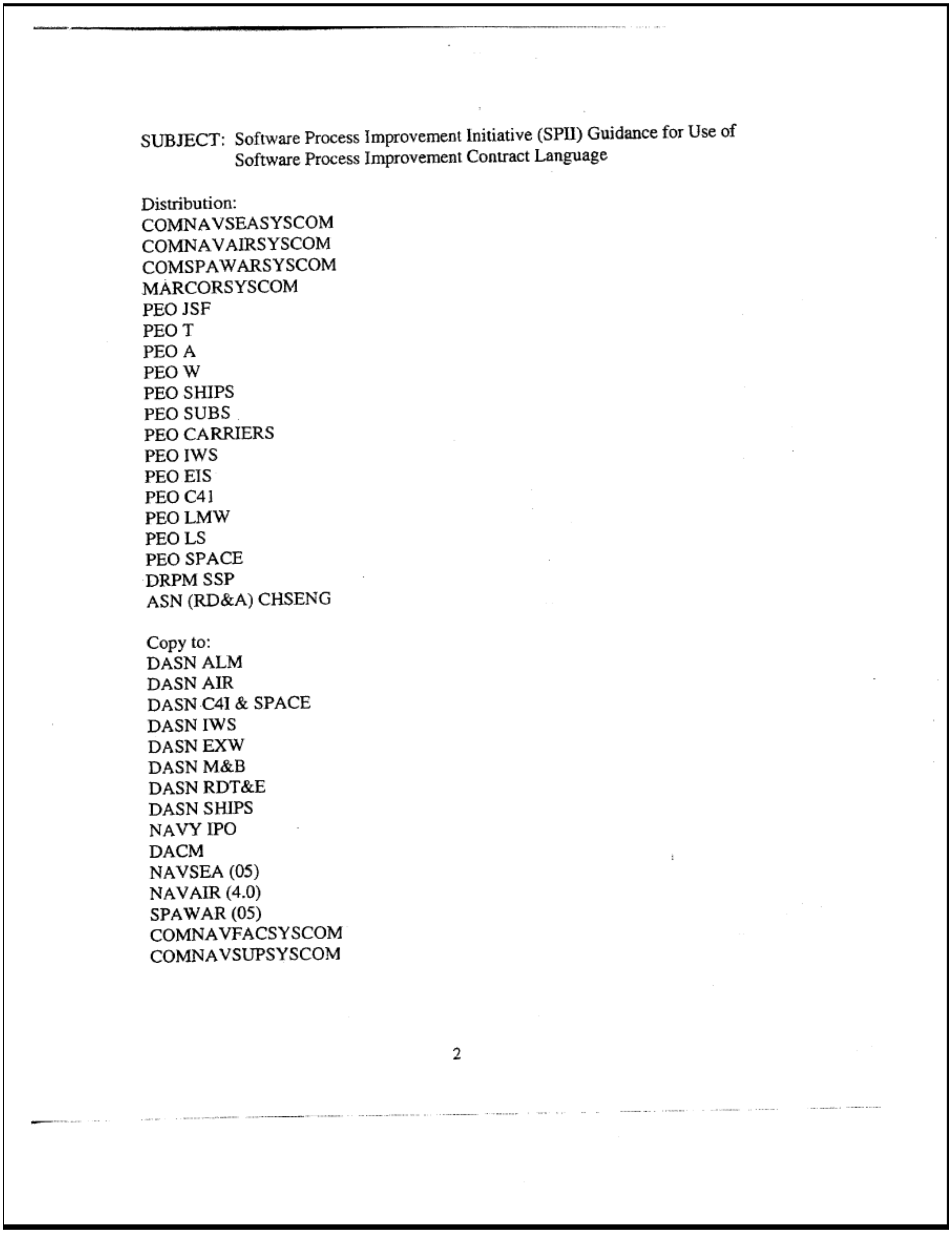
SUBJECT: Software Process Improvement Initiative (SPII) Guidance for Use of Software Process Improvement Contract Language

Under my direction, the SPII captured lessons learned from program office implementation of my 17 November 2006 Software Contract Language policy memo. The attached guidance amplifies my policy by providing further detail regarding tailorable and non-tailorable language as well as assistance in understanding the software engineering processes behind the directed contract language.

Delores M. Etter

Delores M. Etter

Attachment:
As Stated

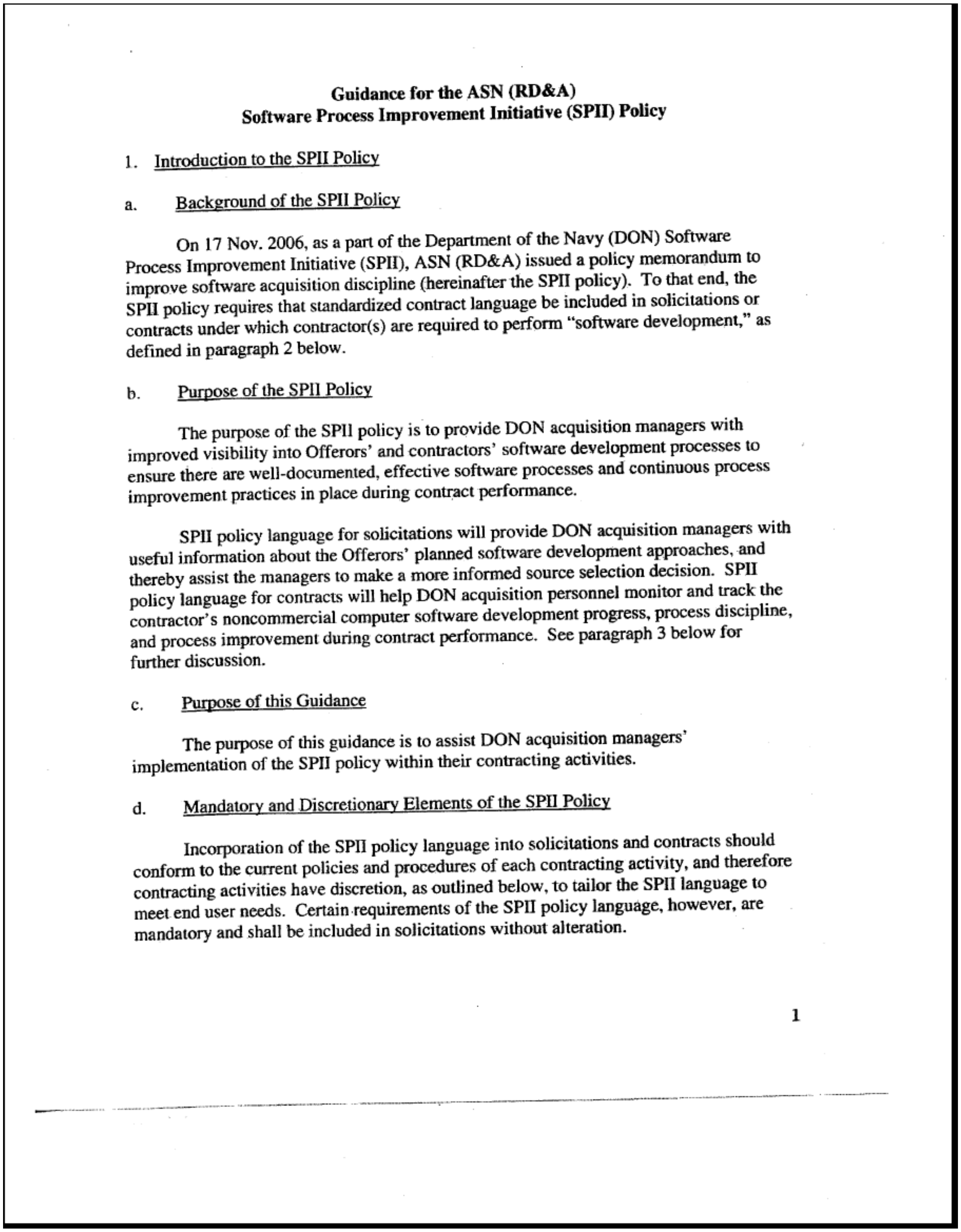


SUBJECT: Software Process Improvement Initiative (SPII) Guidance for Use of Software Process Improvement Contract Language

Distribution:
COMNAVSEASYSCOM
COMNAVAIRSYSCOM
COMSPAWARSYSCOM
MARCORSYSCOM
PEO JSF
PEO T
PEO A
PEO W
PEO SHIPS
PEO SUBS
PEO CARRIERS
PEO IWS
PEO EIS
PEO C4I
PEO LMW
PEO LS
PEO SPACE
DRPM SSP
ASN (RD&A) CHSENG

Copy to:
DASN ALM
DASN AIR
DASN C4I & SPACE
DASN IWS
DASN EXW
DASN M&B
DASN RDT&E
DASN SHIPS
NAVY IPO
DACM
NAVSEA (05)
NAVAIR (4.0)
SPAWAR (05)
COMNAVFACSYSCOM
COMNAVSUPSYSCOM

2



Guidance for the ASN (RD&A) Software Process Improvement Initiative (SPII) Policy

1. Introduction to the SPII Policy

a. Background of the SPII Policy

On 17 Nov. 2006, as a part of the Department of the Navy (DON) Software Process Improvement Initiative (SPII), ASN (RD&A) issued a policy memorandum to improve software acquisition discipline (hereinafter the SPII policy). To that end, the SPII policy requires that standardized contract language be included in solicitations or contracts under which contractor(s) are required to perform "software development," as defined in paragraph 2 below.

b. Purpose of the SPII Policy

The purpose of the SPII policy is to provide DON acquisition managers with improved visibility into Offerors' and contractors' software development processes to ensure there are well-documented, effective software processes and continuous process improvement practices in place during contract performance.

SPII policy language for solicitations will provide DON acquisition managers with useful information about the Offerors' planned software development approaches, and thereby assist the managers to make a more informed source selection decision. SPII policy language for contracts will help DON acquisition personnel monitor and track the contractor's noncommercial computer software development progress, process discipline, and process improvement during contract performance. See paragraph 3 below for further discussion.

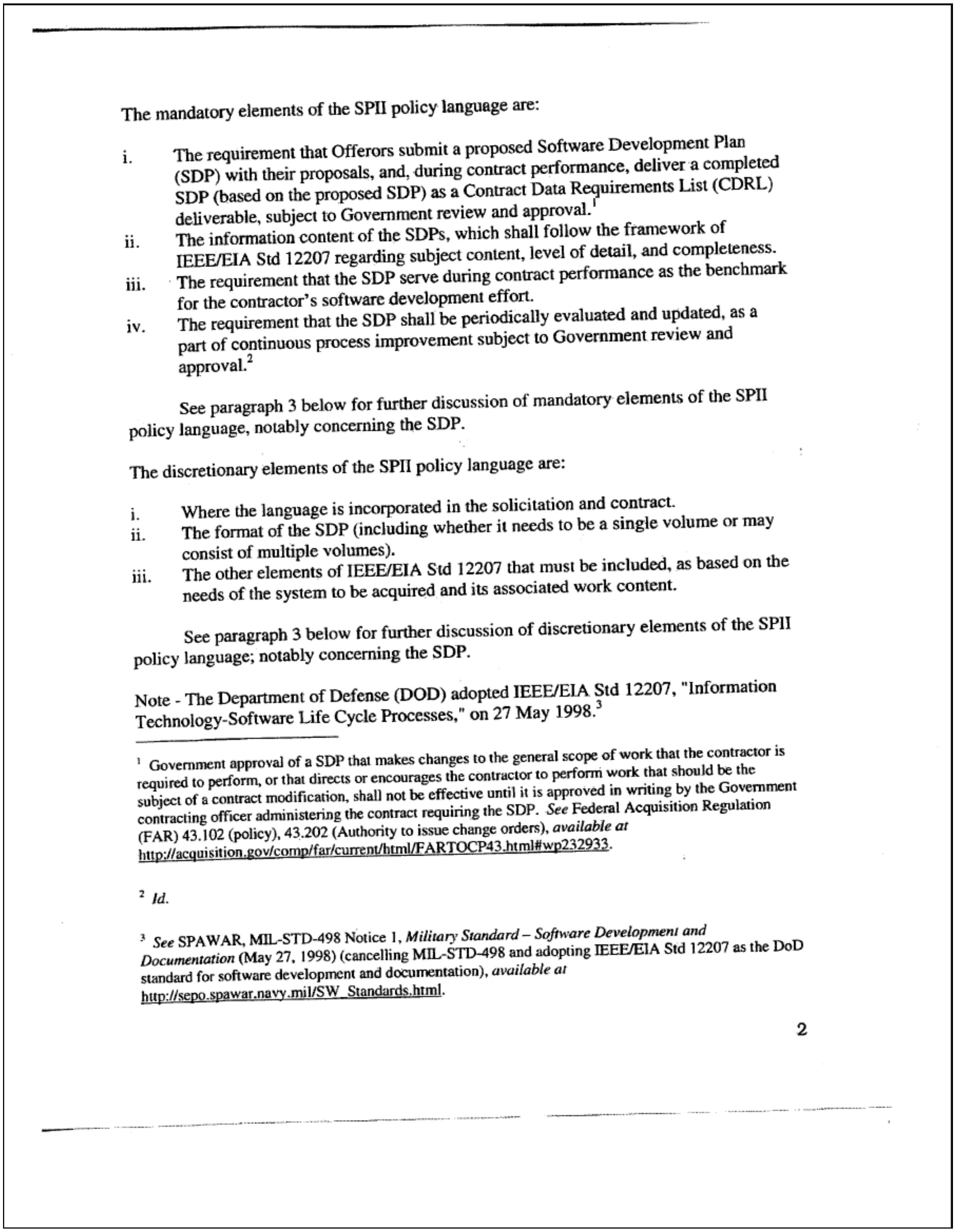
c. Purpose of this Guidance

The purpose of this guidance is to assist DON acquisition managers' implementation of the SPII policy within their contracting activities.

d. Mandatory and Discretionary Elements of the SPII Policy

Incorporation of the SPII policy language into solicitations and contracts should conform to the current policies and procedures of each contracting activity, and therefore contracting activities have discretion, as outlined below, to tailor the SPII language to meet end user needs. Certain requirements of the SPII policy language, however, are mandatory and shall be included in solicitations without alteration.

1



The mandatory elements of the SPII policy language are:

i. The requirement that Offerors submit a proposed Software Development Plan (SDP) with their proposals, and, during contract performance, deliver a completed SDP (based on the proposed SDP) as a Contract Data Requirements List (CDRL) deliverable, subject to Government review and approval.[1]

ii. The information content of the SDPs, which shall follow the framework of IEEE/EIA Std 12207 regarding subject content, level of detail, and completeness.

iii. The requirement that the SDP serve during contract performance as the benchmark for the contractor's software development effort.

iv. The requirement that the SDP shall be periodically evaluated and updated, as a part of continuous process improvement subject to Government review and approval.[2]

See paragraph 3 below for further discussion of mandatory elements of the SPII policy language, notably concerning the SDP.

The discretionary elements of the SPII policy language are:

i. Where the language is incorporated in the solicitation and contract.

ii. The format of the SDP (including whether it needs to be a single volume or may consist of multiple volumes).

iii. The other elements of IEEE/EIA Std 12207 that must be included, as based on the needs of the system to be acquired and its associated work content.

See paragraph 3 below for further discussion of discretionary elements of the SPII policy language; notably concerning the SDP.

Note - The Department of Defense (DOD) adopted IEEE/EIA Std 12207, "Information Technology-Software Life Cycle Processes," on 27 May 1998.[3]

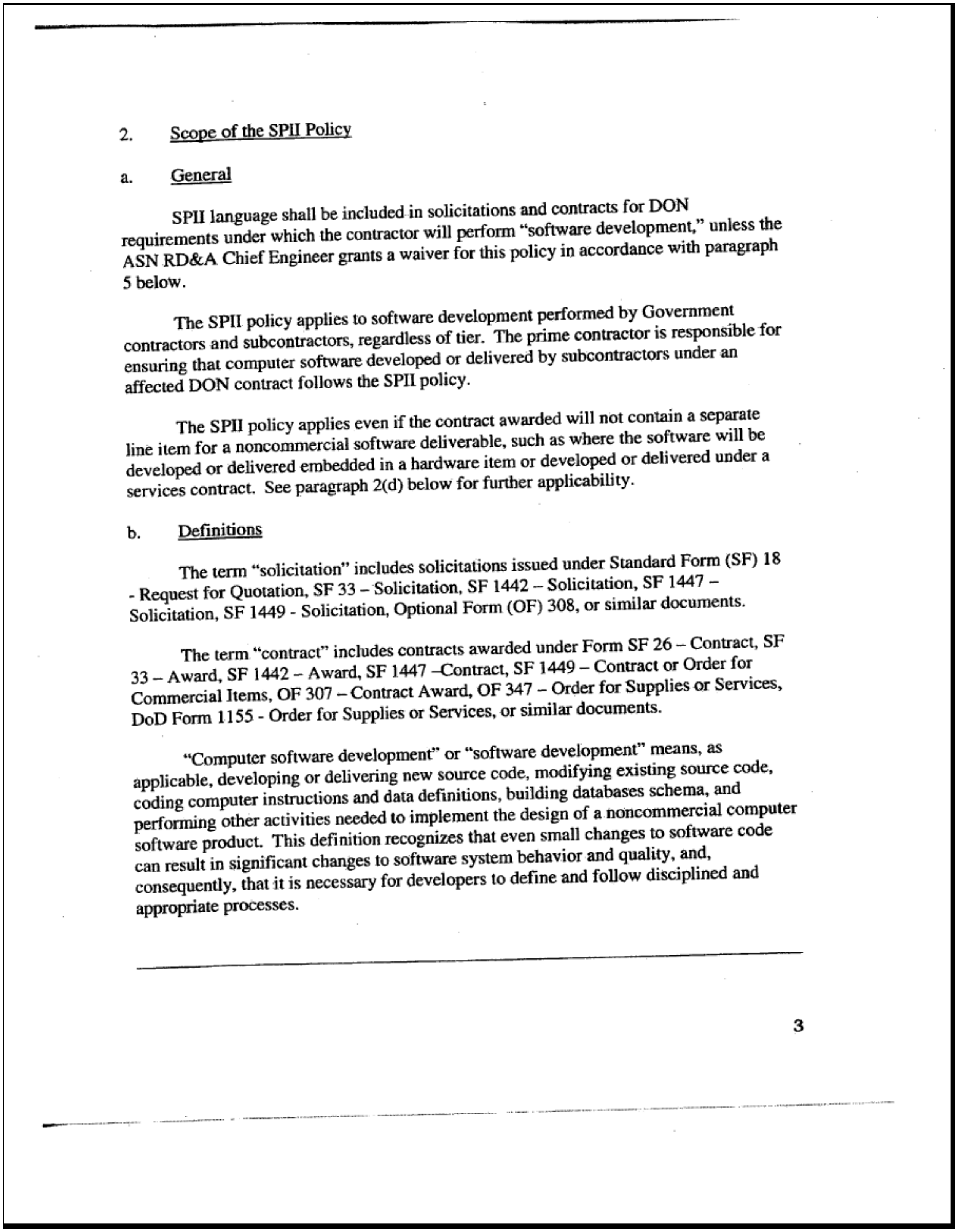
---

[1] Government approval of a SDP that makes changes to the general scope of work that the contractor is required to perform, or that directs or encourages the contractor to perform work that should be the subject of a contract modification, shall not be effective until it is approved in writing by the Government contracting officer administering the contract requiring the SDP. *See* Federal Acquisition Regulation (FAR) 43.102 (policy), 43.202 (Authority to issue change orders), *available at* http://acquisition.gov/comp/far/current/html/FARTOCP43.html#wp232933.

[2] *Id.*

[3] *See* SPAWAR, MIL-STD-498 Notice 1, *Military Standard – Software Development and Documentation* (May 27, 1998) (cancelling MIL-STD-498 and adopting IEEE/EIA Std 12207 as the DoD standard for software development and documentation), *available at* http://sepo.spawar.navy.mil/SW_Standards.html.

2



2. Scope of the SPII Policy

a. General

SPII language shall be included in solicitations and contracts for DON requirements under which the contractor will perform "software development," unless the ASN RD&A Chief Engineer grants a waiver for this policy in accordance with paragraph 5 below.

The SPII policy applies to software development performed by Government contractors and subcontractors, regardless of tier. The prime contractor is responsible for ensuring that computer software developed or delivered by subcontractors under an affected DON contract follows the SPII policy.

The SPII policy applies even if the contract awarded will not contain a separate line item for a noncommercial software deliverable, such as where the software will be developed or delivered embedded in a hardware item or developed or delivered under a services contract. See paragraph 2(d) below for further applicability.

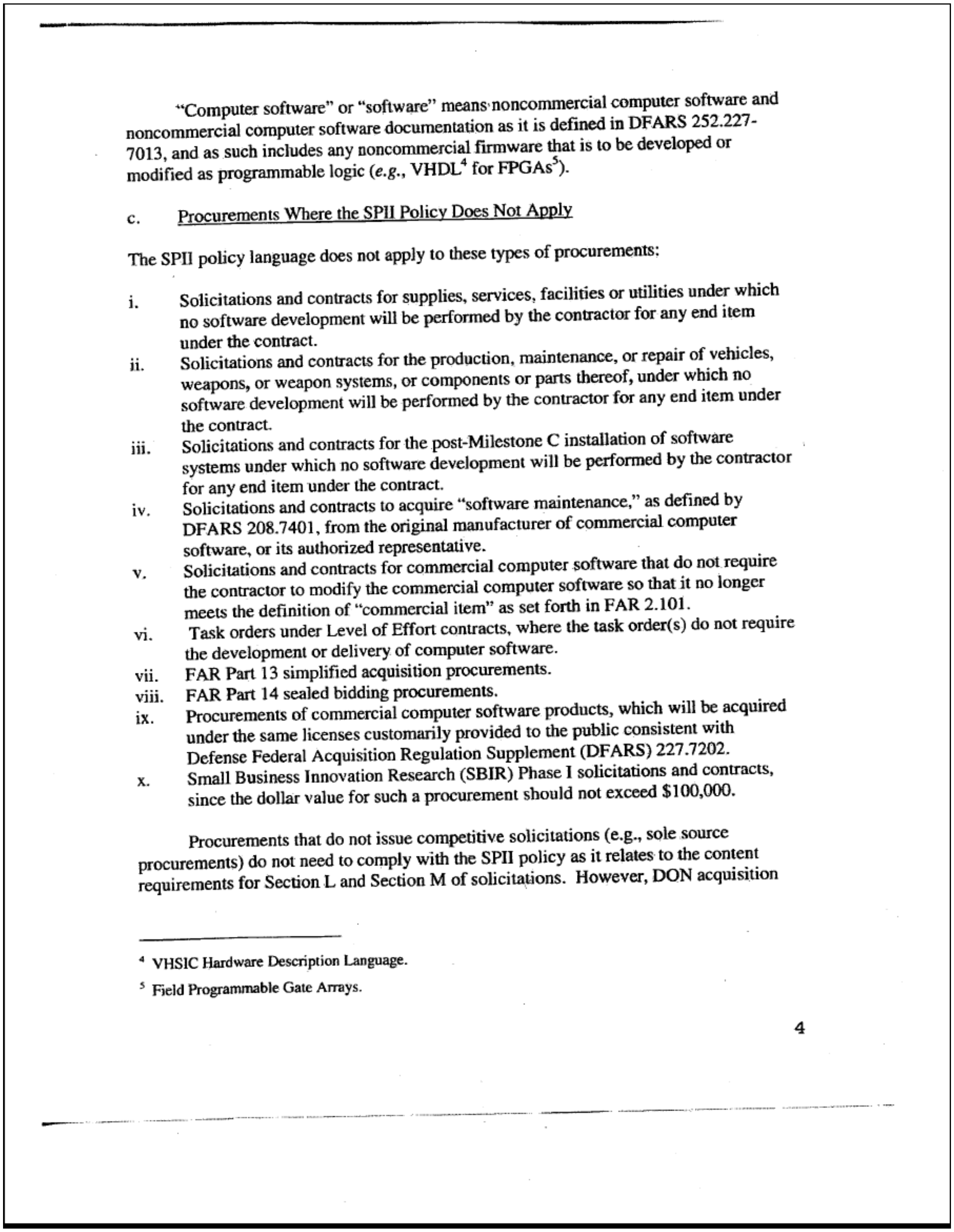
b. Definitions

The term "solicitation" includes solicitations issued under Standard Form (SF) 18 - Request for Quotation, SF 33 – Solicitation, SF 1442 – Solicitation, SF 1447 – Solicitation, SF 1449 - Solicitation, Optional Form (OF) 308, or similar documents.

The term "contract" includes contracts awarded under Form SF 26 – Contract, SF 33 – Award, SF 1442 – Award, SF 1447 –Contract, SF 1449 – Contract or Order for Commercial Items, OF 307 – Contract Award, OF 347 – Order for Supplies or Services, DoD Form 1155 - Order for Supplies or Services, or similar documents.

"Computer software development" or "software development" means, as applicable, developing or delivering new source code, modifying existing source code, coding computer instructions and data definitions, building databases schema, and performing other activities needed to implement the design of a noncommercial computer software product. This definition recognizes that even small changes to software code can result in significant changes to software system behavior and quality, and, consequently, that it is necessary for developers to define and follow disciplined and appropriate processes.

3



"Computer software" or "software" means noncommercial computer software and noncommercial computer software documentation as it is defined in DFARS 252.227-7013, and as such includes any noncommercial firmware that is to be developed or modified as programmable logic (*e.g.*, VHDL[4] for FPGAs[5]).

c. Procurements Where the SPII Policy Does Not Apply

The SPII policy language does not apply to these types of procurements:

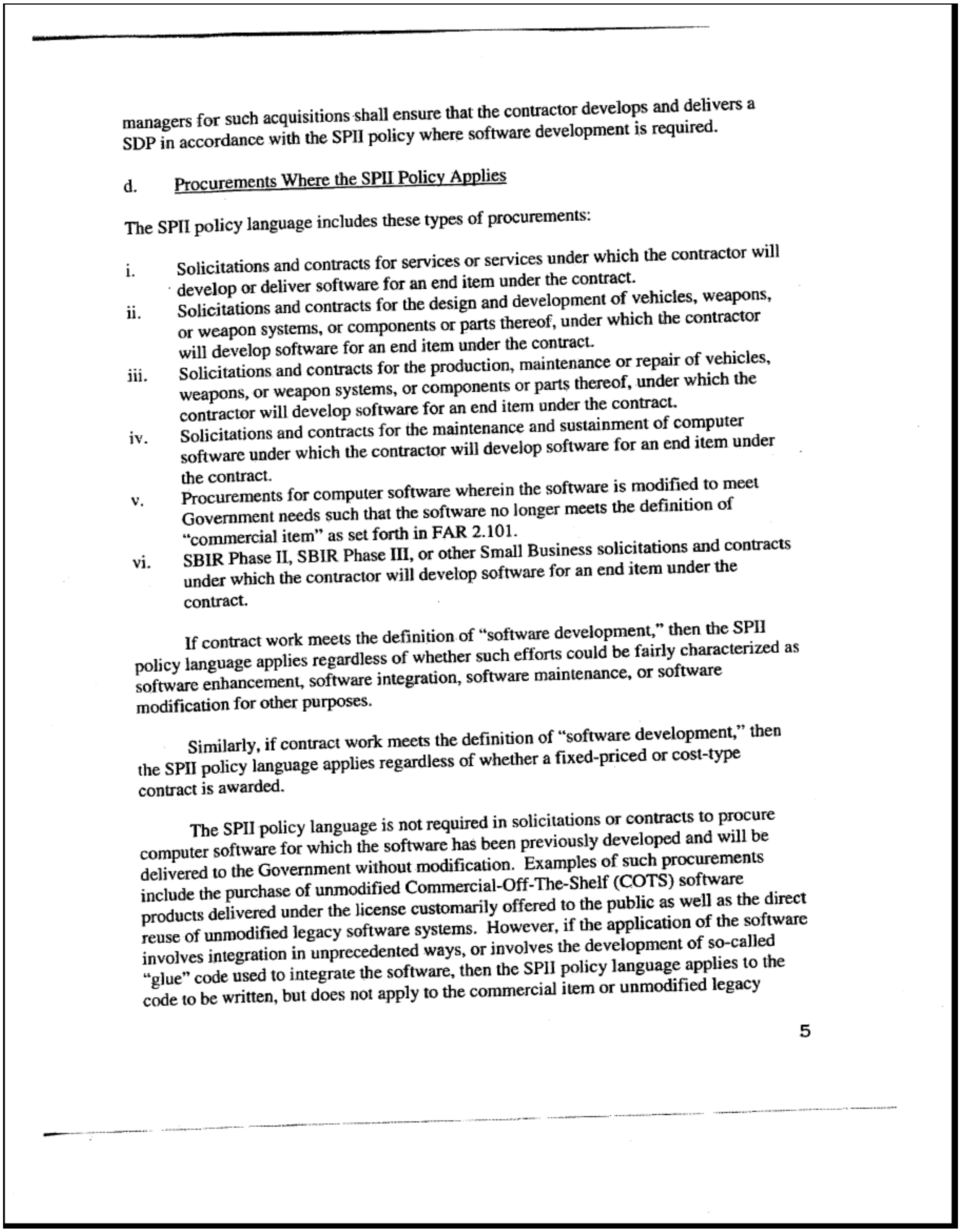
i. Solicitations and contracts for supplies, services, facilities or utilities under which no software development will be performed by the contractor for any end item under the contract.
ii. Solicitations and contracts for the production, maintenance, or repair of vehicles, weapons, or weapon systems, or components or parts thereof, under which no software development will be performed by the contractor for any end item under the contract.
iii. Solicitations and contracts for the post-Milestone C installation of software systems under which no software development will be performed by the contractor for any end item under the contract.
iv. Solicitations and contracts to acquire "software maintenance," as defined by DFARS 208.7401, from the original manufacturer of commercial computer software, or its authorized representative.
v. Solicitations and contracts for commercial computer software that do not require the contractor to modify the commercial computer software so that it no longer meets the definition of "commercial item" as set forth in FAR 2.101.
vi. Task orders under Level of Effort contracts, where the task order(s) do not require the development or delivery of computer software.
vii. FAR Part 13 simplified acquisition procurements.
viii. FAR Part 14 sealed bidding procurements.
ix. Procurements of commercial computer software products, which will be acquired under the same licenses customarily provided to the public consistent with Defense Federal Acquisition Regulation Supplement (DFARS) 227.7202.
x. Small Business Innovation Research (SBIR) Phase I solicitations and contracts, since the dollar value for such a procurement should not exceed $100,000.

Procurements that do not issue competitive solicitations (e.g., sole source procurements) do not need to comply with the SPII policy as it relates to the content requirements for Section L and Section M of solicitations. However, DON acquisition

[4] VHSIC Hardware Description Language.

[5] Field Programmable Gate Arrays.

4



managers for such acquisitions shall ensure that the contractor develops and delivers a SDP in accordance with the SPII policy where software development is required.

d. <u>Procurements Where the SPII Policy Applies</u>

The SPII policy language includes these types of procurements:

i. Solicitations and contracts for services or services under which the contractor will develop or deliver software for an end item under the contract.

ii. Solicitations and contracts for the design and development of vehicles, weapons, or weapon systems, or components or parts thereof, under which the contractor will develop software for an end item under the contract.

iii. Solicitations and contracts for the production, maintenance or repair of vehicles, weapons, or weapon systems, or components or parts thereof, under which the contractor will develop software for an end item under the contract.

iv. Solicitations and contracts for the maintenance and sustainment of computer software under which the contractor will develop software for an end item under the contract.

v. Procurements for computer software wherein the software is modified to meet Government needs such that the software no longer meets the definition of "commercial item" as set forth in FAR 2.101.

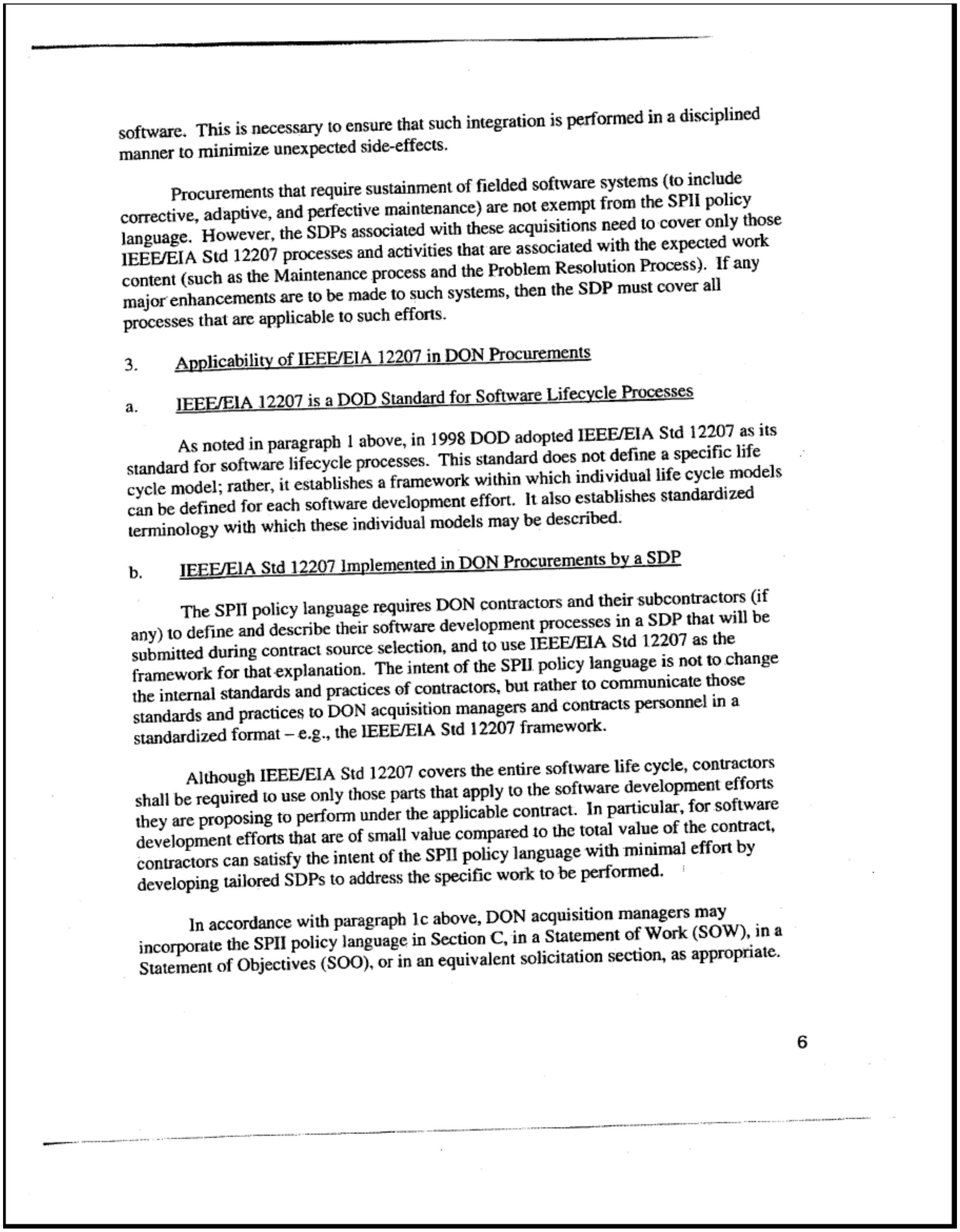
vi. SBIR Phase II, SBIR Phase III, or other Small Business solicitations and contracts under which the contractor will develop software for an end item under the contract.

If contract work meets the definition of "software development," then the SPII policy language applies regardless of whether such efforts could be fairly characterized as software enhancement, software integration, software maintenance, or software modification for other purposes.

Similarly, if contract work meets the definition of "software development," then the SPII policy language applies regardless of whether a fixed-priced or cost-type contract is awarded.

The SPII policy language is not required in solicitations or contracts to procure computer software for which the software has been previously developed and will be delivered to the Government without modification. Examples of such procurements include the purchase of unmodified Commercial-Off-The-Shelf (COTS) software products delivered under the license customarily offered to the public as well as the direct reuse of unmodified legacy software systems. However, if the application of the software involves integration in unprecedented ways, or involves the development of so-called "glue" code used to integrate the software, then the SPII policy language applies to the code to be written, but does not apply to the commercial item or unmodified legacy

5



software. This is necessary to ensure that such integration is performed in a disciplined manner to minimize unexpected side-effects.

Procurements that require sustainment of fielded software systems (to include corrective, adaptive, and perfective maintenance) are not exempt from the SPII policy language. However, the SDPs associated with these acquisitions need to cover only those IEEE/EIA Std 12207 processes and activities that are associated with the expected work content (such as the Maintenance process and the Problem Resolution Process). If any major enhancements are to be made to such systems, then the SDP must cover all processes that are applicable to such efforts.

3. Applicability of IEEE/EIA 12207 in DON Procurements

a. IEEE/EIA 12207 is a DOD Standard for Software Lifecycle Processes

As noted in paragraph 1 above, in 1998 DOD adopted IEEE/EIA Std 12207 as its standard for software lifecycle processes. This standard does not define a specific life cycle model; rather, it establishes a framework within which individual life cycle models can be defined for each software development effort. It also establishes standardized terminology with which these individual models may be described.

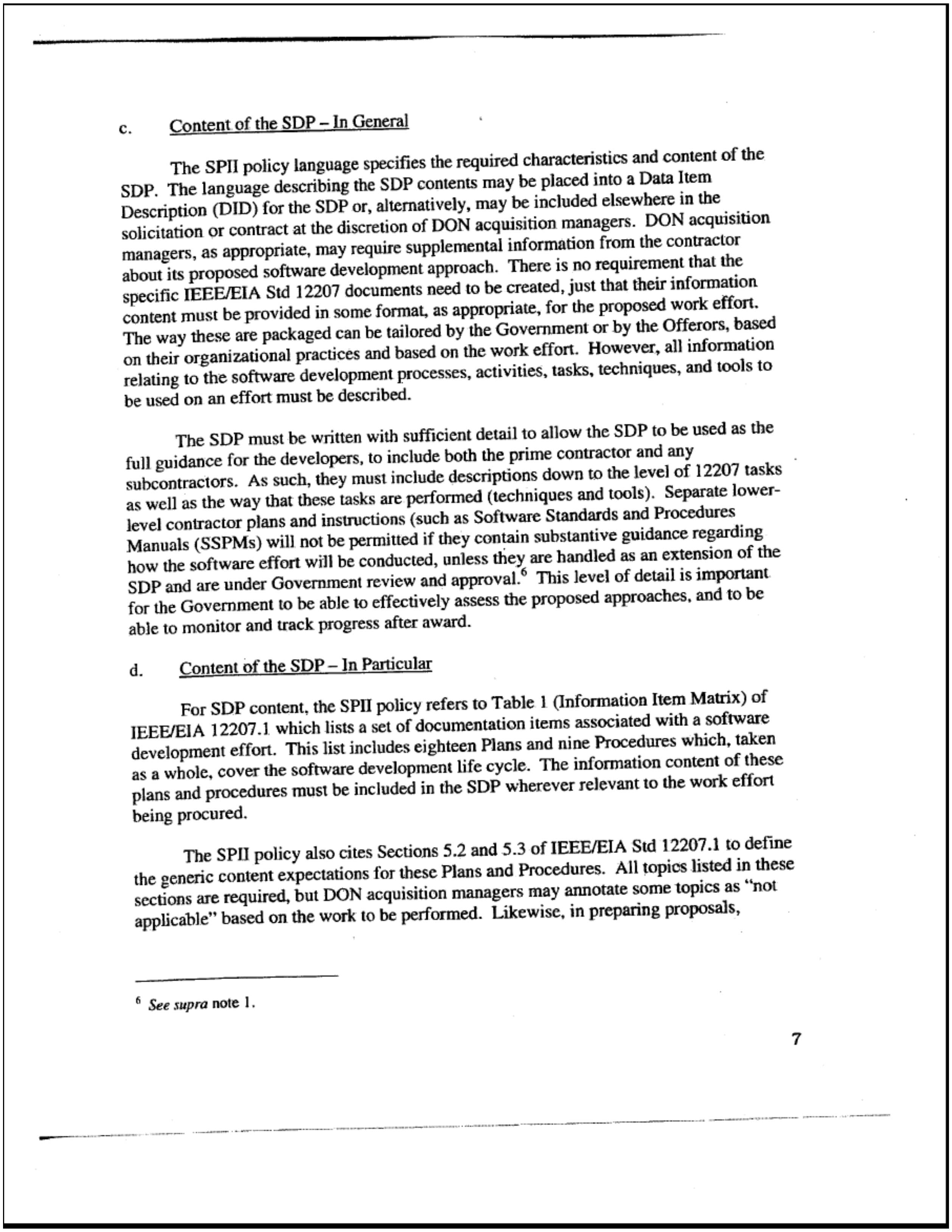
b. IEEE/EIA Std 12207 Implemented in DON Procurements by a SDP

The SPII policy language requires DON contractors and their subcontractors (if any) to define and describe their software development processes in a SDP that will be submitted during contract source selection, and to use IEEE/EIA Std 12207 as the framework for that explanation. The intent of the SPII policy language is not to change the internal standards and practices of contractors, but rather to communicate those standards and practices to DON acquisition managers and contracts personnel in a standardized format – e.g., the IEEE/EIA Std 12207 framework.

Although IEEE/EIA Std 12207 covers the entire software life cycle, contractors shall be required to use only those parts that apply to the software development efforts they are proposing to perform under the applicable contract. In particular, for software development efforts that are of small value compared to the total value of the contract, contractors can satisfy the intent of the SPII policy language with minimal effort by developing tailored SDPs to address the specific work to be performed.

In accordance with paragraph 1c above, DON acquisition managers may incorporate the SPII policy language in Section C, in a Statement of Work (SOW), in a Statement of Objectives (SOO), or in an equivalent solicitation section, as appropriate.

6



c. Content of the SDP – In General

The SPII policy language specifies the required characteristics and content of the SDP. The language describing the SDP contents may be placed into a Data Item Description (DID) for the SDP or, alternatively, may be included elsewhere in the solicitation or contract at the discretion of DON acquisition managers. DON acquisition managers, as appropriate, may require supplemental information from the contractor about its proposed software development approach. There is no requirement that the specific IEEE/EIA Std 12207 documents need to be created, just that their information content must be provided in some format, as appropriate, for the proposed work effort. The way these are packaged can be tailored by the Government or by the Offerors, based on their organizational practices and based on the work effort. However, all information relating to the software development processes, activities, tasks, techniques, and tools to be used on an effort must be described.

The SDP must be written with sufficient detail to allow the SDP to be used as the full guidance for the developers, to include both the prime contractor and any subcontractors. As such, they must include descriptions down to the level of 12207 tasks as well as the way that these tasks are performed (techniques and tools). Separate lower-level contractor plans and instructions (such as Software Standards and Procedures Manuals (SSPMs) will not be permitted if they contain substantive guidance regarding how the software effort will be conducted, unless they are handled as an extension of the SDP and are under Government review and approval.[6] This level of detail is important for the Government to be able to effectively assess the proposed approaches, and to be able to monitor and track progress after award.

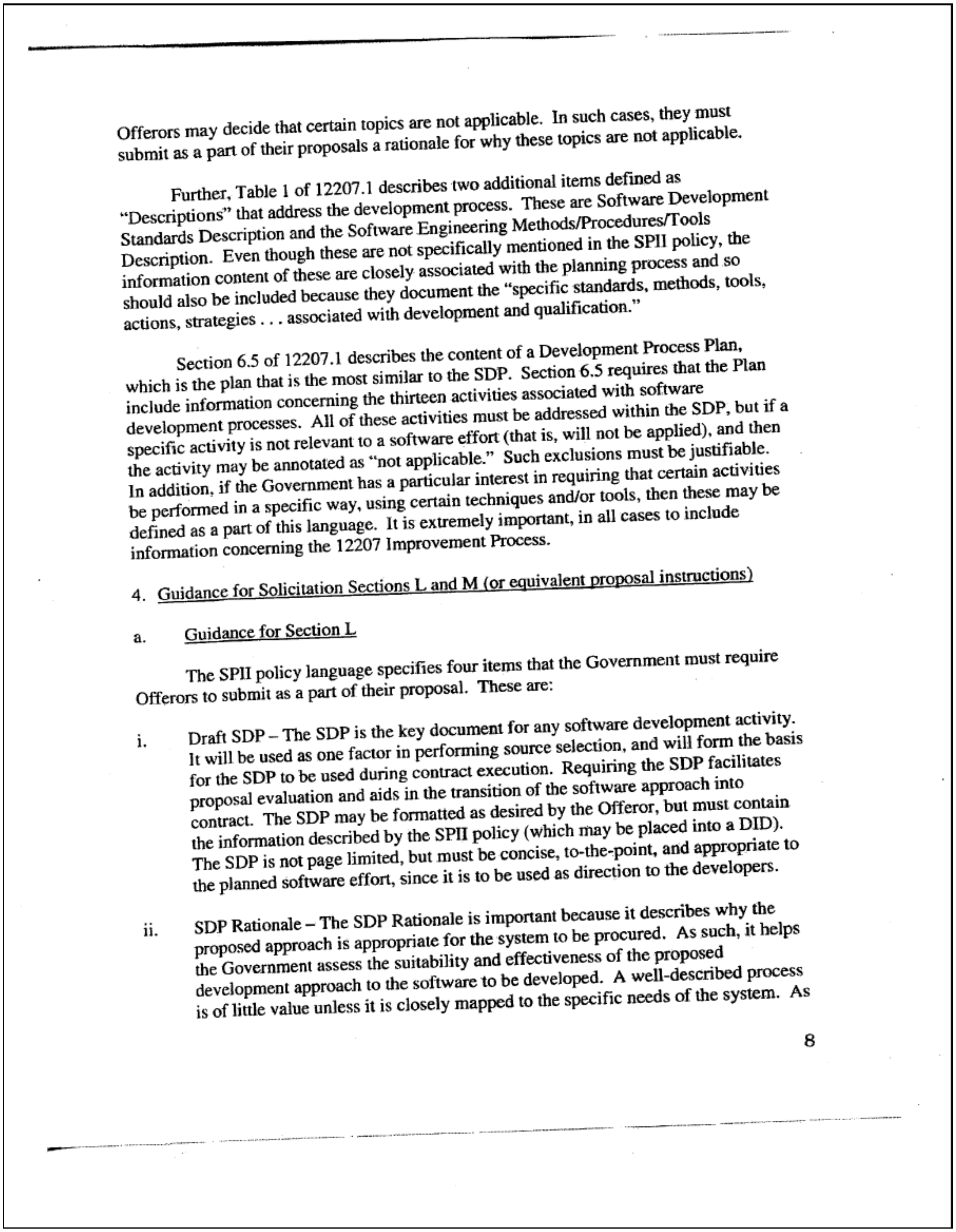
d. Content of the SDP – In Particular

For SDP content, the SPII policy refers to Table 1 (Information Item Matrix) of IEEE/EIA 12207.1 which lists a set of documentation items associated with a software development effort. This list includes eighteen Plans and nine Procedures which, taken as a whole, cover the software development life cycle. The information content of these plans and procedures must be included in the SDP wherever relevant to the work effort being procured.

The SPII policy also cites Sections 5.2 and 5.3 of IEEE/EIA Std 12207.1 to define the generic content expectations for these Plans and Procedures. All topics listed in these sections are required, but DON acquisition managers may annotate some topics as "not applicable" based on the work to be performed. Likewise, in preparing proposals,

[6] *See supra* note 1.

7



Offerors may decide that certain topics are not applicable. In such cases, they must submit as a part of their proposals a rationale for why these topics are not applicable.

Further, Table 1 of 12207.1 describes two additional items defined as "Descriptions" that address the development process. These are Software Development Standards Description and the Software Engineering Methods/Procedures/Tools Description. Even though these are not specifically mentioned in the SPII policy, the information content of these are closely associated with the planning process and so should also be included because they document the "specific standards, methods, tools, actions, strategies . . . associated with development and qualification."

Section 6.5 of 12207.1 describes the content of a Development Process Plan, which is the plan that is the most similar to the SDP. Section 6.5 requires that the Plan include information concerning the thirteen activities associated with software development processes. All of these activities must be addressed within the SDP, but if a specific activity is not relevant to a software effort (that is, will not be applied), and then the activity may be annotated as "not applicable." Such exclusions must be justifiable. In addition, if the Government has a particular interest in requiring that certain activities be performed in a specific way, using certain techniques and/or tools, then these may be defined as a part of this language. It is extremely important, in all cases to include information concerning the 12207 Improvement Process.

4. <u>Guidance for Solicitation Sections L and M (or equivalent proposal instructions)</u>

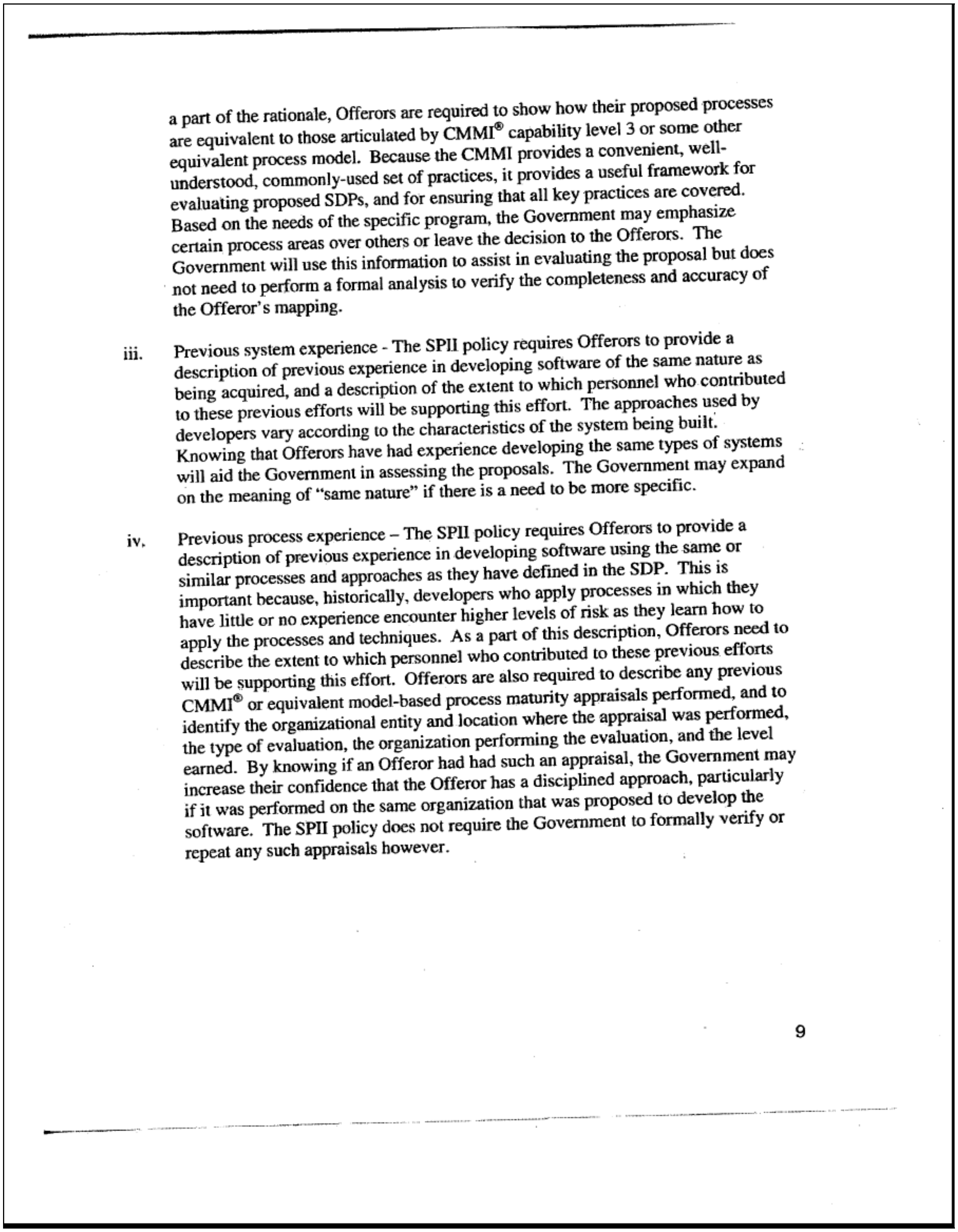
a. <u>Guidance for Section L</u>

The SPII policy language specifies four items that the Government must require Offerors to submit as a part of their proposal. These are:

i. Draft SDP – The SDP is the key document for any software development activity. It will be used as one factor in performing source selection, and will form the basis for the SDP to be used during contract execution. Requiring the SDP facilitates proposal evaluation and aids in the transition of the software approach into contract. The SDP may be formatted as desired by the Offeror, but must contain the information described by the SPII policy (which may be placed into a DID). The SDP is not page limited, but must be concise, to-the-point, and appropriate to the planned software effort, since it is to be used as direction to the developers.

ii. SDP Rationale – The SDP Rationale is important because it describes why the proposed approach is appropriate for the system to be procured. As such, it helps the Government assess the suitability and effectiveness of the proposed development approach to the software to be developed. A well-described process is of little value unless it is closely mapped to the specific needs of the system. As

8

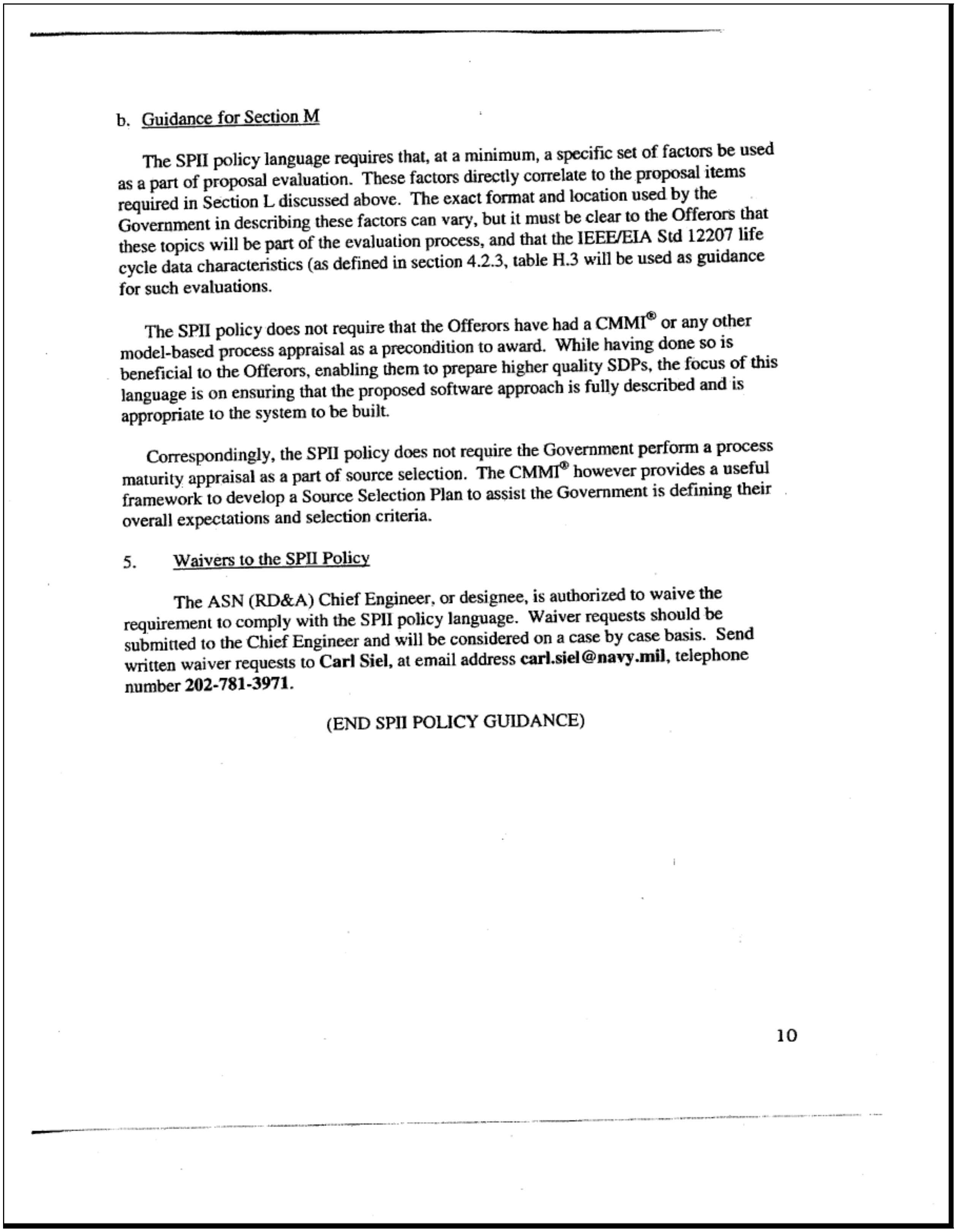


a part of the rationale, Offerors are required to show how their proposed processes are equivalent to those articulated by CMMI® capability level 3 or some other equivalent process model. Because the CMMI provides a convenient, well-understood, commonly-used set of practices, it provides a useful framework for evaluating proposed SDPs, and for ensuring that all key practices are covered. Based on the needs of the specific program, the Government may emphasize certain process areas over others or leave the decision to the Offerors. The Government will use this information to assist in evaluating the proposal but does not need to perform a formal analysis to verify the completeness and accuracy of the Offeror's mapping.

iii. Previous system experience - The SPII policy requires Offerors to provide a description of previous experience in developing software of the same nature as being acquired, and a description of the extent to which personnel who contributed to these previous efforts will be supporting this effort. The approaches used by developers vary according to the characteristics of the system being built. Knowing that Offerors have had experience developing the same types of systems will aid the Government in assessing the proposals. The Government may expand on the meaning of "same nature" if there is a need to be more specific.

iv. Previous process experience – The SPII policy requires Offerors to provide a description of previous experience in developing software using the same or similar processes and approaches as they have defined in the SDP. This is important because, historically, developers who apply processes in which they have little or no experience encounter higher levels of risk as they learn how to apply the processes and techniques. As a part of this description, Offerors need to describe the extent to which personnel who contributed to these previous efforts will be supporting this effort. Offerors are also required to describe any previous CMMI® or equivalent model-based process maturity appraisals performed, and to identify the organizational entity and location where the appraisal was performed, the type of evaluation, the organization performing the evaluation, and the level earned. By knowing if an Offeror had had such an appraisal, the Government may increase their confidence that the Offeror has a disciplined approach, particularly if it was performed on the same organization that was proposed to develop the software. The SPII policy does not require the Government to formally verify or repeat any such appraisals however.

9



b. Guidance for Section M

The SPII policy language requires that, at a minimum, a specific set of factors be used as a part of proposal evaluation. These factors directly correlate to the proposal items required in Section L discussed above. The exact format and location used by the Government in describing these factors can vary, but it must be clear to the Offerors that these topics will be part of the evaluation process, and that the IEEE/EIA Std 12207 life cycle data characteristics (as defined in section 4.2.3, table H.3 will be used as guidance for such evaluations.

The SPII policy does not require that the Offerors have had a CMMI® or any other model-based process appraisal as a precondition to award. While having done so is beneficial to the Offerors, enabling them to prepare higher quality SDPs, the focus of this language is on ensuring that the proposed software approach is fully described and is appropriate to the system to be built.

Correspondingly, the SPII policy does not require the Government perform a process maturity appraisal as a part of source selection. The CMMI® however provides a useful framework to develop a Source Selection Plan to assist the Government is defining their overall expectations and selection criteria.

5. Waivers to the SPII Policy

The ASN (RD&A) Chief Engineer, or designee, is authorized to waive the requirement to comply with the SPII policy language. Waiver requests should be submitted to the Chief Engineer and will be considered on a case by case basis. Send written waiver requests to **Carl Siel**, at email address **carl.siel@navy.mil**, telephone number **202-781-3971**.

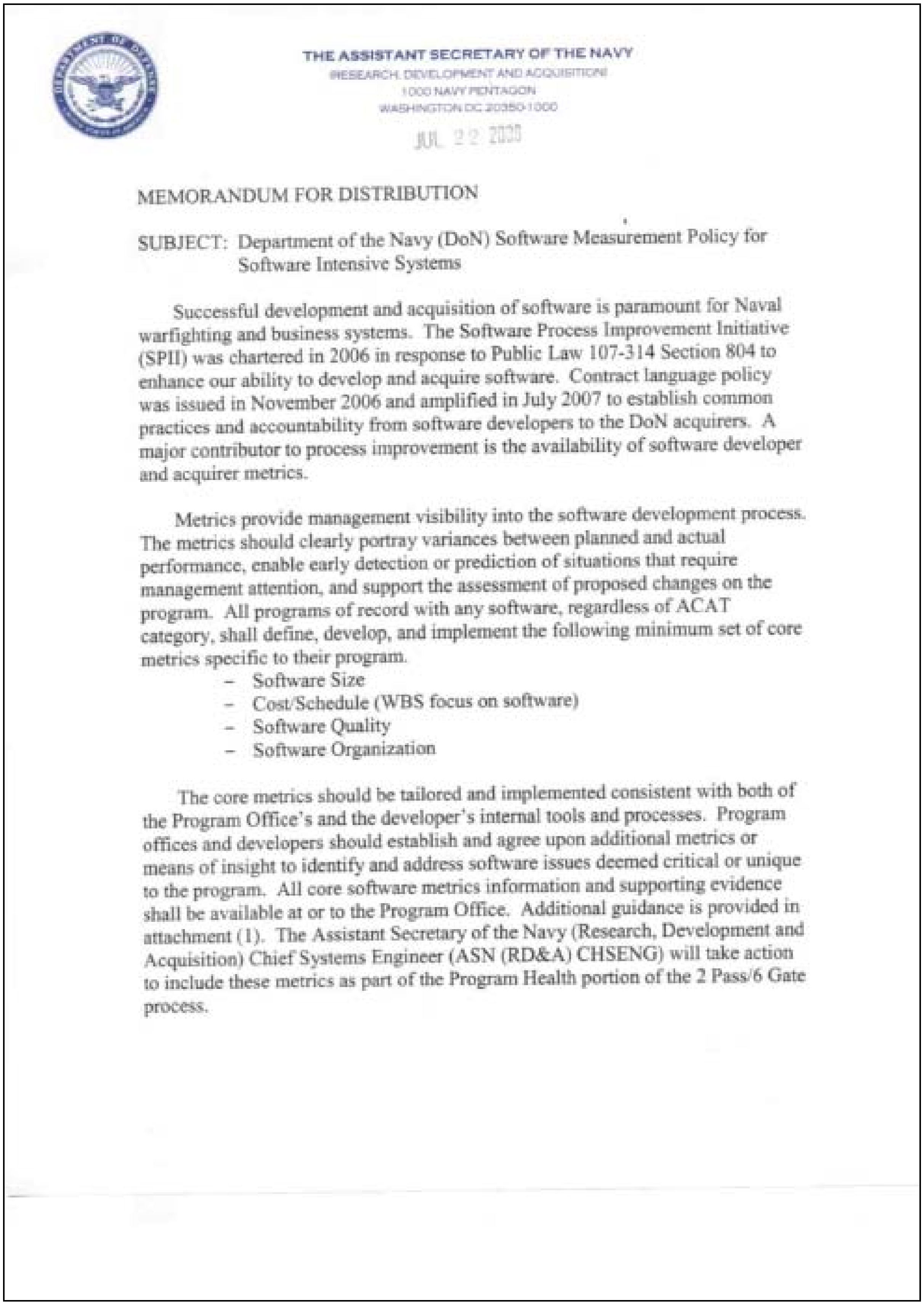
(END SPII POLICY GUIDANCE)

10

This Page Intentionally Left Blank

Appendix D

July 08 Department of the Navy (DoN) Software Measurement Policy for Software Intensive Systems Memo



THE ASSISTANT SECRETARY OF THE NAVY
(RESEARCH, DEVELOPMENT AND ACQUISITION)
1000 NAVY PENTAGON
WASHINGTON DC 20350-1000

JUL 22 2008

MEMORANDUM FOR DISTRIBUTION

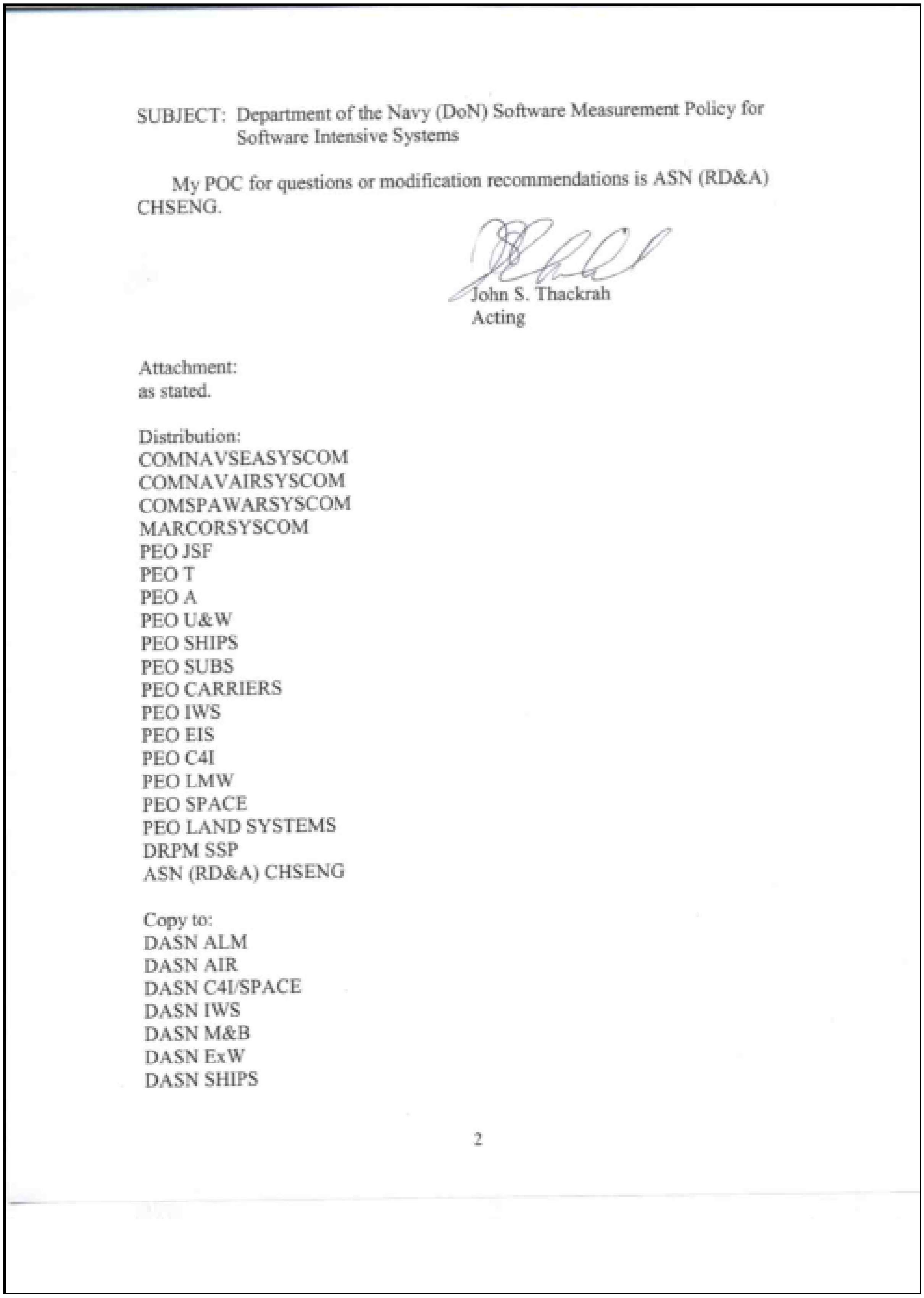
SUBJECT: Department of the Navy (DoN) Software Measurement Policy for Software Intensive Systems

Successful development and acquisition of software is paramount for Naval warfighting and business systems. The Software Process Improvement Initiative (SPII) was chartered in 2006 in response to Public Law 107-314 Section 804 to enhance our ability to develop and acquire software. Contract language policy was issued in November 2006 and amplified in July 2007 to establish common practices and accountability from software developers to the DoN acquirers. A major contributor to process improvement is the availability of software developer and acquirer metrics.

Metrics provide management visibility into the software development process. The metrics should clearly portray variances between planned and actual performance, enable early detection or prediction of situations that require management attention, and support the assessment of proposed changes on the program. All programs of record with any software, regardless of ACAT category, shall define, develop, and implement the following minimum set of core metrics specific to their program.

- Software Size
- Cost/Schedule (WBS focus on software)
- Software Quality
- Software Organization

The core metrics should be tailored and implemented consistent with both of the Program Office's and the developer's internal tools and processes. Program offices and developers should establish and agree upon additional metrics or means of insight to identify and address software issues deemed critical or unique to the program. All core software metrics information and supporting evidence shall be available at or to the Program Office. Additional guidance is provided in attachment (1). The Assistant Secretary of the Navy (Research, Development and Acquisition) Chief Systems Engineer (ASN (RD&A) CHSENG) will take action to include these metrics as part of the Program Health portion of the 2 Pass/6 Gate process.



SUBJECT: Department of the Navy (DoN) Software Measurement Policy for Software Intensive Systems

My POC for questions or modification recommendations is ASN (RD&A) CHSENG.

[illegible]

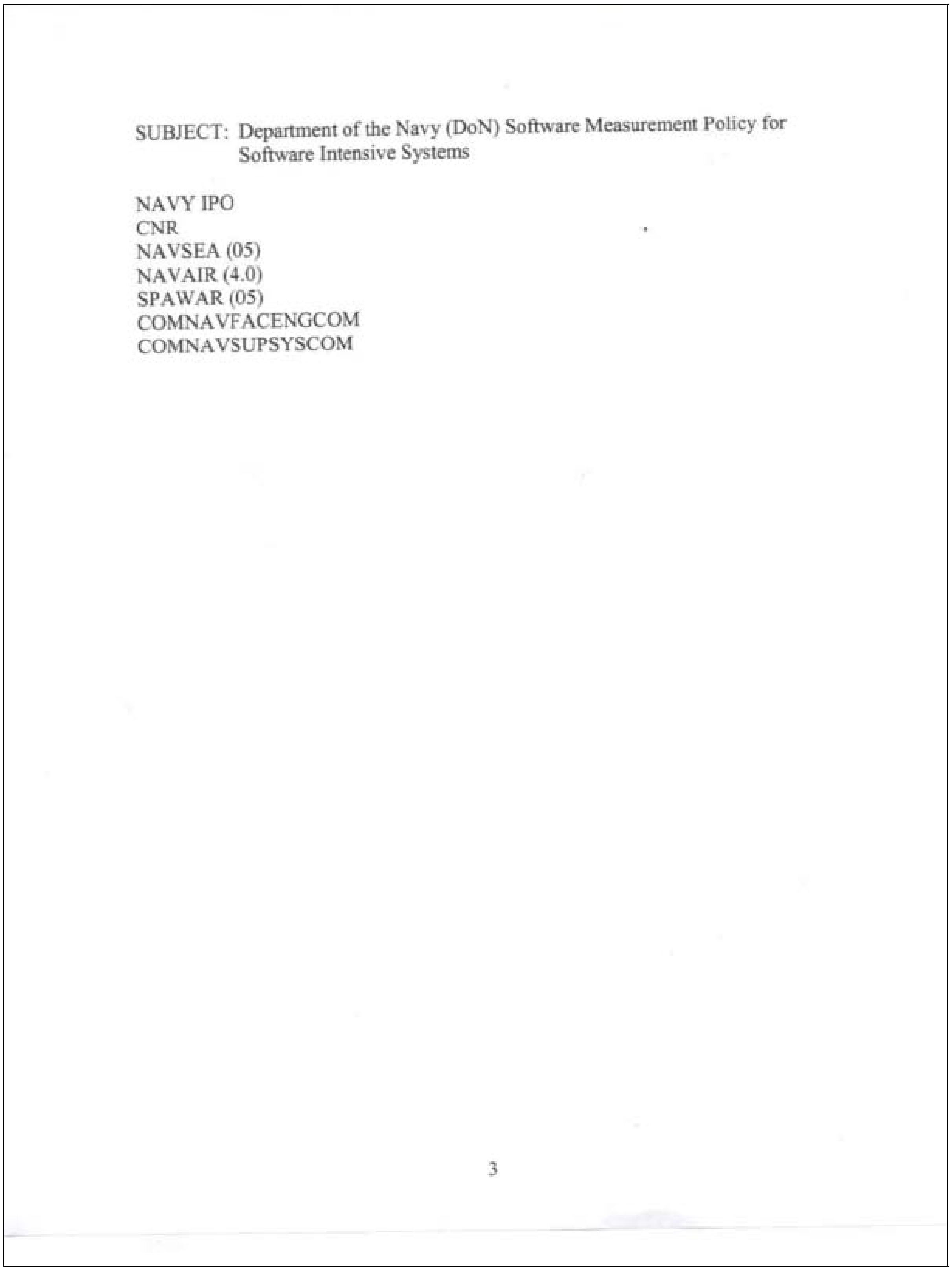
John S. Thackrah
Acting

Attachment:
as stated.

Distribution:
COMNAVSEASYSCOM
COMNAVAIRSYSCOM
COMSPAWARSYSCOM
MARCORSYSCOM
PEO JSF
PEO T
PEO A
PEO U&W
PEO SHIPS
PEO SUBS
PEO CARRIERS
PEO IWS
PEO EIS
PEO C4I
PEO LMW
PEO SPACE
PEO LAND SYSTEMS
DRPM SSP
ASN (RD&A) CHSENG

Copy to:
DASN ALM
DASN AIR
DASN C4I/SPACE
DASN IWS
DASN M&B
DASN ExW
DASN SHIPS

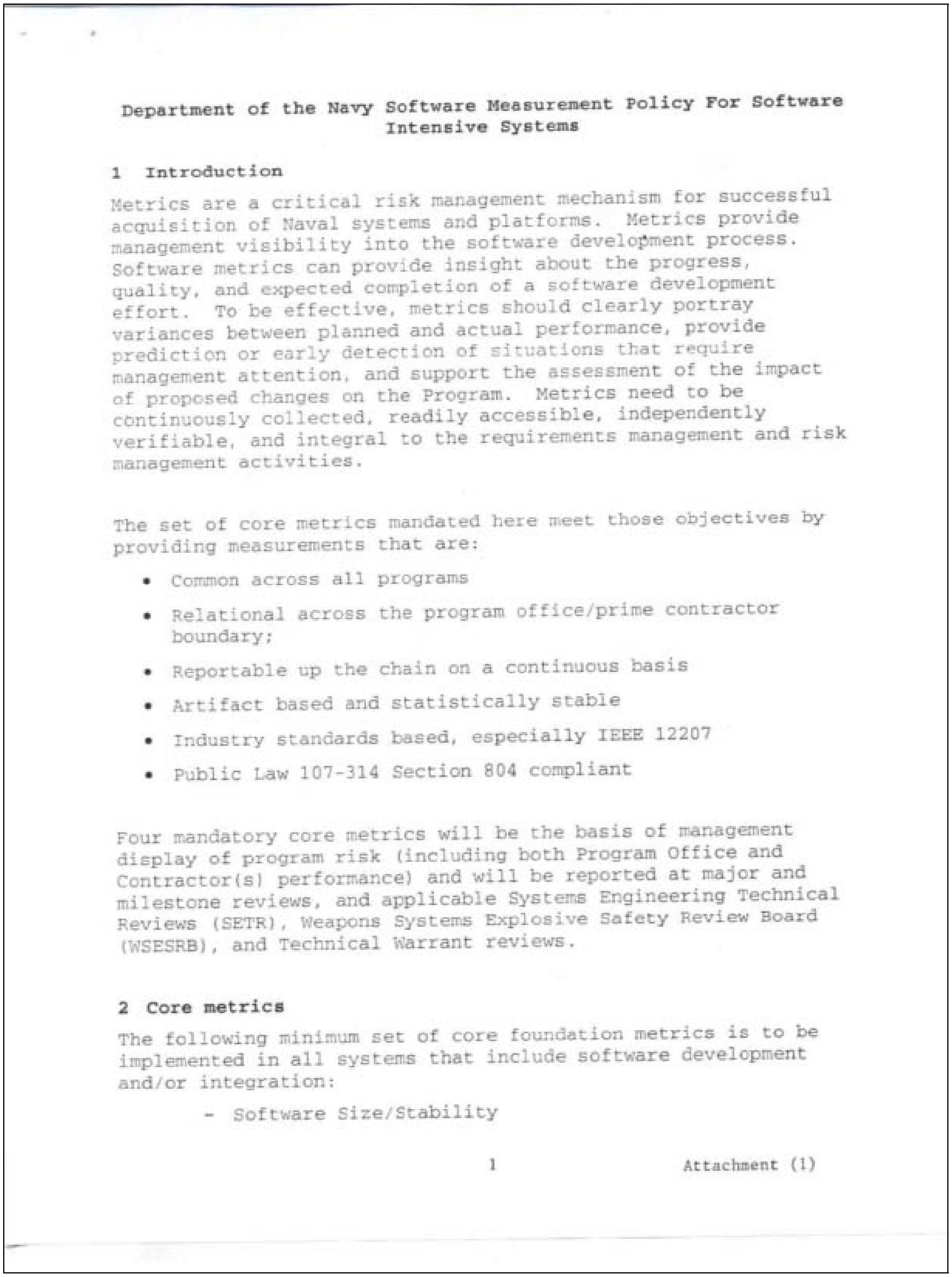
2



SUBJECT: Department of the Navy (DoN) Software Measurement Policy for Software Intensive Systems

NAVY IPO
CNR
NAVSEA (05)
NAVAIR (4.0)
SPAWAR (05)
COMNAVFACENGCOM
COMNAVSUPSYSCOM

3



Department of the Navy Software Measurement Policy For Software Intensive Systems

1 Introduction

Metrics are a critical risk management mechanism for successful acquisition of Naval systems and platforms. Metrics provide management visibility into the software development process. Software metrics can provide insight about the progress, quality, and expected completion of a software development effort. To be effective, metrics should clearly portray variances between planned and actual performance, provide prediction or early detection of situations that require management attention, and support the assessment of the impact of proposed changes on the Program. Metrics need to be continuously collected, readily accessible, independently verifiable, and integral to the requirements management and risk management activities.

The set of core metrics mandated here meet those objectives by providing measurements that are:

- Common across all programs
- Relational across the program office/prime contractor boundary;
- Reportable up the chain on a continuous basis
- Artifact based and statistically stable
- Industry standards based, especially IEEE 12207
- Public Law 107-314 Section 804 compliant

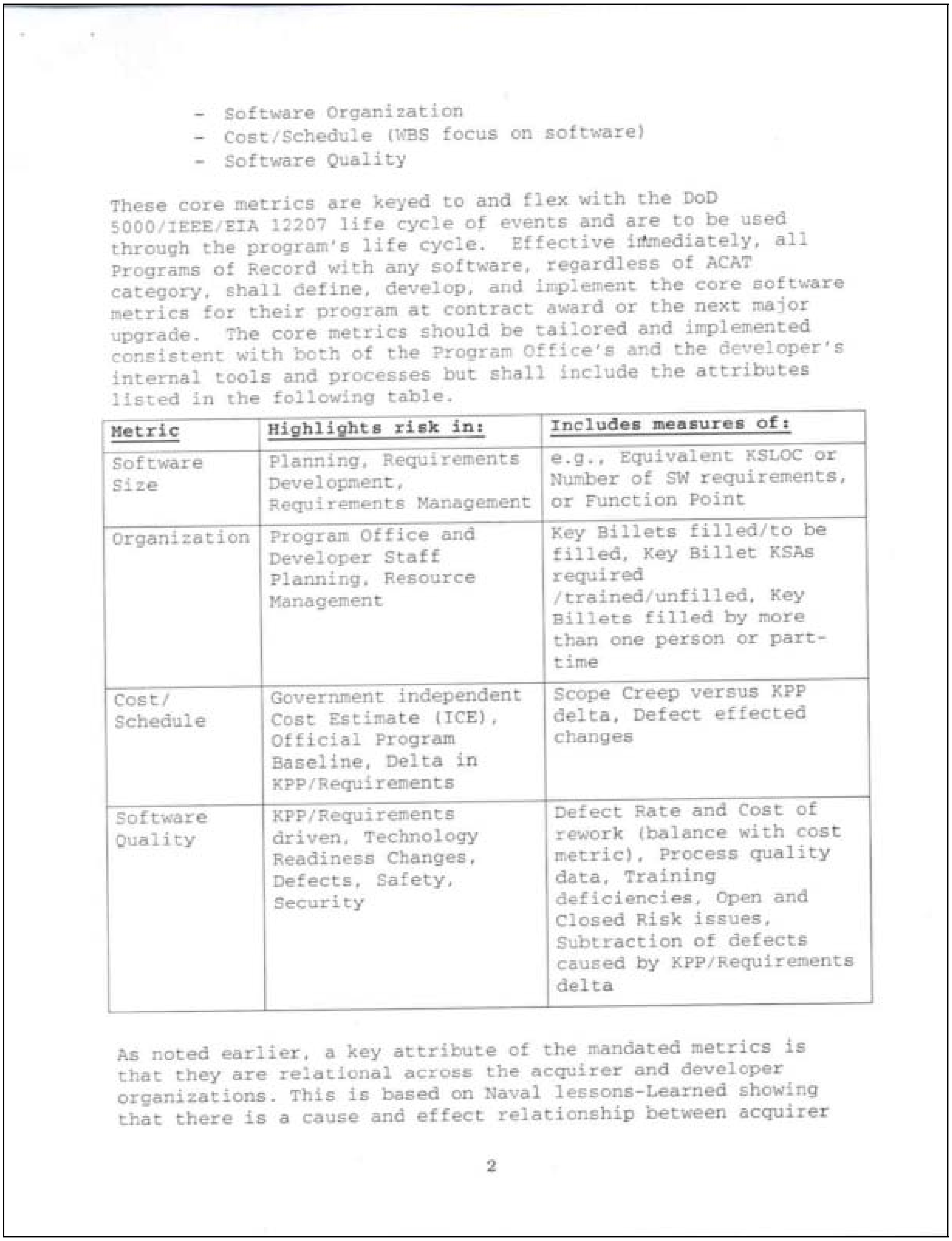
Four mandatory core metrics will be the basis of management display of program risk (including both Program Office and Contractor(s) performance) and will be reported at major and milestone reviews, and applicable Systems Engineering Technical Reviews (SETR), Weapons Systems Explosive Safety Review Board (WSESRB), and Technical Warrant reviews.

2 Core metrics

The following minimum set of core foundation metrics is to be implemented in all systems that include software development and/or integration:

- Software Size/Stability

1 Attachment (1)



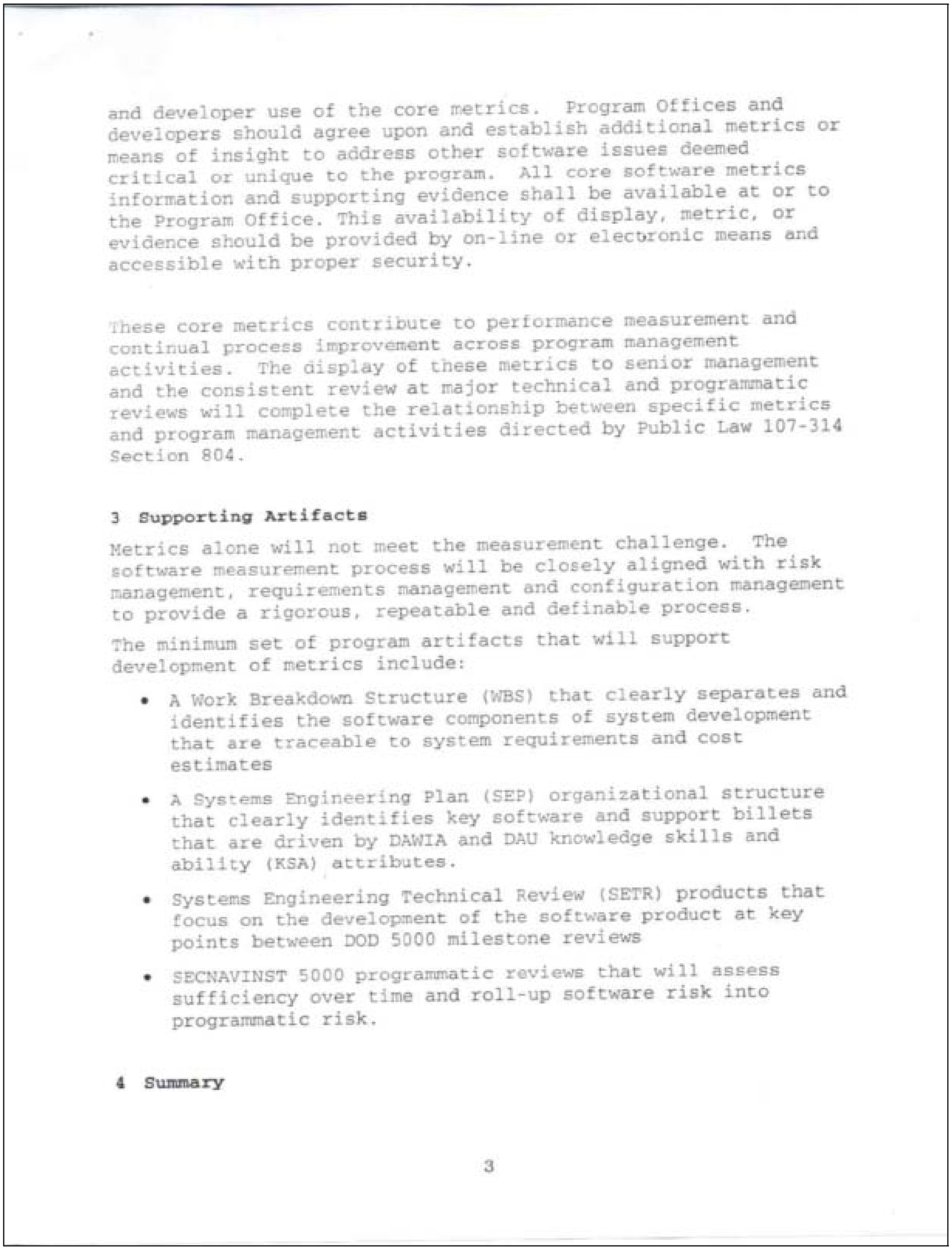
- Software Organization
- Cost/Schedule (WBS focus on software)
- Software Quality

These core metrics are keyed to and flex with the DoD 5000/IEEE/EIA 12207 life cycle of events and are to be used through the program's life cycle. Effective immediately, all Programs of Record with any software, regardless of ACAT category, shall define, develop, and implement the core software metrics for their program at contract award or the next major upgrade. The core metrics should be tailored and implemented consistent with both of the Program Office's and the developer's internal tools and processes but shall include the attributes listed in the following table.

Metric	**Highlights risk in:**	**Includes measures of:**
Software Size	Planning, Requirements Development, Requirements Management	e.g., Equivalent KSLOC or Number of SW requirements, or Function Point
Organization	Program Office and Developer Staff Planning, Resource Management	Key Billets filled/to be filled, Key Billet KSAs required /trained/unfilled, Key Billets filled by more than one person or part-time
Cost/ Schedule	Government independent Cost Estimate (ICE), Official Program Baseline, Delta in KPP/Requirements	Scope Creep versus KPP delta, Defect effected changes
Software Quality	KPP/Requirements driven, Technology Readiness Changes, Defects, Safety, Security	Defect Rate and Cost of rework (balance with cost metric), Process quality data, Training deficiencies, Open and Closed Risk issues, Subtraction of defects caused by KPP/Requirements delta

As noted earlier, a key attribute of the mandated metrics is that they are relational across the acquirer and developer organizations. This is based on Naval lessons-Learned showing that there is a cause and effect relationship between acquirer

2



and developer use of the core metrics. Program Offices and developers should agree upon and establish additional metrics or means of insight to address other software issues deemed critical or unique to the program. All core software metrics information and supporting evidence shall be available at or to the Program Office. This availability of display, metric, or evidence should be provided by on-line or electronic means and accessible with proper security.

These core metrics contribute to performance measurement and continual process improvement across program management activities. The display of these metrics to senior management and the consistent review at major technical and programmatic reviews will complete the relationship between specific metrics and program management activities directed by Public Law 107-314 Section 804.

3 Supporting Artifacts

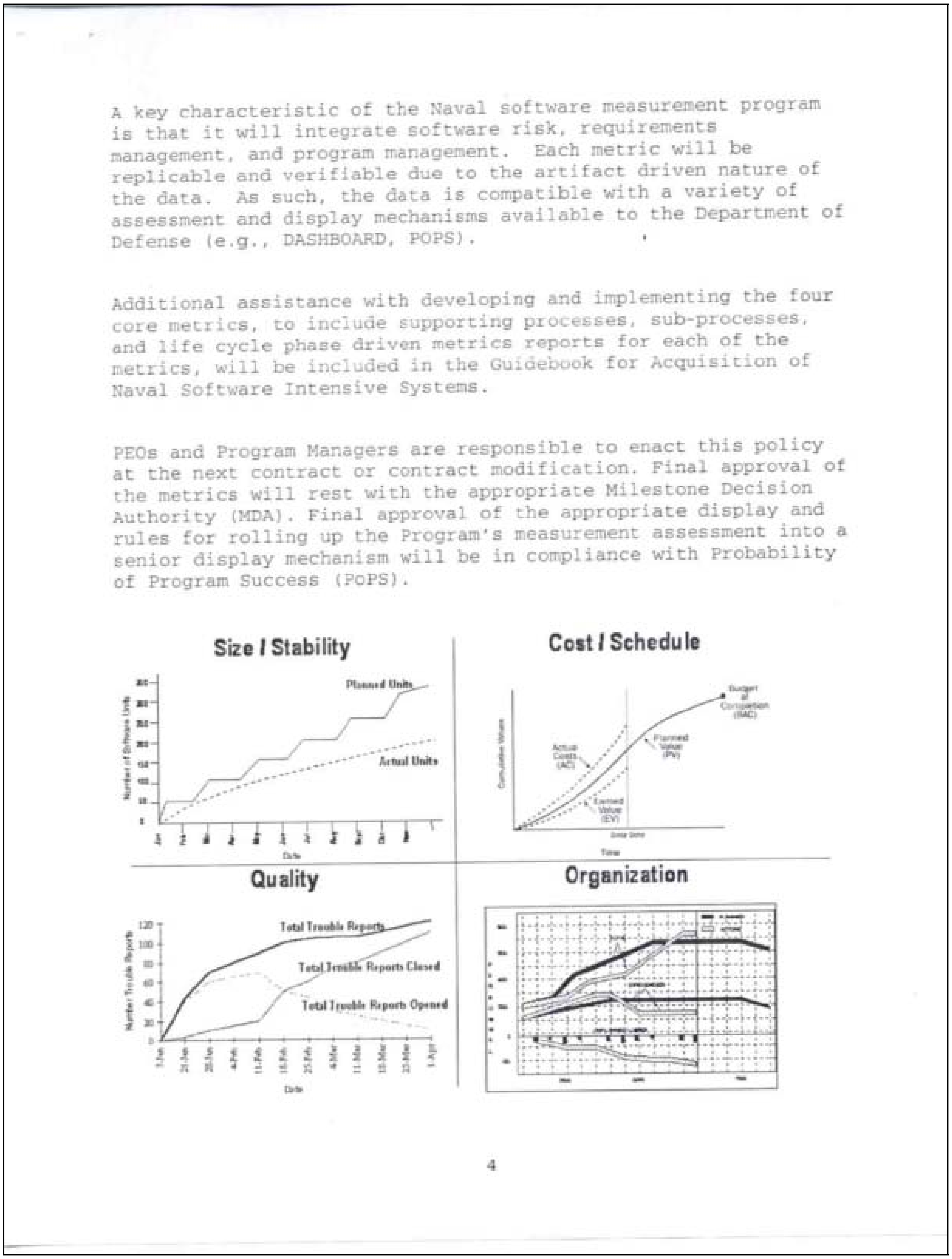
Metrics alone will not meet the measurement challenge. The software measurement process will be closely aligned with risk management, requirements management and configuration management to provide a rigorous, repeatable and definable process.

The minimum set of program artifacts that will support development of metrics include:

- A Work Breakdown Structure (WBS) that clearly separates and identifies the software components of system development that are traceable to system requirements and cost estimates
- A Systems Engineering Plan (SEP) organizational structure that clearly identifies key software and support billets that are driven by DAWIA and DAU knowledge skills and ability (KSA) attributes.
- Systems Engineering Technical Review (SETR) products that focus on the development of the software product at key points between DOD 5000 milestone reviews
- SECNAVINST 5000 programmatic reviews that will assess sufficiency over time and roll-up software risk into programmatic risk.

4 Summary

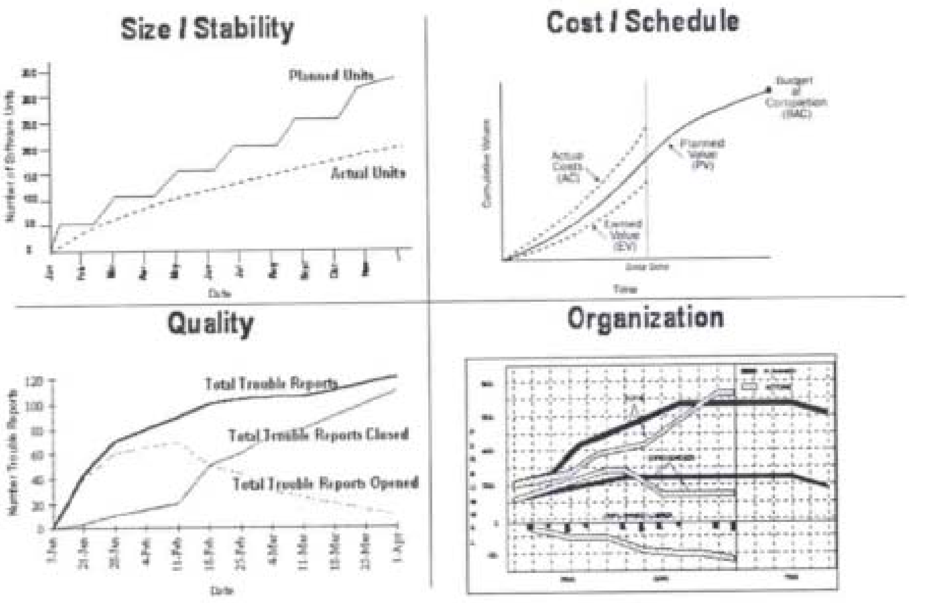
3



A key characteristic of the Naval software measurement program is that it will integrate software risk, requirements management, and program management. Each metric will be replicable and verifiable due to the artifact driven nature of the data. As such, the data is compatible with a variety of assessment and display mechanisms available to the Department of Defense (e.g., DASHBOARD, POPS).

Additional assistance with developing and implementing the four core metrics, to include supporting processes, sub-processes, and life cycle phase driven metrics reports for each of the metrics, will be included in the Guidebook for Acquisition of Naval Software Intensive Systems.

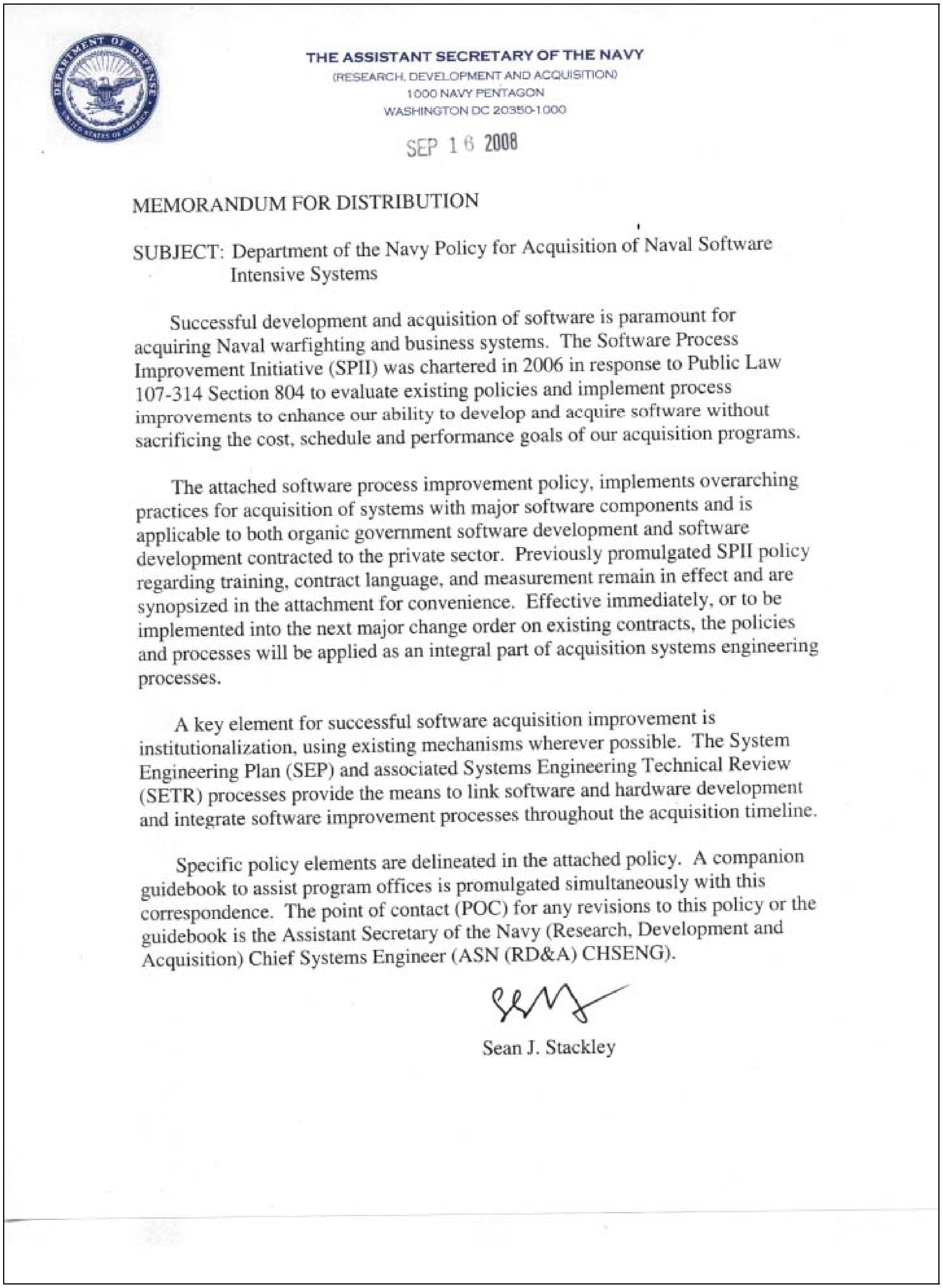
PEOs and Program Managers are responsible to enact this policy at the next contract or contract modification. Final approval of the metrics will rest with the appropriate Milestone Decision Authority (MDA). Final approval of the appropriate display and rules for rolling up the Program's measurement assessment into a senior display mechanism will be in compliance with Probability of Program Success (PoPS).



4

Appendix E

September 08 Department of the Navy Policy for Acquisition of Naval Software Intensive Systems Memo



THE ASSISTANT SECRETARY OF THE NAVY
(RESEARCH, DEVELOPMENT AND ACQUISITION)
1000 NAVY PENTAGON
WASHINGTON DC 20350-1000

SEP 16 2008

MEMORANDUM FOR DISTRIBUTION

SUBJECT: Department of the Navy Policy for Acquisition of Naval Software Intensive Systems

Successful development and acquisition of software is paramount for acquiring Naval warfighting and business systems. The Software Process Improvement Initiative (SPII) was chartered in 2006 in response to Public Law 107-314 Section 804 to evaluate existing policies and implement process improvements to enhance our ability to develop and acquire software without sacrificing the cost, schedule and performance goals of our acquisition programs.

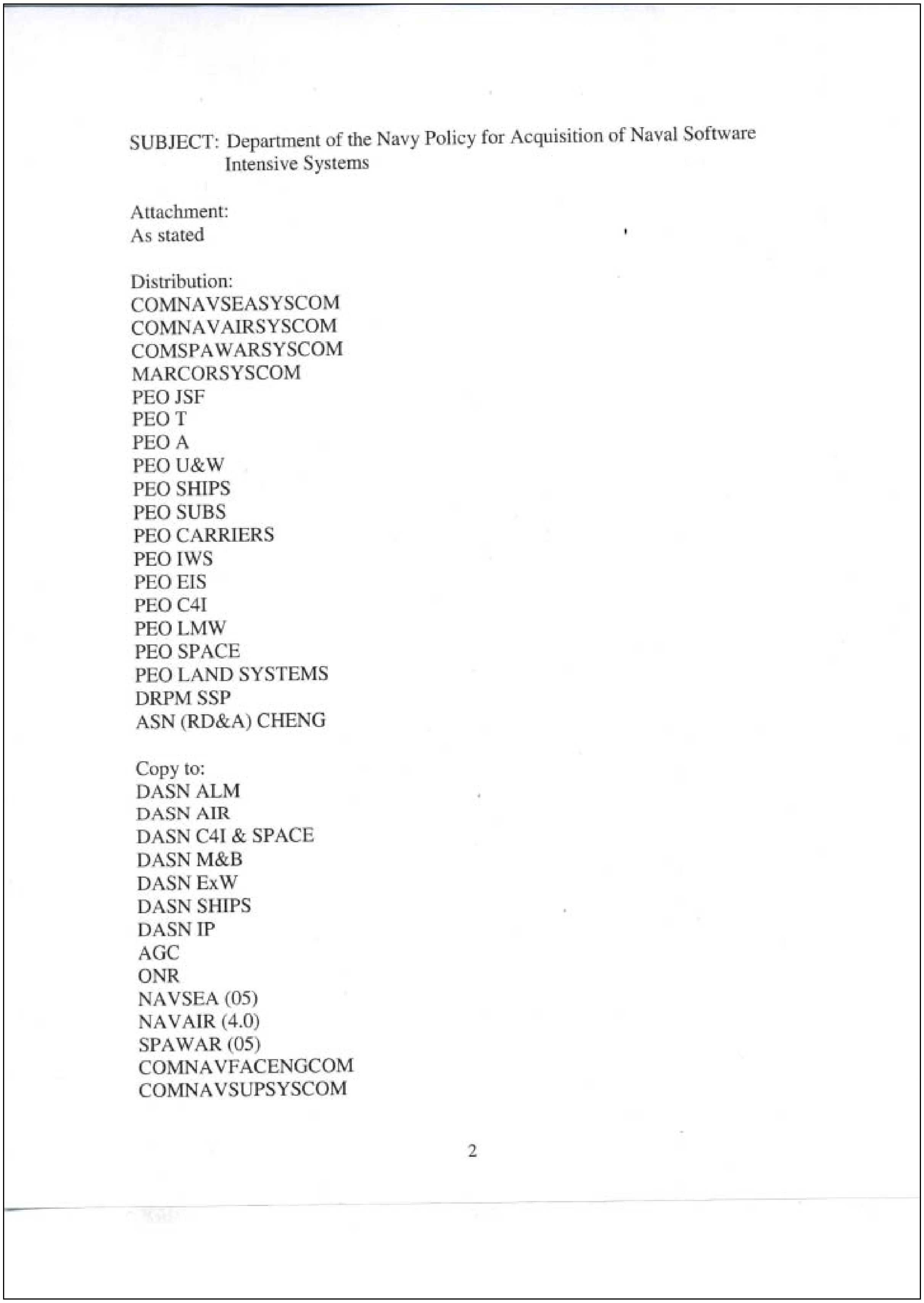
The attached software process improvement policy, implements overarching practices for acquisition of systems with major software components and is applicable to both organic government software development and software development contracted to the private sector. Previously promulgated SPII policy regarding training, contract language, and measurement remain in effect and are synopsized in the attachment for convenience. Effective immediately, or to be implemented into the next major change order on existing contracts, the policies and processes will be applied as an integral part of acquisition systems engineering processes.

A key element for successful software acquisition improvement is institutionalization, using existing mechanisms wherever possible. The System Engineering Plan (SEP) and associated Systems Engineering Technical Review (SETR) processes provide the means to link software and hardware development and integrate software improvement processes throughout the acquisition timeline.

Specific policy elements are delineated in the attached policy. A companion guidebook to assist program offices is promulgated simultaneously with this correspondence. The point of contact (POC) for any revisions to this policy or the guidebook is the Assistant Secretary of the Navy (Research, Development and Acquisition) Chief Systems Engineer (ASN (RD&A) CHSENG).

[illegible]

Sean J. Stackley



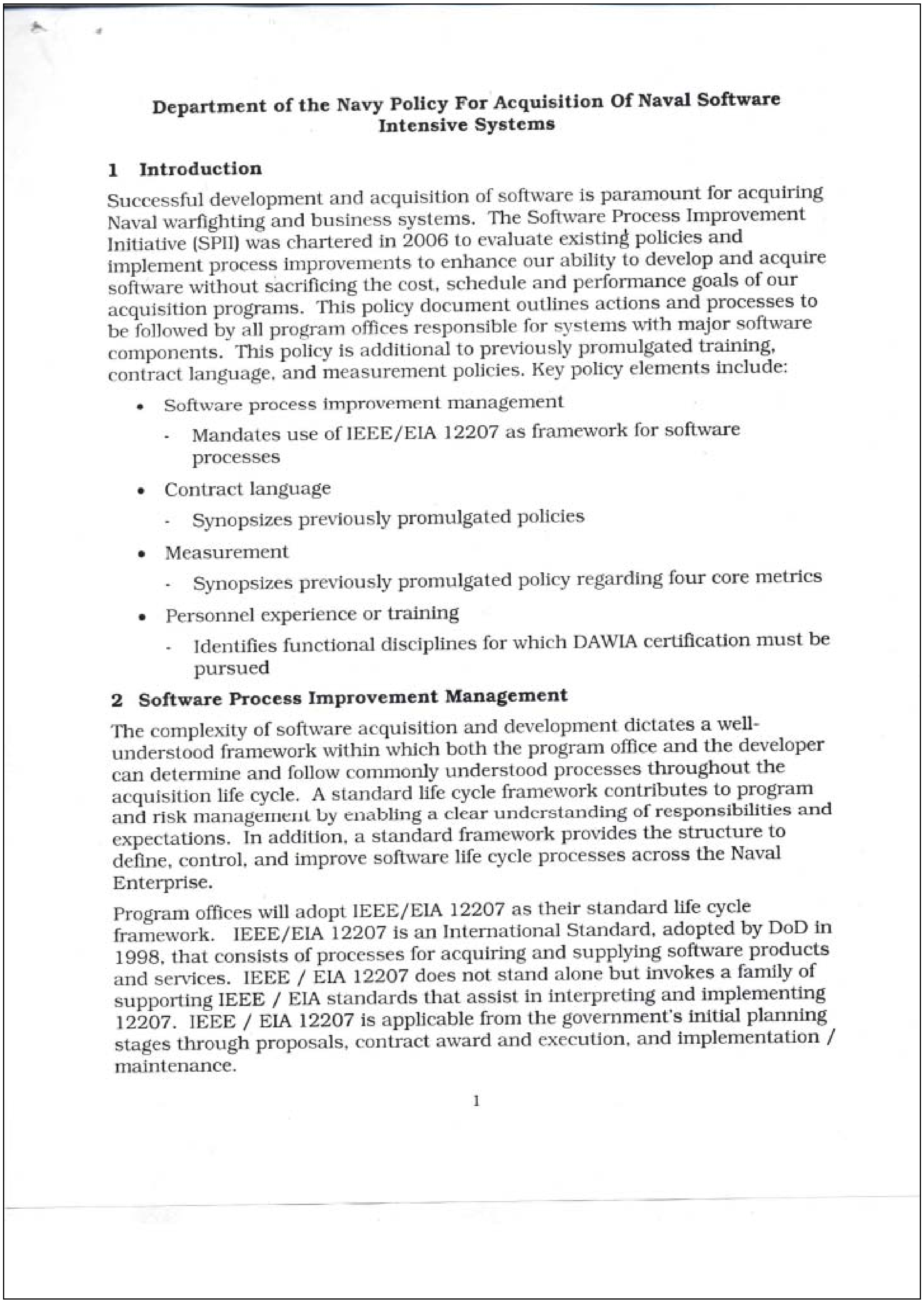
SUBJECT: Department of the Navy Policy for Acquisition of Naval Software Intensive Systems

Attachment:
As stated

Distribution:
COMNAVSEASYSCOM
COMNAVAIRSYSCOM
COMSPAWARSYSCOM
MARCORSYSCOM
PEO JSF
PEO T
PEO A
PEO U&W
PEO SHIPS
PEO SUBS
PEO CARRIERS
PEO IWS
PEO EIS
PEO C4I
PEO LMW
PEO SPACE
PEO LAND SYSTEMS
DRPM SSP
ASN (RD&A) CHENG

Copy to:
DASN ALM
DASN AIR
DASN C4I & SPACE
DASN M&B
DASN ExW
DASN SHIPS
DASN IP
AGC
ONR
NAVSEA (05)
NAVAIR (4.0)
SPAWAR (05)
COMNAVFACENGCOM
COMNAVSUPSYSCOM

2



Department of the Navy Policy For Acquisition Of Naval Software Intensive Systems

1 Introduction

Successful development and acquisition of software is paramount for acquiring Naval warfighting and business systems. The Software Process Improvement Initiative (SPII) was chartered in 2006 to evaluate existing policies and implement process improvements to enhance our ability to develop and acquire software without sacrificing the cost, schedule and performance goals of our acquisition programs. This policy document outlines actions and processes to be followed by all program offices responsible for systems with major software components. This policy is additional to previously promulgated training, contract language, and measurement policies. Key policy elements include:

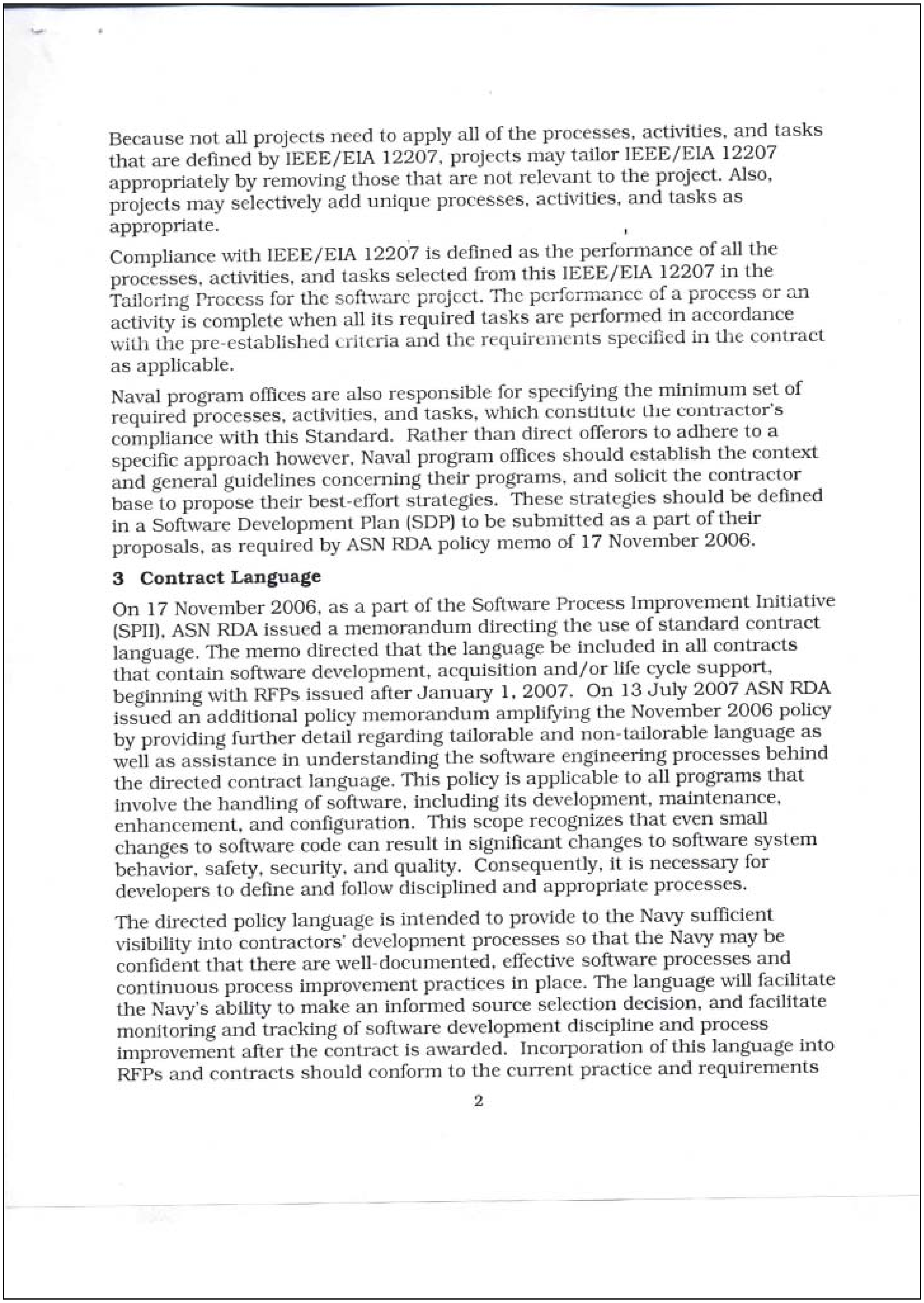
- Software process improvement management
  - Mandates use of IEEE/EIA 12207 as framework for software processes
- Contract language
  - Synopsizes previously promulgated policies
- Measurement
  - Synopsizes previously promulgated policy regarding four core metrics
- Personnel experience or training
  - Identifies functional disciplines for which DAWIA certification must be pursued

2 Software Process Improvement Management

The complexity of software acquisition and development dictates a well-understood framework within which both the program office and the developer can determine and follow commonly understood processes throughout the acquisition life cycle. A standard life cycle framework contributes to program and risk management by enabling a clear understanding of responsibilities and expectations. In addition, a standard framework provides the structure to define, control, and improve software life cycle processes across the Naval Enterprise.

Program offices will adopt IEEE/EIA 12207 as their standard life cycle framework. IEEE/EIA 12207 is an International Standard, adopted by DoD in 1998, that consists of processes for acquiring and supplying software products and services. IEEE / EIA 12207 does not stand alone but invokes a family of supporting IEEE / EIA standards that assist in interpreting and implementing 12207. IEEE / EIA 12207 is applicable from the government's initial planning stages through proposals, contract award and execution, and implementation / maintenance.

1



Because not all projects need to apply all of the processes, activities, and tasks that are defined by IEEE/EIA 12207, projects may tailor IEEE/EIA 12207 appropriately by removing those that are not relevant to the project. Also, projects may selectively add unique processes, activities, and tasks as appropriate.

Compliance with IEEE/EIA 12207 is defined as the performance of all the processes, activities, and tasks selected from this IEEE/EIA 12207 in the Tailoring Process for the software project. The performance of a process or an activity is complete when all its required tasks are performed in accordance with the pre-established criteria and the requirements specified in the contract as applicable.

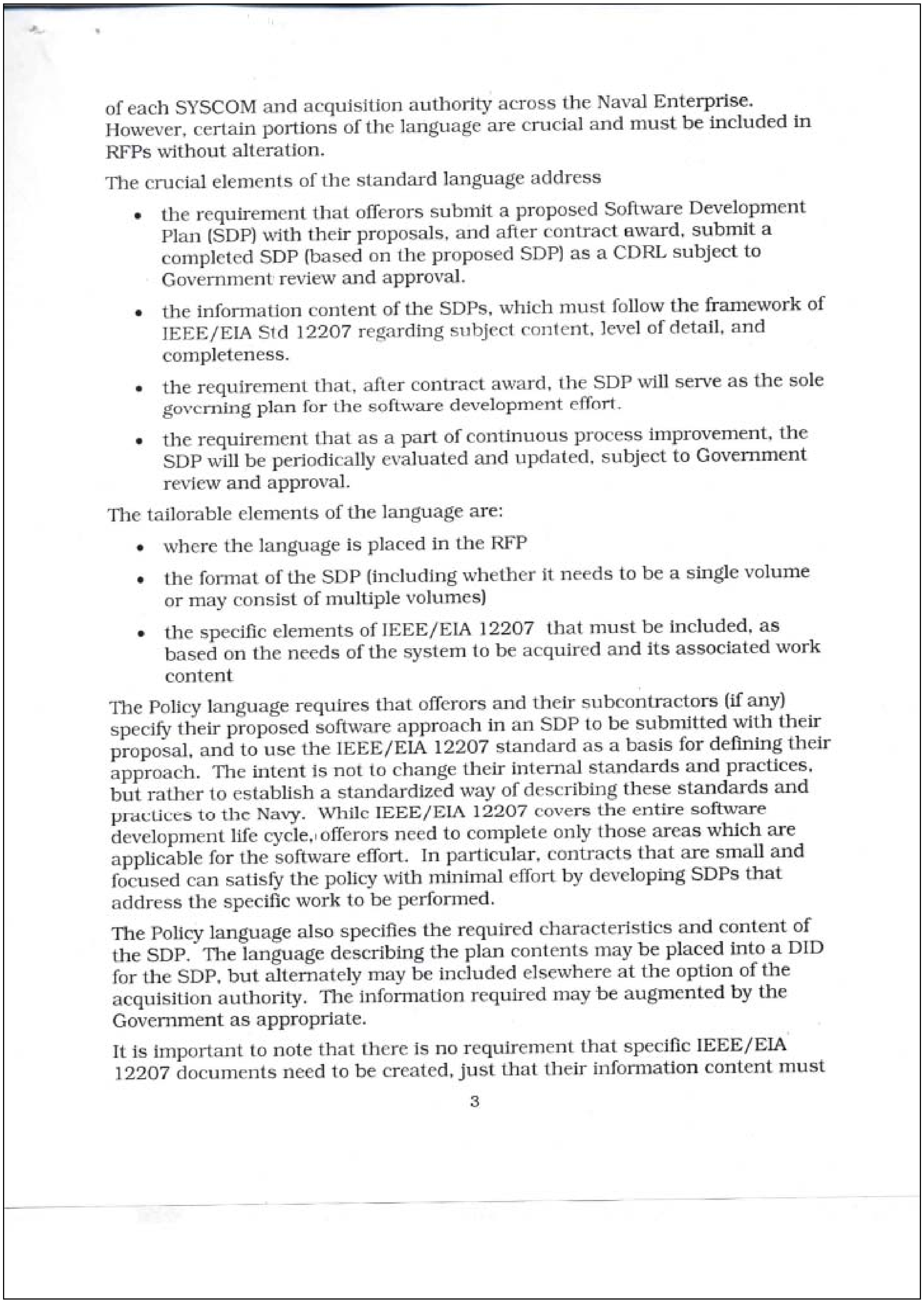
Naval program offices are also responsible for specifying the minimum set of required processes, activities, and tasks, which constitute the contractor's compliance with this Standard. Rather than direct offerors to adhere to a specific approach however, Naval program offices should establish the context and general guidelines concerning their programs, and solicit the contractor base to propose their best-effort strategies. These strategies should be defined in a Software Development Plan (SDP) to be submitted as a part of their proposals, as required by ASN RDA policy memo of 17 November 2006.

3 Contract Language

On 17 November 2006, as a part of the Software Process Improvement Initiative (SPII), ASN RDA issued a memorandum directing the use of standard contract language. The memo directed that the language be included in all contracts that contain software development, acquisition and/or life cycle support, beginning with RFPs issued after January 1, 2007. On 13 July 2007 ASN RDA issued an additional policy memorandum amplifying the November 2006 policy by providing further detail regarding tailorable and non-tailorable language as well as assistance in understanding the software engineering processes behind the directed contract language. This policy is applicable to all programs that involve the handling of software, including its development, maintenance, enhancement, and configuration. This scope recognizes that even small changes to software code can result in significant changes to software system behavior, safety, security, and quality. Consequently, it is necessary for developers to define and follow disciplined and appropriate processes.

The directed policy language is intended to provide to the Navy sufficient visibility into contractors' development processes so that the Navy may be confident that there are well-documented, effective software processes and continuous process improvement practices in place. The language will facilitate the Navy's ability to make an informed source selection decision, and facilitate monitoring and tracking of software development discipline and process improvement after the contract is awarded. Incorporation of this language into RFPs and contracts should conform to the current practice and requirements

2



of each SYSCOM and acquisition authority across the Naval Enterprise. However, certain portions of the language are crucial and must be included in RFPs without alteration.

The crucial elements of the standard language address

- the requirement that offerors submit a proposed Software Development Plan (SDP) with their proposals, and after contract award, submit a completed SDP (based on the proposed SDP) as a CDRL subject to Government review and approval.
- the information content of the SDPs, which must follow the framework of IEEE/EIA Std 12207 regarding subject content, level of detail, and completeness.
- the requirement that, after contract award, the SDP will serve as the sole governing plan for the software development effort.
- the requirement that as a part of continuous process improvement, the SDP will be periodically evaluated and updated, subject to Government review and approval.

The tailorable elements of the language are:

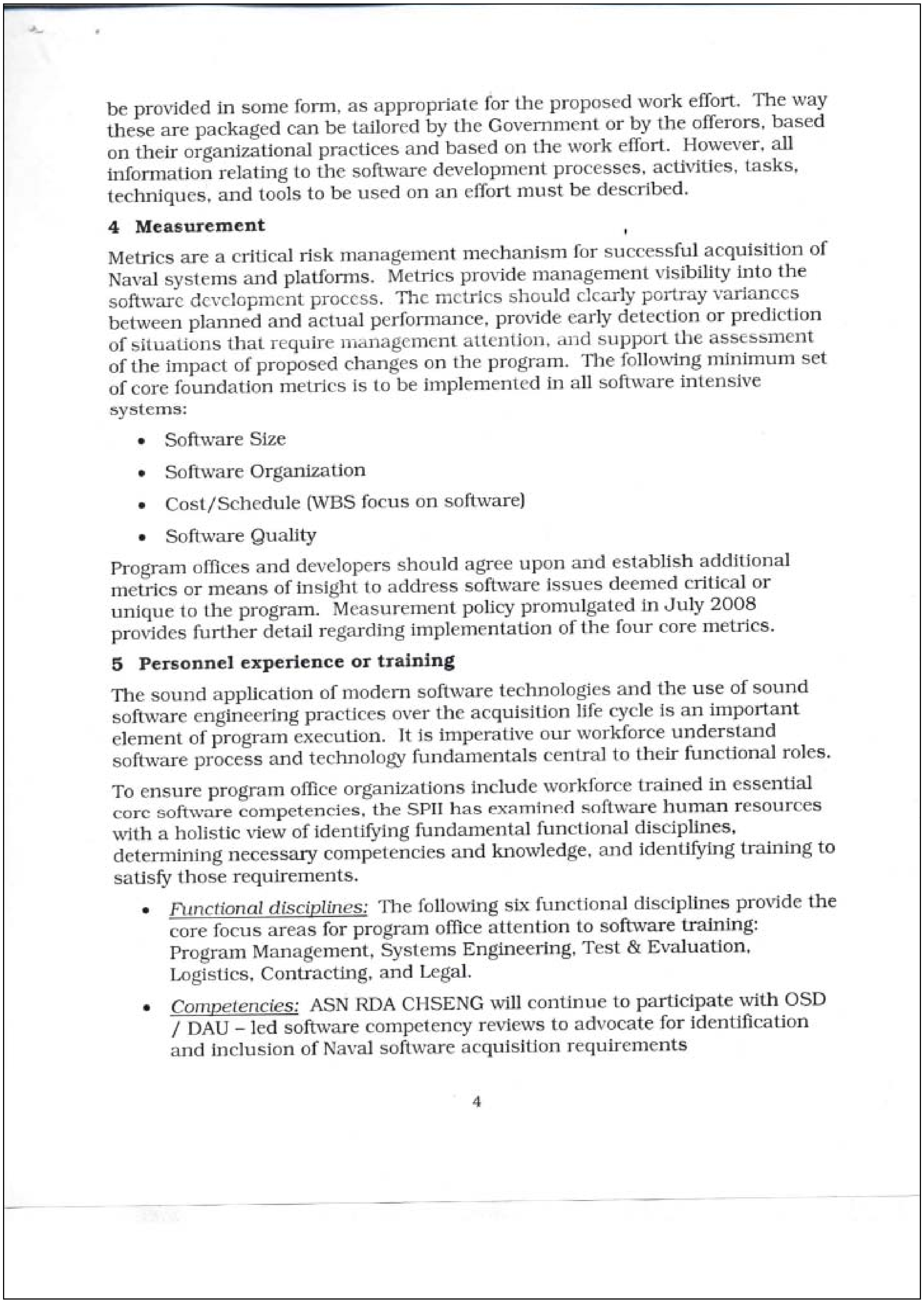
- where the language is placed in the RFP
- the format of the SDP (including whether it needs to be a single volume or may consist of multiple volumes)
- the specific elements of IEEE/EIA 12207 that must be included, as based on the needs of the system to be acquired and its associated work content

The Policy language requires that offerors and their subcontractors (if any) specify their proposed software approach in an SDP to be submitted with their proposal, and to use the IEEE/EIA 12207 standard as a basis for defining their approach. The intent is not to change their internal standards and practices, but rather to establish a standardized way of describing these standards and practices to the Navy. While IEEE/EIA 12207 covers the entire software development life cycle, offerors need to complete only those areas which are applicable for the software effort. In particular, contracts that are small and focused can satisfy the policy with minimal effort by developing SDPs that address the specific work to be performed.

The Policy language also specifies the required characteristics and content of the SDP. The language describing the plan contents may be placed into a DID for the SDP, but alternately may be included elsewhere at the option of the acquisition authority. The information required may be augmented by the Government as appropriate.

It is important to note that there is no requirement that specific IEEE/EIA 12207 documents need to be created, just that their information content must

3



be provided in some form, as appropriate for the proposed work effort. The way these are packaged can be tailored by the Government or by the offerors, based on their organizational practices and based on the work effort. However, all information relating to the software development processes, activities, tasks, techniques, and tools to be used on an effort must be described.

4 Measurement

Metrics are a critical risk management mechanism for successful acquisition of Naval systems and platforms. Metrics provide management visibility into the software development process. The metrics should clearly portray variances between planned and actual performance, provide early detection or prediction of situations that require management attention, and support the assessment of the impact of proposed changes on the program. The following minimum set of core foundation metrics is to be implemented in all software intensive systems:

- Software Size
- Software Organization
- Cost/Schedule (WBS focus on software)
- Software Quality

Program offices and developers should agree upon and establish additional metrics or means of insight to address software issues deemed critical or unique to the program. Measurement policy promulgated in July 2008 provides further detail regarding implementation of the four core metrics.

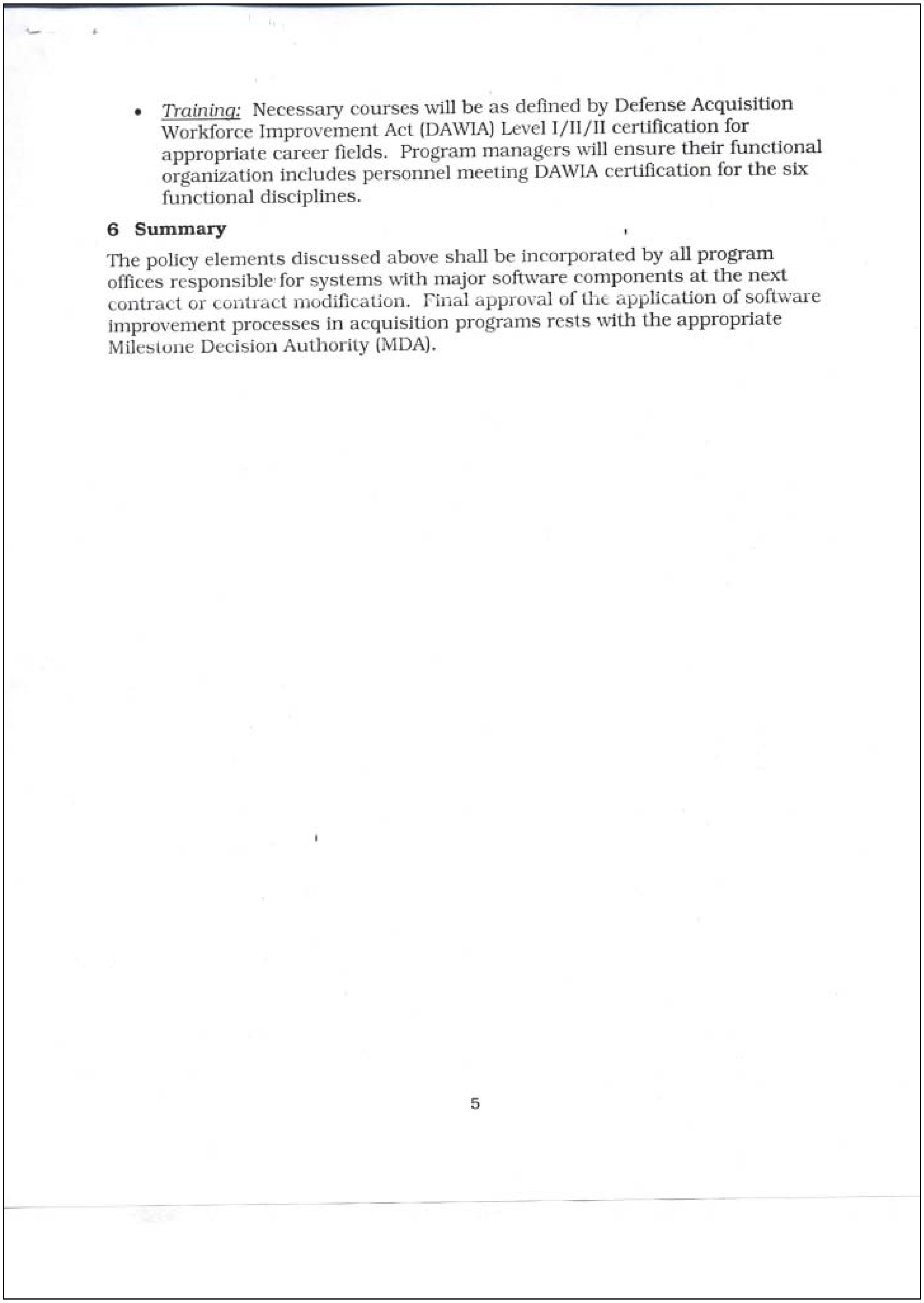
5 Personnel experience or training

The sound application of modern software technologies and the use of sound software engineering practices over the acquisition life cycle is an important element of program execution. It is imperative our workforce understand software process and technology fundamentals central to their functional roles.

To ensure program office organizations include workforce trained in essential core software competencies, the SPII has examined software human resources with a holistic view of identifying fundamental functional disciplines, determining necessary competencies and knowledge, and identifying training to satisfy those requirements.

- *Functional disciplines:* The following six functional disciplines provide the core focus areas for program office attention to software training: Program Management, Systems Engineering, Test & Evaluation, Logistics, Contracting, and Legal.
- *Competencies:* ASN RDA CHSENG will continue to participate with OSD / DAU – led software competency reviews to advocate for identification and inclusion of Naval software acquisition requirements

4



- *Training:* Necessary courses will be as defined by Defense Acquisition Workforce Improvement Act (DAWIA) Level I/II/II certification for appropriate career fields. Program managers will ensure their functional organization includes personnel meeting DAWIA certification for the six functional disciplines.

6 Summary

The policy elements discussed above shall be incorporated by all program offices responsible for systems with major software components at the next contract or contract modification. Final approval of the application of software improvement processes in acquisition programs rests with the appropriate Milestone Decision Authority (MDA).

5

Appendix F
Earned Value Management for Software

Earned Value Management (EVM) is a management technique that integrates cost, schedule, and technical performance measurement and goals. Instead of simply comparing cost incurred to a spend plan EVM incorporates actual work accomplished. As such, EVM is a performance measuring tool and technique and not a cost accounting tool. The Program Manager (PM) should integrate EVM with risk management and other program metrics for effective project management.

EVM = Integrated Management
Work Breakdown Structure - The Key to Integrated Management

The means by which EVM can be used to its full potential for software development is through the creation of an accurate and complete project baseline. That baseline can only be made complete through the development of a Program Work Breakdown Structure (PWBS) that accounts for the software products to be integrated and developed down to the lowest computer software configuration items (CSCI) consistent with the object oriented architecture it is being developed under.

As discussed in the book Project Management, The Common Sense Approach[1], earned value can only be successful if the Program Manager understands the concepts and recognizes the need for a vertical hierarchical relationship between all the units of work to be performed on a software project. This hierarchical relationship is established via the PWBS. Work is performed at the lowest levels of the PWBS; therefore, these critical work subdivision elements have particular significance when it comes to achieving the most beneficial results from using earned value.

The PWBS is a key document that is developed by the Program Manager and systems engineering staff very early in the program planning phase. The PWBS forms the basis for the Statement of Work (SOW), systems engineering plans, Information Management System (IMS), Earned Value Management System (EVMS), and other status reporting.[2]

A PWBS, developed for effective management control, is commonly used to segregate the work scope requirements of the program into definable product elements and related services and data. The PWBS is a direct representation of the work scope defined in the program SOW and breaks that work scope into appropriate elements for cost accounting and work authorization. It is a multi-level hierarchical breakdown that shows how program costs are summarized from the lower elements to the total program level. The extent of detail (breakout and levels) in the PWBS will be determined by program management needs, organizational policies, and contractual agreements.

[1] Lee R. Lambert PMP and Erin Lambert, MBA; 2000 LCG Publishers
[2] See MIL-HDBK-881(current version), Work Breakdown Structure Handbook for further guidance.

As a program progresses from one phase to another, it is a normal process to reassess the PWBS. As a case in point, the product breakdown during a development phase may be different from the product breakdown, the assembly sequence, used in the production phase. If program requirements change, the PWBS will evolve with the program.

A program may elect to prepare a Work Breakdown Structure (WBS) dictionary. The dictionary defines the work scope represented in each element of the WBS. This can be done by summary work scope descriptions or by references to the applicable sections of the SOW. The WBS dictionary does not replace the SOW, but can provide a logical cross reference between it and the WBS. Direct costs are clearly segregated by WBS element without further allocation.

The objectives of EVM are borne in common sense and straightforward:

- Plan all work scope for the project to completion;
- Measure performance based on an objective set of technical criteria;
- Analyze schedule status and projections using a time phased critical path method (CPM) network;
- Analyze the expenditure of funds in light of the work accomplished (i.e. performance);
- Isolate problems by quantifying technical problems within the context of cost and schedule parameters. EVM is not aimed at replacing or changing the process for technical problem detection;
- Forecast completion date and final cost;
- Maintain disciplined control of the performance measurement baseline;
- Integrate project work scope, schedule, and cost objectives into a baseline plan against which accomplishments may be measured;
- Objectively assess accomplishments at the work performance level;
- Analyze significant variances from the plan and forecast impacts; and
- Summarize data to higher levels for management decision making and for implementing corrective action when necessary.

There are eight parts to EVM: Organization, Scheduling, Budgeting, Work Authorization, Data Accumulation and Reporting, Variance Analysis, Estimate At Completion, and Baseline Maintenance and Control. Out of these, schedule is the key parameter of performance that will have the greatest impact on cost and technical performance.

EVM for the Program Manager is important for these reasons:

- Early and accurate identification of trends and problems;
- Accurate picture of contract status - cost, schedule, and technical;
- Basis for course correction;
- Supports mutual goals of contractor and customer; and
- Bring project in on schedule and cost.

The fundamental process that serves as the foundation for implementing and using EVM is simple in theory but requires a disciplined and rigorous approach with the appropriate oversight and support of the Program Manager.

- The contractor establishes a management control system in which it may be required to show that the system meets 32 criteria as defined in ANSI/EIA 748;
- Establish an integrated baseline plan in which work is defined, scheduled, and resources are allocated;
- Work and resources must be driven down to lowest level for execution in the WBS;
- A work authorization system is set up that controls changes to the baseline;
- Budgets are “earned” as work is completed = EARNED VALUE;
- Status is provided against baseline in which schedule and cost variances are isolated;
- Early warning for program risk is identified and affordable, schedule compliant corrective plans can be developed and executed; and
- Early insight is provided into final estimated cost.

The PM has the responsibility to follow current DoD policy in applying EVM and Integrated Master Schedule (IMS) requirements to the proposed contract. EVM system requirements are defined in the contract SOW and in the applicable solicitation/contract clauses.

EVM reporting requirements are defined in the Contract Data Requirements List (CDRL). The PM should tailor reporting requirements based on a realistic assessment of management information needs for effective program control. The PM has the flexibility to tailor requirements that optimize contract visibility while minimizing intrusion into the contractor’s operations. Government reporting requirements are to be specified separately in the contract through the use of a CDRL (DD Form 1423-1, or equivalent). These requirements should be contained in both the solicitation document and in the contract. The PM is also engaged in the evaluation of the proposed EVMS during source selection. The Earned Value Management Implementation Guide (DCMA October 2006) provides more detailed guidance for EVM implementation.

References

Department of Defense. Department Of Defense Handbook - Work Breakdown Structure. July 2005.

Lee R. Lambert PMP and Erin Lambert, MBA; 2000 LCG Publishers

Department of Defense. Defense Contract Management Agency. The Earned Value Management Implementation Guide. October 2006.

G

Appendix
Competencies

See Chapter 3 for a discussion of competencies. Additional information may also be found in the Software Process Improvement Initiative Human Resource Report dated 06 November 2007, which can be accessed at http://acquisition.navy.mil/organizations/dasns/rda_cheng.

Table of Contents

G.1 Program Management

G.1.1 Level I Program Management

Software Acquisition Management Regulatory/Technical Framework Application & Analysis
Give examples of best system strategies for SW intensive systems
Explain the effect of current system Strategies on SW Acquisition Mgmt
Summarize the strengths and weaknesses of current strategies
Explain the impact of acquisition strategy on SW project planning and SW Engineering methods
Explain the impact of Acquisition Reform
Describe the functions of a DoD acquisition strategy and the elements included in a software acquisition.
Describe components of a [Software Acquisition] strategic plan.
Identify the contents of a [Software Acquisition] plan and explain where the information can be obtained
Identify higher guidance and [Software Acquisition] goals for strategic planning
Knowledge of laws, policies, regulations, directives, and guidance impacting DoD [Software Acquisition], including DoD and service specific [Software Acquisition]
Identify the major DoD acquisition policies that apply specifically to software acquisition management and software engineering
Describe the integrated architecture framework; the relationships and roles of the DoD operational, systems, and technical architectures; and the impact of these architectures on the [Software] acquisition process
Recognize software and system architectures
Describe the fundamentals of the DoD Architecture Framework (DoDAF) and address the development, use, governance, and maintenance of architecture data
Describe the program manager's role is managing architecture products and documentation
Identify and describe basic principles of technical standards as they relate to system development and interoperability
Identify interoperability terminology, the importance of planning for interoperability in a [Software] acquisition strategy, and the conceptual components of a [Software] system architecture; and demonstrate the relationship to interoperability
Describe the software Architecture/reuse relationship
Describe risk mitigation through reuse
Identify reuse guidance
Describe domain specific reuse paradigm
Identify existing Reuse repositories
Describe contracting mechanisms for reuse
Describe the impact of Open Systems on software reuse
State COTS/Reuse Issues
Describe portability, through platform independence
Software Risk Management Application & Analysis
Explain software Risk Analysis
Give examples of software Risk management issues (planning, etc.)
Explain varying risk profile through life cycle
Give examples of organizational risk mitigation entities (SEMP, RMWG, TIWG, CRWG, CRLCMP, IPT's, etc.)
Give examples of risk Management guidance
Summarize the concept of a Domain Competent Work Force
List and explain the steps of a risk management process for a [Software] acquisition
Explain the purpose and at least one method for analyzing alternatives
Identify software engineering risks
Identify software risk management methodologies
Describe techniques for attaining safe, secure, and reliable systems.
Explain how to incorporate risk management strategies into software project planning and management
Compare and contrast the commonly accepted standards, tools, and methods used in risk management
Explain how to monitor the status of software engineering risks and common SW risk management issues
Define Software Security
Describe Security Risk Management
Identify Software security guidance (regulations, standards, "orange book")
Describe System Certification
List contemporary security developments
Describe the discipline of Software Safety
Government and Industry Software Acquisition Management Roles
Give examples of standards for Configuration Mgt
Summarize Configuration Mgt Planning
Explain the use of Configuration Mgt throughout SW life-cycle (SMRB, etc.)
Explain the synchronization of HW and SW baselines
Explain Configuration Management CASE tools
Explain the management of Configuration Risks
Explain the purpose for configuration management (CM) and at least four CM functions
Explain the purpose for tracing and managing the configuration of requirements
Summarize Staffing best practices.
Summarize Organizational best practices
Summarize best practices for Matrix Support Groups

Summarize Resource Management best practices
Summarize best practices for Project Control
Summarize best practices for Project Tracking
Explain End User Involvement
Summarize best practices for IPT's and working groups
Summarize best practices for Intergroup Coordination
Give examples of Corrective Actions
Give examples of Lessons Learned
Summarize best practices to deal with Management Issues
Explain the use of teams in managing [Software] acquisition programs and the concepts of team building
Describe [Software] systems and methods for facilitating all aspects of program management
Define organizational and individual roles and responsibilities involved in DoD software acquisition
Reference sources for software acquisition and information technology management policies, standards, and best practices
Reference sources for software acquisition and information technology management policies, standards, and best practices
Describe the impact, roles and opportunities of the DoD Science & Technology Process (e.g. Advanced concept Technology Demonstrations (ACTD) and Advanced Technology Demonstration (ATD)
Unique Software Procurement Requirements Application & Analysis
Explain the development of SW Development Plan (SDP)
Explain the use of SDP in proposal evaluation
Explain the Work Break-down Structure (WBS) for SW
Give examples of Laws/regulation related to SOW and RFP
Give examples of Quality Issues
Explain Contract types and their strengths and weaknesses (for all types of systems)
Give examples of Deliverables (issues and tradeoffs)
Explain the SW portion of Proposal Evaluation
Summarize data and intellectual property rights
Give examples of Commercial & DoD best practices such as Joint Technical Architecture (JTA), Open Systems, COTS, Reuse
Distinguish Model SOWs
Reference sources of DoD policy and guidance on the procurement of intellectual property, including software
Identify the role and elements of electronic commerce in [Software Acquisitions]
Define commercial items and non-developmental items, and explain the commercial items acquisition process
Describe solicitation methods, format, and content and explain the roles of the communications-computer acquisition professional in the solicitation process
Identify the contents of a statement of work/statement of objectives and list sources that would help in their development
Explain the role of evaluation criteria in a [Software Acquisition]
Describe a [Software Acquisition] source selection process
Define contract administration and identify the contract administration responsibilities of various Government officials and organizations for a [Software Acquisition]
Identify the policies, procedures, and management techniques used to establish contract support capabilities for software-intensive systems
Describe appropriate activities to ensure data rights and intellectual property policies are implemented successfully
Describe Open System Migration issues
Identify applicability of Naval Open System architecture policy and guidelines
Identify applicability of Open System architecture policy and guidelines
Identify Open System guidance (Application Portability Profile, regulations, standards)
Describe Open System adaptation effect on acquisition
Identify Commercial Off the Shelf/Non-Developmental item (COTS/NDI) issues
Software Metrics Application & Analysis
Describe the Roles of assessments/evaluations
Identify methods available to assess maturity
Describe the strengths and weaknesses of current methods
Describe the applications of assessments and evaluations
Describe the role of evaluations/assessments in contracting
Identify best practices for the frequency of evaluations/assessments
Describe the responsibilities for evaluations/assessments
Explain the impetus behind the process improvement focus
Describe the structure of the Staged and Continuous representations of CMMI
Describe the general guidelines for selecting either the Staged or Continuous representation
Identify the content of the CMMI Process Areas
Explain where to find more detailed information on applying CMMI
Identify appropriate metrics for visibility into development process, software product, system progress
Describe metrics Collection methodologies
Identify best practices for Metrics Interpretation
Describe bench marking practices
Describe data management technologies and methods for DoD [Software Acquisition] programs
Explain the types and use of measures/metrics in a [Software] acquisition
Analyze Software Technical Life Cycle & Relate It To System Acquisition Process
Identify current approaches (e.g., Functional, Object-Oriented)
Describe strengths and weaknesses of design approaches
Describe the effect of design approach on SW engineering, project planning, CASE selection and use, design reviews & docs
Identify Software Design Guidance (laws, regs, Stds)

Define Technical fundamentals
Identify Development Paradigms (Waterfall, Spiral, Prototyping, Incremental, IE) definitions
List criteria for Paradigm selection
Describe the risks and benefits of each development
Describe paradigm selection resource/management issues
Recognize software measures, development models, paradigms, and strategies appropriate for use in software-intensive acquisitions
Define key [Software] systems and software engineering terms, concepts, and methodologies
Describe how the eight technical processes can be applied in top-down development and bottom up product realization
Describe how the eight technical management processes are used to control and assess systems engineering (SE) activities
Describe the role of a systems model, the work breakdown structure (WBS), standards, top-down design, bottom-up product realization, and the Systems Engineering Plan (SEP)
Describe the role SE management plays in acquisition programs
Explain the relationship between software engineering and systems engineering
Describe the SE process and its application throughout a system's life cycle
Explain the importance of rigorously applying SE principles and practices
Explain the relationship of the software development life cycle to the overall system acquisition process
Recognize the complexity of the software development process to the acquisition life cycle
Identify Cost Factors
List key Software support transition issues
Describe Organic/Outsourcing Post Deployment Software Support considerations
Describe considerations for Software Engineering Environment acquisition & use
Identify DoD Life Cycle Guidance (Directives, Instructions, Standards, etc.)
Describe Support Organization Involvement
Define Continuous process improvement
Describe End User Involvement
Describe Corrective Actions Management
Define Contract Baseline
Describe the relationship with contractor(s)
Identify DoD [Software Acquisition] Management regulations, goals, and procedures
Describe [Software Acquisition] life cycle budget execution goals and objectives
Identify the concepts of change management
Describe examples of the technical, contractual, and personal issues involved in deploying a [Software] system
Identify [Software Acquisition] Life Cycle Management documentation requirements
Recognize the importance of supportability to achieving system readiness requirements and reducing life-cycle costs
Discuss supportability requirements that must be met prior to acquisition or modification of a new/existing [software-intensive] system
Explain the support activities and requirements associated with fielding/deployment and post-production support of software-intensive systems
Identify key software support transition issues
Define Joint Technical Architecture (JTA) [or its equivalent, e.g., DoDAF]
Describe considerations in domain & product line engineering
Identify state of the art software technology topics
Explain at least two [Software] technologies relative to DoD systems development
Describe how modeling and simulation (M&S) can benefit over the entire life cycle of a software-intensive acquisition project
Recognize the integral nature of systems software in modern defense systems and the policies applicable to software intensive systems
Identify and describe modeling and simulation approaches
Software Testing "Best Practices" Application
Define IV&V, and describe benefits and disadvantages
Identify IV&V levels
Identify IV&V guidance
Describe the IV&V relationship to risk management and testing
Describe the IV&V effect on development schedule
Describe the discipline of Software Verification, Validation, and Accreditation (V,V&A)
Give examples of software quality factors
Give examples of software quality guidance
Explain quality improvement methods (Formal Inspection, Walk throughs, Clean room, Peer reviews)
Explain the benefits and risks associated with software quality methods
Give examples of best practices for Software Project Management visibility into software quality (metrics and inspections)
Give examples of Software Product Assessment Techniques
Summarize Software Quality Assurance Planning and Techniques
Identify requirements, methods, and techniques for quality assurance during the system life cycle
Describe the discipline of software Quality
Software Acquisition Management Planning & Status Documentation Analysis
Define Software Requirement management
Identify Requirement Management guidance
Describe Requirement Management responsibilities
Describe User involvement
Identify Requirement Planning issues
Identify types of requirements (derived, explicit, decomposed)
Define software requirements, and describe the benefits and risks of prototyping
Describe Requirements Management issues (baselines, traceability, tool support, life cycle requirements variance)

Describe Requirements/COTS issues
Identify Critical measures of effectiveness for operational issues and criteria
Describe the requirements development process
Define software acquisition and information technology management-specific terms and concepts.
Describe the Government management of reviews and audit process
Identify high interest Software issues and their indicators
Describe Critical Software life cycle reviews
List key Software review questions and data
Identify Entrance & Exit Criteria
Describe the Software review relationship to system reviews
Software Economic Factors Analysis
Describe the strengths and weaknesses of methods and models used for SW cost & schedule estimation
Describe SW cost & schedule reporting
Describe the validation/assessment of fidelity of cost and schedule estimates for SW intensive projects
Identify Life Cycle Costs (incl PDSS)
Identify elements of Planning, Programming, and Budgeting System (PPBS)
Explain the requirements and factors involved in assessing program costs and returns
Identify the purpose and process of Earned Value Management (EVM) and Recognize the value and benefits of EVM in the software acquisition process
Describe the requirements for conducting an economic analysis for a [Software] system in the DoD Life Cycle Management process. Identify examples of the factors included in an economic analysis for a [Software] system
Explain the role, process, and elements of market research in a [Software Acquisition]
Define Business Process Reengineering (BPR)
Identify best practices for adapting maturing technologies
Define the Development Information System/Enterprise
Identify FPI Guidance, Process, Tools
Describe Model Relationship
Describe the impetus behind the process improvement focus

G.1.2 Level II Program Management

Software Acquisition Management Regulatory/Technical Framework Application & Analysis
Give examples of best system strategies for SW intensive systems
Explain the effect of current system Strategies on SW Acquisition Mgmt
Summarize the strengths and weaknesses of current strategies
Explain the impact of acquisition strategy on SW project planning and SW Engineering methods
Explain the impact of Acquisition Reform
Using a software-intensive system, identify acquirer key planning roles and activities. Describe "best practices" for software-intensive systems acquisitions and development that acquirers may use.
Given descriptions of acquisition strategies, issues, risks, software-intensive system, select an appropriate acquisition strategy over the life cycle of the system; select an appropriate software development paradigm within that strategy; explain how modeling, simulation, and prototyping help with this process.
Given materials on applicable Federal laws and DoD acquisition policies, determine legal and policy requirements that apply to a given software-intensive system
Include COTS-based systems where appropriate when formulating software acquisition strategies.
For current laws and policies, identify key software acquisition management activities that should be emphasized during the acquisition of a DoD software intensive system.
Summarize Software Architecture Fundamentals
Explain the relationship of SW to System Architecture
Explain the relationship of Architecture to SW Design
Explain the Impact of architecture on interoperability and reuse
Distinguish between C3I, MCCR, and AIS systems
Summarize best practices for evaluating and acquiring target environments
Give examples of product line & domain engineering considerations (tradeoffs & analysis)
Explain the differences among documentation frameworks (e.g., the Federal Enterprise Architecture Framework (FEAF), the Department of Defense Architecture Framework (DODAF), or the Zachman Framework) and architecture reference models such as those provided in the Federal Enterprise Architecture (FEA).
Describe basic architecture documentation (i.e., work product) methodologies at each level of a commonly used framework (e.g., Zachman, FEAF or DODAF).
Identify the purpose and timing of the SE process outputs over the life cycle, such as program-unique specifications, IT architectures, technical data packages, and other system-specific information.
Give examples of Interoperability and Data Administration Issues
Summarize Interoperability and data administration guidance (Laws, regulation, and standards)
Explain the relationship of Software/System Architecture and interoperability
Explain the Software Architecture/reuse relationship
Explain risk mitigation through reuse
Summarize reuse guidance
Explain Domain specific reuse paradigm
Give examples of existing Reuse repositories
Explain contracting mechanisms for reuse
Explain the impact of Open Systems on software reuse
Give examples of COTS/Reuse Issues
Explain portability, through platform independence

Software Risk Management Application & Analysis
Demonstrate Software Risk Analysis
Solve Software Risk management issues (planning, etc.)
Demonstrate the benefit of varying the risk profile through life cycle
Select Organizational risk mitigation entities (SEMP, RMWG, TIWG, CRWG, CRLCMP, IPT's, etc.) appropriate to a given case situation
Use Risk Management guidance
Summarize the concept of a Domain Competent Work Force
Given programmatic documentation for a given software-intensive system, justify appropriate risk handling methods for that system.
Using a software acquisition system, apply the risk management process as a basis for making sound software acquisition program decisions.
Identify software engineering risks and apply appropriate software risk management methodologies
Incorporate risk management strategies into software project planning and management.
Give examples of Software security considerations
Summarize Security Risk Management
Summarize Software security guidance (regulations, standards, "orange book")
Explain System Certification
Give examples of contemporary security developments
Given a notional software-intensive system, describe software information assurance requirements appropriate to the overall development and acquisition of that system.
Given information about a software-intensive system, identify software safety and reliability issues for the system.
Government and Industry Software Acquisition Management Roles
Utilize Standards for Configuration Mgt
Demonstrate Configuration Mgt Planning
Demonstrate the use of Configuration Mgt throughout SW life-cycle (SMRB, etc.)
Demonstrate the synchronization of HW and SW baselines
Utilize Configuration Management CASE tools
Demonstrate the management of Configuration Risks
Identify the role and functions of configuration management in the acquisition process.
Given a software-intensive system, select software configuration management (CM) activities and issues that are appropriate to the various development phases of a software-intensive system.
Explain the fundamentals of Configuration Management (CM) in software systems.
Demonstrate Staffing best practices.
Demonstrate Organizational best practices
Demonstrate best practices for Matrix Support Groups
Demonstrate Resource Management best practices
Demonstrate best practices for Project Control
Demonstrate best practices for Project Tracking
Demonstrate End User Involvement
Demonstrate best practices for IPT's and working groups
Demonstrate best practices for Intergroup Coordination
Select Corrective Actions
Utilize Lessons Learned
Demonstrate best practices to deal with Management Issues
Given background materials on ISAM course competencies and DoD Acquisition environment, relate ISAM lesson topics to individual learning needs and describe the typical roles played by software management professionals.
Describe the role of the project manager is software project initiating and planning.
Compare the roles and responsibilities of the systems engineering effort across government and contractor boundaries (e.g., Chief Engineer, Lead Systems Engineer, IPT members, etc.) has in regards to the implementation of systems engineering and software engineering.
Interact with software program integrated product teams regarding the application of the systems engineering process to their respective area of expertise.
Unique Software Procurement Requirements Application & Analysis
Prepare a SW Development Plan (SDP)
Use a SDP in a proposal evaluation
Prepare a Work Break-down Structure (WBS) for SW
Select Laws/regulation related to SOW and RFP
Prepare solutions for Quality Issues
Select Contract types based on their strengths and weaknesses (for all types of systems)
Select Deliverables (based on issues and tradeoffs)
Demonstrate a SW Proposal Evaluation
Incorporate Data and intellectual property rights
Incorporate Commercial & DoD best practices such as Joint Technical Architecture (JTA), Open Systems, COTS, Reuse
Prepare Model SOWs
Describe the role of contracts in software acquisition management and software engineering.
Given a software-intensive system and a systems-level acquisition strategy, choose key practices considered essential to contracting for such a system; and identify key activities, tasks, and criteria considered essential for effective proposal evaluation and selection of the best-qualified contractor for that system.
Summarize the role of contracting in the software acquisition process and the major contractual contributions towards managing program risk.
Analyze given proposals and select the best-qualified contractor for a given software-intensive system acquisition.
Analyze given proposals and requirements and select the best-qualified contractor for the acquisition of software development services.
Develop a plan to implement data rights and intellectual property policies within a software-intensive acquisition program.
Give examples of Open System Migration issues
Identify applicability of Naval Open System architecture policy and guidelines
Identify applicability of Open System architecture policy and guidelines
Summarize Open System guidance (Application Portability Profile, regulations, standards)
Explain Open System adaptation effect on acquisition
Give examples of Commercial Off the Shelf/Non-Developmental item (COTS/NDI) issues
Given system requirements and a software application domain, assess life cycle impacts and risks of using COTS and NDI/GOTS as part of computer resource planning and support.

Software Metrics Application & Analysis
Explain the roles of assessments/evaluations
Demonstrate methods available to assess maturity
Distinguish the strengths and weaknesses of current methods
Demonstrate different applications of assessments and evaluations
Demonstrate the role of evaluations/assessments in contracting
Select the frequency of evaluations/assessments
Summarize responsibilities for evaluations/assessments
Compare the three CMMIs - Development, Acquisition, and Services - and their intended environments.
For each PA in maturity levels 2 and 3, describe the typical activities and typical work products that can be expected in an organization that has implemented processes consistent with the PA
Compare and contrast the Software CMM and the CMMI
Explain how CMMI Process Areas (PA) relate to a software or systems engineering life cycles
Describe the CMMI's basic structure and components.
Explain the meaning of capability levels and maturity levels.
Describe the interrelationships between CMMI components.
Identify the CMMI Process Areas.
Locate relevant information in CMMI models.
Describe the environments for which CMMI is best suited.
Describe the role of CMMI-based process discipline in acquisition environments.
Explain the use of process and CMMI appraisals in acquisition.
Identify best practices in using statistics and measures to quantify, plot, and analyze software development in order to manage and improve software acquisition processes.
Select appropriate metrics for visibility into development process, software product, system progress
Select metrics collection methodologies
Demonstrate Metrics Interpretation
Demonstrate Bench marking practices
Develop a Measurement Plan and establish baseline measures.
Evaluate project/program performance metrics as indicators of problems in software-intensive acquisition programs.
Analyze Software Technical Life Cycle & Relate It To System Acquisition Process
Distinguish among current approaches (e.g., Functional, Object-Oriented)
Explain the strengths and weaknesses of design approaches
Explain the effect of design approach on SW engineering, project planning, CASE selection and use, design reviews & docs
Summarize Software Design Guidance (laws, regs, Stds)
Summarize technical fundamentals
Distinguish among Development Paradigms (Waterfall, Spiral, Prototyping, Incremental, IE) definitions
Give examples of criteria for Paradigm selection
Explain the risks and benefits of each development
Summarize paradigm selection resource/management issues
Describe approaches to creating and documenting the structure of a software system.
List common programming or scripting languages.
Using a software-intensive system and software development planning information, identify key practices that can be used by developers to create a quality software product.
Given a software-intensive system and a draft software development plan, analyze the plan for sufficiency and coverage of project specific software acquisition and development issues.
Describe the concept of agile software development.
Given software-intensive system requirements and current DoD policies, assess the impacts of DoD interoperability policies, requirements, applicable architectures and open systems concepts on the acquisition, development, and support of a software-intensive system.
Given requirements documents, acquisition strategy information, risk assessments, and other programmatic documentation for a software-intensive system, develop a feasible build plan for the system.
Explain the importance of accounting for maintenance in software acquisition and development.
Describe the similarities between software maintenance and software development.
Estimate the maintenance effort involved in a software system and evaluate risks associated with continued maintenance vs. redevelopment.
Explain Cost Factors identification
Summarize Key Software support transition issues
Summarize Organic/Outsourcing Post Deployment Software Support considerations
Summarize considerations for Software Engineering Environment acquisition & use
Summarize DoD Life Cycle Guidance (Directives, Instructions, Standards, etc.)
Give examples of Support Organization Involvement
Explain continuous process improvement
Give examples of End User Involvement
Explain Corrective Actions Management
Explain Contract Baseline
Explain the relationship with contractor(s)
Explain how software acquisition activities impact and relate with other functional areas within the software acquisition life cycle.
Describe software lifecycle models.
Describe the phases of the software development life cycle to include requirements analysis, design, implementation, test & evaluation, and maintenance.
Describe key logistics support elements to consider in software product support/sustainability planning and management
Given a software-intensive system in the latter stages of development, identify key issues for deploying it, transitioning its maintenance, and disposing of it.
Distinguish between system development life cycle and the system life cycle.
Select appropriate software lifecycle models for a given system.
Justify the importance of software supportability to achieving system readiness requirements.
Given a software acquisition system, identify critical program management and logistics decisions concerning software system supportability issues and alternatives that would optimize software system design for supportability.
Explain Joint Technical Architecture (JTA) [or its equivalent, e.g., DoDAF]

Explain Domain & product line engineering
Explain state of the art software technology topics
Compare and contrast, in the changing DoD environment, the impacts of major institutional players, major new software acquisition initiatives, and policies specific to defense software acquisition management.
Compare and contrast among modeling and simulation tools, demonstrating that the tools chosen appropriately offer productivity, reliability, availability, and accessibility in support of the organization's missions.
Software Testing “Best Practices” Application
Summarize IV&V definition, benefits, and disadvantages
Explain how to determine IV&V levels
Summarize IV&V guidance
Explain IV&V relationship to risk management and testing
Explain IV&V effect on development schedule
Describe the differing roles of validation, verification, and testing
Explain how V&V and testing fits in the software lifecycle
Identify different V&V techniques and tools
Outline Software quality factors
Outline Software quality guidance
Illustrate Quality improvement methods (Formal Inspection, Walk throughs, Clean room, Peer reviews)
Illustrate the benefits and risks associated with software quality methods
Illustrate best practices for Software Project Management visibility into software quality (metrics and inspections)
Outline Software Product Assessment Techniques
Outline Software Quality Assurance Planning and Techniques
Explain the fundamentals of Software Quality Assurance in software systems.
Interpret evaluations on the quality of software based on factors such as modularity, maintainability, complexity, and algorithm analysis.
Describe the different meanings of software quality and their associated measures.
Distinguish Software testing Phases (DT&E, F/OT&E)
Give examples of appropriate Testing metrics (software maturity, error density)
Give examples of types of Testing (unit, FOT, integration, DT/OT).
Summarize Software integration testing issues
Explain sufficient software testing
Explain Test and Evaluation Master Plan relationship to Testing
Explain High Integrity Systems
Explain the identification of Testing Risks
Describe the discipline of Software Reliability
Describe the different types of Test and Evaluation (T&E), the organizations responsible for them, and the reason for heavy DoD commitment to T&E.
Describe key software testing and evaluation elements to consider in software acquisition management and software engineering.
Given previous instruction on software testing and a software-intensive system, assess software and system test processes for effectiveness.
Discuss available tools, techniques, and metrics for software testing.
Explain how to incorporate software testing and evaluation elements into software project planning and management (Pareto's law and the impact of core requirements - i.e., 80% of the design and testing is up front before coding begins).
Software Acquisition Management Planning & Status Documentation Analysis
Summarize Software Requirement management
Summarize Requirement Management guidance
Summarize Requirement Management responsibilities
Explain User involvement
Give examples of Requirement Planning issues
Distinguish among the types of requirements (derived, explicit, decomposed)
Give examples of software requirements, and describe the benefits and risks of prototyping
Give examples of Requirements Management issues (baselines, traceability, tool support, life cycle requirements variance)
Give examples of Requirements/COTS issues
Explain Critical measures of effectiveness for operational issues and criteria
Given a software-intensive system within an application domain, select appropriate software requirements management methodologies and techniques.
Summarize the Government management of reviews and audit process
Give examples of high interest Software issues and their indicators
Summarize Critical Software life cycle reviews
Give examples of key Software review questions and data
Give examples of Entrance & Exit Criteria
Explain the Software review relationship to system reviews
Software Economic Factors Analysis
Select methods and models for SW cost & schedule estimation based on their strengths and weaknesses
Demonstrate SW cost & schedule reporting
Demonstrate validation/assessment of fidelity of cost and schedule estimates for SW intensive projects
Predict Life Cycle Costs (incl PDSS)
Given knowledge of the software cost and schedule cost estimating process, assess techniques that can be used in preparing cost and schedule estimates for software-intensive systems
Given various cost estimating tools and summary information about a software-intensive system, develop an initial cost and schedule estimate for that system
Given cost estimation tools and preliminary software development cost and schedule estimates for a software-intensive system, justify an appropriate "should cost" estimate for that system
Describe the basics of software size and effort estimates
Describe the basics of creating and monitoring software schedules
Using EVM principles, create detailed work assignments and initialize a metrics tracking system
Determine an appropriate cost-estimating methodology and the types of data required for a software cost estimate
Identify and appropriately apply models for software life-cycle cost estimating
Compare and contrast alternative techniques for software cost estimating
Describe and apply software cost-estimating techniques
Discuss the strengths and weaknesses of software cost-estimating models

Discuss major influences on software cost estimating
Translate software cost estimates into acquisition program budgets
Explain the major activities involved in evaluating and/or negotiating contract proposals
Explain the major activities in conducting market research on a commercial software product to determine product availability and applicability
Explain Business Process Reengineering (BPR)
Explain best practices for adapting maturing technologies
Describe the Development Information System/Enterprise
Give examples of FPI Guidance, Process, Tools
Explain Model Relationship
Describe the impetus behind the process improvement focus
Explain the program manager's role and responsibilities in software process improvement
Explain process improvement and CMMI roles and responsibilities
Describe the impact of leadership in acquisition and process improvement
Describe the value or benefits of model-based process improvement
Describe different process models/methods to apply, and explain how and when to achieve process improvement
Explain the cost of process improvement investment in project or product delivery
Explain how to measure and report process improvement
Describe the knowledge/skills necessary to effectively apply to process improvement
Demonstrate the value of establishing periodic and timely reviews and reporting milestones in which [the software system] performance is evaluated against the [software system] plan
Given programmatic documentation and project-specific measurement data for a software-intensive system, select and analyze performance measures appropriate to the system's acquisition life cycle; appraise tools and techniques available to the program office for planning, measuring and predicting software development, quality and process maturity

G.1.3 Level III Program Management

Software Acquisition Management Regulatory/Technical Framework Application & Analysis
Discriminate best system strategies for SW intensive systems
Analyze the effect of current system Strategies on SW Acquisition Mgmt
Illustrate the strengths and weaknesses of current strategies
Outline the impact of acquisition strategy on SW project planning and SW Engineering methods
Outline the impact of Acquisition Reform
Summarize the strengths and weaknesses of incorporating software product reuse and Commercial Items products into the acquisition strategy of an information intensive system.
Evaluate software acquisition methodology for its ability to support an acquisition strategy.
Employ software acquisition strategies that are characterized by progressively defining requirements and associated design solutions based on evolving user needs.
Summarize Software Architecture Fundamentals
Show the relationship of SW to System Architecture
Show the relationship of Architecture to SW Design
Demonstrate the impact of architecture on interoperability and reuse
Differentiate between C3I, MCCR, and AIS systems
Demonstrate best practices for Evaluating and Acquiring target environments
Illustrate product line & domain engineering considerations (tradeoffs & analysis)
Assess the benefits and limitations that implementing a standards based architecture brings to the acquisition strategy for a software intensive system.
For a given system, defend the decision for an "open system" or "closed system".
Give examples of Interoperability and Data Administration Issues
Summarize Interoperability and data administration guidance (Laws, regulation, and standards)
Explain the relationship of Software/System Architecture and interoperability
Assess interoperability issues and their impacts on software acquisition.
Analyze the Software Architecture/reuse relationship
Show Risk mitigation through reuse
Outline Reuse guidance
Outline Domain specific reuse paradigm
Differentiate existing Reuse repositories
Illustrate contracting mechanisms for reuse
outline the impact of Open Systems on software reuse
Outline COTS/Reuse Issues
Illustrate Portability, through platform independence
Software Risk Management Application & Analysis
Illustrate Software Risk Analysis
Outline Software Risk management issues (planning, etc.)
Illustrate the benefit of varying risk profile through life cycle
Select Organizational risk mitigation entities (SEMP, RMWG, TIWG, CRWG, CRLCMP, IPT's, etc.) appropriate to a given case situation
Outline Risk Management guidance
Illustrate the concept of a Domain Competent Work Force
Analyze the causes of cost, schedule, and performance problems in large software efforts.
Critique the contention that a software crisis exists and current strategies for addressing the crisis.
Apply and evaluate commonly used best practices risk management models.
Demonstrate Software security considerations
Demonstrate Security Risk Management
Utilize Software security guidance (regulations, standards, "orange book")
Describe System Certification
Utilize contemporary security developments
Evaluate the impact of security, safety and integrity requirements on the development of an acquisition strategy for software intensive systems.

Apply appropriate program security techniques to a software acquisition program.
Government and Industry Software Acquisition Management roles
Utilize Standards for Configuration Mgt
Demonstrate Configuration Mgt Planning
Demonstrate the use of Configuration Mgt throughout SW life-cycle (SMRB, etc.)
Demonstrate the synchronization of HW and SW baselines
Utilize Configuration Management CASE tools
Demonstrate the management of Configuration Risks
Demonstrate Staffing best practices.
Demonstrate Organizational best practices
Demonstrate best practices for Matrix Support Groups
Demonstrate Resource Management best practices
Demonstrate best practices for Project Control
Demonstrate best practices for Project Tracking
Demonstrate End User Involvement
Demonstrate best practices for IPT's and working groups
Demonstrate best practices for Intergroup Coordination
Select Corrective Actions
Utilize Lessons Learned
Demonstrate best practices to deal with Management Issues
Evaluate the success factors for creating and sustaining cohesive teams within a software organization.
Analyze the organizational and cultural dynamics of program offices and software development teams.
Evaluate the suitability of alternative organization structures, including integrated product teams.
Describe the appropriate skills mix needed to staff a software project.
Unique Software Procurement Requirements Application & Analysis
Analyze a SW Development Plan (SDP)
Analyze a SDP in a proposal evaluation
Analyze a Work Break-down Structure (WBS) for SW
Outline Laws/regulation related to SOW and RFP
Analyze solutions for Quality Issues
Select Contract types based on their strengths and weaknesses (for all types of systems)
Select Deliverables (based on issues and tradeoffs)
Analyze a SW Proposal Evaluation
Incorporate Data and intellectual property rights
Illustrate Commercial & DoD best practices such as Joint Technical Architecture (JTA), Open Systems, COTS, Reuse
Analyze Model SOWs
Design an acquisition philosophy or model that fits the organization's mission, needs, and culture. Among the factors considered include sourcing issues, type(s) of contract, modular contracting, award fees, use of subcontractors, etc.
Analyze the security implications of software assurance, as it applies to confidentiality, and integrity, including legislation dealing with source manufacturing. Include internal GOTS, external COTS, internet/intranet, legacy codes, applicable legislation regarding source manufacturing, and the types of individuals (US trained, foreign national H-1B visa holders, off-shore workforce, etc.) developing software.
Originate a complete solicitation that effectively communicates the software acquisition strategy and factors for award.
Give examples of Open System Migration issues
Identify applicability of Naval Open System architecture policy and guidelines
Summarize Open System guidance (Application Portability Profile, regulations, standards)
Explain Open System adaptation effect on acquisition
Give examples of Commercial Off the Shelf/Non-Developmental item (COTS/NDI) issues
Software Metrics Application & Analysis
Outline the roles of assessments/evaluations
Analyze methods available to assess maturity
Illustrate the strengths and weaknesses of current methods
Analyze different applications of assessments and evaluations
Outline the role of evaluations/assessments in contracting
Analyze the frequency of evaluations/assessments
Outline responsibilities for evaluations/assessments
Analyze metrics for visibility into development process, software product, system progress
Analyze metrics collection methodologies
Analyze interpretations of metrics
Analyze bench marking practices
Apply data administration and management elements, initiatives, methods, and technologies to an information systems acquisition program.
Evaluate and select software metrics that will provide insight into program status and facilitate early detection of potential problems.
Analyze Software Technical Life Cycle & Relate It To System Acquisition Process
Select current approaches (e.g., Functional, Object-Oriented)
Select design approaches based on their strengths and weaknesses
Predict the effect of design approach on SW engineering, project planning, CASE selection and use, design reviews & docs
Use Software Design Guidance (laws, regs, Stds)
Summarize technical fundamentals
Distinguish among Development Paradigms (Waterfall, Spiral, Prototyping, Incremental, IE) definitions
Prepare for Paradigm selection
Select a development paradigm based on the risks and benefits of each
Summarize paradigm selection resource/management issues
Outline SE activities in the context of the various life cycle phases of the Defense acquisition framework.
Analyze the scope of SE and its relationship to other program management functions across the life cycle.
List important design considerations and their impacts
Identify and explain technical processes that can be applied to control and assess systems engineering (SE) activities for software-intensive systems.
Select an appropriate reengineering strategy to implement, develop, and integrate a software intensive system.
Evaluate and manage a SE process to translate requirements into integrated design solutions, ensuring that solutions both meet current requirements and

facilitate the incorporation of new technologies and capabilities to meet future needs.
Develop key portions of a Systems Engineering Plan.
Describe and analyze the software development and acquisition process.
Illustrate Cost Factors identification
Outline Key Software support transition issues
Outline Organic/Outsourcing Post Deployment Software Support
Analyze considerations for Software Engineering Environment acquisition & use
Outline DoD Life Cycle Guidance (Directives, Instructions, Standards, etc.)
Illustrate Support Organization Involvement
Illustrate Continuous process improvement
Illustrate End User Involvement
Outline Corrective Actions Management
Prepare a Contract Baseline
Explain the relationship with contractor(s)
Evaluate the different parts of the life cycle to achieve a useful and cost effective outcome.
Evaluate acquisition logistics functions and documentation needs over a software system's life cycle, including organic/outsourcing post deployment software issues, and commercial production and support.
Explain Joint Technical Architecture (JTA) [or its equivalent, e.g., DoDAF]
Explain Domain & product line engineering
Explain state of the art software technology topics
Analyze the use of advanced technology tools such as integrated product teams, modeling and simulation, and open systems architectures, to further facilitate management of a developing system.
Evaluate the impact of Congressional and Federal acquisition reform initiatives on acquisition management for software intensive systems.
Assess the impact of current/emerging law upon software acquisition and use.
Formulate and describe strategies to influence Defense software acquisition policies, strategies, plans and procedures.
Evaluate benefits, limitations and tradeoffs of modeling, simulation and prototyping as tools supporting the program life cycle.
Software Testing "Best Practices" Application
Summarize IV&V definition, benefits, and disadvantages
Explain how to determine IV&V levels
Summarize IV&V guidance
Explain IV&V relationship to risk management and testing
Explain IV&V effect on development schedule
Evaluate evidence that a system element meets the defined requirements ("build-to specification) of a given software-intensive system.
Analyze Software quality factors
Modify Software quality guidance
Analyze Quality improvement methods (Formal Inspection, Walk throughs, Clean room, Peer reviews)
Analyze the benefits and risks associated with software quality methods
Create a plan to allow Software Project Management visibility into software quality (metrics and inspections)
Analyze Software Product Assessment Techniques
Create a Software Quality Assurance Plan
Choose appropriate software quality management methodologies based on cost, schedule, and performance risk management considerations.
Differentiate Software testing Phases (DT&E, F/OT&E)
Discriminate appropriate Testing metrics (software maturity, error density)
Discriminate Type of Testing (unit, FOT, integration, DT/OT).
Outline Software integration testing issues
Illustrate sufficient software testing
Outline the Test and Evaluation Master Plan relationship to Testing
Illustrate High Integrity Systems
Illustrate the identification of Testing Risks
Evaluate whether a software testing program adequately supports the quality, mission effectiveness and mission suitability goals of an information intensive acquisition program throughout its life cycle of an information intensive program.
Evaluate methodologies for analyzing, determining, refining, implementing, and testing software intensive system requirements.
Explain the role of testing and evaluation as a feedback mechanism and management tool for the engineering and development of software-intensive systems.
Software Acquisition Management Planning & Status Documentation Analysis
Differentiate Software Requirement management from other SW acquisition management practices
Outline Requirement Management guidance
Differentiate Requirement Management responsibilities
Illustrate User involvement
Outline Requirement Planning issues
Differentiate the types of requirements (derived, explicit, decomposed)
Outline Software requirement definition, benefits, and risks of prototyping
Discriminate Requirements Management issues (baselines, traceability, tool support, life cycle requirements variance)
Outline Requirements/COTS issues
Analyze critical measures of effectiveness for operational issues and criteria
Based on high-level project requirements, create and execute a software management plan to track and control a software-intensive program.
Analyze the requirements process and its impact on the acquisition process, especially in regards to Initial Capabilities Document (ICD), Capabilities Development Document (CDD), Capabilities Production Document (CPD), Acquisition Program Baseline (APB), and related documents (e.g., Command, Control, Communications, computers and Intelligence (C4I), analysis of Alternatives (AOA), etc.).
Evaluate methodologies for analyzing, determining, refining, implementing, and testing software intensive system requirements.
Outline the Government management of reviews and audit process
Illustrate high interest Software issues and their indicators
Outline Critical Software life cycle reviews
Outline key Software review questions and data
Analyze Entrance & Exit Criteria
Outline Software review relationship to system reviews
Present and defend capstone software acquisition case analysis.
Assess Federal and DoD acquisition initiatives

Originate tailored, value added, program documentation (e.g. Acquisition Program Baseline, Risk Management Plan, cost estimates, test results, etc.).
Design a method to ensure that measurement data that has been collected in the assessment process is used in the review and decision making processes.
Software Economic Factors Analysis
Differentiate methods and models used for SW cost & schedule estimation based on their strengths and weaknesses
Outline SW cost & schedule reporting
Illustrate Validation/assessment of fidelity of cost and schedule estimates for SW intensive projects
Outline Life Cycle Costs (incl PDSS)
Evaluate strengths and weaknesses of software cost estimation methods and models.
Evaluate the philosophy, practice, and processes and merits of for determining, refining, and implementing cost as an independent variable (CAIV) and earned value (EV) in managing software intensive systems.
Evaluate, select and apply government and commercial decision tools and evaluation systems used for estimating, measuring, and predicting software cost, schedule and quality as well as in making go/no go decisions.
Describe the ways in which benchmarks may be used to forecast performance of your [software-intensive project/program].
Estimate the risk reserve required for a software intensive system.
Outline Business Process Reengineering (BPR)
Illustrate Adapting maturing technologies
Outline Development Information System/Enterprise
Outline FPI Guidance, Process, Tools
Illustrate Model Relationship
Describe the linkage of technical reviews to technical program management.
Evaluate the impact of selected technologies on the acquisition and development of software-intensive systems.
Assess the revised business orientation reflected in the new DoD acquisition policy.
Examine differences between commercial software acquisition efforts and DoD efforts.
Recognize and selectively adopt commercial best practices.

G.2 SPRDE Systems & Software Engineering Competencies for Levels I, II, & III

SPRDE-SE SMRT Competencies				
Key Competency Area	Sub-competencies	Level I	Level II	Level III
		Generalists	Generalists	Generalists
Acquisition Strategies				
	Best system strategies for SW intensive systems	2	3	4
	Affect of current system Strategies on SW Acquisition Mgmt	2	3	4
	Strengths and weaknesses of current strategies	2	3	4
	Impact of acquisition strategy on SW project planning and SW Engineering methods	2	3	4
	Impact of Acquisition Reform	2	3	4
Architecture				
	Software Architecture Fundamentals	0	2	2
	Relationship of SW to System Architecture	0	2	2
	Relationship of Architecture to SW Design	0	2	2
	Impact of architecture on interoperability and reuse	0	2	2
	Differences in C3I, MCCR, and AIS systems	0	2	2
	Evaluating and Acquiring target environments	0	2	2
	Product line & domain engineering considerations (tradeoffs & analysis)	0	2	2
Contracting Issues				
	Development of SW Development Plan (SDP)	2	3	4
	Use of SDP in proposal evaluation	2	3	4
	Work Break-down Structure (WBS) for SW	2	3	4
	Laws/regulation related to SOW and RFP	2	3	4
	Quality Issues	2	3	4
	Contract types and their strengths and weaknesses (for all types of systems)	2	3	4
	Deliverables (issues and tradeoffs)	2	3	4
	SW portion of Proposal Evaluation	2	3	4
	Data and intellectual property rights	2	3	4
	Commercial & DoD best practices such as Joint Technical Architecture (JTA), Open Systems, COTS, Reuse	2	3	4
	Model SOWs	2	3	4
Configuration Management				
	Standards for Configuration Mgt	2	3	4
	Configuration Mgt Planning	2	3	4
	Use of Configuration Mgt throughout SW life-cycle (SMRB, etc.)	2	3	4
	Synchronization of HW and SW baselines	2	3	4
	Configuration Management CASE tools	2	3	4
	Management of Configuration Risks	2	3	4
Software Cost & Schedule Estimation				
	Strengths and weaknesses of methods and models used for SW cost & schedule estimation	1	3	4
	SW cost & schedule reporting	1	3	4
	Validation/assessment of fidelity of cost and schedule estimates for SW intensive projects	1	3	4
	Life Cycle Costs (incl PDSS)	1	3	4
Program/Project Office organization & relationships				
	Staffing	2	3	4
	Organization	2	3	4
	Matrix Support Groups	2	3	4
	Resource Management	2	3	4
	Project Control	2	3	4
	Project Tracking	2	3	4
	End User Involvement	2	3	4
	IPT's and working groups	2	3	4
	Intergroup Coordination	2	3	4
	Corrective Actions	2	3	4
	Lessons Learned	2	3	4
	Management Issues	2	3	4

Software developing and acquiring maturity				
	Roles of assessments/evaluations	2	3	4
	Methods available to assess maturity	2	3	4
	Strengths and weaknesses of current methods	2	3	4
	Applications of assessments and evaluations	2	3	4
	Role of evaluations/assessments in contracting	2	3	4
	Frequency of evaluations/assessments	2	3	4
	Responsibilities for evaluations/assessments	2	3	4
Engineering Approaches & Methodologies				
	Current approaches (e.g., Functional, Object-Oriented)	2	3	4
	Strengths and weaknesses of design approaches	2	3	4
	Effect of design approach on SW engineering, project planning, CASE selection and use, design reviews & docs	2	3	4
	Software Design Guidance (laws, regs, Stds)	2	3	4
	Technical fundamentals	2	3	4
	Development Paradigms (Waterfall, Spiral, Prototyping, Incremental, IE) definitions	2	3	4
	Criteria for Paradigm selection	2	3	4
	Risks and benefits of each development	2	3	4
	Paradigm selection resource/management issues	2	3	4
Technical Assessments				
	Business Process Reengineering (BPR)	0	1	2
	Adapting maturing technologies	0	1	2
	Development Information System/Enterprise	0	1	2
	FPI Guidance, Process, Tools	0	1	2
	Model Relationship	0	1	2
Interoperability				
	Interoperability and Data Administration Issues	1	3	3
	Interoperability and data administration guidance (Laws, regulation, and standards)	1	3	3
	Relationship of Software/System Architecture and interoperability	1	3	3
Independent Verification and Validation (IV&V)				
	IV&V definition, benefits, and disadvantages	1	2	2
	Determine IV&V levels	1	2	2
	IV&V guidance	1	2	2
	IV&V relationship to risk management and testing	1	2	2
	IV&V effect on development schedule	1	2	2
Life Cycle Management				
	Cost Factors identification	1	3	4
	Key Software support transition issues	1	3	4
	Organic/Outsourcing Post Deployment Software Support	1	3	4
	Software Engineering Environment acquisition & use	1	3	4
	DoD Life Cycle Guidance (Directives, Instructions, Standards, etc.)	1	3	4
	Support Organization Involvement	1	3	4
	Continuous process improvement	1	3	4
	End User Involvement	1	3	4
	Corrective Actions Management	1	3	4
	Contract Baseline	1	3	4
	Relationship with contractor	1	3	4
Metrics				
	Appropriate metrics for visibility into development process, software product, system progress	1	3	4
	Metrics Collection methodologies	1	3	4
	Metrics Interpretation	1	3	4
	Bench marking practices	1	3	4
Open Systems				
	Open System Migration issues	1	2	3
	Open System guidance (Application Portability Profile, regulations, standards)	1	2	3
	Open System adaptation effect on acquisition	1	2	3
	Commercial Off the Shelf/Non-Developmental item (COTS/NDI) issues	1	2	3
Software Quality Management				
	Software quality factors	1	2	3
	Software quality guidance	1	2	3

	Quality improvement methods (Formal Inspection, Walk throughs, Clean room, Peer reviews)	1	2	3
	Benefits and risks associated with software quality methods	1	2	3
	Software Project Management visibility into software quality (metrics and inspections)	1	2	3
	Software Product Assessment Techniques	1	2	3
	Software Quality Assurance Planning and Techniques	1	2	3
Software Requirement Management				
	Software Requirement management definition	1	3	3
	Requirement Management guidance	1	3	3
	Requirement Management responsibilities	1	3	3
	User involvement	1	3	3
	Requirement Planning issues	1	3	3
	Types of requirements (derived, explicit, decomposed)	1	3	3
	Software requirement definition, benefits, and risks of prototyping	1	3	3
	Requirements Management issues (baselines, traceability, tool support, life cycle requirements variance)	1	3	3
	Requirements/COTS issues	1	3	3
	Critical measures of effectiveness for operational issues and criteria	1	3	3
Software Reviews & Audits				
	Government management of reviews and audit process	1	3	3
	High interest Software issues and their indicators	1	3	3
	Critical Software life cycle reviews	1	3	3
	Key Software review questions and data	1	3	3
	Entrance & Exit Criteria	1	3	3
	Software review relationship to system reviews	1	3	3
Software Reuse				
	Software Architecture/reuse relationship	2	3	4
	Risk mitigation through reuse	2	3	4
	Reuse guidance	2	3	4
	Domain specific reuse paradigm	2	3	4
	Existing Reuse repositories	2	3	4
	Contracting mechanisms for reuse	2	3	4
	Impact of Open Systems on software reuse	2	3	4
	COTS/Reuse Issues	2	3	4
	Portability, through platform independence	2	3	4
Software Acquisition Risk Management				
	Software Risk Analysis	2	3	4
	Software Risk management issues (planning, etc.)	2	3	4
	Varying risk profile through life cycle	2	3	4
	Organizational risk mitigation entities (SEMP, RMWG, TIWG, CRWG, CRLCMP, IPT's, etc.)	2	3	4
	Risk Management guidance	2	3	4
	Domain Competent Work Force	2	3	4
Software Security				
	Software security definition	1	3	3
	Security Risk Management	1	3	3
	Software security guidance (regulations, standards, "orange book")	1	3	3
	System Certification	1	3	3
	Contemporary security developments	1	3	3
Software testing Issues				
	Software testing Phases (DT&E, F/OT&E)	2	3	4
	Appropriate Testing metrics (software maturity, error density)	2	3	4
	Type of Testing (unit, FOT, integration, DT/OT).	2	3	4
	Software integration testing issues	2	3	4
	Sufficient software testing	2	3	4
	Test and Evaluation Master Plan relationship to Testing	2	3	4
	High Integrity Systems	2	3	4
	Identification of Testing Risks	2	3	4
Emerging issues & Technologies				
	Joint Technical Architecture (JTA)	1	2	3
	Domain & product line engineering	1	2	3
	Software technology state of the art	1	2	3

G.3 SMRT T&E Engineering Competencies for Levels I, II, & III

Test & Evaluation SMRT Competencies				
Key Competency Area	Sub-competencies	Level I	Level II	Level III
		Generalists	Generalists	Generalists
Acquisition Strategies				
	Best system strategies for SW intensive systems	1	2	3
	Affect of current system Strategies on SW Acquisition Mgmt	1	2	3
	Strengths and weaknesses of current strategies	1	2	3
	Impact of acquisition strategy on SW project planning and SW Engineering methods	1	2	3
	Impact of Acquisition Reform	1	2	3
Architecture				
	Software Architecture Fundamentals	1	3	3
	Relationship of SW to System Architecture	1	3	3
	Relationship of Architecture to SW Design	1	3	3
	Impact of architecture on interoperability and reuse	1	3	3
	Differences in C3I, MCCR, and AIS systems	1	3	3
	Evaluating and Acquiring target environments	1	3	3
	Product line & domain engineering considerations (tradeoffs & analysis)	1	3	3
Contracting Issues				
	Development of SW Development Plan (SDP)	1	3	4
	Use of SDP in proposal evaluation	1	3	4
	Work Break-down Structure (WBS) for SW	1	3	4
	Laws/regulation related to SOW and RFP	1	3	4
	Quality Issues	1	3	4
	Contract types and their strengths and weaknesses (for all types of systems)	1	3	4
	Deliverables (issues and tradeoffs)	1	3	4
	SW portion of Proposal Evaluation	1	3	4
	Data and intellectual property rights	1	3	4
	Commercial & DoD best practices such as Joint Technical Architecture (JTA), Open Systems, COTS, Reuse	1	3	4
	Model SOWs	1	3	4
Configuration Management				
	Standards for Configuration Mgt	2	3	3
	Configuration Mgt Planning	2	3	3
	Use of Configuration Mgt throughout SW life-cycle (SMRB, etc.)	2	3	3
	Synchronization of HW and SW baselines	2	3	3
	Configuration Management CASE tools	2	3	3
	Management of Configuration Risks	2	3	3
Software Cost & Schedule Estimation				
	Strengths and weaknesses of methods and models used for SW cost & schedule estimation	1	3	4
	SW cost & schedule reporting	1	3	4
	Validation/assessment of fidelity of cost and schedule estimates for SW intensive projects	1	3	4
	Life Cycle Costs (incl PDSS)	1	3	4
Program/Project Office organization & relationships				
	Staffing	2	3	3
	Organization	2	3	3
	Matrix Support Groups	2	3	3
	Resource Management	2	3	3
	Project Control	2	3	3
	Project Tracking	2	3	3
	End User Involvement	2	3	3
	IPT's and working groups	2	3	3
	Intergroup Coordination	2	3	3
	Corrective Actions	2	3	3
	Lessons Learned	2	3	3

	Management Issues	2	3	3
Software developing and acquiring maturity				
	Roles of assessments/evaluations	1	2	3
	Methods available to assess maturity	1	2	3
	Strengths and weaknesses of current methods	1	2	3
	Applications of assessments and evaluations	1	2	3
	Role of evaluations/assessments in contracting	1	2	3
	Frequency of evaluations/assessments	1	2	3
	Responsibilities for evaluations/assessments	1	2	3
Engineering Approaches & Methodologies				
	Current approaches (e.g., Functional, Object-Oriented)	2	3	3
	Strengths and weaknesses of design approaches	2	3	3
	Effect of design approach on SW engineering, project planning, CASE selection and use, design reviews & docs	2	3	3
	Software Design Guidance (laws, regs, Stds)	2	3	3
	Technical fundamentals	2	3	3
	Development Paradigms (Waterfall, Spiral, Prototyping, Incremental, IE) definitions	2	3	3
	Criteria for Paradigm selection	2	3	3
	Risks and benefits of each development	2	3	3
	Paradigm selection resource/management issues	2	3	3
Technical Assessments				
	Business Process Reengineering (BPR)	0	1	1
	Adapting maturing technologies	0	1	1
	Development Information System/Enterprise	0	1	1
	FPI Guidance, Process, Tools	0	1	1
	Model Relationship	0	1	1
Interoperability				
	Interoperability and Data Administration Issues	1	3	3
	Interoperability and data administration guidance (Laws, regulation, and standards)	1	3	3
	Relationship of Software/System Architecture and interoperability	1	3	3
Independent Verification and Validation (IV&V)				
	IV&V definition, benefits, and disadvantages	1	2	2
	Determine IV&V levels	1	2	2
	IV&V guidance	1	2	2
	IV&V relationship to risk management and testing	1	2	2
	IV&V effect on development schedule	1	2	2
Life Cycle Management				
	Cost Factors identification	1	3	4
	Key Software support transition issues	1	3	4
	Organic/Outsourcing Post Deployment Software Support	1	3	4
	Software Engineering Environment acquisition & use	1	3	4
	DoD Life Cycle Guidance (Directives, Instructions, Standards, etc.)	1	3	4
	Support Organization Involvement	1	3	4
	Continuous process improvement	1	3	4
	End User Involvement	1	3	4
	Corrective Actions Management	1	3	4
	Contract Baseline	1	3	4
	Relationship with contractor	1	3	4
Metrics				
	Appropriate metrics for visibility into development process, software product, system progress	1	3	4
	Metrics Collection methodologies	1	3	4
	Metrics Interpretation	1	3	4
	Bench marking practices	1	3	4
Open Systems				
	Open System Migration issues	1	2	3
	Open System guidance (Application Portability Profile, regulations, standards)	1	2	3
	Open System adaptation effect on acquisition	1	2	3
	Commercial Off the Shelf/Non-Developmental item (COTS/NDI) issues	1	2	3
Software Quality Management				
	Software quality factors	1	2	3

	Software quality guidance	1	2	3
	Quality improvement methods (Formal Inspection, Walk throughs, Clean room, Peer reviews)	1	2	3
	Benefits and risks associated with software quality methods	1	2	3
	Software Project Management visibility into software quality (metrics and inspections)	1	2	3
	Software Product Assessment Techniques	1	2	3
	Software Quality Assurance Planning and Techniques	1	2	3
Software Requirement Management				
	Software Requirement management definition	1	2	3
	Requirement Management guidance	1	2	3
	Requirement Management responsibilities	1	2	3
	User involvement	1	2	3
	Requirement Planning issues	1	2	3
	Types of requirements (derived, explicit, decomposed)	1	2	3
	Software requirement definition, benefits, and risks of prototyping	1	2	3
	Requirements Management issues (baselines, traceability, tool support, life cycle requirements variance)	1	2	3
	Requirements/COTS issues	1	2	3
	Critical measures of effectiveness for operational issues and criteria	1	2	3
Software Reviews & Audits				
	Government management of reviews and audit process	1	2	3
	High interest Software issues and their indicators	1	2	3
	Critical Software life cycle reviews	1	2	3
	Key Software review questions and data	1	2	3
	Entrance & Exit Criteria	1	2	3
	Software review relationship to system reviews	1	2	3
Software Reuse				
	Software Architecture/reuse relationship	1	3	3
	Risk mitigation through reuse	1	3	3
	Reuse guidance	1	3	3
	Domain specific reuse paradigm	1	3	3
	Existing Reuse repositories	1	3	3
	Contracting mechanisms for reuse	1	3	3
	Impact of Open Systems on software reuse	1	3	3
	COTS/Reuse Issues	1	3	3
	Portability, through platform independence	1	3	3
Software Acquisition Risk Management				
	Software Risk Analysis	1	3	3
	Software Risk management issues (planning, etc.)	1	3	3
	Varying risk profile through life cycle	1	3	3
	Organizational risk mitigation entities (SEMP, RMWG, TIWG, CRWG, CRLCMP, IPT's, etc.)	1	3	3
	Risk Management guidance	1	3	3
	Domain Competent Work Force	1	3	3
Software Security				
	Software security definition	1	3	3
	Security Risk Management	1	3	3
	Software security guidance (regulations, standards, "orange book")	1	3	3
	System Certification	1	3	3
	Contemporary security developments	1	3	3
Software testing Issues				
	Software testing Phases (DT&E, F/OT&E)	2	3	4
	Appropriate Testing metrics (software maturity, error density)	2	3	4
	Type of Testing (unit, FOT, integration, DT/OT).	2	3	4
	Software integration testing issues	2	3	4
	Sufficient software testing	2	3	4
	Test and Evaluation Master Plan relationship to Testing	2	3	4
	High Integrity Systems	2	3	4
	Identification of Testing Risks	2	3	4
Emerging issues & Technologies				
	Joint Technical Architecture (JTA)	1	2	2
	Domain & product line engineering	1	2	2
	Software technology state of the art	1	2	2

G.4 SMRT Acquisition Logistics Competencies for Levels I, II, & III

Logistics SMRT Competencies				
Key Competency Area	Sub-competencies	Level I	Level II	Level III
		Generalists	Generalists	Generalists
Acquisition Strategies				
	Best system strategies for SW intensive systems	1	2	2
	Affect of current system Strategies on SW Acquisition Mgmt	1	2	2
	Strengths and weaknesses of current strategies	1	2	2
	Impact of acquisition strategy on SW project planning and SW Engineering methods	1	2	2
	Impact of Acquisition Reform	1	2	2
Architecture				
	Software Architecture Fundamentals	0	2	2
	Relationship of SW to System Architecture	0	2	2
	Relationship of Architecture to SW Design	0	2	2
	Impact of architecture on interoperability and reuse	0	2	2
	Differences in C3I, MCCR, and AIS systems	0	2	2
	Evaluating and Acquiring target environments	0	2	2
	Product line & domain engineering considerations (tradeoffs & analysis)	0	2	2
Contracting Issues				
	Development of SW Development Plan (SDP)	1	2	3
	Use of SDP in proposal evaluation	1	2	3
	Work Break-down Structure (WBS) for SW	1	2	3
	Laws/regulation related to SOW and RFP	1	2	3
	Quality Issues	1	2	3
	Contract types and their strengths and weaknesses (for all types of systems)	1	2	3
	Deliverables (issues and tradeoffs)	1	2	3
	SW portion of Proposal Evaluation	1	2	3
	Data and intellectual property rights	1	2	3
	Commercial & DoD best practices such as Joint Technical Architecture (JTA), Open Systems, COTS, Reuse	1	2	3
	Model SOWs	1	2	3
Configuration Management				
	Standards for Configuration Mgt	1	2	3
	Configuration Mgt Planning	1	2	3
	Use of Configuration Mgt throughout SW life-cycle (SMRB, etc.)	1	2	3
	Synchronization of HW and SW baselines	1	2	3
	Configuration Management CASE tools	1	2	3
	Management of Configuration Risks	1	2	3
Software Cost & Schedule Estimation				
	Strengths and weaknesses of methods and models used for SW cost & schedule estimation	1	2	3
	SW cost & schedule reporting	1	2	3
	Validation/assessment of fidelity of cost and schedule estimates for SW intensive projects	1	2	3
	Life Cycle Costs (incl PDSS)	1	2	3
Program/Project Office organization & relationships				
	Staffing	2	3	4
	Organization	2	3	4
	Matrix Support Groups	2	3	4
	Resource Management	2	3	4
	Project Control	2	3	4
	Project Tracking	2	3	4
	End User Involvement	2	3	4
	IPT's and working groups	2	3	4
	Intergroup Coordination	2	3	4
	Corrective Actions	2	3	4
	Lessons Learned	2	3	4

	Management Issues	2	3	4
Software developing and acquiring maturity				
	Roles of assessments/evaluations	1	2	3
	Methods available to assess maturity	1	2	3
	Strengths and weaknesses of current methods	1	2	3
	Applications of assessments and evaluations	1	2	3
	Role of evaluations/assessments in contracting	1	2	3
	Frequency of evaluations/assessments	1	2	3
	Responsibilities for evaluations/assessments	1	2	3
Engineering Approaches & Methodologies				
	Current approaches (e.g., Functional, Object-Oriented)	1	2	3
	Strengths and weaknesses of design approaches	1	2	3
	Effect of design approach on SW engineering, project planning, CASE selection and use, design reviews & docs	1	2	3
	Software Design Guidance (laws, regs, Stds)	1	2	3
	Technical fundamentals	1	2	3
	Development Paradigms (Waterfall, Spiral, Prototyping, Incremental, IE) definitions	1	2	3
	Criteria for Paradigm selection	1	2	3
	Risks and benefits of each development	1	2	3
	Paradigm selection resource/management issues	1	2	3
Technical Assessments				
	Business Process Reengineering (BPR)	0	1	2
	Adapting maturing technologies	0	1	2
	Development Information System/Enterprise	0	1	2
	FPI Guidance, Process, Tools	0	1	2
	Model Relationship	0	1	2
Interoperability				
	Interoperability and Data Administration Issues	1	3	3
	Interoperability and data administration guidance (Laws, regulation, and standards)	1	3	3
	Relationship of Software/System Architecture and interoperability	1	3	3
Independent Verification and Validation (IV&V)				
	IV&V definition, benefits, and disadvantages	0	0	2
	Determine IV&V levels	0	0	2
	IV&V guidance	0	0	2
	IV&V relationship to risk management and testing	0	0	2
	IV&V effect on development schedule	0	0	2
Life Cycle Management				
	Cost Factors identification	1	3	4
	Key Software support transition issues	1	3	4
	Organic/Outsourcing Post Deployment Software Support	1	3	4
	Software Engineering Environment acquisition & use	1	3	4
	DoD Life Cycle Guidance (Directives, Instructions, Standards, etc.)	1	3	4
	Support Organization Involvement	1	3	4
	Continuous process improvement	1	3	4
	End User Involvement	1	3	4
	Corrective Actions Management	1	3	4
	Contract Baseline	1	3	4
	Relationship with contractor	1	3	4
Metrics				
	Appropriate metrics for visibility into development process, software product, system progress	0	1	2
	Metrics Collection methodologies	0	1	2
	Metrics Interpretation	0	1	2
	Bench marking practices	0	1	2
Open Systems				
	Open System Migration issues	1	2	2
	Open System guidance (Application Portability Profile, regulations, standards)	1	2	2
	Open System adaptation effect on acquisition	1	2	2
	Commercial Off the Shelf/Non-Developmental item (COTS/NDI) issues	1	2	2
Software Quality				

Management	Software quality factors	1	2	2
	Software quality guidance	1	2	2
	Quality improvement methods (Formal Inspection, Walk throughs, Clean room, Peer reviews)	1	2	2
	Benefits and risks associated with software quality methods	1	2	2
	Software Project Management visibility into software quality (metrics and inspections)	1	2	2
	Software Product Assessment Techniques	1	2	2
	Software Quality Assurance Planning and Techniques	1	2	2
Software Requirement Management				
	Software Requirement management definition	1	2	2
	Requirement Management guidance	1	2	2
	Requirement Management responsibilities	1	2	2
	User involvement	1	2	2
	Requirement Planning issues	1	2	2
	Types of requirements (derived, explicit, decomposed)	1	2	2
	Software requirement definition, benefits, and risks of prototyping	1	2	2
	Requirements Management issues (baselines, traceability, tool support, life cycle requirements variance)	1	2	2
	Requirements/COTS issues	1	2	2
	Critical measures of effectiveness for operational issues and criteria	1	2	2
Software Reviews & Audits				
	Government management of reviews and audit process	1	2	2
	High interest Software issues and their indicators	1	2	2
	Critical Software life cycle reviews	1	2	2
	Key Software review questions and data	1	2	2
	Entrance & Exit Criteria	1	2	2
	Software review relationship to system reviews	1	2	2
Software Reuse				
	Software Architecture/reuse relationship	1	3	3
	Risk mitigation through reuse	1	3	3
	Reuse guidance	1	3	3
	Domain specific reuse paradigm	1	3	3
	Existing Reuse repositories	1	3	3
	Contracting mechanisms for reuse	1	3	3
	Impact of Open Systems on software reuse	1	3	3
	COTS/Reuse Issues	1	3	3
	Portability, through platform independence	1	3	3
Software Acquisition Risk Management				
	Software Risk Analysis	0	2	3
	Software Risk management issues (planning, etc.)	0	2	3
	Varying risk profile through life cycle	0	2	3
	Organizational risk mitigation entities (SEMP, RMWG, TIWG, CRWG, CRLCMP, IPT's, etc.)	0	2	3
	Risk Management guidance	0	2	3
	Domain Competent Work Force	0	2	3
Software Security				
	Software security definition	0	2	2
	Security Risk Management	0	2	2
	Software security guidance (regulations, standards, "orange book")	0	2	2
	System Certification	0	2	2
	Contemporary security developments	0	2	2
Software testing Issues				
	Software testing Phases (DT&E, F/OT&E)	0	2	2
	Appropriate Testing metrics (software maturity, error density)	0	2	2
	Type of Testing (unit, FOT, integration, DT/OT).	0	2	2
	Software integration testing issues	0	2	2
	Sufficient software testing	0	2	2
	Test and Evaluation Master Plan relationship to Testing	0	2	2
	High Integrity Systems	0	2	2
	Identification of Testing Risks	0	2	2
Emerging issues & Technologies				
	Joint Technical Architecture (JTA)	1	2	2

	Domain & product line engineering	1	2	2
	Software technology state of the art	1	2	2

G.5 SMRT Contracting Competencies for Levels I, II, & III

Contracts SMRT Competencies				
Key Competency Area	Sub-competencies	Level I	Level II	Level III
		Generalists	Generalists	Generalists
Acquisition Strategies				
	Best system strategies for SW intensive systems	1	2	2
	Affect of current system Strategies on SW Acquisition Mgmt	1	2	2
	Strengths and weaknesses of current strategies	1	2	2
	Impact of acquisition strategy on SW project planning and SW Engineering methods	1	2	2
	Impact of Acquisition Reform	1	2	2
Architecture				
	Software Architecture Fundamentals	0	1	2
	Relationship of SW to System Architecture	0	1	2
	Relationship of Architecture to SW Design	0	1	2
	Impact of architecture on interoperability and reuse	0	1	2
	Differences in C3I, MCCR, and AIS systems	0	1	2
	Evaluating and Acquiring target environments	0	1	2
	Product line & domain engineering considerations (tradeoffs & analysis)	0	1	2
Contracting Issues				
	Development of SW Development Plan (SDP)	2	4	4
	Use of SDP in proposal evaluation	2	4	4
	Work Break-down Structure (WBS) for SW	2	4	4
	Laws/regulation related to SOW and RFP	2	4	4
	Quality Issues	2	4	4
	Contract types and their strengths and weaknesses (for all types of systems)	2	4	4
	Deliverables (issues and tradeoffs)	2	4	4
	SW portion of Proposal Evaluation	2	4	4
	Data and intellectual property rights	2	4	4
	Commercial & DoD best practices such as Joint Technical Architecture (JTA), Open Systems, COTS, Reuse	2	4	4
	Model SOWs	2	4	4
Configuration Management				
	Standards for Configuration Mgt	0	1	2
	Configuration Mgt Planning	0	1	2
	Use of Configuration Mgt throughout SW life-cycle (SMRB, etc.)	0	1	2
	Synchronization of HW and SW baselines	0	1	2
	Configuration Management CASE tools	0	1	2
	Management of Configuration Risks	0	1	2
Software Cost & Schedule Estimation				
	Strengths and weaknesses of methods and models used for SW cost & schedule estimation	1	2	2
	SW cost & schedule reporting	1	2	2
	Validation/assessment of fidelity of cost and schedule estimates for SW intensive projects	1	2	2
	Life Cycle Costs (incl PDSS)			
Program/Project Office organization & relationships				
	Staffing	0	1	2
	Organization	0	1	2
	Matrix Support Groups	0	1	2
	Resource Management	0	1	2
	Project Control	0	1	2
	Project Tracking	0	1	2
	End User Involvement	0	1	2
	IPT's and working groups	0	1	2
	Intergroup Coordination	0	1	2
	Corrective Actions	0	1	2
	Lessons Learned	0	1	2
	Management Issues	0	1	2
Software		0	1	2

developing and acquiring maturity	Roles of assessments/evaluations	0	1	2
	Methods available to assess maturity	0	1	2
	Strengths and weaknesses of current methods	0	1	2
	Applications of assessments and evaluations	0	1	2
	Role of evaluations/assessments in contracting	0	1	2
	Frequency of evaluations/assessments	0	1	2
	Responsibilities for evaluations/assessments	0	1	2
Engineering Approaches & Methodologies				
	Current approaches (e.g., Functional, Object-Oriented)	1	2	2
	Strengths and weaknesses of design approaches	1	2	2
	Effect of design approach on SW engineering, project planning, CASE selection and use, design reviews & docs	1	2	2
	Software Design Guidance (laws, regs, Stds)	1	2	2
	Technical fundamentals	1	2	2
	Development Paradigms (Waterfall, Spiral, Prototyping, Incremental, IE) definitions	1	2	2
	Criteria for Paradigm selection	1	2	2
	Risks and benefits of each development	1	2	2
	Paradigm selection resource/management issues	1	2	2
Technical Assessments				
	Business Process Reengineering (BPR)	0	1	1
	Adapting maturing technologies	0	1	1
	Development Information System/Enterprise	0	1	1
	FPI Guidance, Process, Tools	0	1	1
	Model Relationship	0	1	1
Interoperability		0	1	1
	Interoperability and Data Administration Issues	0	1	1
	Interoperability and data administration guidance (Laws, regulation, and standards)	0	1	1
	Relationship of Software/System Architecture and interoperability	0	1	1
Independent Verification and Validation (IV&V)				
	IV&V definition, benefits, and disadvantages	1	2	2
	Determine IV&V levels	1	2	2
	IV&V guidance	1	2	2
	IV&V relationship to risk management and testing	1	2	2
	IV&V effect on development schedule	1	2	2
Life Cycle Management				
	Cost Factors identification	0	1	2
	Key Software support transition issues	0	1	2
	Organic/Outsourcing Post Deployment Software Support	0	1	2
	Software Engineering Environment acquisition & use	0	1	2
	DoD Life Cycle Guidance (Directives, Instructions, Standards, etc.)	0	1	2
	Support Organization Involvement	0	1	2
	Continuous process improvement	0	1	2
	End User Involvement	0	1	2
	Corrective Actions Management	0	1	2
	Contract Baseline	0	1	2
	Relationship with contractor	0	1	2
Metrics				
	Appropriate metrics for visibility into development process, software product, system progress	0	1	2
	Metrics Collection methodologies	0	1	2
	Metrics Interpretation	0	1	2
	Bench marking practices	0	1	2
Open Systems				
	Open System Migration issues	1	2	2
	Open System guidance (Application Portability Profile, regulations, standards)	1	2	2
	Open System adaptation effect on acquisition	1	2	2
	Commercial Off the Shelf/Non-Developmental item (COTS/NDI) issues	1	2	2
Software Quality Management				
	Software quality factors	1	2	2
	Software quality guidance	1	2	2

	Quality improvement methods (Formal Inspection, Walk throughs, Clean room, Peer reviews)	1	2	2
	Benefits and risks associated with software quality methods	1	2	2
	Software Project Management visibility into software quality (metrics and inspections)	1	2	2
	Software Product Assessment Techniques	1	2	2
	Software Quality Assurance Planning and Techniques	1	2	2
Software Requirement Management				
	Software Requirement management definition	1	2	2
	Requirement Management guidance	1	2	2
	Requirement Management responsibilities	1	2	2
	User involvement	1	2	2
	Requirement Planning issues	1	2	2
	Types of requirements (derived, explicit, decomposed)	1	2	2
	Software requirement definition, benefits, and risks of prototyping	1	2	2
	Requirements Management issues (baselines, traceability, tool support, life cycle requirements variance)	1	2	2
	Requirements/COTS issues	1	2	2
	Critical measures of effectiveness for operational issues and criteria	1	2	2
Software Reviews & Audits				
	Government management of reviews and audit process	1	2	2
	High interest Software issues and their indicators	1	2	2
	Critical Software life cycle reviews	1	2	2
	Key Software review questions and data	1	2	2
	Entrance & Exit Criteria	1	2	2
	Software review relationship to system reviews	1	2	2
Software Reuse				
	Software Architecture/reuse relationship	1	2	3
	Risk mitigation through reuse	1	2	3
	Reuse guidance	1	2	3
	Domain specific reuse paradigm	1	2	3
	Existing Reuse repositories	1	2	3
	Contracting mechanisms for reuse	1	2	3
	Impact of Open Systems on software reuse	1	2	3
	COTS/Reuse Issues	1	2	3
	Portability, through platform independence	1	2	3
Software Acquisition Risk Management				
	Software Risk Analysis	0	2	2
	Software Risk management issues (planning, etc.)	0	2	2
	Varying risk profile through life cycle	0	2	2
	Organizational risk mitigation entities (SEMP, RMWG, TIWG, CRWG, CRLCMP, IPT's, etc.)	0	2	2
	Risk Management guidance	0	2	2
	Domain Competent Work Force	0	2	2
Software Security				
	Software security definition	0	2	2
	Security Risk Management	0	2	2
	Software security guidance (regulations, standards, "orange book")	0	2	2
	System Certification	0	2	2
	Contemporary security developments	0	2	2
Software testing Issues				
	Software testing Phases (DT&E, F/OT&E)	0	0	0
	Appropriate Testing metrics (software maturity, error density)	0	0	0
	Type of Testing (unit, FOT, integration, DT/OT).	0	0	0
	Software integration testing issues	0	0	0
	Sufficient software testing	0	0	0
	Test and Evaluation Master Plan relationship to Testing	0	0	0
	High Integrity Systems	0	0	0
	Identification of Testing Risks	0	0	0
Emerging issues & Technologies				
	Joint Technical Architecture (JTA)	1	2	2
	Domain & product line engineering	1	2	2
	Software technology state of the art	1	2	2

G.6 Legal Competencies

Defining Rights In Intellectual Property Under Government Procurement Contracts
Principles of Patent law
Define Types of Patents
Discuss Bayh-Dole Act and Implementing Executive Orders
Review Principles of Copyright Law
Discuss Exclusive Rights vs. Limitation on Rights
Discuss Copyrights under Government Contracts
Principles of Trademark Law
Consideration arising from E-Commerce
Rights in Technical Data and Computer Software in Government Contracts
Definitions of technical data and computer software
Explain Regulatory Revisions and Frameworks
Explain Unlimited Rights
Define Limited Rights in Technical Data
Define Government Purpose License rights and Government Purpose Rights
Define non-standard rights
Software Escrow
Define different types of software agreements
Software Escrow benefits and concerns for the Government
Software Escrow benefits and concerns for the Developer
Software Escrow benefits and concerns for the Prime Contractor
Discuss common pitfalls for of SW escrow agreements
Intellectual Property Rights under CRADA's
Define CRADA's
Discuss CRADA marketing considerations
Enforcing Intellectual Property Rights Under Government Contracts
Claims against the Government for Infringement of Misappropriations
Discuss March-In Rights
Emerging Intellectual Property Issues
Relating to Home Land Security
Relating to Open Source Software
Licensing Software and Technology to the Federal Government
Discuss Issues Regarding Government Buys of Intellectual Property
Discuss Issues Regarding Government Licenses of Intellectual Property
Discuss Writing Standard License Agreements
Review Government Purpose & Nonstandard Rights 1988 and 1995 Regulations
Discuss SW rights resulting from Experimental Development
Discuss "Government Purpose Rights"
Discuss Remedies in Bid Protest Cases
Discuss Remedies for Licensing Problems
Discuss the breach of contract damages
Define limited rights in technical data
Define restricted rights in computer software
Discuss warranties and indemnifications

H Appendix
Common Software Staffing Problems

H.1 Ineffective Software Acquisition Management

H.1.1 Background

In an article on the "Blended Workforce," the authors indicated, "Federal workers frequently are co-located with contractor personnel in the same government offices, virtually indistinguishable, and often doing the same or similar work. The emergence of this blended workforce is the result of choice and necessity, outsourcing and privatization policy, and ad hoc acquisition decision-making. Because of efforts to downsize the federal workforce without similarly reducing its functions, a greater reliance on the private sector became inevitable. Nowhere are the potential problems more evident than in acquisition, where the number of federal workers has declined by nearly 50 percent since the mid-1990s and the workload has increased in dollar value and complexity. Recent contracting problems could be due in part to scarce acquisition officials and oversight. Government must develop an experienced, skilled, and balanced workforce in order to improve acquisition practices and integrity." [1] In a recent study, "Strategic Plan, 2007-2012" (GAO-07-1SP),[2] the Government Accountability Office observed: "Greater reliance on third parties…calls for an acquisition process based on realistic and well-defined requirements and contract terms that reflect a careful balancing of risks between the government and its contractors, as well as a skilled acquisition workforce capable of planning, negotiating and managing increasingly complex contracts."

In the June 2006 CrossTalk, Timothy K. Perkins of the Software Technology Support Center points out an issue we have seen with numerous Navy programs in his article titled, "Knowledge: The Core Problem of Project Failure."[3] Mr. Perkins indicates, "Having led and participated in more than 10 Independent Expert Program Reviews (IEPRs) for the Software Technology Support Center and the Tri-Service Assessment Office, and having spent my military career as a project/program manager, several individuals have asked if there is a common thread among programs or projects that are having difficulty. The answer is yes. Some expect the thread to be project planning, others risk management, and others expect one of the other project management themes. However, the root causes can be reduced to two issues: either project managers do not have the knowledge they need, or they do not properly apply the knowledge they have."

[1] Bednar, Richard J. and Gary P. Quigley. "The Blended Workforce." 9 May 2007.
[2] U.S GAO. "DoD Acquisition Outcomes: A Case for Change," GAO-06-257T. GAO, 2006. http://www.gao.gov/new.items/d06257t.pdf
[3] Perkins, Timothy K. "Knowledge: The Core Problem of Project Failure." CrossTalk, The Journal of Defense Software Engineering. June 2006.

In the June 2006 CrossTalk article "Social and Technical Reasons for Software Project Failures,"[4] Capers Jones also pointed out something that we have seen over and over with Navy programs. Mr. Jones says, "One of the most common sources of friction between corporate executives and software managers is the social issue that software project status reports are not accurate or believable. In case after case, monthly status reports are optimistic that all is on schedule and under control until shortly before the planned delivery when it is suddenly revealed that everything was not under control and another six months may be needed. What has long been troubling about software project status reporting is the fact that this key activity is severely underreported in software management literature. It is also undersupported in terms of available tools and methods. The situation of ambiguous and inadequate status reporting was common even in the days of the *waterfall model* of software development. Inaccurate reporting is even more common in the modern era where the *spiral model* and other alternatives such as *agile methods* and the object-oriented paradigm are supplanting traditional methods. The reason is that these nonlinear software development methods do not have the same precision in completing milestones as did the older linear software methodologies. The root cause of inaccurate status reporting is that Program Managers (PMs) are simply not trained to carry out this important activity. Surprisingly, neither universities nor many in-house management training programs deal with status reporting."

H.1.2 Findings

There is no formal or established career path for software acquisition professionals, which would provide trained staff to fill the senior level positions in the program offices.

H.2 Immature Software Acquisition Processes

H.2.1 Background

Many Navy teams have underestimated the budget and schedule needed to establish mature software teams and to train them to be smart buyers. When there are significant budget cuts as with the Global War on Terror, training and process improvement seem to be at the top of the list.

Analysis has shown that DoD tends to award contracts for systems to companies good at hardware, but not known nationally for their software development. In CrossTalk Magazine on software acquisition in the Army, Edgar Dalrymple, the Program Manager for the Future Combat Systems Brigade Combat Team and Associate Director of Software and Distributed Systems, when answering a question on making one change in the way the government procures software, said, "The government, at least the Army, needs to stop buying software exclusively from the traditional defense contracting base. These companies have the overhead costs of manufacturing companies, yet software development should carry a far smaller overhead burden. Most defense contractors are still managed by manufacturing engineers or business managers." [5]

Analyses performed by Navy Subject Matter Experts (SME) in evaluating the executability of the software program have confirmed Mr. Dalrymple's concern. The Navy found that many of the companies awarded large contracts underbid the amount of effort that needs to be applied to software requirements engineering for largely two reasons. One, they bid it like producing hardware which in many cases is easier to build to

[4] Jones, Capers. "Social and Technical Reasons for Software Project Failures©." CrossTalk, The Journal of Defense Software Engineering. June 2006.
[5] Starrett, Elizabeth. "Software Acquisition in the Army." CrossTalk, The Journal of Defense Software Engineering. May 2007.

the specifications. Two, for new contracts they build their software teams from multiple sources some of which do not have experience in building military systems. While the division bidding the contract may have a CMMI® level 3 to 5 rating, the teams are initially working at Level 1. These teams do not have the rigor to their requirements engineering at the beginning because they have yet to become a cohesive team. As such we begin on the wrong foot in two ways. The estimates for cost and schedule do not reflect quality requirements development and analysis and the personnel are forced to meet unrealistic and unexecutable plans for the requirements phase. Sometimes this is recognized at the Systems Requirements Review, meaning that software requirements are already in the reactive/catch-up mode. With an emphasis on Earned Value Management, they become schedule driven and enter the requirements reviews with immature products.

H.2.2 Findings

Government proposal teams with little experience in process maturity/capability too often developed Requests for Proposals (RFPs). To complicate matters, contracting personnel preparing contract documentation were inexperienced and lacked training in how to prepare an RFP package (terms & conditions, data item descriptions, integrated master schedule, etc.) for a software intensive system and then to evaluate proposals received. Acquirers (Program Executive Offices (PEO) and Program Offices) tended to be under-funded, untrained, or inexperienced in monitoring high capability management or systems/software engineering processes of suppliers with high process maturity. Acquirers were not requiring the delivery of measurement data that objectively showed progress to the contract.

H.3 Ineffective Software Requirements Management

H.3.1 Background

In testimony before the U.S. Senate's Committee on Armed Services Subcommittee on AirLand, the Government Accountability Office (GAO) reported, "Our work has shown that DoD's requirements process generates more demand for new programs than fiscal resources can support. DoD compounds the problem by approving so many highly complex and interdependent programs. Moreover, once a program is approved, requirements can be added along the way that increases costs and risks." [6] In an article for CrossTalk Magazine, Capers Jones highlights two of the five major root causes for software project failures as inaccurate estimating and schedule planning are related to requirements engineering. They are "Formal estimates are demanded before requirements fully defined" and "New requirements are added, but the original estimate cannot be changed." [7] It seems that added requirements are an issue that needs to be managed. More importantly, we need to fully define the requirements at the beginning of the program and ensure our cost and schedule estimates allow for changes in requirements up to a certain point in the acquisition cycle.

The GAO report cited above also stated, "To understand why these problems persist, we must look not just at the product development process but at the underlying requirements and budgeting processes to define problems and find solutions." Capers Jones point out, "average change rates of 2 percent per calendar month indicate that the methods used for gathering and analyzing the initial requirements are inadequate and should be improved." Both Capers Jones and the GAO point a finger at the requirements processes.

[6] U.S. GAO. "DoD Acquisition Outcomes: A Case for Change."
[7] Jones, Capers.

Recently the VH-71 Presidential Helo Program was identified as having requirements management issues. “Typically Defense Department programs have aggressive schedules, but the presidential helicopter is even more aggressive than we are used to seeing,” Paul Francis, director of acquisition and sourcing management for the Government Accountability Office, said in an interview.[8] Early on, the Navy and Lockheed Martin “didn't seem to be on the same page in terms of what the requirements were and what exactly Lockheed was required to deliver,” Francis said. “What we are seeing is that they can't deliver the aircraft with those capabilities in that amount of time at those costs.”

H.3.2 Findings

There was insufficient/inadequate training or lack of process application in all aspects of requirements disciplines. In addition, the Requirements Manager was not a critical position/function in the Program Office or among the career development for Acquisition Professionals. Poor communications with stakeholders during requirements elicitation, analysis, and verification, and the development life cycle was uncovered as well.

H.4 High Personnel Turnover in Acquiring Organizations

H.4.1 Background

The Navy’s most experienced software acquisition professionals retiring and in some cases leaving the Navy either to work another agency, or to work private industry, or academia. Workforce shaping initiatives and the low level of recruitment in the software area over the last 10-15 years has caused a shortage of qualified personnel to serve as software leads. This was supported in the Blended Workforce article.[9] The highly experienced software acquisition professionals have been forced to work multiple programs, not giving any one program sufficient time to ensure efficient/effective software acquisition. This contention for the “experts” has caused them to become overworked and more likely to leave government service when eligible for retirement. In some cases software personnel are leaving their functional area for promotions or to take jobs where they can spend more time with their family (less travel).

Because of the high demand, many of the software acquisition professionals are being recruited away from programs and the position cannot be filled when they leave. Some programs are hanging onto people as they develop their talents and keeping the person “stove-piped” in a software position. As result they do not get the diverse training that a software lead needs to be effective. In some cases the software professional just wants to build software and has no desire learn about acquisition or managing a team. In essence, the Navy has not been building the software leaders of the future very well and now have a gap.

The conversion of government retirement from Civil Service Retirement System (CSRS) to Federal Employees Retirement System (FERS) has made the software professional more transportable from job to job. While the intent may have been for government employees to have the ability to go into private industry to gain experience and then return to government service, few return to the government after being in private industry. The CSRS software professionals are retiring or will be within the next 5-10 years.

[8] Capaccio, Tony, Bloomberg News. “Large Cost Overrun Likely In Lockheed Helicopter Contract.” Washington Post, 17 May 2007.
[9] Bednar, Richard.

H.4.2 Findings

The program's staffing requirements were not planned to meet knowledge, skill, and experience requirements needed to perform the functions within the office. Program staffs tend to be created around budget constraints, traditional staffing structure, and availabilities. New members of the staff are assigned to a program, trained, and then too often move to their next job. Rotation cycles of key military personnel (especially those in leadership positions) impede consistency and continuity of program management.

H.5 Unrealistic Cost and Schedule Estimates

H.5.1 Background

In his article for the June 2006 CrossTalk Magazine, Capers Jones point out, "The fifth and last of the major estimating issues – *conservative estimates may be overruled and replaced by aggressive estimates* – is the rejection of conservative or accurate cost estimates and development schedules by clients or top executives. The conservative estimates are replaced by more aggressive estimates that are based on business needs rather than on the capabilities of the team to deliver. For some government projects, schedules may be mandated by Congress or by some outside authority. There is no easy solution for such cases."[10] This occurs because Program Managers and decision makers do not have sufficient knowledge of software development to recognize the validity of the conservative estimates. In addition, in many of the Navy's programs, historical data just doesn't exist for much of our work, which is often unique in domain, application, etc. We have not necessarily been rigorous in using something like a Delphi method to develop a more realistic program estimate.

In the Capers Jones article, he also pointed out, "Since both corporate executives and software managers find estimating to be an area of high risk, what are the factors triggering software cost estimating problems? From analysis and discussions of estimating issues with several hundred managers and executives in more than 75 companies between 1995 and 2006, the following were found to be the major root causes of cost estimating problems:

- Formal estimates are demanded before requirements are fully defined;
- Historical data is seldom available for calibration of estimates;
- New requirements are added, but the original estimate cannot be changed;
- Modern estimating tools are not always utilized on major software projects; and
- Conservative estimates may be overruled and replaced by aggressive estimates.

The most common reason for schedule slippages, cost overruns, and outright cancellation of major systems is that they contain too many bugs or defects to operate successfully."[11]

H.5.2 Findings

Program office personnel lack the expertise (i.e., training, education, tools, and experience) to accurately predict software cost and schedule and to understand and manage the risks associated with those

[10] Jones, Capers.
[11] Jones, Capers.

estimates throughout the software life cycle. Program office personnel, including higher-level management, are not trained to manage and track cost and schedule baselines. Cost department analysts with software experience to support the development of the estimates and the review of the accuracy at regular intervals are not always assigned to programs. Conservative estimates developed by software professionals are overruled by the program office and replaced by aggressive or success-oriented estimates.

H.6 Inconsistent Earned Value Management

H.6.1 Background

Earned Value Management (EVM) is inconsistently applied to software development efforts and often implemented incorrectly. A useful Earned Value Management System (EVMS) is closely tied to good estimating and accurate Work Breakdown Structures (WBS), yet EVMS reporting is not enough to meet management information needs in most software development programs. Unlike hardware, software can appear to meet critical milestones by delivering limited functionality with misleading cost and schedule status. A trained, educated, and experienced supplier and program office staff is required to properly structure a quality set of program measures, including EVM, for the software portion of the program. Guidance and lessons learned in this area are not as prevalent as other software acquisition issues.

H.6.2 Findings

Contracts are inconsistent in the variety of submittals, of quality metrics, and in any attempted EVMS. Even if contracts were better, program office personnel are not trained to structure a quality EVMS for software development. Process assets and metrics for managing cost and schedule baselines are lacking (guidance, lessons learned, standard measures, and training).

H.7 Ignored Best Practices and Lessons Learned

H.7.1 Background

Few programs use the program management and systems/software engineering best practices and lessons learned that are readily available and have been accepted worldwide as industry standards. In a community with a proliferation of policy documents, handbooks, and guidebooks, programs have “too many choices” for guidance, so they rarely use any of them.

H.7.2 Findings

Program staff rarely include personnel with expertise in software acquisition best practices, lessons learned, and related topics. Those experts who are available to provide support are spread thin and tend to migrate to the acquisition category (ACAT) I programs due to the availability of funding. There is a pervasive lack of process awareness in the Navy software acquisition community made worse by a lack of knowing how, or when, to ask for help and from whom (non-attributionally). Programs have insufficient resources to track all policies. Program personnel are not assigned to track, interpret, or advise on implementation of policies.

References

Bednar, Richard J. and Gary P. Quigley. "The Blended Workforce." 9 May 2007.

Capaccio, Tony, Bloomberg News. "Large Cost Overrun Likely In Lockheed Helicopter Contract." Washington Post, 17 May 2007.

Jones, Capers. "Social and Technical Reasons for Software Project Failures©." CrossTalk, The Journal of Defense Software Engineering. June 2006.

Perkins, Timothy K. "Knowledge: The Core Problem of Project Failure." CrossTalk, The Journal of Defense Software Engineering. June 2006.

Starrett, Elizabeth. "Software Acquisition in the Army." CrossTalk, The Journal of Defense Software Engineering. May 2007.

U.S. GAO. "DoD Acquisition Outcomes: A Case for Change," GAO-06-257T. GAO, 2006. <http://www.gao.gov/new.items/d06257t.pdf >.

This Page Intentionally Left Blank

Appendix

Functional Assignment Matrix

This matrix lists each of the core Department of Navy (DoN) acquisition disciplines, the critical billet/positions most often associated with each of these disciplines, and the primary functions performed by each position. This matrix may be used to assess staff coverage, especially in areas that are critical to program success, and to help in assessing overall staff risk exposure at program initiation and thereafter. See Chapter 4 for a discussion of disciplines and staff tailorability.

DISCIPLINE ➔	PROGRAM MANAGEMENT				SYSTEMS ENGINEERING				SOFTWARE ENGINEERING				LOGISTICS		TEST & EVALUATION			CONTRACTS			LEGAL	
NOMINAL POSITION TITLE ➔ FUNCTION ➔	Program Manager	Risk Management Manager	Project (Sub-system) Manager	Process Compliance Manager	Requirements Manager	Chief Systems Engineer	Systems Process Manager	Sub-System Lead Engineer	Software Architect	Chief Software Engineer	Software Process Manager	Sub-System Software Engineer	Logistics Manager		Lead T&E Engineer	DT Lead Engineer	OT Lead Engineer	Procurement Contracting Officer	Administrative Contracting Officer	In-Plant Program Rep or DCMAO	Counsel	
Planning																						
Task Planning (WBS, SMP/PP)	X		X																			
Acquisition Planning	X												X					X				
Progress/Earned Value: Planning			X			X		X				X						X				
Acquisition Plan	X																					
All Test Planning															X							
Communications						X																
Cost estimating			X																			
Develop Software Development Plan												X										
Develop the SEP						X																
Ensure tech reviews scheduled and executed				X																		
Establish Data Repository				X																		
File Nomenclature				X																		
IBR lead			X																			
Identify and obtain in-house and CSS support	X																					
Identify required metrics				X																		
IPT Organizational Structure								X														
Master schedule	X																					
Monitoring and Controlling the Plan				X																		
Overall Cost	X																					
Overall Schedule	X																					

DISCIPLINE ➔	PROGRAM MANAGEMENT				SYSTEMS ENGINEERING				SOFTWARE ENGINEERING				LOGISTICS		TEST & EVALUATION			CONTRACTS			LEGAL	
NOMINAL POSITION TITLE ➔ / FUNCTION ⬇	Program Manager	Risk Management Manager	Project (Sub-system) Manager	Process Compliance Manager	Requirements Manager	Chief Systems Engineer	Systems Process Manager	Sub-System Lead Engineer	Software Architect	Chief Software Engineer	Software Process Manager	Sub-System Software Engineer	Logistics Manager		Lead T&E Engineer	DT Lead Engineer	OT Lead Engineer	Procurement Contracting Officer	Administrative Contracting Officer	In-Plant Program Rep or DCMAO	Counsel	
Provide direction as to future goals of the platforms - (e.g. commonality of software, NCW, OA, IA)									X													
Provide the technical expertise to acquire and monitor the hardware development.						X																
Provides the technical expertise to acquire and monitor software intensive systems										X												
Software Acquisition Process			X																			
Software Development Life Cycle									X													
SOW development												X										
Staff Planning			X																			
Stakeholder ID and Involvement				X																		
Risk																						
Software Risk Identification										X												
Software Risk Monitoring										X												
Facilitate Risk Identification Workshops		X																				
Risk modeler		X																				
Risk process training		X																				
Risk Register		X																				
Establish/Chair Risk Review Board		X																				
Prepare Risk Status Reports						X				X												
Mitigation Strategies						X				X												
Risk Tool Training		X																				
Management																						
Conducting Milestone Reviews			X																			

DISCIPLINE ➔	PROGRAM MANAGEMENT				SYSTEMS ENGINEERING				SOFTWARE ENGINEERING				LOGISTICS		TEST & EVALUATION			CONTRACTS			LEGAL	
NOMINAL POSITION TITLE ➔ FUNCTION ⬇	Program Manager	Risk Management Manager	Project (Sub-system) Manager	Process Compliance Manager	Requirements Manager	Chief Systems Engineer	Systems Process Manager	Sub-System Lead Engineer	Software Architect	Chief Software Engineer	Software Process Manager	Sub-System Software Engineer	Logistics Manager		Lead T&E Engineer	DT Lead Engineer	OT Lead Engineer	Procurement Contracting Officer	Administrative Contracting Officer	In-Plant Program Rep or DCMAO	Counsel	
Staffing/Team Selection			X																			
Project Process Monitoring				X																		
Acquisition Monitoring	X																					
Progress/Earned Value: Tracking																		X				
Project Status Monitoring/Reporting			X																			
Size and Complexity Estimation										X												
Software Cost and Schedule Estimation																						
Measurement for Project Management												X										
Measurement for Improvement				X																		
Software Assurance												X										
Process Improvement Leadership				X																		
Award Fee assessment																		X				
Coordination of GAO reporting	X																					
Customer advocate		X																				
Customer interface	X																					
Earned Value Management (software)										X												
IPAR/CPAR assessment																						
Keeper of technical budgets								X														
Manage and track cost baselines																		X				
Manage and track schedule baselines																		X				
Measurement and Analysis		X																				
Monitoring of contractor technical/financial progress																		X				
Program Management Reviews			X																			

DISCIPLINE ➔	PROGRAM MANAGEMENT				SYSTEMS ENGINEERING				SOFTWARE ENGINEERING				LOGISTICS		TEST & EVALUATION			CONTRACTS			LEGAL	
NOMINAL POSITION TITLE ➔ FUNCTION ⬇	Program Manager	Risk Management Manager	Project (Sub-system) Manager	Process Compliance Manager	Requirements Manager	Chief Systems Engineer	Systems Process Manager	Sub-System Lead Engineer	Software Architect	Chief Software Engineer	Software Process Manager	Sub-System Software Engineer	Logistics Manager		Lead T&E Engineer	DT Lead Engineer	OT Lead Engineer	Procurement Contracting Officer	Administrative Contracting Officer	In-Plant Program Rep or DCMAO	Counsel	
Software Engineering Management										X												
Systems Engineering Management						X																
Software Acquisition																		X				
Software cost estimate review												X										
Software Measurement										X												
Stakeholder Coordination	X																					
Systems Measurement			X																			
Requirements																						
Requirements Management					X																	
Requirements and Operations Concepts Definition, Analysis, and Validation									X													
Intergroup Collaboration/Liaison					X																	
COTS/GOTS Evaluation										X												
Decision Analysis									X													
Tradeoffs, Tailoring, and Prioritizing										X												
System Architecture									X													
Prototyping, Analysis, Simulation, and Testing Approaches											X											
Software Technology Awareness									X													
4.0P requirements					X																	
Allocated Baseline					X																	
Certification requirements (e.g. Lk 16)																						
Derive requirements from customer needs					X																	
Ensure bi-directional traceability of requirements					X																	

DISCIPLINE ➔	PROGRAM MANAGEMENT				SYSTEMS ENGINEERING				SOFTWARE ENGINEERING				LOGISTICS		TEST & EVALUATION			CONTRACTS			LEGAL	
NOMINAL POSITION TITLE ➔ FUNCTION ⬇	Program Manager	Risk Management Manager	Project (Sub-system) Manager	Process Compliance Manager	Requirements Manager	Chief Systems Engineer	Systems Process Manager	Sub-System Lead Engineer	Software Architect	Chief Software Engineer	Software Process Manager	Sub-System Software Engineer	Logistics Manager		Lead T&E Engineer	DT Lead Engineer	OT Lead Engineer	Procurement Contracting Officer	Administrative Contracting Officer	In-Plant Program Rep or DCMAO	Counsel	
Functional Baseline					X																	
IA requirements									X													
IA/AT requirements									X													
Identifies software support requirements													X									
Identify external product interfaces and ensure reflected in architecture																						
Logistics Support Requirements													X									
Requirements Owner, allocator and maintainer					X																	
System Safety Requirements						X																
Software acquisition requirements																		X				
Requirements Repository					X																	
Change Review Board					X																	
Contractual requirements																		X				
Software Configuration Management																						
Configuration Management Planning											X											
CM Identification												X										
CM Change Control												X										
Configuration Status Accounting and Auditing											X											
Configuration Management												X										
Configuration Management Plan											X											
Identify configuration items												X										
Software Quality Assurance																						

DISCIPLINE ➔	PROGRAM MANAGEMENT				SYSTEMS ENGINEERING				SOFTWARE ENGINEERING				LOGISTICS		TEST & EVALUATION			CONTRACTS			LEGAL	
NOMINAL POSITION TITLE ➔ FUNCTION 🡇	Program Manager	Risk Management Manager	Project (Sub-system) Manager	Process Compliance Manager	Requirements Manager	Chief Systems Engineer	Systems Process Manager	Sub-System Lead Engineer	Software Architect	Chief Software Engineer	Software Process Manager	Sub-System Software Engineer	Logistics Manager		Lead T&E Engineer	DT Lead Engineer	OT Lead Engineer	Procurement Contracting Officer	Administrative Contracting Officer	In-Plant Program Rep or DCMAO	Counsel	
Process Compliance											X											
QA Plan												X										
Quality evaluation															X							
Process																						
Process Engineering				X																		
Process Deployment				X																		
Capturing Lessons Learned				X																		
Life Cycle Models/Process Tailoring											X											
Ensure appropriate process related CDRLs are included in the contract																		X				
Establish and Maintain Organizational Process Assets				X																		
Process Asset Library							X															
Process Compliance				X																		
Process Improvement for the program office	X																					
Process Improvement models, methods and tools										X												
Software Training																						
Maintain skills matrix										X												
Plan & coordinate training											X											
Facilitate Workshops											X											
Contracts																						
Tailor and manage the contract																		X				

DISCIPLINE ➔	PROGRAM MANAGEMENT				SYSTEMS ENGINEERING				SOFTWARE ENGINEERING				LOGISTICS		TEST & EVALUATION			CONTRACTS			LEGAL	
NOMINAL POSITION TITLE ➔ FUNCTION ⬇	Program Manager	Risk Management Manager	Project (Sub-system) Manager	Process Compliance Manager	Requirements Manager	Chief Systems Engineer	Systems Process Manager	Sub-System Lead Engineer	Software Architect	Chief Software Engineer	Software Process Manager	Sub-System Software Engineer	Logistics Manager		Lead T&E Engineer	DT Lead Engineer	OT Lead Engineer	Procurement Contracting Officer	Administrative Contracting Officer	In-Plant Program Rep or DCMAO	Counsel	
Assure that team members meet CDRL review deadlines				X																		
Contract award, delivery orders, Contract mods, etc																		X				
Contract performance evaluation (CPARS)																		X				
Contract software requirements (e.g. 12207, SDP)																		X				
Process and Performance CDRLS				X																		
Data rights																		X				
Documentation requirements																		X				
Engineering																						
Software Architecture									X													
Software Design Methods										X												
COTS/GOTS-Based Design										X												
Software Reliability and Safety												X										
Software Implementation												X										
Programming Languages									X													
Reusable Software Development										X												
Unit Testing												X										
Software and System Integration										X												
Software Development Environments and Facilities												X										
Plan engineering activities						X																

DISCIPLINE ➔	PROGRAM MANAGEMENT				SYSTEMS ENGINEERING				SOFTWARE ENGINEERING				LOGISTICS		TEST & EVALUATION			CONTRACTS			LEGAL	
NOMINAL POSITION TITLE ➔ FUNCTION 🡻	Program Manager	Risk Management Manager	Project (Sub-system) Manager	Process Compliance Manager	Requirements Manager	Chief Systems Engineer	Systems Process Manager	Sub-System Lead Engineer	Software Architect	Chief Software Engineer	Software Process Manager	Sub-System Software Engineer	Logistics Manager		Lead T&E Engineer	DT Lead Engineer	OT Lead Engineer	Procurement Contracting Officer	Administrative Contracting Officer	In-Plant Program Rep or DCMAO	Counsel	
Define Technical Performance Measurement (TPM) requirements									X													
Ensure appropriate software related CDRLs are included in the contract																		X				
Ensure commonality and reuse of software across PMA platforms/efforts.									X													
Ensure that the process for developing all software are consistent (e.g. SDP is acceptable, 12207).											X											
Integration										X												
Owner of internal interfaces									X													
Performance modeler									X													
Provide lessons learned from common PMA platforms.											X											
Review of SCM/SQA plans											X											
Software Engineering processes											X											
Software integration										X												
Software Safety												X										
System Architect						X																
System modeler and simulator							X															
Systems Engineering processes						X																
Systems Integration						X																
Systems Integrator						X																
Understand how all the pieces interact (e.g. hardware, software, airframe, certification, IA, OA)						X																

DISCIPLINE ➔	PROGRAM MANAGEMENT				SYSTEMS ENGINEERING				SOFTWARE ENGINEERING				LOGISTICS		TEST & EVALUATION			CONTRACTS			LEGAL	
NOMINAL POSITION TITLE ➔ FUNCTION ⬇	Program Manager	Risk Management Manager	Project (Sub-system) Manager	Process Compliance Manager	Requirements Manager	Chief Systems Engineer	Systems Process Manager	Sub-System Lead Engineer	Software Architect	Chief Software Engineer	Software Process Manager	Sub-System Software Engineer	Logistics Manager		Lead T&E Engineer	DT Lead Engineer	OT Lead Engineer	Procurement Contracting Officer	Administrative Contracting Officer	In-Plant Program Rep or DCMAO	Counsel	
Ensure hardware development/availability and software development plays together.						X																
Logistics																						
Development Software Support (PDSS) Plan													X									
Performance Based approach													X									
Acquisition support													X									
Disposal													X									
Distribution													X									
Life cycle software support plan													X									
Logistics													X									
Maintenance													X									
Operations													X									
Operator training manuals													X									
Sustainability													X									
User Manuals													X									
Test and Evaluation																						
Requirements Management					X																	
Requirements and Operations Concepts Definition, Analysis, and Validation					X																	
Intergroup Collaboration/Liaison							X															
COTS/GOTS Evaluation										X												
Decision Analysis									X													
Owner of system tests															X							
Test Plan															X							

DISCIPLINE ➔	PROGRAM MANAGEMENT				SYSTEMS ENGINEERING				SOFTWARE ENGINEERING				LOGISTICS		TEST & EVALUATION			CONTRACTS			LEGAL	
NOMINAL POSITION TITLE ➔ FUNCTION ↓	Program Manager	Risk Management Manager	Project (Sub-system) Manager	Process Compliance Manager	Requirements Manager	Chief Systems Engineer	Systems Process Manager	Sub-System Lead Engineer	Software Architect	Chief Software Engineer	Software Process Manager	Sub-System Software Engineer	Logistics Manager		Lead T&E Engineer	DT Lead Engineer	OT Lead Engineer	Procurement Contracting Officer	Administrative Contracting Officer	In-Plant Program Rep or DCMAO	Counsel	
Manage Test activities															X							
In-service tests															X							
Operational tests															X							
Developmental tests															X							
Prototypes												X										
System sign-off						X																
Test Resource ID and Allocation															X							
Validation and Verification												X										
Business/Financial																						
Business Analyst																		X				
Cost accounts, Cost Performance Report and IBR																		X				
Ensure availability of funds																		X				
Funds acceptance, distribution, tracking	X																					
GAO required report for software										X												

This Page Intentionally Left Blank

J

Appendix
Comparability of Predictive and Adaptive Techniques

Applied Development Technique	Technique Definition	Basic Feature(s)	Basic Advantages	Basic Disadvantages
Predictive	Software development effort based upon a predictive model that was calibrated, to some extent, with actual results.	Stable requirements and similarity experience basis. Sometimes the design is already developed.	Estimates are usually based upon actual cost encountered in a previous endeavor. Does not necessarily require the expense of creative people.	Not accurate unless external conditions are known perfectly.
Sequential Coding	Sequential lines of code in a high-level procedural language like COBOL, FORTRAN, C, etc.	Program is executed sequentially throughout elements. These programs are well represented by flowcharts.	Coding techniques are mature; libraries are well developed and known. A mature software developer can facilitate reuse by adding functionality to library.	Software developed by sequential techniques are not well suited for event driven functionality applications.
Object Oriented Programming (OOP)	OOP centers on the development of small, reusable program routines (modules) that are linked together and to other objects to form a program.	Event driven objects are developed using a hierarchal class structure. Each class features parent-child relationships that facilitate reuse. Child objects understand codes inherent to their parents, plus their own unique code.	Suitable for event driven functionality. Code reuse facilitated by inheritance and upgrades to short modular code modules in model, view, or controller paradigms. Investment of usable repository promotes resue and serious reductions in development times. Maintenance of that software will be much simpler and much less error prone as consequence of up front design investment in careful designs.	Lack of standardized coding procedures. Requires skilled developers to exploit the power of OOP. It is possible to write sequential code in an OOP program. Also useful in adaptive category when exploiting reuse.

Table J-1. Comparability for Some Popular Existing Predictive Software Development Techniques[1]

[1] Davis, Noopur, and Julia Mullaney. The Team Software Process^{SM} (TSP^{SM}) in Practice: A Summary of Recent Results. CMU/SEI-2003-TR-014. Pittsburgh: Carnegie Mellon University, Software Engineering Institute, 2003.

Applied Development Technique	Technique Definition	Basic Feature(s)	Basic Advantages	Basic Disadvantages
Rapid Application Development	A short time duration development technique with compressed phases of initiation, development, and implementation.	Requirements identification phase is quick and the capabilities should remain unchanged throughout the development.	Good for design or redevelopment of low-complexity applications when selected as a predictive technique. Given acceptable risk, RAD may be used within development and design phases of another technique.	Bad for complex high risk applications when used for predictive cases. However, the method is fine when selected to preceede an another technique.
(CASE) Tools	Computer Aided Software Engineering Tools	Provides mechanisms for automated software development. For example a graphical user interface can be developed with CASE tools. When the result is right the code is auto generated.	Can increase productivity, reduce cost, and improve product quality.	Incompatibilities among vendors. High start-up costs. Requires management patience for long-term ROI. Requires similar security coordination and disciplined configuration management as other tools (see section 2.3 for I/A concerns).
Lean S/W Development	Lean Software development is a translation of lean manufacturing principles and practices to the software development domain. It can be applied both the predictive and adaptive techniques.	Lean SW development relies on a CMMI-like infrastructure for development and improvement (as so meets CMMI goals). Focus is on the core task of identifying and eliminating waste while driving key decisions outward in the development cycle.	Highly responsive to customer needs, and provides a best-in-class answer to requirements volatility issues	Often requires a facilitator to realize productive gains. Senior management support is a must.

Table J-1 (cont.). Comparability for Some Popular Existing Predictive Software Development Techniques

Applied Development Technique	Technique Definition	Basic Feature(s)	Basic Advantages	Basic Disadvantages
MDA	Model Driven Architecture is useful to exploit design patterns of modularity and reuse at the architectural level.	Models expressed in a well-defined notation. Organized around a set of models by imposing a series of transformations between models. Requires industry standards to provide openness to consumers, and fosters competition among vendors. Facilitates meaningful integration and transformation among models, and is the basis for automation. Requires industry standards to provide openness to consumers, and fosters competition among vendors.	MDD elements are easy to synthesize. Inherits reusable functionality from architectural design.	Requires an up front investment strategy. Management must be patient for a ROI beyond two-year scope of government procurement cycles. Requires similar security coordination and disciplined configuration management as other tools (see section 2.3 for I/A concerns).
MDD	Model Driven Development is useful to get the correct scope on total cost of ownership early in a programs life-cycle.	Useful for modeling systems and subsystems at a component level for requirements validation (i.e. to control requirements creep).	Up front programmatic risks mitigation. Applies to codes synthesized from MDA, and non-MDA paradigms. Sometimes supports auto code generation. Provides early feedback to the development team.	Can be used inappropriately if not properly configuration managed. Requires an investment strategy to get the models to the appropriate level of fidelity. Models need to be maintained and be consistent with implemented code.

Table J-1 (cont.). Comparability for Some Popular Existing Predictive Software Development Techniques

Applied Development Technique	Technique Definition	Basic Feature(s)	Basic Advantages	Basic Disadvantages
Adaptive	Employ incremental and /or iterative cycles intended to converge upon meeting capability requirements.	Develop Large and complex systems via incremental and iterative development through capability demonstrations.	Can converge upon meeting requirements while managing risk incrementally with assessment points. Process will converge upon what works for the team and the project. Useful in controlling unpredictable processes.	The cost of iterations necessary during the RDT&E program phases requires more front-end investment. Processes in the end phases may be completely different than processes initiated at startup.
Agile	Most agile class of methods attempt to minimize risk by developing software in short time boxes, called iterations, which typically last one to four weeks. Each iteration is like a miniature software project of its own, and includes all of the tasks necessary to release the mini-increment of new functionality: planning, requirements analysis, design, coding, testing, and documentation.[2]	Used to produce sufficiently advanced models. Also inherits adaptive characteristics.	Can converge upon meeting requirements while managing risk incrementally with assessment points. Life cycle costs are actually driven down, delivery time is shorter, and quality is higher.	May require non recurring investment on first implementation.

Table J-2. Comparability for Some Popular Existing Adaptive Software Development Techniques[2,]

[2] The Team Software Process^{SM} (TSP^{SM}) in Practice: A Summary of Recent Results.

Applied Development Technique	Technique Definition	Basic Feature(s)	Basic Advantages	Basic Disadvantages
Crystal Family	A technique that puts focus on roles of the development team (i.e. sponsors, designers, users, etc), and communication between the team members. The ideal team is 4 to 6 people.	Use of processes specifically tailored to the skills of the team. Select methods of appropriate weight and tailor them based on project size and criticality.	Exploits unique qualities from within the developer's organization. Usually successful with a small development team.	Does not accommodate for missing skill sets.
Dynamic Systems Development Method	The result of a United Kingdom Industry Consortium extracting best practices and mapping them into a framework.	Fix time and resources then develop the functionality while adjusting functionality with the design build iterations.	On time delivery of products. Empowered software development team build around user.	Not accommodative to requirements that are not flexible.
Extreme Programming	XP is a discipline of software development applicable to small teams who need to produce quick results in dynamic environments.	Assumes requirements are frozen then implements a cycle of planning, design, coding and testing. Good for rapid prototyping.	Customer focused and can produce successfully developed software despite vague or changing requirements. Life cycle costs are actually driven down, delivery time is shorter, and quality is higher.	Difficult to apply in organizations where the acquisition personnel (developers) are not co-located from the users.

Table J-2 (cont.). Comparability for Some Popular Existing Adaptive Software Development Techniques

Applied Development Technique	Technique Definition	Basic Feature(s)	Basic Advantages	Basic Disadvantages
Lean S/W Development	Lean Software development is a translation of lean manufacturing principles and practices to the software development domain. It can be applied both the predictive and adaptive techniques.	Lean SW development relies on a CMMI-like infrastructure for development and improvement (as so meets CMMI goals). Focus is on the core task of identifying and eliminating waste while driving key decisions outward in the development cycle.	Highly responsive to customer needs, and provides a best-in-class answer to requirements volatility issues	Often requires a facilitator to realize productive gains. Senior management support is a must.
Feature Driven Development	(FDD) is an iterative and incremental software development process. It is one of a number of Agile methods for developing software and forms part of the Agile Alliance.	Iterative development of business systems, and an instance of agile technique. FDD blends a number of industry-recognized best practices into a cohesive whole. These practices are all driven from a client-valued functionality perspective.	Frequent and tangible deliverables. Its main purpose is to deliver tangible, working software repeatedly in a timely manner.	The cost of iterations necessary during the RDT&E program phases requires more front-end investment.
Internet Speed Development	An Agile Software Development method using a combined spiral model/waterfall model with daily builds aimed at developing a product with high speed.	Closely aligned to agile development principles. Management oriented framework for fast releases based on time drivers, quality dependencies and process adjustments.	Copes well with fast changing requirements.	Iterations will increase the cost of RT&E program phases.

Table J-2 (cont.). Comparability for Some Popular Existing Adaptive Software Development Techniques

Applied Development Technique	Technique Definition	Basic Feature(s)	Basic Advantages	Basic Disadvantages
Object Oriented Programming (OOP)	OOP centers on the development of small, reusable program routines (modules) that are linked together and to other objects to form a program.	Event driven objects are developed using a hierarchal class structure. Each class features parent-child relationships that facilitate reuse. Child objects understand codes inherent to their parents, plus their own unique code.	Suitable for event driven functionality. Code reuse facilitated by inheritance and upgrades to short modular code modules in model, view, or controller paradigms. Investment of usable repository promotes resue and serious reductions in development times. Maintenance of that software will be much simpler and much less error prone as consequence of up front design investment in careful designs.	Lack of standardized coding procedures. Requires skilled developers to exploit the power of OOP. Also useful in adaptive category when exploiting reuse.
Scrum	Scrum is an agile, lightweight process that can be used to manage and control software and product development using iterative, incremental practices. Wrapping existing engineering practices, including Extreme Programming and RUP	Frequent management activity to identify deficiencies or impediments coupled with developer's selection of techniques. SCRUM is actually a daily mgt. technique applied in agile development efforts.	Prioritized backlog most important items get worked. Supports scalability concepts through OOP approach. It produces a potentially shippable set of functionality at the end of every iteration.	Requires a balance of management but not micro-management. Substantial front-end architectural preparation required. Some coders not comfortable with level of responsibility brought by scrum.

Table J-2 (cont.). Comparability for Some Popular Existing Adaptive Software Development Techniques

References

Davis, Noopur, and Julia Mullaney. The Team Software Process^{SM} (TSP^{SM}) in Practice: A Summary of Recent Results. CMU/SEI-2003-TR-014. Pittsburgh: Carnegie Mellon University, Software Engineering Institute, 2003.

This Page Intentionally Left Blank

Appendix K
Evaluating Tool Types

K.1 Configuration Management Tools

Configuration management tools can be effectively applied throughout the Software Development Life Cycle (SDLC). Configuration management tools should be interoperable with requirements management tools, for ease of tracing requirements to baselines. An acquiring agency can require that offerors describe their actual or intended processes and tools for configuration management.

K.1.1 What It Is

Software Configuration Management (SCM) is a methodology designed to:

- Identify software configuration items;
- Provide a change control process;
- Provide status accounting reports describing baselines, version numbers, open and closed trouble reports, related documentation; and
- Support configuration audits.

K.1.2 Why It Is Needed

The introduction to the IEEE "Standard for Software Configuration Management Plans" says this about SCM:

> SCM constitutes good engineering practice for all software projects, whether phased development, rapid prototyping, or ongoing maintenance. It enhances the reliability and quality of software by:
>
> - Providing structure for identifying and controlling documentation, code, interfaces, and databases to support all life-cycle phases;
> - Supporting a chosen development/maintenance methodology; and
> - Producing management and product information concerning the status of baselines, change control, tests, releases, audits, etc. [1]

[1] IEEE 828-1998: IEEE Standard for Software Configuration Management Plans: Institute of Electrical and Electronics Engineers, 1998.

K.1.3 How to Evaluate the Tool

Determine which of the following are important to the program and can be supported by a candidate tool. Does the SCM Tool:

- Identify functional, allocated, and product baseline configuration items?
- Relate functional, allocated, and product baseline configuration items?
- Associate configuration items with system requirements?
- Provide status reports describing software and documentation configuration items and related trouble reports and change requests?
- Support online status reports?
- Support online ad-hoc query?
- Provide help tools for users?
- Track all trouble reports by configuration item?
- Track all trouble reports by location (if multiple)?
- Allow online submission of trouble reports?
- Allow online submission of change requests?
- Track changes requested and approved for each configuration item?
- Allow users to submit change requests?
- Allow users to prioritize change requests?
- Simplify or automate change request notification to members of the configuration control board?
- Support input from subject matter experts (SME's)?
- Support cost/schedule impact input from SME's?
- Track prioritization changes recommended by configuration board after SME input?
- Track configuration control board recommended action for each change request (fix, defer, etc.)?
- Track configuration control board manager's decision for each change request?
- Track configuration control board recommendation for fixes and enhancements slated for next release?
- Track configuration items by date and location for fielded systems?

K.2 Requirements Definition and Management Tools

Software requirements management tools are essentially tools that track system requirements, derived requirements, and any changes. The acquirer can use the RFP to emphasize the importance of requirements traceability, and can evaluate the offeror's proposal for description of capability and process. A requirements tool that supports tracking projected costs for associated development can help a project or program manager to make tradeoff decisions (between enhancements and fixes) for future releases.

K.2.1 What It Is

Requirements management tools support a continuous process of tracking requirements to the capability provided and tested. Requirements tools can be used to show how a system's requirements map to mission and joint capabilities, as well as how the requirements are mapped into system design, test procedures, and baselined systems

K.2.2 Why It Is Needed

Rework can account for up to 40% or more of a development organization's total outlay - time and money. Correcting requirements defects can be 50 to 200 times more expensive after test or deployment than during the early design stages. By eliciting, specifying, analyzing, and validating requirements early, costly rework can be reduced later in the development lifecycle. SDLC tools that address collaborative software requirements definition and management are necessary for:

- Solidification of actual requirement;
- Avoiding requirement creep; and
- Traceability from requirements to functionality.

K.2.3 How to Evaluate the Tool

Evaluation should consider whether the software requirements definition and management tool address the following areas:

- Elicitation: Method(s) for –
  - Customizing templates or forms (or surveys) for requirements definition, which can include user scenarios in easily understandable forms.
- Specification: Method(s) for –
  - Documented requirements.
  - Traceability from top-level to derived requirements.
  - Defining a hierarchy of requirement types, attributes, and relevant data, and providing some traceability and reporting of status.
- Analysis: Method(s) for –
  - Capturing, displaying, and aggregating individual and group evaluation of derived requirements.
- Validation: Method(s) for –
  - Capturing requirements review, signoff, and creation of a baseline.
- Management: Method(s) for –
  - Tracking changes and associated approval.
  - Tracking implementation status, re-work, specific groups of critical requirements into configuration items, testing, and releases while producing measurements consistent with naval metrics policies.
  - Establishing processes for managing changes (requests, impact analysis, and communicating changes).
  - Focusing resources and project planning.
- Verification: Method(s) for –

- Tracing requirements to test plans, processes, and test results.

K.3 System Analysis and Modeling Tools

System analysis/modeling tools may be proposed or in use by a contractor. The acquirer will need to evaluate advantages described against costs for licenses, training, and required expertise. The return on investment for such tools may be difficult to assess, and a significant learning curve is often associated initially. The tools should be evaluated for use of interface standards (standard input and output formats to facilitate data sharing and also to make it possible to change to another tool or process if necessary. The products of the tool should be understandable to any stakeholders that should be involved in reviewing the design or process captured. If the user/stakeholder cannot walk through and validate the captured design, then the tool may be impeding common understanding and agreement of design decisions. Naval and DoD policies require certification of all models, simulators, and stimulators for which there are no further checks of their output or assessment "downstream." Use DODINST 5000.61[2] for this process.

K.3.1 What It Is

Modeling establishes a way to visualize the design and compare it against requirements before coding begins. Modeling involves describing desired features and operations in detail, including:

- Screen layouts;
- Business rules;
- Process diagrams;
- Pseudocode;
- Autocode; and
- Other documentation.

The Unified Modeling Language (UML) is common among developers for specifying, visualizing, and documenting software design. The UML diagrams may be difficult for non-developers, so this can be a major consideration if the design needs to be briefed or "walked through" with users or other stakeholders.

Autocode is currently poorly used and has led to serious safety anomalies. Best practices do exist for small uses in specific areas but not for safety or security requirements without significant additional requirements being made on the program by Technical Warrants.

K.3.2 Why It Is Needed

Acquirers will want to be familiar with any tool and product that is used to capture design. The more eyes on the design, the more errors will be caught at this early stage. Design errors that are not identified in design walk-throughs too often result in multiple seemingly unrelated problems, many of which will be caught only in the field, and the fixes to those problems will introduce more bugs. Thus the design tool, and ability to use it with all appropriate stakeholders during design walk-through, is key to reducing rework.

[2] DODINST 5000.61, 13MAY2003, "DoD Modeling and Simulation (M&S) Verification, Validation, and Accreditation (VV&A).

K.3.3 How to Evaluate the Tool

Evaluation should consider the following questions:

- Who the tool will serve, and obtain input?
- Which computing (host) platform(s) will be considered and which will ultimately be selected?
  - Will the software development be for Windows or Unix or both, etc?
  - On which platform will the development be conducted?
  - License cost.
- Does the modeling tool address the following areas?
  - Repository Support – Provides data sharing and concurrency control features.
  - HTML Documentation – Provides a static view of the object model that can be referred to quickly in a browser, without having to launch the modeling tool itself.
  - Round-Trip Engineering – The ability to both forward and reverse engineer source code.
  - Debugging – The ability to debug with either the UML (For UML tools) or with code
  - Data Modeling – Allows integration of data modeling facilities while also supporting the synchronization of data and object models after each iteration of design.
  - (For UML tools) Full UML Support – Diagrams which should be supported are the Use Case, Class, Collaboration, Sequence, Package, and State diagrams.
  - Versioning – Allows versions to be saved so that as subsequent iterations are created, previous versions are available for restore.
  - Scripting – Supports the generation of scripts for the purpose of directly accessing the object model to create other artifacts (i.e. metrics, reports, documentation).
  - Diagram Views w/Export – Facilitates customization of the view of a class and its details and the capability to export into common formats such as word or web page format.
  - Printing Support – Allows accurate renditions of large diagrams to be produced through multi-page printing.
  - Metrics – Provides feedback on the viability of a particular model.
- Does the system analysis tool address the following areas?
  - Database Schema;
  - Data Flow Diagrams;
  - Entity Relationship Diagrams;
  - Program Specifications;
  - User Documentation.

K.4 Development Tools

Numerous tools are available for use in development. The combination of tools will vary by type, size, scale, and complexity of a development effort. There is no single best solution, and many developer organizations will have established practices and tools that evolve over time. The developer's plan should include a number of risk-reducing steps within the development phase. Development tools are available to support these important steps of code inspection, code review, and unit testing. A Capability Maturity Model Integration (CMMI®) of Level 3 (or higher), or an equivalent standard software process, is an indication that the developer has the required experience and employs tools appropriately. Higher risk programs may stipulate higher levels of capability maturity.

K.4.1 What It Is

Development tools such as code generators, code inspection tools, debuggers, and tools accommodating test generation compliment implementation processes. Automatic Software Inspection (ASI) tools verify that the software is compliant with coding standards.

K.4.2 Why It Is Needed

Errors are always easier to isolate and cheaper to fix the earlier they are found. Development tools make it easier for the developer to move from design to code, track the progress of each code unit, check each unit for compliance with local and general coding standards, and support communications among the developers.

K.4.3 How to Evaluate the Tool

Note: This area is intended for the developer community; the acquirer will normally not be involved at this detail.

- Identify whether the tools full capabilities span the activities of multiple SDLC phases.
  - **Positive:** Provide integrated functionality.
  - **Negative:** Can be expensive.
- Code Generation
- Identify what role a Code Generator will play in code creation/revision.
  - Will manual coding also be needed?
  - Will integration of manual coding and auto generated code be necessary?
  - How will code debugging be conducted?
- Identify whether the tool is a stand-alone code generator.
  - **Positive:** Less expensive.
  - **Negative:** Can be effort intensive because of the need to move back and forth between modeling tool and code generator.
- Code Review/Software Inspection
- Identify whether code reviews/inspections are being done with an ASI tool.
  - **Positive:** The cost of a complete inspection becomes less prohibitive as a code base grows.
  - **Negative:** The cost and effort required to find true defects using ASI tools is high, because a large number of false positives must be manually evaluated and eliminated.
- Identify whether code reviews/inspections are being done with an ASI service.
  - **Positive:** ASI services provide code review in significantly less time and at a dramatically lower cost than manual inspection or internal use of inspection tools.
  - **Negative:** The cost and effort required to find true defects using ASI tools is high, because a large number of false positives must be manually evaluated and eliminated.

- Does the ASI tool/services address the following areas?
  - Identify the location and describing the circumstances under which the defects will occur.
  - Identify the parts of the code with the greatest risk.
  - Compare code quality with a benchmark.
  - The breakdown of defects by defect class.
  - Identify syntax and interface errors.
  - Identify potential for reducing code volume via redundant or unused code.

K.5 Defect Tracking Tools

The acquirer may require that the use of some type of defect tracking tools by the supplier be included in the SOW. The size, scale, complexity, program visibility and/or risk may drive the acquirer's interest in defect tracking. The number and type of defects can be important to identify and correct root cause and thus improve quality control. Defect tracking tools can be applied throughout the SDLC, and the tool reports can be effectively used by the acquirer.

K.5.1 How to Evaluate the Tool

Evaluation should consider:

- Does the defect management tool address the following areas?
  - Enable the user to track defects:
    - By source unit.
    - By programmer.
    - By date.
    - By type defect.
  - Enable the categorization of Defects.
  - Enable customization of Defect content.
  - Support standard and customized reports.

K.6 Source Code (Security) Analysis Tools

The size, scale, complexity, and other system/software development components will determine the level and type of source code analysis tools required to be employed by the supplier. The acquirer will want to identify security and source code vulnerability as an important area of concern, and may want to specify threat so that offerors can describe security analysis tool selection and application if appropriate. Source Code Analysis (SCA) tools can be applied throughout development and maintenance.

K.6.1 What It Is

SCA tools in general provides risk management through an automated method which is utilized to eliminate coding errors and design flaws. Security Analysis tools manage security risk for coding errors, design flaws and policy violations.

K.6.2 Why It Is Needed

Source code security analysis is supported within a number of tools. Source code vulnerabilities are difficult and expensive to identify through manual inspection. Tools continue to improve and evolve along with the threat. Use of the tools will increase cost and schedule, and project estimates must consider these costs. Risk analyses may be conducted to determine the potential impact of realized risk (vulnerabilities being found and exploited) before determining level of investment and application in such tools.

K.6.3 How to Evaluate the Tool

Evaluation should consider:

- Does the security analysis tool address the following areas?
  - Inspects calls to identify potential “Insecure” library functions.
  - Identifies bounds-checking errors and scalar type confusion.
  - Identifies type confusion among references/pointers.
  - Detection for memory allocation errors.
  - Detection of vulnerabilities which involve sequences of operations (Control-Flow Analysis).
  - Perform data-flow analysis.
  - Perform pointer-aliasing analysis.
  - Provide customizable detection capabilities.

K.7 Testing Tools

The IEEE Standard Glossary of Software Engineering Terminology defines verification and validation (V&V) as “The process of determining whether the requirements for a system or component are complete and correct, the products of each development phase fulfill the requirements or conditions imposed by the previous phase, and the final system or component complies with specified requirements.”

The V-model (shown in Figure K-1) illustrates that test planning as a part of requirements, specification, design and coding efforts should render acceptance, system, integration and unit test plans which are compliant with design.

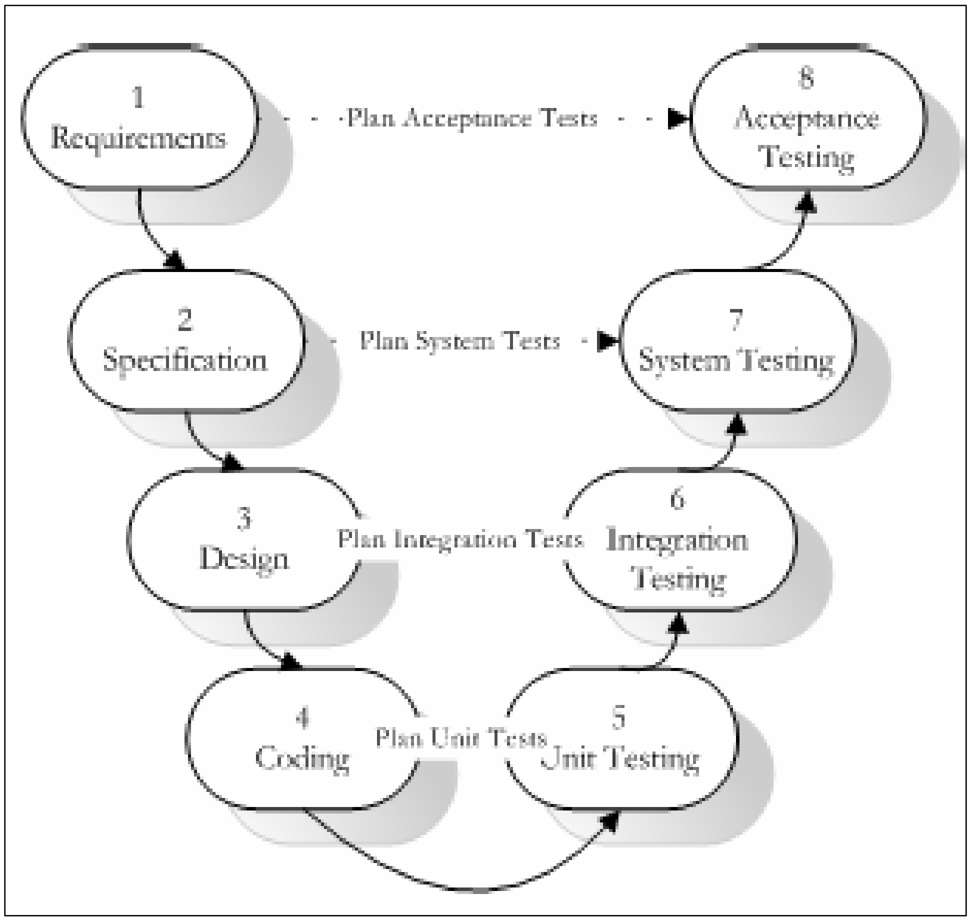


Figure K-1. The V model

- Testing tools can be used throughout the SDLC, beginning with tracing requirements and derived requirements to test design or plan.
- The acquirer will look for CMMI® Level 3 (or higher), or an equivalent standard software process, to ensure repeatable processes including test processes.
- The acquirer will likely have independent and/or oversight testing and certification steps that may require acquirer-level investment or understanding of test tools.

K.7.1 What It Is

Test tools provide support in the areas of:

- Test planning and monitoring;
- Designing test cases;
- Constructing test cases;
- Executing test cases;
- Capturing and comparing test results;
- Reporting test results;

- Tracking software problem reports/defects; and
- Managing the test ware.

K.7.2 Why It Is Needed

Test tools can improve the test team ability to conduct repeatable tests, identify defects, track defects to code modules, and produce test reports. While it is impossible to fully test any sizeable computer program, test tools can automate some tests, increasing the number of tests that can be conducted and eliminating some sources of human error.

K.7.3 How to Evaluate the Tool

Evaluation should consider:

- Does the testing tool address the following areas?
  - Test Management - Enable the user to author and maintain requirements:
    - Support the authoring of test requirements.
    - Support the maintenance of test requirements.
    - Support controlled access to test requirements.
    - Support discrete grouping or partitioning of test requirements.
    - Support traceability of requirements to test cases and defects.
    - Support canned and user defined queries against test requirements.
    - Support canned and user defined reports against test requirements.
    - Support coverage analysis of test requirements against test cases.
    - Support the integration of other toolsets via a published API or equivalent capacity.
  - Test Automation - Enable the user to author, maintain, and execute automated test cases by:
    - Support the creation, implementation, and execution of automated test cases.
    - Support controlled access to test automation.
    - Support data driven automated test cases.
    - Support keyword enabled test automation.
    - Integrate with all Tier 1 and 2 test management tools that support integration.
    - Integrate with all Tier 1 and 2 defect management tools that support integration.
    - Enable test case design within a non-technical framework.
    - Enable test automation and verification of Web, GUI, .NET, and Java applications.
    - Support the integration of other toolsets via a published API or equivalent capacity.

Appendix L
Sample SDP DID Outline, Template and Guidance

L.1 Sample SDP DID Outline and Template

This Data Item Description (DID) is provided for reference as discussed in Chapters 1, 5, and 7 of this guidebook. Although the specific format is not mandated, this sample reflects software process and contracting mandates and guidelines particularly with respect to the Institute of Electrical & Electronics Engineers/Electronic Industries Alliance (IEEE/EIA) Standard 12207. Other software DIDs in use by Department of the Navy (DoN) acquisition organizations (e.g., the software DID under SPAWAR configuration management) are also acceptable as long as they reflect the policy mandates noted in this guidebook, specifically by including all information items specified in the policy to the level of detail required.

Title Page

Signature Page

Change History

Preface

Table of Contents

List of Figures

List of Tables

1. Overview of Plan

1.1 Purpose, Scope, and Objectives

Describe the role of the plan, its purpose and objectives, and what it covers. Describe the relationship of the Plan to other plans, such as a Systems Engineering Plan, that are intended to comprise the complete "SDP" or reference it.

1.2 Plan Assumptions and Constraints

Describe the assumptions that are in place behind the development of this plan.

2 Product Overview

Provide an overview of the project to establish the context for this plan.

2.1 Product Role and Mission

Where the product will be used, for what purpose, and where it fits into a larger context (system of systems). Refer to IEEE Std.. 1058, section 4.1.1.1 for guidance and content. Provide architecture diagrams per the Naval "Systems of Systems" Systems Engineering Guidebook series.

2.2 Product Modes of Operation

To characterize the role of the system and how it is to operate in the perspective of the users, describe the system modes of operation. Include information from the IEEE/EIA 12207.1 Concept of Operations description (section 6.3).

2.3 Special Considerations

Describe product features/requirements that require additional and focused attention, such as software systems safety and security requirements, high reliability and availability expectations, performance requirements, harsh operating environments, legal and statuary constraints, privacy, etc. Include any linkage to the Program Protection Plan (PPP), Anti-Tamper appendix to that plan, and any Critical Program Information (CPI).

3 Project Overview

3.1 Project Deliverables

Refer to IEEE Std.. 1058, section 4.1.1.3 for guidance and content. Describe all project deliverables. Describe the intellectual property rights associated with each deliverable, as well as the form of each item and the mechanism to be used for delivery. Include projected COTS and the definition of COTS intended for this program.

3.2 Overall Schedule and Budget

Describe the overall project schedule, highlighting when the project deliverables will be provided and what tailorable organization staffing headcounts will be expected. Also summarize overall project budgeting, indicating allocation of resources across the schedule. If the program is under Earned Value Management System (EVMS), include what and how the software will be included. Refer to IEEE Std.. 1058, section 4.1.1.4 for guidance and content.

3.3 Known Project Risks

Describe the known risks associated with the project. This list will be incorporated into the risk management process and tracked throughout development. Risk should be characterized for critical requirements, safety-critical requirements, critical staffing, and integration risks.

3.4 Assumptions and Constraints

Refer to IEEE Std. 1058, section 4.1.1.2 for guidance and content.

4 References

This section shall list the number, title, revision, and date of all documents referenced in this plan. This section shall also identify the source for all documents not available through normal Government stocking activities. Refer to IEEE Std.. 1058, section 4.2 for guidance and content.

5 Definitions of Terms, Acronyms, and Abbreviations

6 Project Management

This section covers the information requirements of the Management Process, IEEE/EIA 12207.0 section 7.1. Include integrated approach to requirements management, risk management, open architecture, subcontractor management (see 6.7 below), COTS management, and, if CPI is indicated, supplier assurance.

6.1 Project Organization

Conforms with IEEE/EIA 12207.1, section 6.11.3.b. Refer to IEEE Std. 1058, section 4.4 for guidance and content. Include internal organization with roles and responsibilities, as well as external organizations and interfaces on whom the project depends and with whom the project will interact. Indicate which billets are considered critical and how they will be managed and scheduled. Include descriptions of any planned interactions with the acquirer and with the user community (IEEE/EIA 12207.1, sections 6.11.3.j and 6.11.3.k). Describe the organization features that facilitate management of global software issues and policies (such as global architecture features, exception and error handling policies, coding standards, etc.)

6.1.2 Internal Structure

6.1.3 External Interfaces and Dependencies

6.1.4 Roles and Responsibilities

6.2 Start-up Plan

To include assigning staff, staff training, and initial allocation of resources. Indicate which billets are critical to software. Refer to IEEE 1058, section 4.5.1 for guidance.

6.3 Work Breakdown Structure

In accordance with IEEE/EIA 12207.1, section 6.11.3.d, and IEEE/EIA 12207.0, section 5.2.4.5.c.. Include any EVMS/WBS lines which make software visible.

6.3.1 Work Activities

6.3.2 Schedule Allocation

6.3.3 Resource Allocation

6.3.4 Budget Allocation

6.4 Reporting Plan

Describe the plans for reporting progress, metrics, and activities to the acquirer. Refer to IEEE 1058, section 4.5.3.5 for guidance.

6.5 Training Plan

Describe the approach to training of project staff, in accordance with IEEE/EIA 12207.0, section 7.4, Training Process. Refer to IEEE 1058, section 4.5.1.4 for guidance. Emphasis on software critical billets and maintaining relevant staff competency throughout the life cycle.

6.6 Metrics Plan

Describe the measures that the developer plans to collect, including both management and technical metrics, and maintain evidence for independent validation, if required. Indicate which ones will be regularly reported, and which ones will be available for additional information. These metrics need to include measures addressing software process as well as product attributes. Refer to IEEE 1058, section 4.5.3.6 for guidance.

6.7 Acquisition Process

If the developer will be depending on additional suppliers, this section must be included, and needs to describe the developer's approach to management of these suppliers (subcontractor management). The relevant IEEE/EIA 12207.0 process is Acquisition, described in section 5.1 of IEEE/EIA 12207.0. Include information from the IEEE/EIA 12207.1 Acquisition Plan. Guidance is also provided by IEEE 1058, section 4.7.7. The included activities that need to be described are:

1. Initiation
2 Request-for-Proposal Preparation

3 Contract preparation and update
4 Supplier monitoring
5 Acceptance and completion

If subcontractors and/or vendors will not be used, this section of the SDP may be annotated with N.A. (not applicable).

6.8 Risk Management Plan

Describe Management Process, Planning Activity, 7.1.2.1f. Describe the risks associated with each process and aspect of the software to be developed (including development, test, management, etc.). Describe the definitions (risk, issue, ranking), approach towards, and management of, project risks. Refer to IEEE 1058, section 4.5.4 for guidance. Also IEEE/EIA 12207.1, section 6.11.3.l.

6.9 Closeout plan

Describe Management Process, Closure Activity (IEEE/EIA 12207.0, section 7.15). Describe plans for closeout of the software development effort, including staff reassignment, archiving data items, etc. Refer to IEEE 1058, section 4.5.5 for guidance.

7. Development Process

Development is covered by the IEEE/EIA 12207.0 Development Process, as described in IEEE/EIA 12207.0 section 5.3, and in this guidebook (see software development techniques). This section shall include specific standards, methods, tools, actions, reuse strategy, and responsibility associated with the development and qualification of all requirements, including software systems safety and security. Note that if any safety is involved then the naval enterprise requires a Software Systems Safety Technical Review Panel (SSSTRP and Weapons Systems Explosive Safety Review Board (WSESRB) be scheduled for each milestone and firing exercise or release. If security is involved, then the naval enterprise requires the SDP link to their PPP, with mandatory Anti-Tamper appendix, and compliance with DODD 5200.39, regardless of a Platform IT certification or full accreditation. If different processes are to be employed for different software Configuration Item (CI), then this section should specify these differences in the appropriate activity subsection. Include information IAW IEEE/EIA 12207.1, section 6.5, Development Process Plan. Use IEEE Std.. 1058 section 4.6 as additional guidance.

7.1 Overview of Approach

7.1.1 Overall Approach

Provide a brief description of the overall development approach, including rationale for its choice, its technical basis and suitability, and its key features.

7.1.2 Reuse

Describe the extent of reuse to be achieved and the approach to be used, including plans for acceptance of the reused products. Describe the approach to be used for identifying and accepting reused assets and COTS. Include all types of reused assets, including requirements, designs, code, test, etc. Include warranty and obsolescence management processes and deliverables.

7.1.3 Open Systems

Describe the approach to be followed for achieving open system and open architectures.

7.1.4 Critical Requirements

Describe the approach for handling critical requirements, introduced in section 2.3. These include safety, security, and information assurance among others.

7.2 Summary of Methods, Tools, and Techniques

In this section, the developer is to describe those standards, methods, tools, and programming languages to be used as a part of development, in support of the defined Development Process. The task for this is IEEE/EIA 12207.0 section 5.3.1.3. Based on IEEE/EIA 12207.1 *Software engineering methods/procedures/ tools description* and *Software Development Standards Description*, IEEE/EIA 12207.1 section 6.17. Include techniques for ensuring that all critical requirements will be met, including safety, security, performance, and others. Describe all non-deliverable software artifacts that will be used to develop the product. Describe compliance with DODI5000.61 for models, simulators, and stimulators, especially for safety.

7.3 Life Cycle Model

This section describes the life cycle model chosen by the developer, as part of process implementation of the Development Process. Map the processes, activities, and tasks of IEEE/EIA 12207.0 onto the life cycle model. Describe the product build plan. Based on 12207.1 *Software life cycle model description* and IEEE/EIA 12207.0 section 5.3.1.1. Refer to IEEE Std.. 1058, section 4.6.1 and IEEE Std.1074 for additional guidance.

7.4 Software Development Infrastructure

This section describes the planned development infrastructure, and is covered by the IEEE/EIA 12207.0 Infrastructure Process, described in section 7.2 of IEEE/EIA 12207.0. Include description of how information is to be shared with the customer. Describe how and where the different project artifacts will be maintained, including the system executables, system documentation, software development files, etc.

7.5 Verification Plan

Describe how the results of each activity are verified to be in compliance with the requirements and conditions imposed on them upon entrance to that activity. Refer to IEEE/EIA 12207.0 section 6.4, Verification Process. Include approach for verifying behavioral and quality requirements, including critical requirements.

7.6 Validation Plan

Describe how the final products will be validated to be in compliance with the overall system requirements and conditions. Include approach for validating both behavioral and quality requirements, including critical requirements. Covers the Validation Process. Refer to IEEE/EIA 12207.0 section 6.4, Verification Process.

7.7 System Requirements Analysis

Describe how the system requirements will be defined. Describe the Analysis of Alternatives (AOA) and tradeoff process used to optimize requirements. Describe the requirements flowdown approach to track to and from operational needs and operational concepts. Describe how all responsible SMEs will be involved in the process (such as hardware, software engineers, domain engineers, …) Describe how the operational scenarios are derived from operational concepts. Include information on this activity in accordance with IEEE/EIA 12207.0 section 5.3.2. Include information for how verification will be performed, ensuring that the results from applying this activity meets its goals. Describe how considerations for reuse and open architectures are employed.

7.8 System Architectural Design

Describe how the system architecture will be defined. Describe the AOA and tradeoff process used to optimize the architecture. Describe the requirements flowdown approach to track to and from system requirements. Describe how the operational scenarios will be applied to the architecture. Describe how responsible SMEs will be involved in the process (such as hardware, software engineers, domain engineers, …) Include information on this activity in accordance with IEEE/EIA 12207.0 section 5.3.3. Include information for how verification will be performed, ensuring that the results from applying this activity meets its goals. Describe how considerations for reuse and open architectures are employed.

7.9 Software Requirements Analysis

Describe how the software requirements will be defined. Describe the tradeoff process used to balance requirements across all software CIs. Describe the requirements flowdown approach to track to and from system requirements and test. Include information on this activity in accordance with IEEE/EIA 12207.0 section 5.3.4. Ensure that the approach to handling all special requirements are covered, including security, software systems safety, performance, and human interface. Include information for how verification will be performed, ensuring that the results from applying this activity meets its goals. Describe any intended model and simulation. Describe how considerations for reuse and open architectures are employed.

7.10 Software Architectural Design

Include information on this activity in accordance with IEEE/EIA 12207.0 section 5.3.5. Ensure that the approach to open systems and open architecture is covered. Include information for how verification will be performed, ensuring that the results from applying this activity meets its goals. Describe how considerations for reuse and open architectures are employed. Describe how allocation of mission-critical, safety, or security requirements, as applicable, will be visible within the design.

7.11 Software Detailed Design

Include information on this activity in accordance with IEEE/EIA 12207.0 section 5.3.6. Include information for how verification will be performed, ensuring that the results from applying this activity meets its goals. Describe how considerations for reuse and open architectures are employed. Describe any special designs for mission-critical, safety, security, and anti-tamper requirements.

7.12 Software Coding And Testing

Describe the approach to be followed for coding and unit testing. Specify the coverage criteria to be applied for this testing. Include information on this activity in accordance with IEEE/EIA 12207.0 section 5.3.7. Include information for how verification will be performed, ensuring that the results from applying this activity meets its goals. Describe how considerations for reuse and open architectures are employed. Describe how reused and COTS components are verified and changes are managed. Describe regression testing definitions and triggers.

7.13 Software Integration

Include information on this activity in accordance with IEEE/EIA 12207.0 section 5.3.8. Ensure that this is consistent with the system and software build plan, described in section 7.3. Refer to Software Integration Plan, IEEE/EIA 12207.1 section 6.18. Include information for how verification will be performed, ensuring that the results from applying this activity meets its goals. Describe how integration will be performed with any models and simulators or human-in-the-loop.

7.14 Software Qualification Testing

Include information on this activity in accordance with IEEE/EIA 12207.0 section 5.3.9.

7.15 System Integration

Include information on this activity in accordance with IEEE/EIA 12207.0 section 5.3.10. Include information for how verification will be performed, ensuring that the results from applying this activity meets its goals. Describe how integration will be performed with any models and simulators or human-in-the-loop.

7.16 System Qualification Testing

Include information on this activity in accordance with IEEE/EIA 12207.0 section 5.3.11. Include information for how verification will be performed, ensuring that the results from applying this activity meets its goals.

7.17 Software Installation

If the work effort includes this activity, include information in accordance with IEEE/EIA 12207.0 section 5.3.12. If not, annotate this section with 'n/a'. Include information for how verification will be performed, ensuring that the results from applying this activity meets its goals. Describe how installation will be integrated with naval enterprise, such as release for Flight Clearance, WSESRB, platform certification, certification and accreditation, etc., as appropriate.

7.18 Software Acceptance Support

If the work effort includes this activity, include information in accordance with IEEE/EIA 12207.0 section 5.3.13. If not, annotate this section with 'n/a'. Include information for how verification will be performed, ensuring that the results from applying this activity meets its goals.

8 Operation Process

If the work effort includes this process, describe the Operational Process, as defined in IEEE/EIA 12207.0, section 5.4, Operation Process. If not, annotate this section with 'n/a'. Base on IEEE/EIA 12207.1 section 6.9, *Operation Process Plan*. Also include information from the *Concept of Operations Description*, IEEE/EIA 12207.1 section 6.3.

8.1 Process Implementation

Include information on this activity in accordance with IEEE/EIA 12207.0 section 5.4.1.

8.2 Operational Testing

Include information on this activity in accordance with IEEE/EIA 12207.0 section 5.4.2.

8.3 System Operation

Include information on this activity in accordance with IEEE/EIA 12207.0 section 5.4.3.

8.4 User Support

Include information on this activity in accordance with IEEE/EIA 12207.0 section 5.4.4.

9 Maintenance Process

If the scope of effort includes maintenance activities (post deployment), include information IAW IEEE/EIA 12207.1, section 6.8, Maintenance Process Plan and IEEE/EIA 12207.0 section 5.5, Maintenance Process. If not, annotate this section with N.A.

9.1 Process Implementation

9.2 Problem and Modification analysis

9.3 Modification Implementation

9.4 Maintenance Review/Acceptance

9.5 Migration

9.6 Software Retirement

10 Supporting Processes

10.1 Documentation Process

Describe way that the project information will be documented. Includes specific documents to be produced as well as other forms of information capture, such as automated tools, databases, reports, briefings, engineering notebooks, software development files and libraries, etc. Describe the format and/or standard that these documents are based on. Describe any documentation required to support measures and metrics. Ensure that this section is consistent with section 6 of this SDP. Describe the intellectual property rights associated with each type of data. Describe how Controlled Unclassified Information (CUI) will be marked, handled, and delivered.

10.2 Configuration Management Plan

Describe how project artifacts will be managed and controlled. Based on IEEE/EIA 12207.1 section 6.14, Software Configuration Management Plan and IEEE/EIA 12207.0, section 6.11.3.r. *Plan may be supplied separately*

10.3 Quality Assurance Plan

Based on Software Quality Assurance Plan, IEEE/EIA 12207.1 section 6.20. Include information from the Audit Process (IEEE/EIA 12207.0, section 6.7). *This plan may be supplied separately.*

10.4 Joint Review Plan

Describe the approach for performing joint reviews, both at the management level and the technical level. Covers IEEE/EIA 12207.0, Joint Review Process. Include information from IEEE/EIA 12207.0 section 6.6, and IEEE/EIA 12207.1, *Joint Review Procedure*.

10.4.1 Overview of Reviews Planned

Describe the general approach to planning for and conducting reviews. Describe how the customer will be involved. Describe how the information for each review will be provided to the attendees.

10.4.2 Management Reviews

Describe which reviews will be conducted, when, and what criteria for entry and exit will be applied.

10.4.3 Technical Reviews

Describe which reviews will be conducted, when, and what criteria for entry and exit will be applied.

10.5 Problem Resolution Plan

Covers the IEEE/EIA 12207.0 Problem Resolution Process. Describe in accordance with IEEE/EIA 12207.0 section 6.8 and IEEE/EIA 12207.1, section 6.10. Describe how the software IPT, described in this guidebook, will be involved in this process.

10.6 Process Improvement Plan

To describe the Improvement Process, IEEE/EIA 12207.0, section 7.3. This section is fundamental to ASN(RD&A) software process improvement mandates; source selection evaluations should consider it a significant shortfall if not included in the offeror's response.

10.6.1 Process Assessment

Describe the approach to be used to assess the current process, to include its level of adherence as well as its effectiveness.

10.6.2 Process Improvement

Describe the plan for how to perform process improvement, including the process for updating this SDP.

10.7 Test/Validation Plans

Describe in accordance with IEEE/EIA 12207.0 section 6.8 and IEEE/EIA 12207.1, section 6.27. Coordinate with the Software Integration Plan and the Software Verification Plan. This plan may be supplied separately and may be deferred until after the requirements for the software CIs have been defined and their SRSs have stabilized.

11 Additional Plans

If any additional plans are required, list these in this section.

Annexes

Index

L.2 Guidance for Applying and Tailoring the SDP DID

L.2.1 Guidelines for Tailoring the SDP DID

Section L.2.2 describes the standard (default) content of the Software Development Plan (SDP), as defined by the SDP DID. The default requires the inclusion of all IEEE/EIA Standard 12207.0 processes and activities, unless the offeror deems them to be not applicable to the work effort.

Section L.2.3 describes the steps that may be taken to tailor the content of the SDP. This tailoring is accomplished by specifically listing those processes and activities that are not applicable to the work effort. This list may be incorporated into the DID itself, or may be included in the SOW as a part of its role of defining the expected work effort. Some processes and activities are essential for process improvement and hence cannot be tailored out of the SDP.

L.2.2 Standard Content

The DoN is using IEEE/EIA Standard 12207 as the underlying framework for software development work efforts. In the context of IEEE/EIA Standard 12207.0, a life cycle model for a specific development effort consists of the set of processes, activities, and tasks taken as a whole that result in the production of the intended product. IEEE/EIA Standard 12207.0 defines a set of standard processes and associated activities and tasks that cover the entire software development life cycle. These are grouped into three categories: primary, supporting, and organizational. The specific processes and activities within each group are listed in Tables L-1, L-2, and L-3 below.

IEEE/EIA Standard 12207.0 describes the task content of these process and activities but does not define the way tasks are performed. As a part of their SDP, developers are required to describe the specific techniques, tools, methods, and other details regarding how each of the tasks are to be performed.

If an offeror deems any process, activity, or task to be not applicable for the work effort (that is, is not necessary for the accomplishment of the work effort), and then they may annotate such activity as “not applicable.” Such exclusions must be justified in the SDP Rationale document which is to be submitted along with the proposal (see RFP section L).

If the desire is to require the offerors to fully describe how they will accomplish all of the IEEE/EIA Standard 12207.0 activities, no additional language is required. If an acquisition decides that certain processes/activities/tasks are not appropriate for a work effort, then they are to tailor the SDP DID as appropriate. If an offeror feels that a specific element is not required for the work effort, they are to annotate the SDP appropriately.

L.2.3 Tailored Content

Many programs will not need to employ all of the IEEE/EIA Standard 12207.0 processes. For example, if a program solely involves development, with deployment covered under a different contractual vehicle, then the activities relating to the Operation Process may be tailored out of the required SDP content.

Likewise, if the government has a particular interest in requiring that certain activities be performed in a specific way, using preferred techniques and/or tools, then these may be defined to tailor the SDP as a part of this language. It is extremely important, in all cases to include information concerning the IEEE/EIA Standard 12207 Improvement Process, as it is a core element of the DoN Software Improvement Initiative. This tailoring is to be accomplished by providing in the SDP a table specifying the activities and processes that are relevant to the work effort and that must be addressed in the SDP. This tailoring table is to be based on Tables L-1, L-2 and L-3 contained in this guidance.

IEEE/EIA Standard 12207.0 Primary Processes and Activities	
Process	*Activities*
Acquisition	▪ Initiation ▪ Request-for-proposal [-tender] preparation ▪ Contract preparation and update ▪ Supplier monitoring ▪ Acceptance and completion
Supply	▪ Initiation ▪ Preparation of response ▪ Contract ▪ Planning ▪ Execution and control ▪ Review and evaluation ▪ Delivery and completion
Development	▪ Process implementation ▪ System requirements analysis ▪ System architectural design ▪ Software requirements analysis ▪ Software architectural design ▪ Software detailed design ▪ Software coding and testing ▪ Software integration ▪ Software qualification testing ▪ System integration ▪ System qualification testing ▪ Software installation ▪ Software acceptance support
Operation	▪ Process implementation ▪ Operational testing ▪ System operation ▪ User support
Maintenance	▪ Process implementation ▪ Problem and modification analysis ▪ Modification implementation ▪ Maintenance review/acceptance ▪ Migration ▪ Software retirement

Table L-1. IEEE/EIA Standard 12207.0 Primary Processes and Activities

IEEE/EIA Standard 12207.0 Supporting Processes and Activities	
Process	*Activities*
Documentation process	▪ Process implementation ▪ Design and development ▪ Production ▪ Maintenance
Configuration management process	▪ Process implementation ▪ Configuration identification ▪ Configuration control ▪ Configuration status accounting ▪ Configuration evaluation ▪ Release management and delivery
Quality assurance process	▪ Process implementation ▪ Product assurance ▪ Process assurance ▪ Assurance of quality systems
Verification process	▪ Process implementation ▪ Verification - Contract verification - Process verification - Requirements verification - Design verification - Code verification - Integration verification - Documentation verification
Validation process	▪ Process implementation ▪ Validation
Joint review process	▪ Process implementation ▪ Project management reviews ▪ Technical reviews
Audit process	▪ Process implementation ▪ Audit
Problem resolution process	▪ Process implementation ▪ Problem resolution

Table L-2. IEEE/EIA Standard 12207.0 Supporting Processes and Activities

IEEE/EIA Standard 12207.0 Organizational Life Cycle Processes and Activities	
Process	*Activities*
Management Process	▪ Initiation and scope definition ▪ Planning ▪ Execution and control ▪ Review and evaluation ▪ Closure
Infrastructure Process	▪ Process implementation ▪ Establishment of the infrastructure ▪ Maintenance of the infrastructure
Improvement Process	▪ Process establishment ▪ Process assessment ▪ Process improvement
Training Process	▪ Process implementation ▪ Training material development ▪ Training plan implementation

Table L-3. IEEE/EIA Standard 12207.0 Organizational Life Cycle Processes and Activities

Appendix M

Introduction to IEEE/EIA Standard 12207

M.1 Introduction

Electrical and Electronics Engineers/Electronic Industries Alliance (IEEE/EIA) Standard 12207, IEEE Standard for Information Technology – Life Cycle Processes, provides a common framework for developing and managing software. While "12207" is generally used to refer to this set of standards, it is important to understand that there are three standards that comprise the IEEE/EIA Standard 12207 supporting framework: 12207.0, 12207.1, and 12207.1 (Figure M-1).

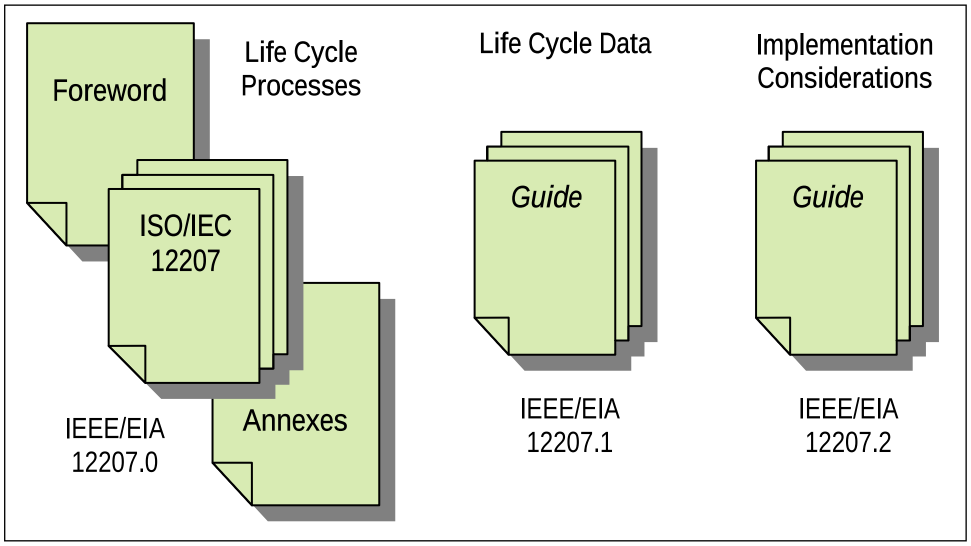


Figure M-1. Overview of IEEE/EIA Standard 12207

M.1.1 IEEE/EIA Standard 12207.0

IEEE/EIA 12207.0-1996: Industry Implementation of International Standard ISO/IEC 12207: 1995 – (ISO/IEC 12207) Standard for Information Technology – *Software life cycle processes.*

IEEE/EIA Standard 12207.0 contains "concepts and guidelines to foster better understanding and application of the standard."[1] The goal of this standard is to provide a common basis for software practices that would be useable for both national and international business. The standard includes "clarifications, additions, and changes accepted by the Institute of Electrical and Electronics Engineers and the Electronic Industries Alliance as formulated by a joint project of these two organizations." Table M-1 provides an overview of IEEE/EIA Standard 12207.0 document structure.

Overview of IEEE/EIA Standard 12207.0 Document Structure	
Clause 1 – Scope	Annex C – Guidance on processes and organizations
Clause 2 – Normative references	Annex D – Bibliography
Clause 3 – Definitions	Annex E – Basic concepts of ISO/IEC 12207
Clause 4 – Application of these International Standards	Annex F – Compliance
Clause 5 – Primary life cycle processes	Annex G – Life cycle processes objectives
Clause 6 – Supporting processes	Annex H – Life cycle data objectives
Clause 7 – Organizational life cycle processes	Annex I – Relationships
Annex A – Tailoring process	Annex J – Errata
Annex B – Guidance on tailoring	

Table M-1. Overview of IEEE/EIA Standard 12207.0 Document Structure[2]

M.1.2 IEEE/EIA Standard 12207.1

IEEE/EIA Guide 12207.1-1997: Industry Implementation of International Standard ISO/IEC 12207: 1995 – (ISO/IEC 12207) Standard for Information Technology – *Software life cycle processes – life cycle data.*

IEEE/EIA Standard 12207.1 provides guidance for recording life cycle data resulting from the life cycle processes of IEEE/EIA Standard 12207.0. Plainly stated, this standard provides descriptions for many key plans and artifacts that are required for software lifecycle support.[3] Table M-2 provides an overview of 12207.1 document structure.

Overview of IEEE/EIA Standard 12207.1 Document Structure	
Clause 1 – Scope	Clause 5 – Generic information item content guidelines
Clause 2 – Normative references	Clause 6 – Specific information item content guidelines
Clause 3 – Definitions	Annex A – References
Clause 4 – Life cycle data	
Clause 4.1 – Overview	
Clause 4.2 – Life cycle data objectives	
Clause 4.3 – Information item matrix	
Clause 4.4 – Compliance	

Table M-2. Overview of IEEE/EIA Standard 12207.1 Document Structure[4]

[1] IEEE/EIA 12207.0-1996; Industry Implementation of International Standard ISO/IEC 12207:1995 (ISO/IEC 12207) Standard for Information Technology-Software life cycle processes – Software Life Cycle Processes, IEEE/EIA 12207.0 (Mar 1998), IEEE Press, New York, NY, USA.
[2] IEEE/EIA Standard 12207.0-1996.
[3] IEEE/EIA 12207.1-1997; Industry Implementation of International Standard ISO/IEC 12207:1995 (ISO/IEC 12207) Standard for Information Technology-Software life cycle processes – Software Life Cycle Data, IEEE/EIA Standard 12207.0 (April 1998), IEEE Press, New York, NY, USA.
[4] IEEE/EIA Standard 12207.1-1997.

M.1.3 IEEE/EIA Standard 12207.2

IEEE/EIA Guide 12207.2-1997: Industry Implementation of International Standard ISO/IEC 12207: 1995 – (ISO/IEC 12207) Standard for Information Technology – *Software life cycle processes – Implementation considerations.*

IEEE/EIA Standard 12207.2 provides implementation considerations guidance for the normative clauses of IEEE/EIA Standard 12207.0.[5] This guidance is based on software industry experience with the life cycle processes of IEEE/EIA Standard 12207.0 and provides a summary of best practices for implementing primary, supporting, and organizational lifecycle processes defined in clauses 5, 6, and 7 of ISO/IEC 12207. Table M-3 provides an overview of IEEE/EIA Standard 12207.2 document structure.

Overview of IEEE/EIA Standard 12207.2 Document Structure	
Clause 1 – Scope	Annex E – IEEE/EIA 12207.0 Annex J Errata
Clause 2 – Normative references	Annex F – Use of reusable software products
Clause 3 – Definitions	Annex G – Candidate joint management reviews
Clause 4 – Application	Annex H – Software measurement categories
Clause 5 – Primary life cycle processes	Annex I – Guidance on development strategies and build planning
Clause 6 – Supporting processes	Annex J – Category and priority classifications for problem reporting
Clause 7 – Organizational life cycle processes	Annex K – Software product evaluations
Annex A – IEEE/EIA 12207.0 Annex A Tailoring process	Annex L – Risk management
Annex B – IEEE/EIA 12207.0 Annex F Compliance	Annex M – Life cycle processes references
Annex C – IEEE/EIA 12207.0 Annex G Life cycle processes objectives	
Annex D – IEEE/EIA 12207.0 Annex H Life cycle data objectives	

Table M-3. Overview of IEEE/EIA Standard 12207.2 Document Structure[6]

M.1.4 IEEE/EIA 12207 Supporting Standards and Process Frameworks

IEEE/EIA Standard 12207 differs from most of the other standards in the IEEE Computer Society software and systems engineering standards collection. Where this set of standards provides a common implementation framework, other standards in the collection provide users with insight into recommended software engineering best practices. This combination of the IEEE/EIA Standard 12207 common frameworks, along with the supporting process guidance provided by the other standards in the collection, can act as a catalyst when used effectively in support of software process definition and deployment.[7]

In addition, an organization should employ an organizational process to establish, control, and improve life cycle processes using a process improvement methodology such as Capability Maturity Model Integration (CMMI®). Figure M-2 provides an overview of how these framework and supporting standards support continuous software process improvement.

[5] IEEE/EIA 12207.2-1998; Industry Implementation of International Standard ISO/IEC 12207:1995 (ISO/IEC 12207) Standard for Information Technology-Software life cycle processes – Implementation Considerations, IEEE/EIA 12207.0 (April 1998), IEEE Press, New York, NY, USA.

[6] IEEE/EIA Standard 12207.2-1998.

[7] Land, Susan K., and John W. Walz. Practical Support for CMMI®-SW Software Project Documentation – Using IEEE Software Engineering Standards. New York: Wiley-IEEE Computer Society Press, 2005.

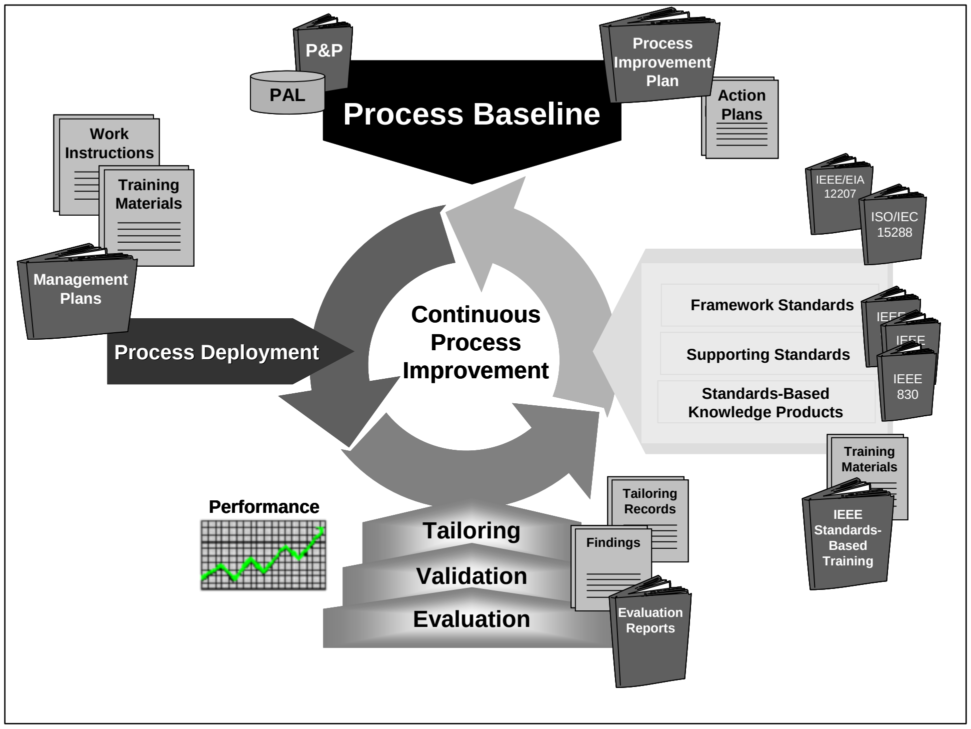


Figure M-2. Standards Support for Continuous Software Process Improvement[8]

M.2 IEEE/EIA Standard 12207 - Primary Processes

As previously stated, IEEE/EIA Standard 12207 standards establish a framework for the life cycle of software. The life cycle begins with an idea or need that can be partly or wholly satisfied by software and ends with the retirement of that software. The framework architecture is built with a set of processes. Interrelationships exist among these processes and each process should be placed under the responsibility of an organization or a party in the software life cycle. Figure M-3 provides an overview of these processes.

IEEE/EIA Standard 12207 groups the activities that may be performed during the life cycle of software into five Primary processes, eight Supporting processes, and four Organizational processes. Each life cycle process is divided into a set of activities; each activity is further divided into a set of tasks.

The five Primary processes serve primary parties during the life cycle of software. A primary party is one that initiates or performs the development, operation, or maintenance of software products. These primary parties are the acquirer, the supplier, the developer, the operator, and the maintainer of software products.

[8] Croll, Paul, "How to Use Standards as Best Practice Information Aids for CMMI-Compliant Process Engineering", 14th Annual DoD Software Technology Conference, USA, 2002.

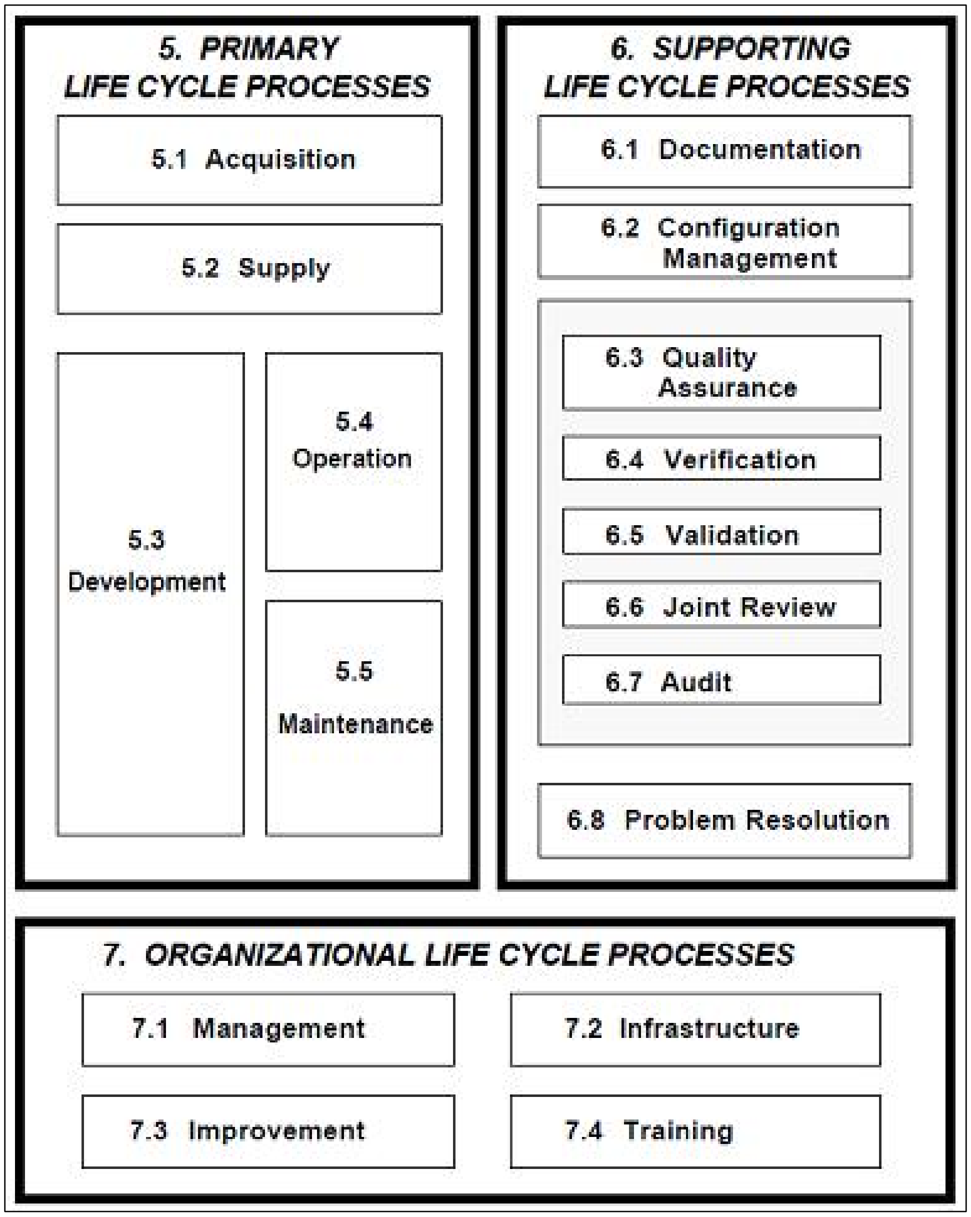


Figure M-3. Diagram of the Primary, Supporting, and Organizational Processes[9]

The eight Supporting life cycle processes support other processes as an integral part with a distinct purpose and contribute to the success and quality of the software project. A Supporting process is employed and executed, as needed, by another process. The four Organizational life cycle processes are employed by an organization to establish and implement an underlying structure made up of associated life cycle processes and personnel and continuously improve the structure and processes. They are typically employed outside the realm of specific projects and contracts; however, lessons from such projects and contracts contribute to the improvement and overall success of an organization.

[9] IEEE/EIA Standard 12207.0-1996.

M.2.1 Acquisition Process

The Acquisition Process defines the activities of the acquirer, the organization that acquires a systems, software product or software service. The process begins with the definition of the need to acquire a system, software product or software service. The process continues with the preparation and issue of a request for proposal, selection of a supplier, and management of the acquisition process through to the acceptance of the system, software product or software service.[10] Table M-4 provides a list of the Acquisition Process objectives:

Acquisition Process Objectives

a) Develop and deliver a Request for Proposal;
b) Develop a contract, including tailoring of the standard, that clearly expresses the expectation, responsibilities, and liabilities of both the acquirer and the supplier;
c) Obtain products and/or services that satisfy the customer need;
d) Manage the acquisition so that specified constraints (e.g., cost, schedule and quality) and goals (e.g., degree of software reuse) are met;
e) Establish a statement of work to be performed under contract;
f) Qualify potential suppliers through an assessment of their capability to perform the required software;
g) Select qualified suppliers to perform defined portions of the contract;
h) Establish and manage commitments to and from the supplier;
i) Regularly exchange progress information with the supplier;
j) Assess compliance of the supplier against the agreed upon plans, standards and procedures;
k) Assess the quality of the supplier's delivered products and services;
l) Establish and execute acceptance strategy and conditions (criteria) for the software product or service being acquired;
m) Establish a means by which the acquirer will assume responsibility for the acquired software product or service.

Table M-4. Acquisition Process Objectives[11]

The type and extent of control applied to the supplier and purchased product depends upon the effect of the product on the subsequent realization processes or the final product. For Commercial-Off-The-Shelf (COTS) software, almost no control is applied to the supplier. On the other hand, for custom software development, the acquirer takes on the role of the knowledgeable customer, deeply involved with the supplier through their software life cycle.

This involvement begins with customer review of requirements related to the product. For either extreme, COTS or custom software development, this process requires evaluation and selection of the supplier based on established criteria. In either case, records should be kept on the evaluations and subsequent actions taken.

In the case of COTS software, the objective evidence of supplier selection can be documented in a decision matrix. When selecting suppliers for more complex custom software development efforts, requirements must be communicated to the supplier in a Request For Proposal (RFP). For more extensive automation of operations a Concept of Operations (CONOPS) can be used to supplement the RFP and define the current system or situation, describing the justification for and nature of changes, concepts for

[10] IEEE/EIA Standard 12207.2-1998.
[11] Land, Susan K., and John W. Walz. Practical Support for ISO 9001 Software Project Documentation – Using IEEE Software Engineering Standards. New York: Wiley-IEEE Computer Society Press, 2006.

the proposed system, operational scenarios, summary of impacts, and analysis of the proposed system. Figure M-4 provides an overview of the acquisition process as described by IEEE/EIA Standard 12207.2.

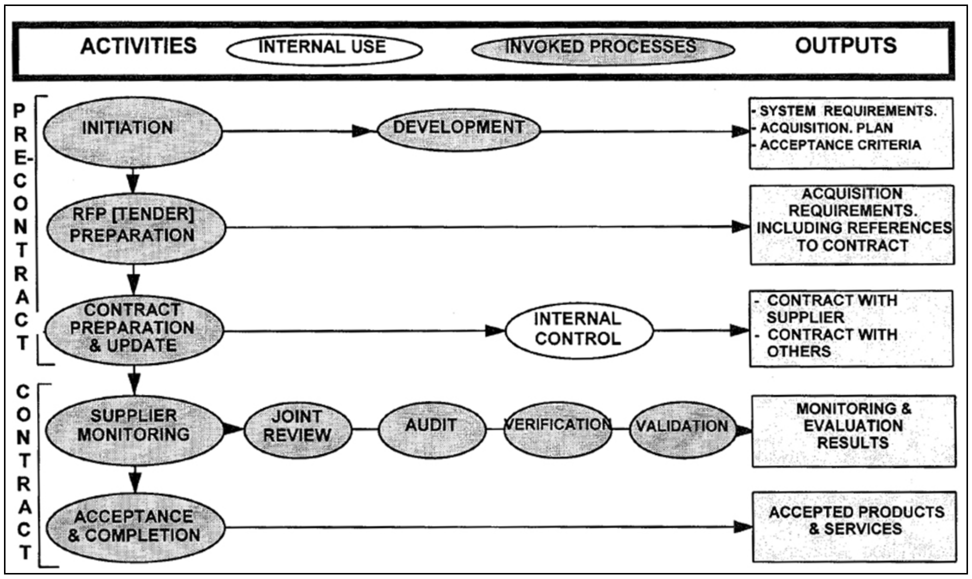


Figure M-4. IEEE/EIA Standard 12207.2 Acquisition Process[12]

M.2.2 Supply Process

The Supply Process contains the activities and tasks of the supplier, the organization that provides the system, software product or software service to the acquirer. The process may be initiated either by a decision to prepare a proposal to answer an acquirer's request for proposal or by signing and entering into a contract with the acquirer to provide the system, software product or software service. The process continues with the determination of procedures and resources needed to manage and assure the project, including development of project plans and execution of the plans through delivery of the system, software product or software service to the acquirer. Table M-5 provides a list of the Supply Process objectives.

Supply Process Objectives
a) Establish clear and ongoing communication with the customer; b) Define documented and agreed customer requirements, with managed changes; c) Establish a mechanism for ongoing monitoring of customer needs; d) Establish a mechanism for ensuring that customers can easily determine the status and disposition of their requests; e) Determine requirements for replication, distribution, installation, and testing of the system containing software or stand-alone software product; f) Package the system containing software or the stand-alone software product in a way that facilitates its efficient and effective replication, distribution, installation, testing, and operation; g) Deliver a quality system containing software or stand-alone software product to the customer, as defined by the requirements, and install in accordance with the identified requirements.

Table M-5. Supply Process Objectives[13]

[12] IEEE/EIA Standard 12207.2-1998.
[13] Practical Support for ISO 9001 Software Project Documentation.

The supplier is responsible for working with the customer to define and clearly understand all requirements associated with the software development effort. This may include delivery and post-delivery activities and items not identified in the RFP. In addition, the supplier must also determine any statutory and regulatory requirements related to the product. The supplier organization is responsible for conducting a thorough review of the software product requirements before committing to supply the software product to the customer in order to:

- Ensure software product requirements are defined;
- Resolve any requirements differing from those previously expressed; and
- Ensure its ability to meet the requirements.[14]

The supplier is responsible for software product planning. The supplier must plan and manage the processes needed for product realization. Figure M-5 provides an overview of the Supply Process as described by IEEE/EIA Standard 12207.2.

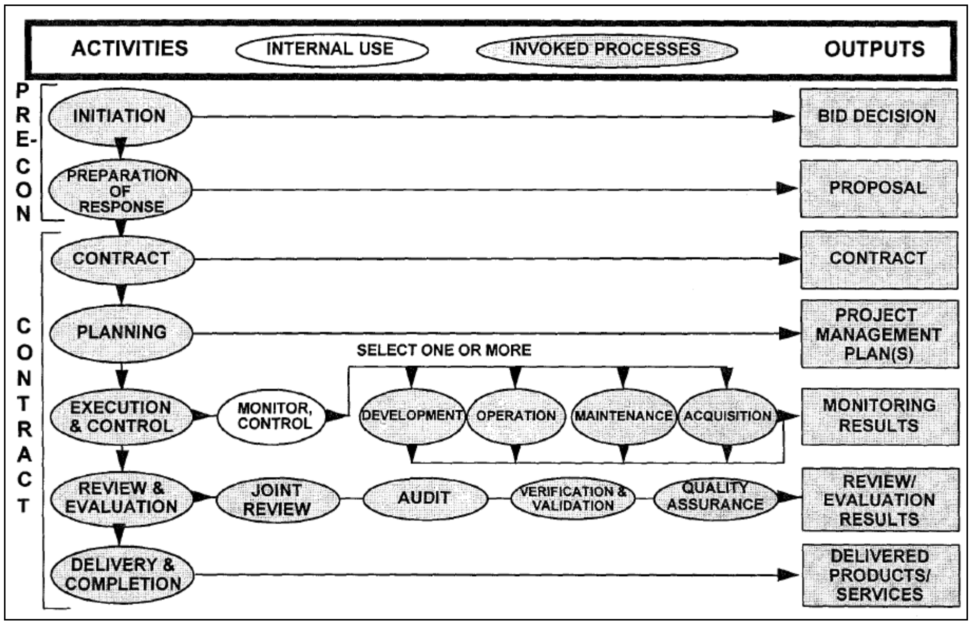


Figure M-5. IEEE/EIA Standard 12207.2 Supply Process[15]

M.2.3 Development Process

The Development Process contains the activities and tasks of the developer (the organization that defines and develops software products) including managing the Development Process at the project level following the Management Process, Infrastructure Process, and Tailoring Process. The developer also manages the process at the organizational level following the Improvement Process and the Training Process. Finally the developer performs the Supply Process if it is the supplier of developed software products. Table M-6 provides a list of the development process objectives.

[14] International Standard 9001, Quality management systems – Requirements, ISO 9001:2000(E), 2000, Switzerland.
[15] IEEE/EIA Standard 12207.2-1998.

Development Process Objectives

a) Develop requirements of the system that match the customer's stated and implied needs;
b) Propose an effective solution that identifies the main elements of the system;
c) Allocate the defined requirements to each of those main elements;
d) Develop a software and/or system release strategy;
e) Communicate the requirements, proposed solution and their relationships to all affected parties;
f) Define the requirements allocated to software components of the system and their interfaces to match the customer's stated and implied needs;
g) Develop software requirements that are analyzed, correct, and testable;
h) Understand the impact of software requirements on the operating environment;
i) Develop a software release strategy;
j) Approve and update the software requirements, as needed;
k) Communicate the software requirements to all affected parties;
l) Develop an architectural design;
m) Define internal and external interfaces of each software component;
n) Establish traceability between system requirements and design and software requirements, between software requirements and software design, and between software requirements and tests;
o) Define verification criteria for all software units against the software requirements;
p) Produce software units defined by the design;
q) Accomplish verification of the software units against the design;
r) Develop an integration strategy for software units consistent with the release strategy;
s) Develop acceptance criteria for software unit aggregates that verify compliance with the software requirements allocated to the units;
t) Verify software aggregates using the defined acceptance criteria;
u) Verify integrated software using the defined acceptance criteria;
v) Record the results of the software tests;
w) Develop a regression strategy for retesting aggregates, or the integrated software, should a change in components be made;
x) Develop an integration plan to build system unit aggregates according to the release strategy;
y) Define acceptance criteria for each aggregate to verify compliance with the system requirements allocated to the units;
z) Verify system aggregates using the defined acceptance criteria;
aa) Construct an integrated system demonstrating compliance with the system requirements (functional, nonfunctional, operations and maintenance);
bb) Record the results of the system tests;
cc) Develop a regression strategy for retesting aggregates or the integrated system should a change in components be made;
dd) Identify transition concerns, such as availability or [sic] work products, availability of system resources to resolve problems and adequately test before fielding corrections, maintainability, and assessment of transitioned work products.

Table M-6. Development Process Objectives[16]

The Development Process is the largest of the 17 processes in IEEE/EIA Standard 12207. The development activities are:

- Process implementation;
- System requirements analysis;
- System architectural design;
- Software requirements analysis;
- Software architectural design;

[16] Practical Support for ISO 9001 Software Project Documentation.

- Software detailed design;
- Software coding and testing;
- Software integration;
- Software qualification testing;
- System integration;
- System qualification testing;
- Software installation; and
- Software acceptance support.[17]

Depending upon the type of contract, the Development Process begins with process implementation and continues through to the software acceptance support. Unless stipulated in the contract, the developer should define the software life cycle model for the project. This requires planning the necessary processes, documents, and resources, followed up by records as evidence the processes and resulting product meet requirements. Process implementation should result in a definition of what products are to be produced, who is to produce them, and when they are to be produced and verified.

Each software requirement must be determined and records must be maintained. Each requirement must be reviewed for sufficiency and completeness. Any incomplete, ambiguous, or conflicting requirement must be resolved. The requirements should include:

- Functional and performance requirements;
- Applicable statutory, safety, and regulatory requirements; and
- Essential requirements.

The developer is responsible for the definition of the software design and the documentation of all associated development outputs so as to enable verification against the inputs to the design and Development Process.

The developer must also define requirements, design and development outputs so that as to enable verification against the inputs to the design and development process. All reviews, test planning and execution should be performed in accordance with the planned software life cycle so that as to confirm the resulting software product is capable of meeting the requirements for its specified application or intended use. When contractually required, the validation must be completed before software delivery or implementation. The results of the verification and subsequent follow-up actions must be maintained. Figure M-6 provides a description of the Development Process as presented by IEEE/EIA Standard 12207.2.

[17] IEEE/EIA Standard 12207.0-1996.

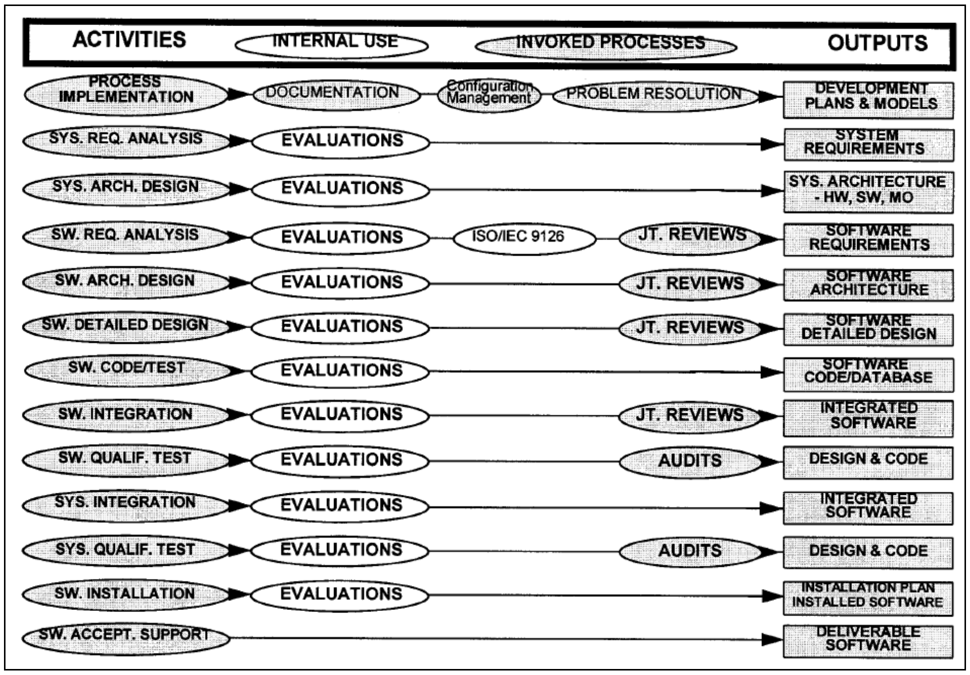


Figure M-6. IEEE/EIA Standard 12207.2 Development Process[18]

M.2.4 Operation Process

The Operation Process contains the activities and tasks of the operator, the organization that provides the service of operating a computer system in its live environment for its users. This process covers the operation of the software product and operational support to users.

The operator manages the Operation Process at the project level following the Management Process, which is instantiated in this process; establishes an infrastructure under the process following the Infrastructure Process; tailors the process for the project following the Tailoring Process; and manages the process at the organizational level following the Improvement Process and the Training Process. Table M-7 describes the operation process objectives.

Operation Process Objectives
a) Identify and mitigate operational risks for the software introduction and operation;
b) Operate the software in its intended environment according to documented procedures;
c) Provide operational support by resolving operational problems and handling user inquires and requests;
d) Provide assurance that software (and host system) capacities are adequate to meet user needs.
e) Identify customer support service needs on an ongoing basis;
f) Assess customer satisfaction with both the support services being provided and the product itself on an ongoing basis;
g) Deliver needed customer services.

Table M-7. Operation Process Objectives[19]

[18] IEEE/EIA Standard 12207.2-1998.
[19] Practical Support for ISO 9001 Software Project Documentation.

If customer communication during operations and maintenance is required by contract, the supplying organization determines and implements effective arrangements for communicating with customers on:

- Software product information (e.g., warranties);
- Inquiries, contracts, or order handling; and
- Customer feedback (i.e., support, complaints, satisfaction).

The contract may also add provisions for the control of production and service. These provisions may direct the software producer to plan and control operations, including:

- The need to set up a help desk to conduct telephone or other electronic communication with the customer(s).
- Arrangements for ensuring continuity of support, such as disaster recovery, security and backup.

Figure M-7 provides a description of the Operation Process as presented by IEEE/EIA Standard 12207.2.

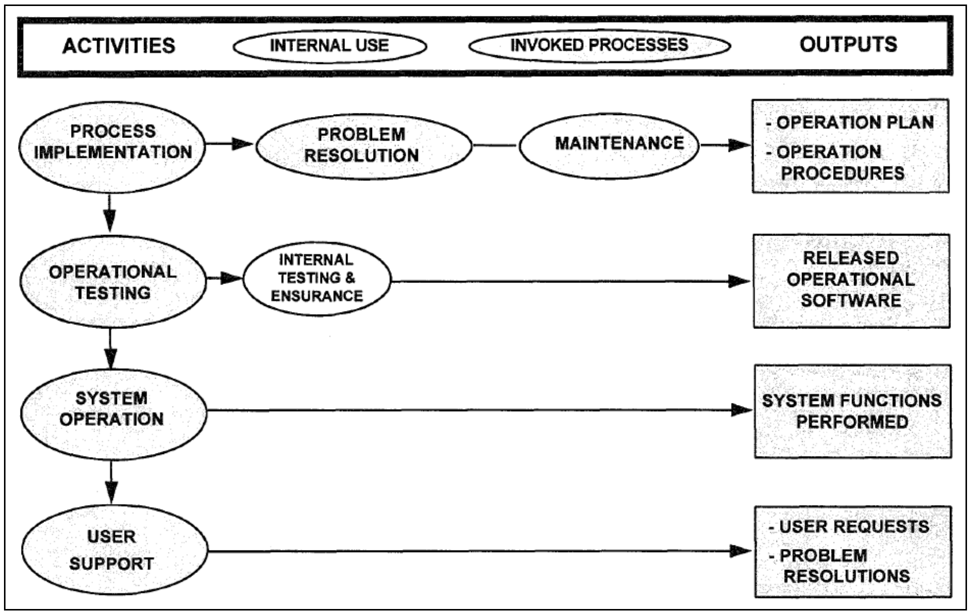


Figure M-7. IEEE/EIA Standard 12207.2 Operation Process[20]

[20] IEEE/EIA Standard 12207.2-1998.

M.2.5 Maintenance

The Maintenance Process contains the activities and tasks of the maintainer, the organization that provides the service of maintaining the software product; that is, managing modifications to the software product to keep it current and in operational fitness. The objective is to modify existing software product while preserving its integrity.

The maintainer manages the Maintenance Process at the project level following the Management Process, which is instantiated in this process; establishes an infrastructure under the process following the Infrastructure Process; tailors the process for the project following the Tailoring Process; and manages the process at the organizational level following the Improvement Process and the Training Process. Table M-8 describes the maintenance process objectives.

Maintenance Process Objectives
a) Define the impact of organization, operations, and interfaces on the existing system in operation;
b) Identify and update appropriate life cycle data;
c) Develop modified system components with associated documentation and tests that demonstrate that the system requirements are not compromised;
d) Migrate system and software upgrades to the user's environment;
e) Ensure fielding of new systems or versions does not adversely affect ongoing operations;
f) Maintain the capability to resume processing with prior versions.

Table M-8. Maintenance Process Objectives[21]

This process consists of the following activities:

- Process implementation;
- Problem and modification analysis;
- Modification implementation;
- Maintenance review/acceptance;
- Migration; and
- Software retirement.[22]

The requirements in support of the maintainer of both software products and services involved the support and control of design and development changes. The maintainer should be required to identify and manage records on all design and development changes. These changes must be reviewed, verified, validated and approved before implementation. Changes must be evaluated in terms of their effect on constituent parts and products already delivered. The results of the change review and subsequent follow-up actions must be maintained. Figure M-8 provides a description of the Maintenance Process as presented by IEEE/EIA Standard 12207.2.

[21] Practical Support for ISO 9001 Software Project Documentation.

[22] International Standard 9001.

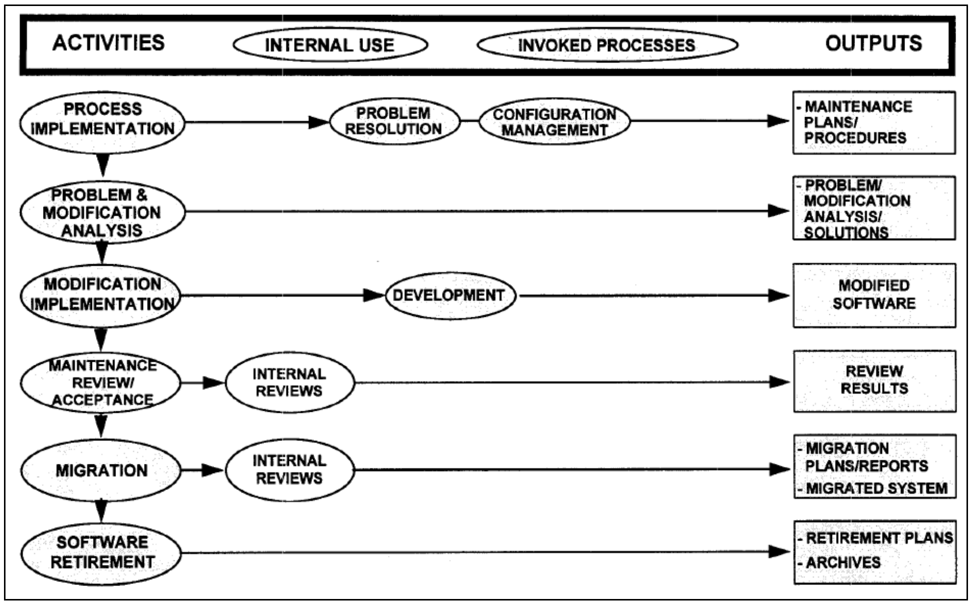


Figure M-8. IEEE/EIA Standard 12207.2 Maintenance Process[23]

M.3 IEEE/EIA Standard 12207 – Supporting Processes

IEEE/EIA Standard 12207 describes eight supporting processes that support the project from the organizational level:

- Documentation;
- Configuration management;
- Quality assurance;
- Verification;
- Validation;
- Joint review;
- Audit; and
- Problem resolution.

The organization employing and performing a supporting process:

- Manages it at the project level following the Management Process;
- Establishes an infrastructure under it following the Infrastructure Process;
- Tailors it for the project following the Tailoring Process;
- Manages it at the organizational level following the Improvement Process and the Training Process; and
- Assures its quality through Joint Reviews, Audits, Verification, and Validation processes.

[23] IEEE/EIA Standard 12207.2-1998.

M.3.1 Documentation

The Documentation Process is a process for recording information produced by a life cycle process or activity. This process contains the set of activities to plan, design, develop, produce, edit, distribute, and maintain those documents needed by all concerned such as managers, engineers, and users of the software product. Table M-9 provides a list of the Document Process objectives.

Document Process Objectives
a) Identify all documents to be produced by the process or project;
b) Specify the content and purpose of all documents and plan and schedule their production;
c) Identify the standards to be applied for development of documents;
d) Develop and publish all documents in accordance with identified standards and in accordance with nominated plans;
e) Maintain all documents in accordance with specified criteria.

Table M-9. Document Process Objectives[24]

M.3.2 Configuration Management

The Configuration Management Process works throughout the software life cycle to manage Configuration Item (CI) where the CI is defined within a configuration that satisfies an end use function and the CI can be uniquely identified at a given reference point. Table M-10 provides a list of the Configuration Management Process objectives:

Configuration Management Objectives
a) Identify, define, and control all relevant items of the project;
b) Control modifications of the items;
c) Record and report the status of items and modification requests;
d) Ensure the completeness of the items;
e) Control storage, handling, release, and delivery of the items.

Table M-10. Configuration Management Process Objectives[25]

M.3.3 Quality Assurance

The Quality Assurance (QA) Process provides adequate assurance that the software products and processes in the project life cycle conform to their specified requirements and adhere to their established plans. The QA process must assure that each process, activity, and task required by the contract or described in plans are being performed in accordance with the contract and with those plans. QA also provides assurances that each software product has undergone software product evaluation, testing, and problem resolution. The Quality Assurance process may make use of the results of other supporting processes, such as Verification, Validation, Joint Review, Audit, and Problem Resolution. Table M-11 provides a list of the Quality Assurance Process objectives:

[24] Practical Support for ISO 9001 Software Project Documentation.
[25] Practical Support for ISO 9001 Software Project Documentation.

Quality Assurance Process Objectives
a) Identify, plan, and schedule quality assurance activities for the process or product;
b) Identify quality standards, methodologies, procedures, and tools for performing quality assurance activities and tailor to the project;
c) Identify resources and responsibilities for the performance of quality assurance activities;
d) Establish and guarantee the independence of those responsible for performing quality assurance activities;
e) Perform the identified quality assurance activities in line with the relevant plans, procedures, and schedules;
f) Apply organizational quality management systems to the project.

Table M-11. Quality Assurance Process Objectives[26]

M.3.4 Verification

The Verification Process determines whether the software products of an activity fulfill the requirements or conditions imposed on them in the previous activities. This process may include analysis, review and test. Verification should be integrated, as early as possible, with the process that employs it. Verification is defined as confirmation by examination and provision of objective evidence that specified requirements have been fulfilled. Table M-12 provides a list of the Verification Process objectives.

Verification Process Objectives
a) Identify criteria for verification of all required work products;
b) Perform requirements verification activities;
c) Find and remove defects from products produced by the project.

Table M-12. Verification Process Objectives[27]

M.3.5 Validation

The Validation Process is a process for determining whether the requirements and the final system or software product fulfills its specific intended use. Validation is defined as confirmation by examination and provision of objective evidence that the particular requirements for a specific intended use are fulfilled. Validation is often performed by testing, conducted at several levels or approaches. Table M-13 provides a list of the Validation Process objectives.

Validation Process Objectives
a) Identify criteria for validation of all required work products;
b) Perform required validation activities;
c) Provide evidence that the work products, as developed, are suitable for their intended use.

Table M-13. Validation Process Objectives[28]

[26] Practical Support for ISO 9001 Software Project Documentation.
[27] Practical Support for ISO 9001 Software Project Documentation.
[28] Practical Support for ISO 9001 Software Project Documentation.

M.3.6 Joint Review

The Joint Review Process evaluates the status and products of an activity of a project as appropriate. Joint Reviews are at both project management and technical levels and are held throughout the life of the contract. Joint Reviews may include the customer and / or the supplier. Examples are project management reviews and technical reviews. Throughout the Development Process, the technical reviews complete the activities of:

- Software requirements analysis;
- Software architectural design;
- Software detailed design;
- Software integration; and
- Software acceptance support.

Table M-14 provides a list of the Joint Review Process objectives.

Joint Review Process Objectives
a) Evaluate the status and products of an activity of a process through joint review activities between the parties to a contract;
b) Establish mechanisms to ensure that action items raised are recorded for action.

Table M-14. Joint Review Process Objectives[29]

M.3.7 Audit

The Audit Process determines the degree of organizational and project compliance with the processes, requirements, plans, and contract. This process is employed by one party (auditing party) who audits the software products or activities of another party (audited party). The audit produces a list of detected issues or problems, which are recorded and entered into the Problem Resolution Process. Table M-15 provides a list of the Audit Process objectives.

Audit Process Objectives
a) Determine compliance with requirements, plans, and contract, as appropriate;
b) Arrange the conduct of audits of work products or process performance by a qualified independent party, as specified in the plans;
c) Conduct follow-up audits to assess corrective action(s), closure, and root cause actions.

Table M-15. Audit Process Objectives[30]

M.3.8 Problem Resolution

The Problem Resolution Process analyzes and resolves the problems (including non-conformances), whatever their nature or source, that are discovered during the execution of development, operation, maintenance, or other processes. Normally problems found in products under author control are resolved by the author, not in this process. Table M-16 provides a list of the Problem Resolution Process objectives.

[29] Practical Support for ISO 9001 Software Project Documentation.
[30] Practical Support for ISO 9001 Software Project Documentation.

Problem Resolution Process Objectives
a) Provide a timely, responsive, and documented means to ensure that all discovered problems are analyzed and resolved; b) Provide a mechanism for recognizing and acting on trends in problems identified.

Table M-16. Problem Resolution Process Objectives[31]

M.4 IEEE/EIA Standard 12207 – Organizational Processes

IEEE/EIA Standard 12207 describes four Organizational life cycle processes (Figure M-9): the Management Process, Infrastructure Process, Improvement Process, and Training Process. These processes work at the organizational level for all projects and should be in place prior to performing the five primary life cycle processes of Acquisition, Supply, Development, Maintenance, and/or Operation Processes and the eight Supporting processes. The organization employing and performing a Primary and Supporting process needs to:

- Manage it at the project level following the Management Process;
- Establish an infrastructure under it following the Infrastructure Process;
- Tailor it for the project following the Tailoring Process;
- Manage it at the organizational level following the Improvement Process and the Training Process; and
- Assure its quality through Joint Reviews, Audits, Verification, and Validation.

Organizational life cycle processes evaluate whether a new, changed, or outsourced process could be supported.

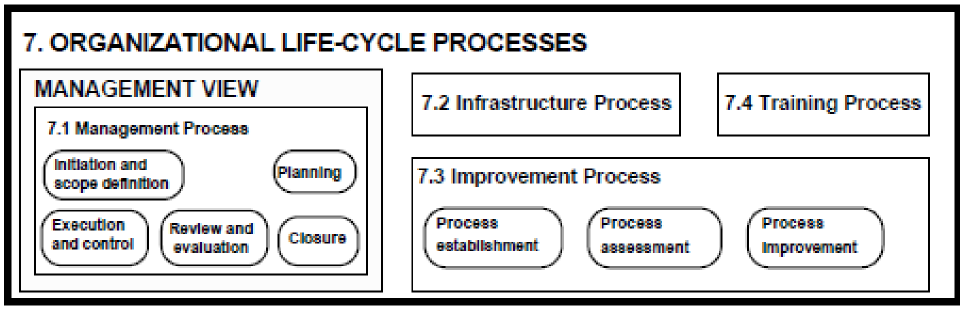


Figure M-9. IEEE/EIA Standard 12207 Organizational Life Cycle Processes[32]

M.4.1 Management

The Management Process contains the generic activities and tasks managing its respective processes. The manager is responsible for product management, project management, and task management of the applicable processes, such as the Acquisition, Supply, Development, Operation, Maintenance, or Supporting Process. This Table M-17 provides a list of the management process objectives.

[31] Practical Support for ISO 9001 Software Project Documentation.
[32] IEEE/EIA Standard 12207.0-1996.

Management Process Objectives
a) Define the project work scope;
b) Identify, size, estimate, plan, track, and measure the tasks and resources necessary to complete the project;
c) Identify and manage interfaces between elements in the project and with other projects and organizational units;
d) Take corrective action when project targets are not achieved;
e) Establish quality goals, based on the customer's quality requirements, for various checkpoints within the project's software life cycle;
f) Establish product performance goals, based on the customer's requirements, for various checkpoints within the project's software life cycle;
g) Define and use measures that reflect the results of project activities or tasks, at checkpoints within the project's life cycle, to assess whether the technical, quality, and product performance goals have been achieved;
h) Establish criteria, measures, and procedures for identifying software engineering practices and integrate improved practices into the appropriate software life cycle processes and methods;
i) Perform the identified quality activities and confirm their performance;
j) Take corrective action when technical, quality, and product performance goals are not achieved;
k) Determine the scope of risk management to be performed for the project;
l) Identify risks to the project as they develop;
m) Analyze risks and determine the priority in which to apply resources to mitigate those risks;
n) Define, implement, and assess appropriate risk mitigation strategies;
o) Define, apply, and assess risk measures to reflect the change in the risk state and the progress of the mitigation activities;
p) Establish an environment that supports effective interaction between individuals and groups;
q) Take corrective action when expected progress is not achieved.

Table M-17. Management Process Objectives[33]

M.4.2 Infrastructure

The Infrastructure Process is a process to establish and maintain the infrastructure needed for any other process. The infrastructure may include hardware, software, tools, techniques, standards, and facilities for development, operation, or maintenance. Table M-18 describes the Infrastructure Process objectives.

Infrastructure Process Objectives
a) Establishing and maintaining a well-defined software engineering environment, consistent with, and supportive of, the set of standard processes and organizational methods and techniques;
b) Tailoring the software engineering environment to the needs of the project and the project team;
c) Developing a software engineering environment that supports project team members regardless of the performance location of process activities;
d) Implementing a defined and deployed strategy for reuse.

Table M-18. Infrastructure Process Objectives[34]

[33] Practical Support for ISO 9001 Software Project Documentation.
[34] Practical Support for ISO 9001 Software Project Documentation.

M.4.3 Improvement

The Improvement Process is a process for establishing, assessing, measuring, controlling, and improving a software life cycle process. The Improvement Process uses software life cycle data as it provides a history of what happened during development and maintenance. Table M-19 describes the Improvement Process objectives.

Improvement Process Objectives
a) Establish a well-defined and maintained standard set of processes, along with a description of the applicability of each process;
b) Identify the detailed tasks, activities, and associated work products for each standard process, together with expected criteria;
c) Establish a deployed specific process for each project, tailored from the standard process in accordance with the needs of the project;
d) Establish and maintain information and data related to the use of the standard process for specific projects;
e) Understand the relative strengths and weaknesses of the organization's standard software processes;
f) Make changes to standard and defined processes in a controlled way;
g) Implement planned and monitored software process improvement activities in a coordinated manner across the organization.

Table M-19. Improvement Process Objectives[35]

M.4.4 Training

The Training Process provides and maintains trained personnel. As all software engineering processes are largely dependent upon knowledgeable and skilled personnel, , therefore, it is imperative that training be planned and implemented early so that trained personnel are available as the software product is acquired, supplied, developed, operated, or maintained. The Training Process objectives are described in Table M-20.

Training Process Objectives
a) Identify the roles and skills required for the operations of the organization and the project;
b) Establish formal procedures by which talent is recruited, selected, and transitioned into assignments in the organization;
c) Design and conduct training to ensure that all individuals have the skills required to perform their assignments;
d) Identify and recruit or train, as appropriate, individuals with the required skills and competencies to perform the organizational and project roles;
e) Establish a work force with the skills to share information and coordinate their activities efficiently;
f) Define objective criteria against which unit and individual training performance can be measured, to provide performance feedback, and to enhance performance continuously.

Table M-20. Training Process Objectives[36]

[35] Practical Support for ISO 9001 Software Project Documentation.
[36] Practical Support for ISO 9001 Software Project Documentation.

M.5 Summary

IEEE/EIA Standard 12207 provides a common framework supporting the joint communication between acquiring and supplying organizations and their expressed expectations for the performance of work. This standard describes the guidelines set for what are widely accepted as good principles and practices in support of software development. Most importantly, this standard can be used to define relations and negotiate agreements between all parties in the software life cycle.

If used by individual organizations, IEEE/EIA Standard 12207.0, and its supplemental guides, provide a basis for determining consistent and acceptable minimum levels of quality, performance, safety (low risk) and reliability.

References

IEEE/EIA 12207.0-1996; Industry Implementation of International Standard ISO/IEC 12207:1995 (ISO/IEC 12207) Standard for Information Technology-Software life cycle processes – Software Life Cycle Processes, IEEE/EIA 12207.0 (Mar 1998), IEEE Press, New York, NY, USA.

IEEE/EIA 12207.1-1997; Industry Implementation of International Standard ISO/IEC 12207:1995 (ISO/IEC 12207) Standard for Information Technology-Software life cycle processes – Software Life Cycle Data, IEEE/EIA 12207.0 (April 1998), IEEE Press, New York, NY, USA.

IEEE/EIA 12207.2-1998; Industry Implementation of International Standard ISO/IEC 12207:1995 (ISO/IEC 12207) Standard for Information Technology-Software life cycle processes – Implementation Considerations, IEEE/EIA 12207.0 (April 1998), IEEE Press, New York, NY, USA.

International Standard 9001, Quality management systems – Requirements, ISO 9001:2000(E), 2000, Switzerland.

Land, Susan K., and John W. Walz. Practical Support for CMMI®-SW Software Project Documentation – Using IEEE Software Engineering Standards. New York: Wiley-IEEE Computer Society Press, 2005.

Land, Susan K., and John W. Walz. Practical Support for ISO 9001 Software Project Documentation – Using IEEE Software Engineering Standards. New York: Wiley-IEEE Computer Society Press, 2006.

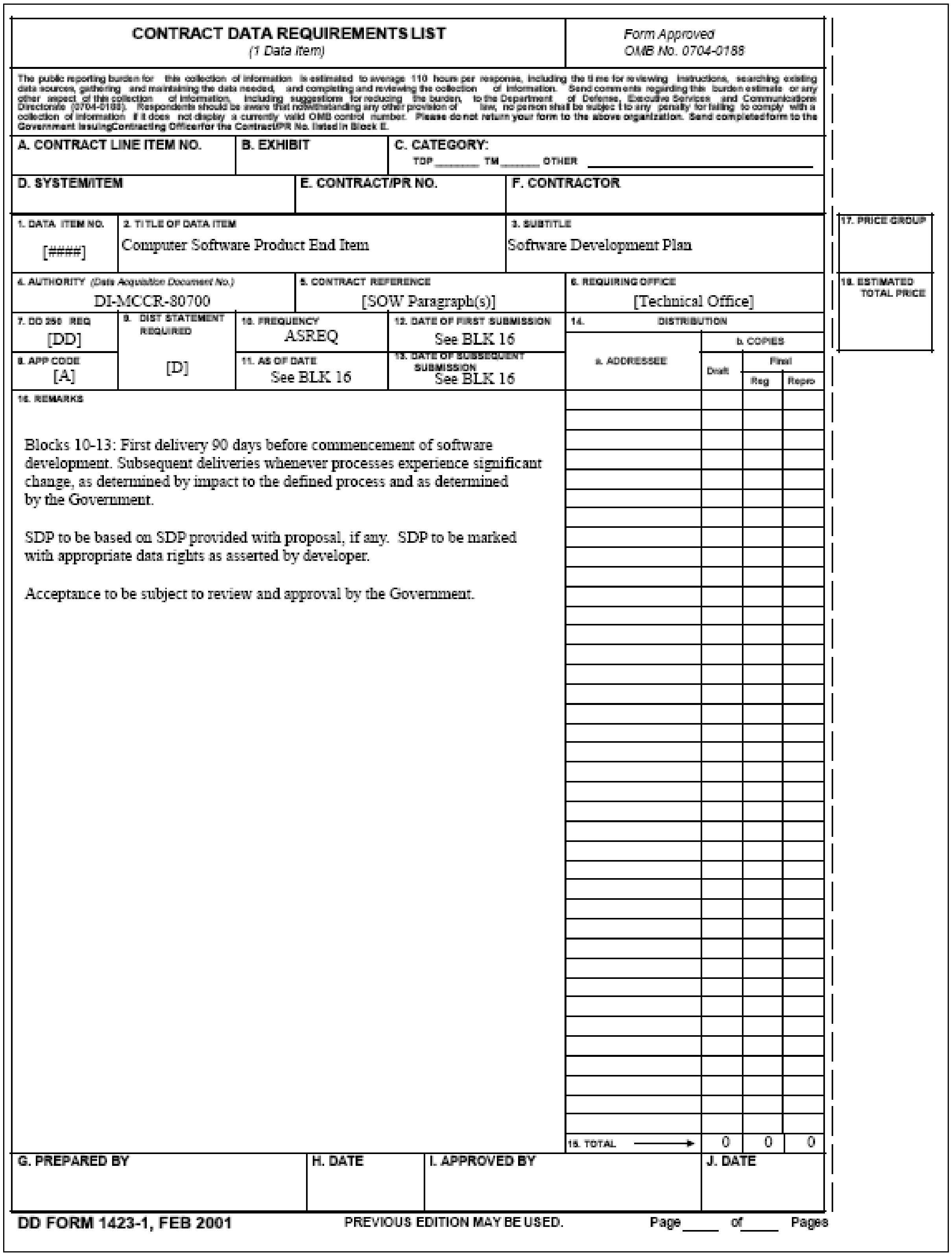
Croll, Paul, “How to Use Standards as Best Practice Information Aids for CMMI-Compliant Process Engineering”, 14th Annual DoD Software Technology Conference, USA, 2002.

IEEE, Software and Systems Engineering Standards Committee Charter Statement, http://standards.computer.org/sesc/index.htm.

N Appendix

Sample DD-1423s

Sample CDRL-SDP



CONTRACT DATA REQUIREMENTS LIST
(1 Data Item)

Form Approved
OMB No. 0704-0188

The public reporting burden for this collection of information is estimated to average 110 hours per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to the Department of Defense, Executive Services and Communications Directorate (0704-0188). Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. Please do not return your form to the above organization. Send completed form to the Government Issuing Contracting Officer for the Contract/PR No. listed in Block E.

A. CONTRACT LINE ITEM NO.	B. EXHIBIT	C. CATEGORY: TDP ______ TM ______ OTHER ______
D. SYSTEM/ITEM	E. CONTRACT/PR NO.	F. CONTRACTOR

1. DATA ITEM NO. [####]	2. TITLE OF DATA ITEM Computer Software Product End Item	3. SUBTITLE Software Development Plan	17. PRICE GROUP
4. AUTHORITY *(Data Acquisition Document No.)* DI-MCCR-80700	5. CONTRACT REFERENCE [SOW Paragraph(s)]	6. REQUIRING OFFICE [Technical Office]	18. ESTIMATED TOTAL PRICE
7. DD 250 REQ [DD]	9. DIST STATEMENT REQUIRED [D]	10. FREQUENCY ASREQ	12. DATE OF FIRST SUBMISSION See BLK 16
8. APP CODE [A]		11. AS OF DATE See BLK 16	13. DATE OF SUBSEQUENT SUBMISSION See BLK 16

14. DISTRIBUTION

a. ADDRESSEE	b. COPIES Draft	Final Reg	Final Repro
15. TOTAL →	0	0	0

16. REMARKS

Blocks 10-13: First delivery 90 days before commencement of software development. Subsequent deliveries whenever processes experience significant change, as determined by impact to the defined process and as determined by the Government.

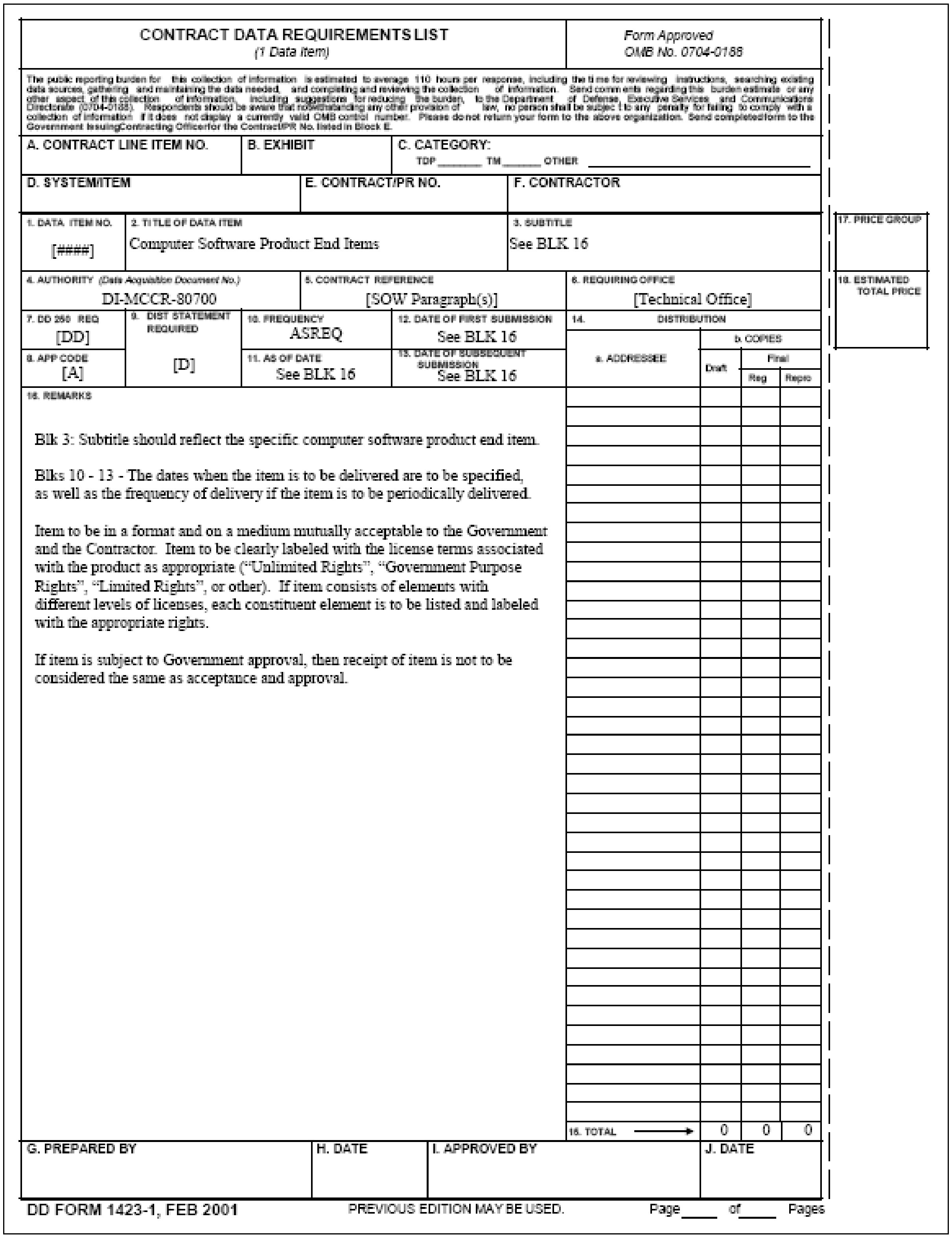
SDP to be based on SDP provided with proposal, if any. SDP to be marked with appropriate data rights as asserted by developer.

Acceptance to be subject to review and approval by the Government.

G. PREPARED BY	H. DATE	I. APPROVED BY	J. DATE

DD FORM 1423-1, FEB 2001 PREVIOUS EDITION MAY BE USED. Page ____ of ____ Pages

Sample CDRL-Software End Product



CONTRACT DATA REQUIREMENTS LIST
(1 Data Item)

Form Approved
OMB No. 0704-0188

The public reporting burden for this collection of information is estimated to average 110 hours per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to the Department of Defense, Executive Services and Communications Directorate (0704-0188). Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. Please do not return your form to the above organization. Send completed form to the Government Issuing Contracting Officer for the Contract/PR No. listed in Block E.

A. CONTRACT LINE ITEM NO.	B. EXHIBIT	C. CATEGORY: TDP _____ TM _____ OTHER _____
D. SYSTEM/ITEM	E. CONTRACT/PR NO.	F. CONTRACTOR

1. DATA ITEM NO. [####]	2. TITLE OF DATA ITEM Computer Software Product End Items	3. SUBTITLE See BLK 16
4. AUTHORITY *(Data Acquisition Document No.)* DI-MCCR-80700	5. CONTRACT REFERENCE [SOW Paragraph(s)]	6. REQUIRING OFFICE [Technical Office]

17. PRICE GROUP

18. ESTIMATED TOTAL PRICE

7. DD 250 REQ [DD]	9. DIST STATEMENT REQUIRED [D]	10. FREQUENCY ASREQ	12. DATE OF FIRST SUBMISSION See BLK 16	14. DISTRIBUTION
8. APP CODE [A]		11. AS OF DATE See BLK 16	13. DATE OF SUBSEQUENT SUBMISSION See BLK 16	

14. DISTRIBUTION			
	b. COPIES		
a. ADDRESSEE	Draft	Final	
		Reg	Repro
15. TOTAL →	0	0	0

16. REMARKS

Blk 3: Subtitle should reflect the specific computer software product end item.

Blks 10 - 13 - The dates when the item is to be delivered are to be specified, as well as the frequency of delivery if the item is to be periodically delivered.

Item to be in a format and on a medium mutually acceptable to the Government and the Contractor. Item to be clearly labeled with the license terms associated with the product as appropriate ("Unlimited Rights", "Government Purpose Rights", "Limited Rights", or other). If item consists of elements with different levels of licenses, each constituent element is to be listed and labeled with the appropriate rights.

If item is subject to Government approval, then receipt of item is not to be considered the same as acceptance and approval.

G. PREPARED BY	H. DATE	I. APPROVED BY	J. DATE

DD FORM 1423-1, FEB 2001 PREVIOUS EDITION MAY BE USED. Page____ of____ Pages

Sample Source Code Headers

Unlimited

```
////////////////////////////////////////////////////////////////////////////////
/// SECURITY CLASSIFICATION: UNCLASSIFIED
////////////////////////////////////////////////////////////////////////////////
Copyright (C) (Date & Company) (if required use this statement)
```

Notwithstanding any copyright notice, U.S. Government rights in this work are defined by DFARS 252.227-7013 or DFARS 252.227-7014 as detailed below. Use of this work other than as specifically authorized by the U.S. Government may violate any copyrights that exist in this work.

```
/// UNLIMITED RIGHTS
/// DFARS Clause reference: 252.227-7013 (a)(15) and 252.227-7014 (a)(15)
/// Unlimited Rights. The Government has the right to use, modify, reproduce, perform,
/// display, release or disclose this (technical data or computer software) in whole or in part, in
/// any manner, and for any purpose whatsoever, and to have or authorize others to do so.
///
/// Distribution Statement D. Distribution authorized to the Department of Defense and
/// U.S. DoD Contractors only in support of US DoD efforts. Other requests shall be
/// referred to JPEO JTRS.
///
/// Warning: - This document contains data whose export is restricted by the Arms Export
/// Control Act (Title 22, U.S.C., Sec 2751, et seq.) as amended, or the Export Administration
/// Act (Title 50, U.S.C., App 2401 et seq.) as amended. Violations of these export laws
/// are subject to severe criminal and civil penalties. Disseminate in accordance with
/// provisions of DoD Directive 5230.25.
```

Government Purpose Rights

///
/// SECURITY CLASSIFICATION: UNCLASSIFIED
///

Copyright (C) (Date & Company) (if required use this statement)

Notwithstanding any copyright notice, U.S. Government rights in this work are defined by 252.227-7013 (f)(2) and 252.227-7014 (f)(2) *as detailed below. Use of this work other than as specifically authorized by the U.S. Government may violate any copyrights that exist in this work.*

/// GOVERNMENT PURPOSE RIGHTS

///Rights in Technical Data, computer software & documentation in non-commercial items

///DFARS Clause: 252.227-7013 (f)(2) and 252.227-7014 (f)(2)

Government purpose rights. The Government's rights to use, modify, reproduce, release, perform, display, or disclose these technical data are restricted by paragraph (b)(2) of the Rights in Technical Data-Noncommercial Items clause contained in the below identified contract. No restrictions apply after the expiration date shown below. Any reproduction of technical data or portions thereof marked with this legend must also reproduce the markings.

Contract No.

Contractor Name

Contractor Address

Expiration Data

///
/// Distribution Statement D. Distribution authorized to the Department of Defense and
/// U.S. DoD Contractors only in support of US DoD efforts. Other requests shall be
/// referred to JPEO JTRS.
///
/// Warning: - This document contains data whose export is restricted by the Arms Export
/// Control Act (Title 22, U.S.C., Sec 2751, et seq.) as amended, or the Export Administration
/// Act (Title 50, U.S.C., App 2401 et seq.) as amended. Violations of these export laws
/// are subject to severe criminal and civil penalties. Disseminate in accordance with
/// provisions of DoD Directive 5230.25.

Specially Negotiated License Rights

//
/// SECURITY CLASSIFICATION: UNCLASSIFIED
//

Copyright (C) (Date & Company) (if required use this statement)

Notwithstanding any copyright notice, U.S. Government rights in this work are defined by 252.227-7013 (f)(2) and 252.227-7014 (f)(2) *as detailed below. Use of this work other than as specifically authorized by the U.S. Government may violate any copyrights that exist in this work.*

/// Specially Negotiated License Rights (Special GPR)

///Rights in Technical Data, computer software & documentation in non-commercial items

///DFARS Clause: 252.227-7013 (f)(2) and 252.227-7014 (f)(2)

The Government's rights to use, modify, reproduce, release, perform, display, or disclose these technical data and computer software are restricted by the specially negotiated Government Purpose Rights license contained in the below identified agreement at clause H- . Any reproduction of technical data or portions thereof marked with this legend must also reproduce the markings.

Contract No. (SRW Contract number)

Contractor Name: ITT Aerospace/Communications Division

Contractor Address: 1919 West Cook Road, Fort Wayne, IN 46801

Expiration Data: Perpetual.

///
/// Distribution Statement D. Distribution authorized to the Department of Defense and
/// U.S. DoD Contractors only in support of US DoD efforts. Other requests shall be
/// referred to JPEO JTRS.
///
/// Warning: - This document contains data whose export is restricted by the Arms Export
/// Control Act (Title 22, U.S.C., Sec 2751, et seq.) as amended, or the Export Administration
/// Act (Title 50, U.S.C., App 2401 et seq.) as amended. Violations of these export laws
/// are subject to severe criminal and civil penalties. Disseminate in accordance with
/// provisions of DoD Directive 5230.25.

Appendix O
Sample Best Value Checklist

This checklist provides a list of questions for consideration during the software source selection decision process to help government select the proposal that offers the “best value” to the government. “Best value” refers to the expected outcome of an acquisition that, in the government’s estimation, provides the greatest overall benefit to the government in response to the government’s requirement.

- Did you foster a pre-solicitation dialog with industry to:
  - Ensure a mutual understanding of the government’s need and industry’s capabilities;
  - Minimize inclusion of non-value added requirements; and
  - Promote a more effective best value process?
- Did you select to use best value to evaluate and compare factors in addition to cost in order to identify and select the most advantageous offer?
- Did you request only the information needed to evaluate proposals against the evaluation criteria. (You never ask for information you do not intend to evaluate.)
- Did you structure evaluation criteria and their relative order of importance to clearly reflect your needs and facilitate preparation of proposals that best satisfy that need?
- Were you sure to limit evaluation criteria to those areas which will reveal substantive differences or risk levels among competing offers?
- Did you clearly outline the basis for the best value decision in the solicitation?
- Are you sure you used cost or price as evaluation factors?
- Did you include the offerors’ relevant past performance as an evaluation factor?
- Did you ensure consistency among the objectives of the acquisition, the contracting strategy, the plan for selecting a source, the solicitation, and the evaluation and selection?
- Were discussions meaningful in that they identified to the offeror all deficiencies and significant weaknesses in the proposal, including any weaknesses that when accumulated, have a significant adverse impact on a proposal’s overall rating? (You want to avoid technical leveling or transfusion.)
- Are you sure that the best value decision:
  - Is based on a comparative analysis of the proposals;
  - Is consistent with stated evaluation criteria; and
  - Considers whether or not perceived benefits are worth any price premium?

- Did the team make the decision on a rational basis and set it forth in an independent, stand-alone defensible document?
- Were offerors debriefed promptly, at their request, as to the basis for the selection decision? (Candidly explain the results of the government's evaluation of their proposal without making any point-by-point comparisons with other proposals.)
- Did you release information on a fair and equitable basis consistent with regulatory and legal restrictions?