

Software Acquisition Best Practices: Experiences from the Space Systems Domain

30 September 2004

Prepared by

R. J. ADAMS, S. ESLINGER, K. L. OWENS, and M. A. RICH
Software Engineering Subdivision

Prepared for

SPACE AND MISSILE SYSTEMS CENTER
AIR FORCE SPACE COMMAND
2430 E. El Segundo Boulevard
Los Angeles Air Force Base, CA 90245

Engineering and Technology Group

This report was submitted by The Aerospace Corporation, El Segundo, CA 90245-4691, under Contract No. FA8802-04-C-0001 with the Space and Missile Systems Center, 2430 E. El Segundo Blvd., Los Angeles Air Force Base, CA 90245. It was reviewed and approved for The Aerospace Corporation by M. A. Rich, Principal Director, Software Engineering Subdivision. Michael Zambrana was the project officer for the Mission-Oriented Investigation and Experimentation (MOIE) program.

This report has been reviewed by the Public Affairs Office (PAS) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nationals.

This technical report has been reviewed and is approved for publication. Publication of this report does not constitute Air Force approval of the report's findings or conclusions. It is published only for the exchange and stimulation of ideas.

A handwritten signature in cursive script, reading "Michael Zambrana", is positioned above a horizontal line.

Michael Zambrana
SMC/AXE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. 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 ADDRESS.

1. REPORT DATE (DD-MM-YYYY) 30-09-2004		2. REPORT TYPE		3. DATES COVERED (From - To)	
4. TITLE AND SUBTITLE Software Acquisition Best Practices: Experiences from the Space Systems Domain				5a. CONTRACT NUMBER FA8802-04-C-0001	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) R. J. Adams, S. Eslinger, K. L. Owens, and M. A. Rich				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) The Aerospace Corporation Software Engineering Subdivision El Segundo, CA 90245-4691				8. PERFORMING ORGANIZATION REPORT NUMBER TR-2004(8550)-1	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Space and Missile Systems Center Air Force Space Command 2450 E. El Segundo Blvd. Los Angeles Air Force Base, CA 90245				10. SPONSOR/MONITOR'S ACRONYM(S) SMC	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S) SMC-TR-05-02	
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT This report describes a comprehensive set of software acquisition best practices that the Software Acquisition MOIE research team has identified based on their experience with numerous space programs over many years. These best practices address pre-contract award activities, post-contract award activities, and full life cycle activities for the acquisition of large, complex, software-intensive systems.					
15. SUBJECT TERMS Software, Software acquisition, Best practices, Software engineering, Software risk management					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES 34	19a. NAME OF RESPONSIBLE PERSON Suellen Eslinger
a. REPORT UNCLASSIFIED	b. ABSTRACT UNCLASSIFIED	c. THIS PAGE UNCLASSIFIED			19b. TELEPHONE NUMBER (include area code) (310)336-2906

Acknowledgements

The preparation of this report was funded by the Mission-Oriented Investigative Experimentation (MOIE) program's Software Acquisition task. The authors wish to thank their USAF Space and Missile Systems Center counterpart and MOIE sponsor, Mike Zambrana (SMC/AXE, Directorate of Systems Engineering) for his valuable input and support.

The authors also wish to thank the following members of the Software Engineering Subdivision for graciously providing their time and effort to review the paper and provide valuable input.

Linda A. Abelson, Engineering Specialist, Software Acquisition and Process Office

John C. Cantrell, Senior Engineering Specialist, Software Architecture and Engineering Department

Peter Hantos, Senior Engineering Specialist, Software Acquisition and Process Office

Contents

1. Introduction.....	1
2. Definition of Software Acquisition Best Practices	3
2.1 Software Acquisition Processes.....	3
2.2 Characteristics of Software Acquisition Best Practices.....	4
3. The Space Systems Software Acquisition Best Practices Roadmap	5
3.1 Pre-Contract Award Space System Software Acquisition Best Practices	5
3.1.1 Establishing the Program Baseline	6
3.1.2 Obtaining Contractual Insight.....	9
3.1.3 Obtaining Contractual Commitment.....	11
3.1.4 Selecting a Capable Software Contractor Team	13
3.1.5 Providing Tools for Contract Management	16
3.1.6 A Software Acquisition Best Practice Contract.....	18
3.2 Post-Contract Award Space System Software Acquisition Best Practices.....	19
3.2.1 Performing Technical Product Reviews	19
3.2.2 Performing Software Process Reviews.....	21
3.2.3 Managing the Contract.....	22
3.3 Full Life Cycle Software Acquisition Best Practices	25
4. Conclusions.....	29
References.....	31
Acronyms and Abbreviations	33

Figures

1. Software acquisition and software engineering domains.....	3
2. Space systems software acquisition best practice roadmap.....	5
3. A “typical” software acquisition best practice contract.....	18

1. Introduction

Prior to the acquisition reform movement of the mid to late 1990s, DoD system acquisition used a highly structured paradigm based upon detailed system requirements specifications, military standards compliance, and formalized deliverable technical documentation using detailed Data Item Descriptions (DIDs), which often required Government approval. During the 1990's acquisition reform movement, the acquisition environment underwent a radical change from this very structured paradigm to a "faster, better, cheaper" philosophy based upon high-level system objectives and performance specifications, commercial standards, contractor-determined processes, and few, if any, deliverable technical documents.

The lessons learned over many years of system acquisition had been captured in the military specifications, standards, and DIDs in use prior to the 1990's acquisition reform movement. These lessons were essentially discarded by acquisition reform with the cancellation of military standards, the prohibition of the use of any process standards on contracts, the deletion of virtually all technical deliverable documentation, and the introduction of new concepts such as Total System Performance Responsibility (TSPR). The acquisition reform practices were based on the belief that eliminating perceived excessive contractual requirements would result in the achievement of the "faster, better, cheaper" objectives. Recent experiences with failures on software-intensive acquisition reform programs have proved this belief to be false.

Acquisition policy has recently undergone another major change with the approval of a new series of DoD 5000 policy documents, a new capabilities generation process (replacing the former requirements generation process), and new acquisition policy unique to space systems. Many previous restrictions imposed under acquisition reform have been lifted. During the transition to these changes, space system programs began asking for guidance on the "right" set of system acquisition practices that they should be using.

The acquisition of large, complex, software-intensive systems has historically been fraught with major problems, including performance deficiencies, extensive software defects, and cost and schedule overruns. With the focus on new acquisition practices, the opportunity exists for identifying and implementing a comprehensive set of software acquisition best practices designed to reduce risk in the acquisition of software-intensive systems. The solution, however, is not as simple as returning to the practices in use prior to acquisition reform. This report describes a comprehensive set of software acquisition best practices developed by the authors based on experiences in the space systems domain.

2. Definition of Software Acquisition Best Practices

2.1 Software Acquisition Processes

The term “software acquisition” is defined to mean the set of processes (i.e., the methods, tools, techniques, procedures, etc.) used by the Government to acquire the software portion of software-intensive systems. Software acquisition covers the activities of the entire program life cycle, from the identification of needed capability through system retirement, including pre-contract award and post-contract award activities through all program life cycle phases. The term “software engineering,” on the other hand, is defined to be the set of processes used by the developers to build the software portion of software-intensive systems. Figure 1 illustrates the different domains of software acquisition and software engineering.

One of the principal components of a successful software development project is the quality of the software engineering processes used. This statement is based on the well-established fact that the quality of a software product is highly dependent upon the quality of the processes used to develop and maintain that product [Paulk, M. et al., 1994, p. 8]. However, software acquisition processes are also very influential in achieving a successful software development project. The software acquisition processes used can positively encourage, or adversely constrain, the developers in their application of high-quality software engineering processes to a software development effort. The application of software acquisition best practices, therefore, can reduce risk in the acquisition of software-intensive systems.

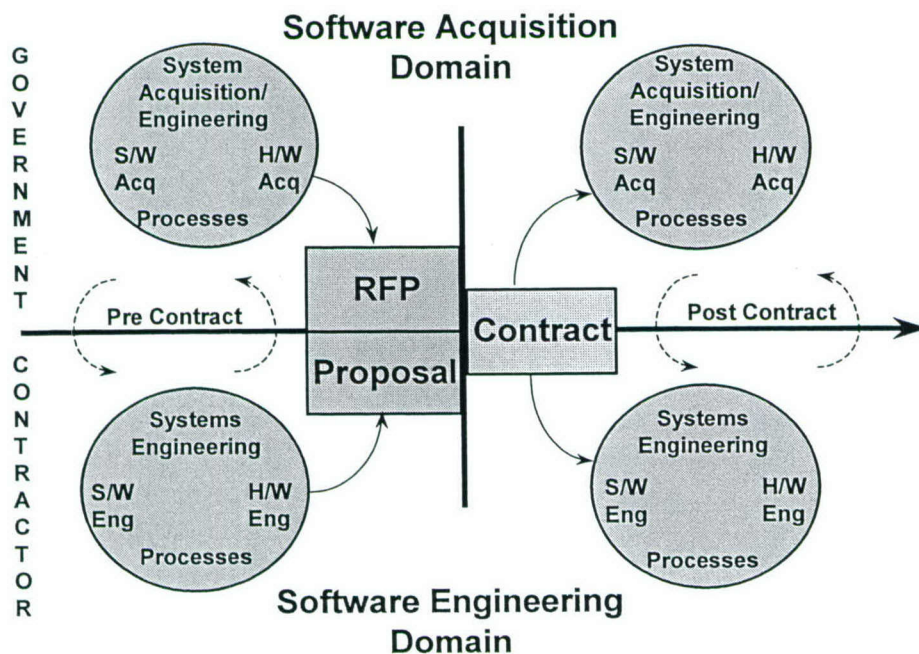


Figure 1. Software acquisition and software engineering domains.

2.2 Characteristics of Software Acquisition Best Practices

Software acquisition best practices are, by definition, practices that people with recognized software acquisition expertise have identified through experience as being significant contributors to the successful acquisition of software-intensive systems. Both negative experiences and positive experiences on past programs can be used to identify software acquisition best practices. However, care must be taken when using negative experiences so as not to be trapped by logical fallacies. If practice “A” is used, and the resulting experience “B” is not desirable, this does not imply that if practice “A” is not used, then “B” will not occur and the resulting experience will be desirable. ($A \Rightarrow B$ is not equivalent to $\text{Not } A \Rightarrow \text{Not } B$.)

Since best practices are experientially based, a best practice has never been proven to be “best” in any analytical sense. Best practices also never form an exhaustive set; there is always the possibility of additional best practices being identified. In addition, they are not static. Best practices change based on new experiences and new technologies.

A comprehensive set of software acquisition best practices must provide a consistent and integrated approach to software acquisition throughout the program life cycle, both pre- and post-contract award. In addition, because software always exists within the context of the system, the software acquisition best practices must be consistent and integrated with a comprehensive set of system acquisition best practices. Finally, the set of best practices must be suitable for acquiring today’s complex software systems that will be developed using the latest software development process and product technologies.

Today’s software-intensive space systems are large systems with multiple-satellite constellations and multiple ground elements, frequently worldwide. These systems involve complex combinations of hardware and software with complex external and internal interfaces. They are usually unprecedented and have high reliability and integrity requirements. The size of the software in space systems now under development is on the order of 10^5 Source Lines of Code (SLOC) onboard and 10^6 to 10^7 SLOC on the ground. Space systems software acquisition best practices must be able to effectively support the acquisition of these systems.

A comprehensive set of space system software acquisition best practices satisfying the above criteria has been developed by the authors based on experiences from the space systems domain. This set of software acquisition best practices was synthesized from the authors’ experiences supporting the United States Air Force (USAF) Space and Missile Systems Center (SMC) and the National Reconnaissance Office (NRO) in the acquisition of software-intensive space systems over a 20-year period. The authors have over 60 collective years of experience in the acquisition of software-intensive space systems while at The Aerospace Corporation, in addition to substantial prior experience in industry developing space and similar software-intensive systems.

3. The Space Systems Software Acquisition Best Practices Roadmap

The recommended set of 24 space system software acquisition best practices is organized into ten categories, as shown in Figure 2. Five of these categories address pre-contract award activities, three address post-contract award activities, and two cover activities performed throughout the entire pre- and post-contract award periods.

Each of these categories of best practices is described in more detail below. This report, however, is not intended to be a tutorial in the application of these best practices. Rather, it briefly describes the best practices, the rationale for their inclusion, and essential elements for their effective application. It should be noted that the recommended software acquisition best practices are not independent of each other; in fact, effective use of a particular software acquisition best practice may require concurrent use of other software acquisition best practices.

3.1 Pre-Contract Award Space System Software Acquisition Best Practices

The pre-contract award space system software acquisition best practices address the following areas: establishing the program baseline, obtaining contractual insight, obtaining contractual commitment, selecting a capable contractor team, and providing contract management tools. The best practices in each of these areas are discussed in the following paragraphs. In addition, a typical software acquisition best practice contract that embodies these best practices is described.

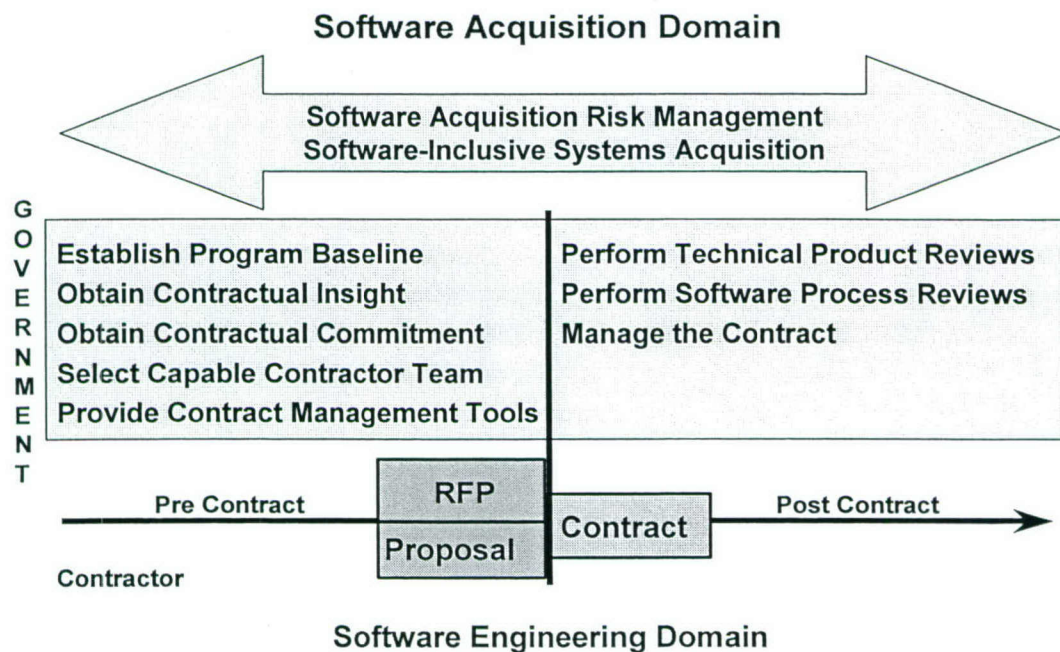


Figure 2. Space systems software acquisition best practice roadmap.

3.1.1 Establishing the Program Baseline

There are three software acquisition best practices associated with establishing the program baseline as described below.

Include Software in System Performance Requirements

This best practice involves the inclusion of software in the specification of the system performance requirements for the software-intensive system to be developed. Examples of performance requirements that should have both hardware and software components specified are specialty engineering requirements (including dependability, reliability, maintainability, and availability (DRMA); supportability, including testability and integrated system diagnostics; safety; security; and human systems integration); Key Performance Parameters (KPPs) (e.g., mission performance timelines, accuracies); computer resource reserves (especially onboard margins for processor throughput, memory, storage, and communication bandwidth); and interoperability (including open systems interface requirements). The most effective application of this best practice requires the participation of acquisition team² personnel knowledgeable in specifying software-inclusive system performance requirements in the system performance requirements definition process.

Since the system performance requirements become the contractual requirements, it is very important that they have a complete system perspective, including both hardware and software. The system performance requirements should not be specified so as to only reflect the hardware contribution. For example, including software in the system availability requirements will help ensure that the availability actually experienced during operations will meet the specified requirements. When availability requirements include only hardware, the operationally experienced availability will be significantly less than specified in the requirements due to failures or inefficiencies caused by software. This can result in the software-intensive system not being suitable for operations.

Perform Software Architecture-Inclusive Trade Studies

This best practice involves the inclusion of the software architecture along with and as an integral part of the system architecture trade studies. Architectural decisions made in isolation from the software implications of those decisions can adversely affect program executability in terms of the ability to develop the required software within the program's cost and schedule baseline. Decisions that involve software as well as hardware include space-ground and hardware-software allocation trades, onboard computer hardware sizing, and reuse of major legacy system components.

² The Government software acquisition team for most space systems consists of Government personnel (military and civilian), Federally Funded Research and Development Center (FFRDC) personnel and personnel from Systems Engineering and Technical Assistance (SETA) contractors. To simplify terminology, the term "acquisition team" will be used throughout this report to refer to members of the Government software acquisition team, unless the context requires different terminology for clarity.

Of particular importance is performing a cost-benefit trade study for reusing legacy software that considers architectural constraints of the legacy software, re-engineering needed, and life cycle maintenance implications. The most effective application of this best practice requires the participation of acquisition team personnel knowledgeable in space system software architecting along with the space system hardware-knowledgeable engineers.

Including software architecture as an integral part of the system architecture trade studies supports the establishment of a robust, integrated Government hardware/software architecture baseline, which is essential in order to obtain an accurate cost and schedule estimate for the entire system to be developed. If the hardware and software architectures are not integrated and consistent with each other, the cost and schedule estimates most likely will be too low since there is a high probability that an inconsistent, non-integrated architecture will not satisfy all of the system requirements. The software portion of the integrated hardware/software architecture baseline must address Commercial Off-the-Shelf (COTS), reuse and newly developed software for all space and ground elements.

Determine Realistic, Independent Baseline Software Estimates

This best practice involves making realistic software size, effort, cost, and schedule estimates based upon the system's software architecture, valid software historical data, and the appropriate use of software cost models. The contractual cost and schedule constraints imposed by the Government upon the contractor must then be based upon these realistic software cost and schedule estimates.

In the program's early acquisition life cycle phases, software cost and schedule estimates should be based upon the acquisition team's software architecture baseline and historical data collected on similar past programs. Later in the acquisition life cycle, updates to the Government's software cost and schedule estimates can incorporate the contractor's³ evolving software architecture and judicious use of data collected from the actual software development effort. Care should be taken to ensure that the Government's estimates include all work that must be accomplished and all costs that must be incurred in order to develop the software portion of the software-intensive system for the entire development life cycle, from system requirements definition through transition to operations and maintenance. This includes efforts for developing new software, re-engineering reuse software, and integrating the new, reuse, and COTS software into the system. Work not easily estimated by standard software cost models must also be included (e.g., graphical user interface screen development, database population, knowledge base population). Finally, realistic estimates of cost and schedule risk must be included in the cost and schedule estimates based on the

³ Large, complex space systems usually require a large contractor team consisting of the prime contractor and multiple team members (e.g., subcontractors, other intra-corporate organizations, vendors). To simplify terminology, the term "contractor" will be used throughout this report to refer to the entire contractor team, unless the context requires different terminology for clarity.

uncertainty in the parameters input to the estimating process. Software cost and schedule risk are always large in the program's early acquisition phases when it is not possible to accurately estimate the software size.

An important part of this best practice is that the Government's software cost and schedule estimates should be independent; that is, the acquisition team should perform its own software cost and schedule estimation and not rely solely upon the contractor's estimates. Data from the contractor should provide important input into the acquisition team's estimation process. However, the independence of the acquisition team's estimates is necessary to eliminate the biases present in the contractor's estimates (e.g., underestimation of amount of new code to be developed, overestimation of amount of COTS and reuse code, overly optimistic parameters used in the software cost models, and overly optimistic productivity data).

Realistic software cost and schedule estimation not only increases the predictability of the cost and schedule, and, therefore, decreases the probability of overruns; it also contributes to improved quality of the software product. Sufficient time and effort to develop the software products, including performing the quality checks and correcting any identified defects, are essential to reducing downstream rework. When extreme cost or schedule constraints are imposed upon the developer, neither the cost nor the schedule is likely to be met. In addition, the resulting cost or schedule pressure will cause poor software product quality since the contractual cost or schedule constraints will force the developer to take shortcuts in the software development processes. According to the DoD Software Program Manager's Network, "attempts to compress a schedule to less than 80% of its nominal schedule aren't usually successful" [DoD Software Program Manager's Network, p.22].

Realistic software cost and schedule estimation is especially critical for defining a feasible evolutionary acquisition strategy. Defining the capabilities allocated to the evolutionary blocks without consideration of the software that must be developed to support each block can lead to a program that is not executable. For space systems in particular, defining the capabilities strictly in terms of the space segment can lead to a situation where nearly all of the ground software (millions of SLOC) must be developed in the first evolution to support the first block of satellites, resulting in severe up-front cost and schedule constraints for this software that cannot be met.

3.1.2 Obtaining Contractual Insight

There are three software acquisition best practices associated with obtaining contractual insight as described below.

Require Key Software Technical and Management Deliverables

This best practice involves requiring key software technical and management products as contract deliverables (i.e., as Contract Data Requirements List (CDRL)

items). Requiring this documentation to be delivered ensures that the contractor actually performs the work necessary to prepare these critically important products and that the contractor performs this work according to a minimum defined standard (i.e., the DID). It also ensures that the acquisition team has the visibility it needs into the evolving state of the software development effort and enables the acquisition team to perform in-depth technical reviews of the delivered products.

Clearly not all software products should be formally delivered to the Government by the contractor. The software products recommended for formal delivery consist of those with the highest risk reduction potential. Three types of software products are recommended for formal delivery. The first type consists of plans and reports that provide the acquisition team with a thorough understanding of how the software development project is being managed and of what the actual state of the software development effort is throughout the life cycle. The Software Development Plan (SDP), the most important of the planning documents, should be an integrated plan covering all of the software development being performed by all members of the contractor team. Other important planning documents are the Software Master Build Plan, which specifies the contents of each build of software when using iterative software development life cycle models (e.g., the spiral model), and the Software Transition Plan, which describes the planning for transitioning the software to operations and maintenance. Periodic metrics reports (e.g., monthly) in addition to other periodic management reporting provide additional insight into the status of the software development effort and the state of the evolving software product.

The second type of deliverable software products consists of the software requirements, architecture, and test products. Delivery of these technical products reduces risk because of increased contractor emphasis on performing the activities to produce these products and because of expected defect reduction from acquisition team technical reviews. Since the software requirements drive the entire software development effort, specifying and documenting the software requirements (including the software interface requirements) in a clear and well-organized manner will assist both the contractor and the acquisition team in analyzing these requirements for correctness, completeness, and consistency. A well-defined and documented software architecture, including multiple architectural views that describe various aspects of the architecture, is another critically important technical product for risk reduction since the software architecture provides the framework for all subsequent software design and development. Similarly, developing formal verification test plans, procedures, and reports for the software and interface requirements helps ensure that these requirements are being adequately verified, and thus reduces risk by helping to ensure that the software performs as specified.

The third type of deliverable software products consists of documented information essential for delivery, installation, operations, and maintenance of the software products. The exact set of documents needed in these areas is program-unique, depending upon the requirements of the users and operators for operational documentation, the acquisition strategy for software maintenance, and the site requirements for delivery

and installation. Examples of these products are Version Description Documents, Software Installation Guides, Software User's Manuals, Operator Positional Handbooks, Software Maintenance Manuals, Software Product Specifications, and training materials.

Change control and baseline management of these important products are critical contractor activities. It is especially important that changes to the software requirements, architecture, and test products be strictly managed as the software development effort proceeds. Deliveries of updates to the deliverable software products at significant milestones and as changes occur should be required in the contract.

Requiring Government approval for deliverable products is an additional mechanism that can be used to help ensure that the contractor produces high-quality deliverable software products. The following types of software products, at a minimum, are recommended for approval coding: plans (especially the Software Development Plan and Software Transition Plan), requirements (Software and Interface Requirements Specifications), test documents (Software Test Plans and Reports), and operations and maintenance documentation (e.g., the Software Product Specifications and Software User's Manuals).

Electronic delivery is the preferred mechanism for all deliverable products. The following best practice discusses this in more detail.

Require Timely Electronic Access to All Software Products

There are a large number of software products prepared during any software development effort that are not included in the set recommended for delivery. Some examples of these products are the detailed design information, the evolving software code, unit test documentation, build scripts, and software integration test documentation. In addition, all software products (both deliverable and non-deliverable) generally go through interim versions that are internally reviewed before the product is finalized. A mechanism needs to exist whereby the acquisition team can examine any interim or final software products without having them formally delivered. On most development efforts, there are critical, high risk areas of the software where evaluation of the non-deliverable software products is needed. In addition, there is frequently the need for the acquisition team to examine areas of the software where significant problems have been found.

The recommended mechanism for achieving this insight is to include an electronic access clause in the contract requiring the contractor to establish a mechanism for the customer to be able to access any technical or management interim or final products produced under the contract. The electronic access clause needs to also require that the Government have timely access to these products since any significant delay to access impedes the potential risk reduction benefit of the acquisition team's review.

Such clauses are in common use at SMC. In addition, the contract needs to require all CDRL items to be delivered electronically.

Require Software Level Technical and Management Reviews

This best practice involves contractually requiring software-level technical and management reviews in addition to the system- and program-level reviews. Most current contracts for space systems require major system reviews (e.g., System Design Review (SDR), System Preliminary Design Review (PDR), System Critical Design Review (CDR), System Test Readiness Review (TRR)). These major system reviews provide little opportunity to review the details of the software architecture, design, and test readiness.

Software-level technical reviews are needed to ensure that all stakeholders in the software development (e.g., system users and operators; the acquisition team; contractor systems engineers; and contractor hardware, software, and test engineers) receive technical information about the evolving software products at a sufficient level of detail to find defects and reduce risk. Most software development is now performed using iterative software life cycle models, such as the incremental or spiral models. Using iterative life cycle models, the software is developed in a sequence of builds, each of which adds capability to the preceding build, and each build consists of portions of the Computer Software Configuration Items (CSCIs). Software-level technical reviews, therefore, need to be performed for each build, rather than for completed CSCIs.

Periodic software-level management reviews are also important for understanding the actual state of the evolving software product. Program management reviews and Integrated Product Team (IPT) meetings too frequently have the unfortunate characteristic of being hardware focused. Contractually requiring software-level management reviews will ensure that the program management reviews and IPT meetings include software to a level of importance commensurate with its risk.

3.1.3 Obtaining Contractual Commitment

There are two software acquisition best practices associated with obtaining contractual commitment as described below.

Mandate Compliance With a Robust Full Life Cycle Software Development Standard

This best practice involves contractually mandating compliance with a robust full life cycle standard for software development that is suitable for developing software-intensive systems with high reliability and integrity requirements. A robust full life cycle software process standard defines the software development life cycle activities that must be performed and the tasks that must be accomplished for each of those activities. It also specifies the required minimum content of the most important soft-

ware development products produced by those activities. Rigorous adherence to a robust full life cycle software standard can result in a higher quality software product since it increases the probability that all necessary software engineering tasks will be accomplished and all necessary engineering work will be performed. Requiring contractual compliance with such a standard reduces the possibility of the contractor cutting corners that would result in increased risk to achieving a successful software acquisition.

A joint SMC/NRO Software Process IPT, as part of the SMC/NRO Mission Assurance Improvement Task Force, has recently completed a study of full life cycle software development standards. Based on this study, EIA/IEEE J-STD-016-1995, the commercial version of MIL-STD-498, was recommended for use as a compliance document on SMC and NRO contracts for software-intensive space systems. This standard specifies a rigorous set of software development activities and products that are suitable for developing high-assurance software, such as that in space systems. Since this standard specifies only “what” must be performed, not “how” it must be performed, it can be used with any software development methods and tools. The software Process IPT also developed tailoring that should be applied when using J-STD-016 on contracts [Adams, R. J. et al., 2004a]. In addition, the IPT prepared a software standard based on MIL-STD-498 with this tailoring incorporated that can be used on SMC and NRO contracts instead of the tailored J-STD-016 [Adams, R. J. et al., 2004b].

Require Contractor Commitment to the Software Development Plan

This best practice involves having the contractor contractually commit to following their SDP. This can be accomplished contractually in several ways. The SDP can be cited as a compliance document in the contractor’s Integrated Management Plan (IMP) or in the Statement of Work (SOW). Both the IMP and the SOW eventually become part of the contract. With a contractually compliant SDP, the initial version and any subsequent modifications should require approval by the Government before being put on the contract. This will help ensure that the SDP defines sufficiently high-quality software engineering processes to support development of the required software.

The SDP describes the contractor’s plans for accomplishing the software portion of the contract and contains the results of both qualitative and quantitative planning for the software development effort. It identifies the software to be built and the characteristics of that software that drive the planning process (e.g., size, type). It describes the technical and management activities to be performed, and the processes, methods, and tools to be used for each. It specifies the life cycle model (or models) to be used and their associated sequencing of activities. It allocates roles and responsibilities for all of the software development work to be performed. The SDP also provides (usually by reference to information stored in program management tools) the software development top level and detailed schedules with critical paths and the required effort by job category over time. In addition, it describes any other resources (e.g.,

facilities, computer hardware, tools) necessary to perform the software development. The SDP needs to be an integrated plan covering all of the software development being performed by members of the contractor team. It is especially important that the SDP specify a well-defined set of processes for those activities that cross team member boundaries.

The SDP thus contains the details as to how the contractor intends to perform the required software development effort and is complementary to the full life cycle standard described above, which specifies what must be performed. Since the quality of the software products is highly dependent upon the quality of the processes used to develop them, adherence to mature, well-disciplined software engineering processes is essential to delivery of a high-quality software-intensive system. Contractual commitment to both the robust full life cycle standard and the SDP helps to ensure that the contractor will consistently use high-quality software engineering processes throughout the contract.

3.1.4 Selecting a Capable Software Contractor Team

There are three software acquisition best practices associated with selecting a capable software contractor team as described below.

Perform a Software Capability Appraisal as Part of the Source Selection

This best practice consists of performing a formal appraisal of the contractor's software development capability as part of the source-selection process. The primary purpose for performing a software capability appraisal is to increase the likelihood of selecting a contractor team that is capable of developing the required software within the program constraints. It is well established that risk in software development is reduced by selecting a contractor with mature software engineering processes. A secondary purpose for performing a software capability appraisal is to identify risks associated with the selected contractor to facilitate managing these risks beginning at contract award. Another secondary purpose is to obtain a contractual commitment from the selected contractor to adopt processes that instill and support effective software engineering discipline.

For a software capability appraisal to be effective as a discriminator among the offerors, the appraisal must occupy a position of sufficient importance in the source selection evaluation criteria. It is strongly recommended that the software capability appraisal be its own separate subfactor within the Mission Capability factor, and that the weight of this subfactor be commensurate with the program's software risk. This position of software in the Mission Capability factor also allows the Past Performance factor to include relevant software development performance for the prime contractor and significant software team members.

Two principal methods have been used in the past for performing formal software capability appraisals: the Software Engineering Institute's (SEI's) Software Capability Evaluation (SCESM), which is based upon the Capability Maturity Model[®] for Software (SW-CMM[®]), and the USAF's Software Development Capability Evaluation (SDCE).³ Both methods have proven to be effective tools for obtaining insight into the offerors' software development processes, and both methods provide strengths, weaknesses, and risks for use in the source selection evaluation.

Recently the new Capability Maturity Model Integration[®] (CMMI[®]) models were introduced that are replacing the SW-CMM.⁴ A new appraisal method is used with the CMMI models, the Standard CMMI Appraisal Method for Process Improvement (SCAMPISM).⁵ The currently defined SCAMPI appraisal method is a "Class A" method that is very time consuming and resource intensive, and therefore not easily used during the source selection process. An SEI initiative is currently in progress to define "Class B" and "Class C" SCAMPI appraisal methods that are less time consuming and resource intensive. These methods, which are expected to be available in Fall 2004, promise to be useful for performing software capability appraisals during source selection.

Whatever method is used for performing the software capability appraisal, there are two important attributes that the appraisal must have in order to meet its primary objective of selecting a contractor that is capable of developing the required software within the program constraints. First, the appraisal must evaluate the processes proposed for use by the contractor on the program under bid. It is not sufficient to evaluate corporate process descriptions and processes used on past programs since the processes proposed for this program may be quite different. Second, the appraisal must evaluate the contractor as a team, not each team member individually. The set of team members with software development responsibility must have well-integrated processes in order to perform effectively as a team.

It should be emphasized that applying this best practice means that the acquisition team performs a formal appraisal of the offerors as part of the source selection. The contractor's SW-CMM or CMMI levels previously obtained by self appraisal (using internal staff or personnel hired by the contractor as the appraisal team members) should never be used as a substitute. A contractor's self appraised level is frequently higher than the level at which the organization as a whole is actually performing due to the ability to "cherry pick" the projects used for the appraisal. In addition, an existing self appraisal will not have been performed on the processes proposed for the specific program under bid and is limited to a single contractor organization, not the entire bidding team.

3 SCE is a service mark of Carnegie Mellon University (CMU). CMM and Capability Maturity Model are registered in the U.S. Patent and Trademark Office by CMU.

4 CMMI and Capability Maturity Model Integration are registered in the U.S. Patent and Trademark Office by CMU.

5 SCAMPI is a service mark of CMU.

Evaluate Software Architecture with System Design

This best practice involves evaluating the contractor's proposed software architecture as part of the evaluation of the proposed system architectural design during source selection. The Government evaluates the proposed space system design as an important, highly weighted part of the source selection evaluation criteria (e.g., as a subfactor of the Mission Capability factor) in virtually every space system design or development contract. The most effective application of this best practice requires the joint participation by acquisition team personnel knowledgeable in space systems software architecting as well as space system hardware-knowledgeable engineers in this part of the source selection evaluation.

For software-intensive space systems, software is an integral part of the space system design, equally important to hardware. Large inadequacies and risks in a proposed space system design can be underestimated or missed entirely if the evaluation does not include the software portion of that design. Care must be taken in developing the RFP to ensure that the source selection evaluation criteria appropriately address the major hardware and software architecture issues important to the program. Examples of such issues are space-ground trades, hardware-software allocations of system requirements, onboard processing requirements, and appropriateness of reuse of legacy system components. The expertise gained in developing a robust, integrated Government hardware/software architecture baseline (see paragraph 3.1.1) will provide a good foundation for defining a set of source selection evaluation criteria that fully addresses the program's major hardware and software architecture issues and risks.

Evaluate Realism of Cost and Schedule Bids

This best practice involves evaluating the realism of the contractor team's proposed software cost and schedule for the software-intensive system under bid. Cost and schedule are always important factors in any source selection. This is one area, however, where less is not necessarily better! Unquestioning acceptance of unreasonably low software cost and schedule bids will not only lead to contract overruns; it will also result in a poor quality software product since the contractor will take shortcuts in software quality enhancing activities (e.g., peer reviews, robust software testing) in order to meet the contractual cost and schedule constraints.

The application of this best practice requires that the acquisition team perform a detailed evaluation of the assumptions underlying the software cost and schedule estimates during the source selection process. Low estimates of new lines of code, large amounts of reuse and COTS software, high productivity estimates, and overly optimistic cost drivers being used in the software cost models all must be examined for realism. In addition, the software cost estimate must be examined to determine whether all software-related effort has been included, especially those items not easily estimated using software cost models. Furthermore, the systems engineering,

program management, and integration and test effort related to software must be examined for adequacy. Since these activities are usually estimated by taking a percentage of the development effort, an unreasonably low estimate of software development effort will lead to an inadequate amount of effort being applied in these other important areas. The expertise gained in developing a realistic, independent Government baseline of software size, effort, cost, and schedule estimates (see paragraph 3.1.1) will provide the foundation for performing this detailed evaluation of the software cost and schedule bids during source selection.

3.1.5 Providing Tools for Contract Management

There are two software acquisition best practices associated with providing tools for contract management as described below.

Provide Contract Incentives for Software Quality, Not Just Cost and Schedule

This best practice involves the use of award fees and other contractual incentives to reward software quality by positively motivating the contractor to use software engineering best practices. The use of the term “quality” in this context means producing software work products that require very little, if any, rework in successor activities. The application of this best practice requires building criteria related to software quality into the contractual award and incentive fee plans.

Award fee is generally given at predetermined time intervals throughout the contract duration. Award fee should be used to reward the contractor for adherence to their defined software processes and for implementing software process improvement on the program to improve the effectiveness of their processes. Award fee should also be used to reward the contractor for producing high-quality software products, i.e., products with low rework rates in subsequent activities. The contractor should also be rewarded for the timeliness and adequacy of their responsiveness to comments from acquisition team software product and process reviews. Incentive fees based on performance of the delivered software product in its operational environment can also be used to positively motivate the use of software engineering best practices during development. Incentive fees post-delivery or -launch can be based upon meeting system performance measures that include both hardware and software (e.g., system reliability and availability) during operations. Incentive fees can also be based upon defect removal effectiveness measures such as the ratio of the number defects found during development to the number of defects found during operations.

Frequently, programs include target cost and schedule incentives without also including quality incentives. The more constrained the contractual target cost and schedule, the more likely there are to be large incentives dependent upon meeting those constraints. Having target cost and schedule incentives without also having product quality incentives sends the message to the contractor that product quality is not important to the Government. The contractor will then be motivated to take shortcuts in their software development processes to earn the associated incentives,

such as by reducing the effort spent on quality enhancement activities (e.g., peer reviews, software quality assurance reviews) and reducing the robustness of the test program. Such shortcuts are always counterproductive since the associated increase in latent defects will eventually result in increased cost and schedule due to increased rework and the necessity for operational workarounds. The Government should be especially concerned about cost and schedule incentives that result in infeasible cost and schedule constraints for the software development effort.

Effective application of this best practice requires commitment from the Government in their execution of the software quality incentives. The Government must be willing to allocate sufficient amounts of funds to the software quality portion of the award fees and incentives so that the contractor is significantly rewarded for use of software engineering best practices. In addition, the Government should also ensure that the contractor is significantly penalized for process non-compliance and poor quality software products. Independent technical reviews of software products and processes by the acquisition team (see paragraphs 3.2.1 and 3.2.2) are effective mechanisms for providing input on software product and process quality to the award fee process.

Mandate Periodic Team Software Capability Appraisals

This best practice involves requiring periodic formal software capability appraisals of the contractor during the contract period of performance. The objective of this best practice is to ensure that the contractor is actually using their defined software development processes and is applying software engineering best practices on the software being implemented for the program. The requirement for the contractor to undergo periodic software capability appraisals must be contained in the contract. There are several places one can implement this in a contract, such as in the SOW or in a contract special provision (Section H). In any case, for this best practice to be most effective, the results of the appraisals and the timeliness and adequacy of any resulting improvement actions must be coupled to the award fee and included in the award fee plan.

The contract must specify the frequency with which the appraisals are to be performed and the appraisal technique to be used to allow the contractor to accurately bid the time and effort needed. Appraisals used to evaluate the contractor's software development processes that are actually being used on a contract are called "contract process monitoring" appraisals. The same appraisal methods are used for contract process monitoring as for source selection (see paragraph 3.1.4). Thus, the choices currently consist of the SCE and SDCE and are soon expected to include the SCAMPI "Class B" and "Class C" appraisals. As with a software capability appraisal performed during source selection, the contract process monitoring appraisals must evaluate the contractor as a team, not each software team member individually.

3.1.6 A Software Acquisition Best Practice Contract

The performance of many of the pre-contract award software acquisition best practices described in paragraphs 3.1.1 through 3.1.5 above requires the inclusion of appropriate requirements in the contract. Figure 3 shows the various sections of a typical DoD contract and which best practice requirements are included in each section.

In some procurements, the Government includes a Statement of Objectives (SOO) in the RFP package but does not include a Government-prepared SOW. The contractor is then asked to prepare a Contractor Statement of Work (CSOW) in addition to the IMP for inclusion in the contract. In this case, the best practice requirements allocated to the IMP/SOW in Figure 3 would be included in some portion of the RFP package other than the Government SOW, such as the SOO or the Special Provisions. Care must be taken during the source selection to review the IMP and CSOW to ensure that they contain all of the necessary tasks and events to support these best practice requirements.

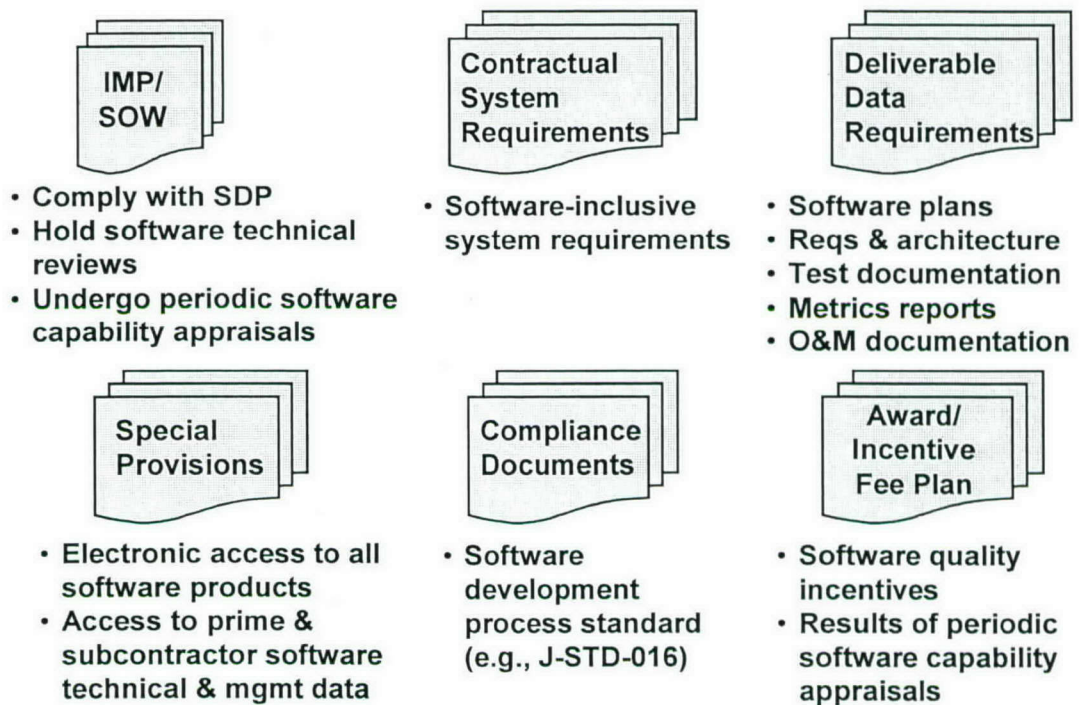


Figure 3. A "typical" software acquisition best practice contract.

3.2 Post-Contract Award Space System Software Acquisition Best Practices

The post-contract award space system software acquisition best practices address performing technical product reviews, performing software process reviews, and managing the contract. The best practices in each of these areas are discussed in the following paragraphs. Many of these best practices involve using contractual elements established by the pre-contract award best practices described in paragraph 3.1 and its subparagraphs.

3.2.1 Performing Technical Product Reviews

There are three software acquisition best practices associated with performing technical product reviews as described below.

Perform In-Depth Technical Reviews of Software Products

This best practice involves the acquisition team performing in-depth technical reviews of the contractor's software products throughout the development life cycle. While the ideal would be to perform in-depth technical reviews of all of the key software products, in practice, this is generally not feasible due to the limited acquisition team workforce. Therefore, the acquisition team's technical resources must focus on the areas of the software development effort with highest technical risk. An effective software acquisition risk management process (see paragraph 3.3) is required to adequately define those risk areas. Results of reviewing a sampling of software products can be used to predict quality trends across all software products produced by the same software team member.

The objective of in-depth technical reviews of the contractor's software products is to assist in identifying defects in the software products. Technical reviews of software products by the acquisition team generally identify defects not found by the contractor's quality checks (e.g., defects due to misinterpretations of requirements, defects due to incorrect use of new technologies, defects due to lack of attention to operations and sustainment issues). This is due to the difference in perspective of the acquisition team compared to that of the contractor's development personnel.

A good working relationship and good communication between the acquisition team and the contractor's software developers will enhance the effectiveness of the technical reviews of software products. The most effective application of these technical reviews occurs when the results of the acquisition team's reviews are used by the contractor to correct and improve their products. This behavior on the part of the contractor can be motivated by rewarding quality via award and/or incentive fees (see paragraph 3.1.5).

Monitor Software Integration and Verification Adequacy

This best practice involves the acquisition team monitoring the contractor's software integration and verification testing. The purpose of this best practice is to determine the adequacy of the contractor team's software integration and verification testing so that corrective actions can be taken, if needed, to improve the fidelity of the software test program. Another purpose is for the acquisition team to assist the contractor in identifying defects discovered during testing. As with technical product reviews, the acquisition team generally must focus its limited technical resources where monitoring of testing provides the highest risk reduction potential. Results of monitoring a sampling of software integration and verification testing can be used to predict quality trends across all software testing performed by the same software team member. The effectiveness of this best practice is increased when the software test plan and procedures have been reviewed in-depth by the acquisition team before the formal testing takes place (see above best practice).

The purpose of software integration testing is to find defects so that they can be fixed before the software is used in higher levels of testing (e.g., space vehicle, segment, system), while the purpose of software verification testing is to verify that the software meets its requirements. The most effective application of acquisition team resources for this best practice is for the acquisition team to monitor the build integration testing for each build and to monitor the software verification testing.

Monitoring of testing involves ensuring that the test procedures are conducted as written, that any deviations from what is expected are documented, and that all results are captured. In addition, the acquisition team personnel should participate in the data analysis activity to help determine whether the software performs as expected and whether the software meets its requirements. Frequently, defects that are not recognized by the contractor's test personnel are identified by the acquisition team. This is due to the acquisition team's experience and operational perspective. One of the most important goals for the acquisition team during monitoring software integration and verification testing at the build level is obtaining early performance analysis results and early determination as to whether the KPPs are being met. The acquisition team needs to ensure that the software integration and verification testing addresses these important issues.

Include Users and Operators in All Technical Review Activities

This best practice involves including user and operator personnel in the technical software product reviews throughout the life cycle. Qualified user and operator stakeholders should be involved in the in-depth technical reviews of the software products [especially the software requirements, software architecture and design (including the human-computer interface design, algorithm design, and critical interface design), the software verification test products, and the O&M products], and in monitoring the integration and verification testing of the software. The effective

application of this best practice requires consistent and continuous participation of qualified user and operator personnel and a good working relationship between the acquisition team, contractor, and user and operator personnel.

Close involvement of the user and operator stakeholders throughout the software development life cycle will reduce rework due to misinterpretations by the contractor of user and operator requirements and needs. In addition, participation by qualified user and operator personnel and their technical support contractors in software product reviews can assist the contractor in identifying and correcting product defects early in the life cycle. Close involvement of the user and operator stakeholders in software verification testing can also assist the contractor in defect identification, and can provide an early assessment of the suitability of the software products for meeting operational needs.

3.2.2 Performing Software Process Reviews

There are two software acquisition best practices associated with performing software process reviews as described below.

Review the Effectiveness of the Contractor's Software Processes

This best practice involves the acquisition team evaluating the effectiveness of the software processes being used by the contractor throughout the development life cycle. These best practice's software process reviews are informal process reviews performed by the acquisition team as they work with the contractor throughout the software development effort (rather than formal software capability appraisals). The objective of acquisition team reviews of the contractor's software processes is to assist the contractor in improving their software development processes. Since the quality of the software products is highly dependent on the quality of the processes used to develop those products, performing acquisition team software processes reviews is a software quality enhancement activity.

There are two aspects to be addressed in these acquisition team software process reviews. The first aspect is whether the contractor is adhering to the software processes they have defined and documented for the program (e.g., in the SDP and other program software process documents). For this aspect of the review, the acquisition team identifies process adherence deficiencies and assists the contractor in correcting these deficiencies. The second aspect is whether the contractor's defined software development processes are effective. For this aspect of the review, the acquisition team reviews the contractor's defined software processes for the program to determine whether these processes are effective, identifies process deficiencies, and recommends improvement actions. A frequently encountered example is peer reviews that are performed but are not effective at finding significant defects (only trivial defects are found). An acquisition team software process review may determine that the participants do not have enough time to prepare for the peer reviews, that too much material is being reviewed in a single meeting, that adequate review checklists

are not available, or that the review meetings are held without all of the essential review participants being present. Each of the identified causes for lack of effectiveness should lead to an improvement recommendation by the acquisition team. It should be noted here that process adherence to SW-CMM or CMMI processes at Level 3 or above that are defined by each contractor team member performing software development may not be effective for the program due to the problem of integrating these processes across the team member boundaries.

The most effective application of this best practice occurs when the contract incentives (e.g., award fee) reward the contractor team for adherence to their defined software development processes, for implementing software process improvement on the program to improve the effectiveness of their processes, and for the timeliness and adequacy of their responsiveness to comments from Government software process reviews (see paragraph 3.1.5).

Perform Periodic Software Capability Appraisals

This best practice involves conducting formal contract process monitoring software capability appraisals of the contractor at periodic intervals throughout the development life cycle. The objective of this best practice is to ensure that the contractor is actually using the program's defined software development processes and is applying software engineering best practices on the software being implemented for the program.

To apply this best practice, the contract must require the contractor to undergo these appraisals (see paragraph 3.1.5). Once this contractual requirement for periodic software capability appraisals is established, the acquisition team needs to follow through and conduct the appraisals as specified in the contract. It is important that these appraisals be conducted on the entire contractor team performing software development, rather than on each software team member individually. While the frequency of the periodic appraisals must be specified in the contract, the actual schedule for conducting the appraisals is usually negotiated with the contractor. On the one hand, the appraisals should be conducted so as not to interfere with major contract events, but on the other hand, they should be scheduled so that they support significant program or award fee milestones. The most effective application of this best practice occurs when the results of the appraisals and the timeliness and adequacy of any resulting improvement actions are included in the contract incentives (e.g., the award fee plan). More information on these appraisals is found in paragraph 3.1.5 above.

3.2.3 Managing the Contract

There are four software acquisition best practices associated with managing the contract as described below.

Ensure Satisfaction of Software-Inclusive Requirements

This best practice involves ensuring that the contractor includes allocating the software-inclusive contractual requirements (see paragraph 3.1.1) to software components as well as hardware components when designing and developing the system. In addition, this best practice includes ensuring that software is included as well as hardware when verifying that the software-inclusive system requirements are met. The objective of this best practice is to ensure that the system, which includes both hardware and software, meets its requirements when fielded in the operational environment. To meet this objective, the acquisition team must ensure that the contractor is not taking a hardware-only approach to any of the software-inclusive system requirements as the design, implementation, integration, and verification proceed throughout the development life cycle. The acquisition team must be especially watchful of the contractor's approach to the satisfaction of software-inclusive KPPs since not meeting the KPPs can be used as justification for program cancellation.

Contractors have historically been especially deficient about including software in the analyses of the system's specialty engineering requirements (including DRMA; supportability, including testability and integrated system diagnostics; safety; security; and human systems integration). In particular, contractors frequently assume that software does not contribute to dependability, reliability, maintainability, and availability; and, therefore, they allocate these requirements only to hardware components. Because of this assumption, failures caused by software are not included in the estimations of dependability and reliability, and the times for returning the software to an operational state following hardware or software failures are not included in the maintainability and availability estimates. If the acquisition team agrees with this approach and allows the DRMA requirements to be satisfied with only hardware contributions, the operationally experienced DRMA will be significantly less than specified in the requirements due to the contributions of software. This can result in the software-intensive system not being suitable for operations. A hardware-only approach to DRMA can be especially detrimental if any of the DRMA requirements are program KPPs.

Aggressively Use Contract Incentives to Reward Software Quality

This best practice involves the Government aggressively using the contract incentives to motivate the contractor to use software engineering best practices and produce high-quality software products. The application of this best practice requires criteria related to software quality to have been built into the contractual award and incentive fee plans (see paragraph 3.1.5). This best practice, therefore, consists of actually using the contract management tools of award and incentive fees to achieve the desired result of software that is of sufficiently high quality to be suitable for use in a high-reliability, high-integrity, software-intensive space system.

Effective application of this best practice requires the Government to commit to aggressive action in using the contract incentives. The Government must be willing to significantly reward the contractor for using software engineering best practices and, correspondingly, to significantly penalize the contractor for process non-compliance and poor quality software products. Technical reviews of software products and processes by the acquisition team (see paragraphs 3.2.1 and 3.2.2) are effective mechanisms for providing input on software product and process quality to the award and incentive fee process.

Perform Periodic Independent Assessments

This best practice involves the Government establishing an independent assessment team periodically throughout the development life cycle. The usual charter of an independent assessment team includes identifying problems and risks in the software development project, assessing actual software status and performance, and developing solutions to critical problems. Frequently, programs constitute independent assessment teams only when significant problems in the software development effort emerge. While this is an appropriate use of an independent assessment team, it is a reactive use rather than a proactive one. This best practice requires the Government to establish independent assessment teams periodically throughout the development effort (e.g., at significant program or award fee milestones) before problems have become apparent.

An independent assessment team consists of highly qualified technical personnel who are not directly involved in supporting the program on a routine basis. Frequently, acquisition team personnel who are deeply involved in the program on a day-to-day basis become too close to the program to step back and take an objective, truly independent view. The different perspective of the independent assessment team and absence of close program ties can allow the team to identify risks and problems and develop mitigation actions and solutions that the acquisition team and contractor personnel have not been able to see.

Effective application of this best practice requires the Government to take strong action based on the findings of the independent assessment team. Timely response can allow effective risk mitigation to be put into place and problems to be solved earlier rather than later when the solutions are more difficult and costly.

Apply Proactive Quantitative Management

This best practice involves the acquisition team proactively using contractor metrics data to manage the software acquisition. This best practice requires acquisition team personnel with metrics expertise to analyze the delivered monthly metrics reports (see paragraph 3.1.2) in the context of what they see happening day-to-day on the software development effort. This includes cross-metric analysis, as well as analysis of individual metrics, for the entire software development effort for all software team

members. Software and system metrics analysis can identify potential and actual problem areas that can then be examined in more detail to put effective risk mitigation actions and solutions in place. Metrics data provide high-level visibility into the health and status of the evolving software system and lower level visibility for timely problem detection, isolation, and impact assessment. In addition, metrics data provide a summarization of the planning assumptions into quantitative data that allow clear visibility of these assumptions and their implications. To be proactive, this best practice requires action on the part of the acquisition team to address risks and problems identified by the metrics analysis.

The effective application of this best practice requires the contractor to have a robust, comprehensive software and system metrics program. The DoD Practical Software and System Measurement (PSM) initiative⁶ has developed excellent guidance for implementing such a metrics program. The acquisition team must work closely with the contractor in defining the metrics data to be collected and reported to ensure that the contractor's metrics program addresses the acquisition team's information needs as well as the contractor's. The metrics data that are collected and reported need to be balanced across all information categories. The PSM information categories are size and stability, resources and cost, schedule and progress, process performance, product quality, technical effectiveness, and customer satisfaction. Frequently the contractor's metrics program emphasizes schedule and progress metrics to the exclusion of other important information. This is sometimes indicative of a Government emphasis on meeting schedule and of contract incentives that reward meeting schedule and do not reward quality. In particular, the collected and reported metrics data need to include leading indicators of quality problems that are effective predictors of downstream rework. Earned value data alone are not sufficient for managing the acquisition of large, complex software-intensive systems. The SMC/NRO Software Process IPT has developed guidance for the acquisition team on metrics-based software acquisition management [Abelson, L. A. et al., 2004].

3.3 Full Life Cycle Software Acquisition Best Practices

There are two software acquisition best practices that span the entire program life cycle: software acquisition risk management and software systems acquisition.

Software Acquisition Risk Management

This best practice involves the Government performing software acquisition risk management as an integral part of its program acquisition risk management. Software acquisition risk management is an effective mechanism for reducing the impact of potential problems on the acquisition of a software-intensive system. Software acquisition risk management involves a continuous process of risk identification, assessment, prioritization, mitigation, and control throughout the life cycle of the program, from the identification of needed capability through retirement.

⁶ See www.psmc.com for more information about the Practical Software and System Measurement initiative.

The use of software acquisition risk management enables the acquisition team to understand its risks (i.e., future problems that might occur) and, where possible, to mitigate the risks that are assessed to have the largest potential impact on the software-intensive system acquisition. Effective software acquisition risk management can result in reducing the cost and schedule impact of problems by having alternative solutions or workarounds identified before problems occur. Software acquisition risk management by the acquisition team is necessary in addition to software development risk management by the contractor. The contractor's software development risks are almost always software acquisition risks. However, the Government acquisition organization responsible for acquiring the software-intensive system generally has additional risks as well (e.g., risks related to staffing of the acquisition team and risks related to program Key Decision Points). Furthermore, the acquisition team frequently will identify additional software development risks and will assess the importance of the contractor-identified software development risks differently than the contractor. In addition, risk mitigation efforts performed by the acquisition team can assist the contractor in their risk mitigation efforts.

To be effective, the software acquisition risk management process must be practiced by all acquisition team personnel and at all levels of the software acquisition project. Large programs frequently have a risk management process at the program level with the top program risks being closely monitored by Government program management. Just as frequently, however, the only risk management process practiced on large, complex software-intensive programs is the program-level process. An effective software acquisition risk management process must be practiced at every level of the software acquisition project, including the lowest level of the acquisition team. The acquisition team should use the software acquisition risk management process to identify its risks, mitigate and control those risks that are within its scope of control, and elevate those risks with sufficiently high impact to the program-level risk management process.

Software-Inclusive Systems Acquisition

This best practice involves the inclusion of software acquisition as an integral part of the acquisition of software-intensive systems. Software acquisition team personnel must be knowledgeable and must participate in the systems acquisition processes throughout the entire life cycle, from the identification of needed capability through retirement. In addition, the software acquisition processes must be consistent and integrated with the systems acquisition processes.

It is very important for software acquisition to be an integral part of the system acquisition's pre-contract award activities, especially in defining the system acquisition and support strategies and in preparing the system performance requirements and RFP. Without effective participation of software acquisition team personnel in the pre-contract award activities, the selected contractor may not be capable of performing the software development effort, the system performance requirements may not include necessary software-related requirements, and the contract resulting from the

procurement may not be structured to encourage the contractor to follow well-disciplined software development processes and produce high-quality software products. Post-contract award software acquisition must be an integral part of the system acquisition contract management activities, especially for award and incentive fee determination, to encourage the best software development performance from the contractor.

The effective incorporation of software acquisition as an integral part of the systems acquisition processes requires positive action by the Government program management. Government program management must establish an environment where the software acquisition is a highly respected part of the program, equivalent in importance to the hardware acquisition. In addition, the Government program must be structured to provide effective lines of communication, responsibility, and authority among the software acquisition team and the other program teams involved in the systems acquisition processes.

4. Conclusions

This report describes a set of software acquisition best practices that the authors have identified, through experience, as being significant contributors to the successful acquisition of software-intensive systems. These software acquisition best practices, however, are not a panacea; that is, they do not guarantee a successful acquisition since there are a great many influences that can adversely affect large, complex, software-intensive system acquisitions. Rather, this set of best practices reduces risk in software-intensive system acquisition by providing tools that: (1) enable software to be properly integrated into the system development contract, (2) provide appropriate levels of insight into the contractor team's technical and management software products and processes, and (3) support proactive contract management.

This report does not advocate returning to either the pre-acquisition reform environment of the 1980s to early 1990s or the subsequent acquisition reform era that began in the mid 1990s. The software acquisition best practices described in this report strike a balance between these two extremes by recommending contractual requirements and oversight in those areas determined to provide the largest risk reduction benefit.

The software acquisition best practices are most effectively implemented within the context of a software acquisition process improvement program. Using the framework of a software acquisition process improvement program, software acquisition processes based on best practices can be defined, documented, measured, and improved. The most rigorous type of process improvement program uses a formal process model, such as the Capability Maturity Models developed by the SEI. The Software Acquisition Capability Maturity Model (SA-CMM) is the formal process model currently in use for the software acquisition discipline since the disciplines of software and system acquisition have not yet been incorporated into the CMMI models. The SEI has recently published an acquisition module for the CMMI [Bernard, T. et al., 2004]. However, this module is not yet fully developed.

In FY03, the U. S. Congress required each military department to establish software acquisition process improvement programs (Section 804 of the Bob Stump National Defense Authorization Act of Fiscal Year 2003). Subsequent DoD direction for implementing Section 804 was provided in a policy memo from the Undersecretary of Defense for Acquisition, Technology and Logistics and the Assistant Secretary of Defense for Command, Control, Communication and Intelligence[Aldridge and Stenbrit, 2003]. Per this memorandum, the software acquisition process improvement programs are to address the following software acquisition process areas, at a minimum:

- Acquisition Planning
- Requirements Development and Management
- Configuration Management
- Risk Management

- Risk Management
- Project Management and Oversight
- Test and Evaluation
- Integrated Team Management
- Solicitation and Source Selection

Each of the best practices in this report applies to at least one of these software acquisition process areas.

References

- Abelson, L. A., R. J. Adams, and S. Eslinger, "Metrics-Based Software Acquisition Management," The Aerospace Corporation, TOR-2004(3909)-3405, 5 May 2004.
- Adams, R. J., S. Eslinger, P. Hantos, K. L. Owens, L. T. Stephenson, and R. Weiskopf, "Recommended Software Standards for Space Systems," The Aerospace Corporation, TOR-2004(3909)-3406, 5 May 2004a.
- Adams, R. J., S. Eslinger, P. Hantos, K. L. Owens, L. T. Stephenson, J. Tagami, and R. Weiskopf, "Software Development Standard for Space Systems, The Aerospace Corporation, TOR-2004(3909)-3537, 20 July 2004b.
- Air Force Materiel Command, Software Development Capability Evaluation, Volumes 1 and 2, AFMCP 63-103, 15 June 1994.
- Aldridge, E., and J. Stenbit, Software Acquisition Process Improvement Programs, Office of the Secretary of Defense Memorandum, 21 March 2003.
- Barbour, R., M. Benhoff, B. Gallagher, S. Eslinger, T. Bernard, L. Ming, L. Rosa, and C. Ryan, Standard CMMI[®] Appraisal Method for Process Improvement (SCAMPISM), Version 1.1: Method Implementation Guidance for Government Source Selection and Contract Process Monitoring, Software Engineering Institute, Carnegie-Mellon University, CMU/SEI-2002-HB-002, September 2002.
- Bernard, T., B. Gallagher, R. Bate, and H. Wilson, "CCMI[®] Acquisition Module (CMMI-AM)," Version 1.0, Software Engineering Institute, Carnegie Mellon University, CMU/SEI-2004-TR-001, February 2004.
- Byrnes, P., and M. Phillips, Software Capability Evaluation (SCESM) Version 3.0 Method Description, Software Engineering Institute, Carnegie-Mellon University, CMU/SEI-96-TR-2, April 1996.
- Chrissis, M., M. Konrad, and S. Shrum, CMMI[®]: Guidelines for Process Integration and Product Improvement, Addison-Wesley, 2003.
- DoD Software Program Manager's Network, The Program Manager's Guide to Software Acquisition Best Practices, Version 2.31.
- Gallagher, B., C. Alberts, and R. Barbour, Software Acquisition Risk Management Key Process Area (KPA) - A Guidebook, Version 1.0, Software Engineering Institute, Carnegie Mellon University, CMU/SEI-97-HB-002, August 1997.
- IEEE/EIA Interim Standard J-STD-016-1995, Standard for Information Technology, Software Life Cycle Processes, Software Development Acquirer-Supplier Agreement, 30 September 1995.

- Paulk, M., C. Weber, B. Curtis, and M. Chrissis, The Capability Maturity Model[®]: Guidelines for Improving the Software Process, Addison-Wesley, 1994.
- J. McGarry, D. Card, C. Jones, B. Layman, E. Clark, J. Dean, and F. Hall, Practical Software Measurement: Objective Information for Decision Makers, Addison-Wesley, 2001.
- Software Engineering Institute. Capability Maturity Model[®] IntegrationSM, Version 1.1, CMMI[®] for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing (CMMI[®]-SE/SW/IPPD/SS, V1.1), Continuous Representation, (SEI-2002-TR-011), March 2002.
- Software Engineering Institute. Capability Maturity Model[®] IntegrationSM, Version 1.1, CMMI[®] for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing (CMMI[®]-SE/SW/IPPD/SS, V1.1), Staged Representation, (SEI-2002-TR-012), March 2002.
- Software Engineering Institute, Software Acquisition Capability Maturity Model[®] (SA-CMM[®]), Version 1.03, Software Engineering Institute, Carnegie-Mellon University, No. CMU/SEI-2002-TR-010, March 2002.
- Software Engineering Institute. Standard CMMI[®] Appraisal Method for Process Improvement (SCAMPISM), Version 1.1: Method Definition Document, CMU/SEI-2001-HB-001, December 2001.
- U.S. Congress, Bob Stump National Defense Authorization Act for Fiscal Year 2003.

Acronyms and Abbreviations

®	Registered Trademark
Acq.	Acquisition
AFMCP	Air Force Materiel Command Pamphlet
CDR	Critical Design Review
CDRL	Contract Data Requirements List
CMM	Capability Maturity Model
CMMI	Capability Maturity Model Integration
CMU	Carnegie Mellon University
COTS	Commercial Off-the-Shelf
CSCI	Computer Software Configuration Item
CSOW	Contractor Statement of Work
DID	Data Item Description
DoD	Department of Defense
DRMA	Dependability, Reliability, Maintainability and Availability
EIA	Electronic Industries Alliance
Eng.	Engineering
FFRDC	Federally Funded Research and Development Center
FY	Fiscal Year
H/W	Hardware
HB	Handbook
IEEE	Institute of Electrical and Electronics Engineers, Inc.
IMP	Integrated Management Plan
IPPD	Integrated Product and Process Development
IPT	Integrated Product Team
J	Joint
KPA	Key Process Area
KPP	Key Performance Parameter
MIL	Military
MOIE	Mission-Oriented Investigation and Experimentation
NRO	National Reconnaissance Office
PDR	Preliminary Design Review
PSM	Practical Software and System Measurement
RFP	Request for Proposal
S/W	Software
SA-CMM	Software Acquisition Capability Maturity Model
SCAMPI	Standard CMMI Appraisal Method for Process Improvement
SCE	Software Capability Evaluation
SDCE	Software Development Capability Evaluation
SDP	Software Development Plan
SDR	System Design Review
SE	Systems Engineering
SEI	Software Engineering Institute

SETA	Systems Engineering and Technical Assistance
SLOC	Source Lines of Code
SM	Service Mark
SMC	Space and Missile Systems Center
SOO	Statement of Objectives
SOW	Statement of Work
SS	Supplier Sourcing
STD	Standard
SW	Software
SW-CMM	Capability Maturity Model for Software
TR	Technical Report
TRR	Test Readiness Review
TSPR	Total System Performance Responsibility
U.S.	United States
USAF	United States Air Force