

Chapter 8

Contracting for Success

Contents

8.1 Team Building: Attacking The Lion	8-3
8.1.1 Building High-Performance Teams	8-4
8.2 Contract Type Selection	8-5
8.3 Developing the RFP	8-6
8.3.1 RFP Development Team Building	8-7
8.3.2 Statement of Objectives (SOO)	8-9
8.3.3 Contractual Data Requirements List	8-9
8.3.4 Subcontracting	8-9
8.3.5 Joint Venture Partnerships	8-12
8.4 Special Software RFP Considerations	8-13
8.4.1 Commercial-off-the-Shelf (COTS) Software	8-15
8.4.1.1 COTS Advantage	8-15
8.4.1.2 COTS Integration	8-16
8.4.1.3 COTS Integration Lessons-Learned	8-18
8.4.1.4 More Cautions About COTS	8-19
8.4.1.5 Cautions About Modifying COTS	8-21
8.4.2 Data Rights	8-23
8.5 Source Selection Factors	8-25
8.6 Source Selection	8-26
8.6.1 Source Selection Team Building	8-27
8.6.2 Source Selection Planning	8-27
8.6.3 Advisory Multi-Step Process	8-27
8.7 Proposal Evaluation and Contract Award	8-28
8.7.1 Proposal Evaluation	8-28
8.7.2 Best-Value versus the Cost of Poor Quality	8-29
8.7.3 Present Findings to the SSA	8-30
8.7.4 SSA Decision and Contract Award	8-30
8.8 Protests	8-31
8.9 Navy Seawolf Lessons-Learned	8-33
8.10 References	8-35

8.1 Team Building: Attacking The Lion

“Four brave men who do not know each other will not dare to attack a lion. Four less brave, but knowing each other well, sure of their reliability and consequently for their mutual aid, will attack resolutely.” — Colonel Charles Ardnant du Picq [DuPICQ80]

Large, complex, software-intensive military applications are beyond the intellectual comprehension of any one individual. A single creative developer can produce all the elements required for a simple PC application, but one individual cannot fully understand a large-scale software development, such as an automated air traffic control system, the avionics for a stealth fighter, or a complex command, control communications, and intelligent (C3I) network. Nor can one person manage its design, development, integration, and testing without help. These activities require groups of highly-skilled, experienced professionals.

Software engineering, more than any other engineering discipline, is an *extremely human endeavor*. Seasoned managers know that software development programs ultimately succeed or fail, not on the sophistication and power of the tools used by their teams — but on the skills and performance of those teams. There are case studies of organizations that assign two equally matched, independent development groups to a software program to minimize the risk of failure. Team A will be given a sophisticated set of automated tools, finishing the task 20% faster and producing better code than Team B. Then the teams switch tools, and Team A will still complete the task 20% faster, with fewer defects than Team B. Experienced managers are aware that this phenomenon reflects the situation throughout the entire software industry. The teamwork factor impacts significantly on your management ability to *attack the lion*. [ALIC92]

The formation of a Computer Resource Working Group (CRWG) should be your first step in team building. Though the CRWG is no longer required by regulation, it is still a good idea. CRWG members include the implementing agency, the using agency, the supporting agency, and other DoD and Service stakeholders. Other players such as the testers, contracting officers, in-house software developers, and software engineering laboratories are also included. The CRWG should be formed as soon as an official program is designated. The CRWG works together to ensure the Request for Proposal (RFP) addresses all areas of interest and/or concern for all CRWG members. After contract award, the final team member, the developer, is brought onboard. This team member is probably the most crucial and important team member. The software contractor you select can literally “*make you or break you*.” Figure 8-1 illustrates the composition of a typical software acquisition/development team.

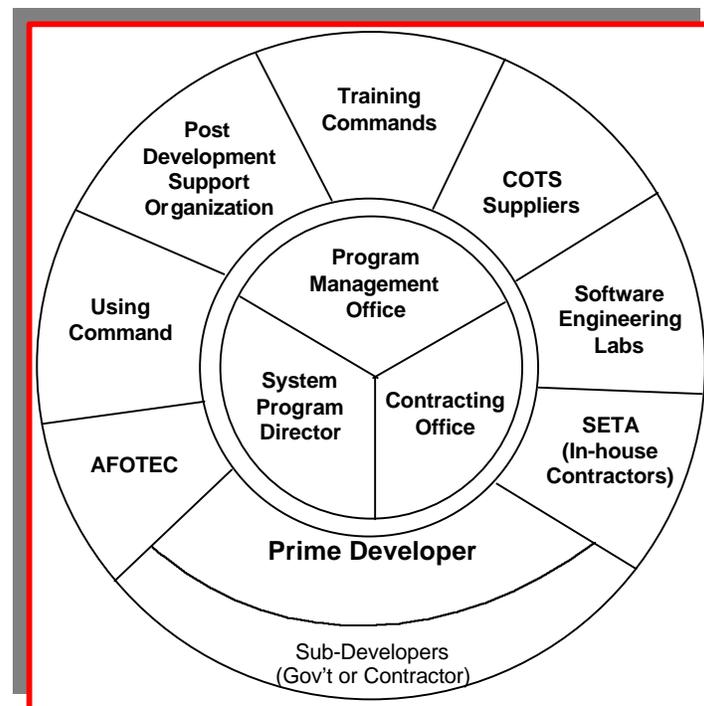


Figure 8-1. Typical Software Acquisition/Development Team

8.1.1 Building High-Performance Teams

Recognizing the importance of each team member (contractor and Government) is essential in the accomplishment of your mission. Too often government program managers have relied totally on the contract as their management vehicle. Their approach considers problems that arise as the contractor's problem and exerts contractual pressure to solve them. A "hold-the-contractor's-feet-to-the-fire" approach has been a common mistake where the them-versus-us mindset has held the contractor at fault when problems occur. This intractable approach has proven unworkable time and again, because it plays against the teamwork goals of communication, cooperation, mutual respect, and trust. [MARCINIAK90]

While the contract should not be used as the only management vehicle, it should be used to set the limits. It represents the best form of communication, a clear statement of the requirement at the start of development, and a framework for establishing a common sense of purpose among government/industry team members. This is where effective team management must start. Keep in mind that the purpose of the development effort is the delivery of an optimum solution — not the exercise of the contract. To deliver a successful solution, successful teamwork is a must. [MARCINIAK90] Team productivity and morale are enhanced when the contract states clear and attainable goals, provides a mechanism for trust and open communications, and defines a breakdown of tasks and resources that allow the team to function as a cohesive unit. When acquisitions fail, government program managers often immediately point their fingers at the hapless contractor; i.e., the contractor is the enemy. Sadly, those program managers do not realize that they, themselves, are frequently the enemy. Do not let this happen to you. Build a team from the outset and develop a common purpose and a productive working relationship with your contractor. *When programs fail there is no innocent party — both sides are guilty.* [MOSEMANN94]

Figure 8-2 illustrates the acquisition organization contracting process.

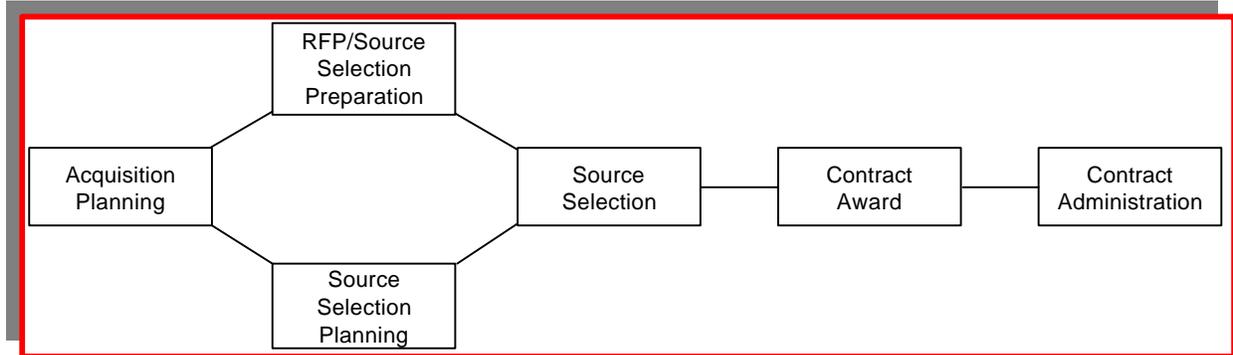


Figure 8-2. Acquisition Organization Contracting Process

8.2 Contract Type Selection

The degree of interaction between you and your contractor depends on the nature of the development effort and the type of contract used. When software requirements are well-defined (such as upgrading or enhancing an existing system) and the risk of development is low, a fixed-price type contract is probably the right choice. Where requirements are ill-defined and development risk is high, a more flexible cost-reimbursable contract may be more appropriate.

A way to mitigate high technology risk is to consider multiple award contracts, or contracts with a provision for event-driven task orders. This contracting strategy allows for a closer working relationship, and provides a better chance for arriving at a satisfactory solution because the risk is shared among government and industry team members. [MARCINIAK 90] [FAR, Part 16 provides basic principles and policy guidelines for contract type selection.]

The same factors influencing cost uncertainty also influence schedule uncertainty. However, there is no equivalent acquisition practice for establishing flexibility in the program baseline schedule. Although baseline schedules are often driven by the initial operational capability (IOC) need, the IOC is frequently established before assessments of technical risk and relative uncertainty, requirements definition, design solution, and effort (i.e., schedule) have been made. This is particularly true for unprecedented systems. Program level schedules are, too often, success oriented, do not reflect past actual schedules, and are, in fact, unachievable. Offerors have no real choice during competitive source selection but to respond to these schedules, even though meeting them represents inordinately high risk. *Success-oriented schedules are seldom successfully achieved.* Schedule pressure often compromises sound engineering and management practices — increasing the risk of poor product quality. One way to deal with unrealistic schedules is to allow partial delivery of the required functionality, with additional functionality delivered in later phases (i.e., evolutionary acquisition).

Another solution is to include the concept of schedule-plus in your RFP as an approach for establishing program schedule baselines. The schedule-plus approach is selected for the same reasons the cost-plus approach is used, where technology and other factors have sufficient uncertainty and risk. This is equivalent to setting up a management reserve for cost, extended to schedule. It is also warranted if you are planning to use a cost-plus approach, if similar past

programs have had schedule performance problems, if a comprehensive Program Definition and Risk Reduction phase has not been accomplished, or if you face significant challenges in the areas of performance and/or supportability requirements.

A schedule-plus contract includes a baseline (minimum) schedule plus a delta range. This is determined by comparing your proposed schedule to past actual schedules achieved on programs similar to yours. Your schedule must then be evaluated and adjusted to within the delta, as necessary, at program milestones. Schedules adjusted beyond the delta range will require re-approval of the baseline schedule estimate. You can include incentives for the contractor to stay within the planned schedule delta, such as a profit share line tied to the schedule, completion fees, and award fees. Your RFP must also identify specific schedule review elements to determine schedule status. These reviews are based on measurable events and actual data (metrics). The schedule-plus approach is also integrated with the systems engineering master schedule and plan. Firm performance requirements are clearly differentiated from goals, allowing offerors to bid to realistic schedules. Your schedule-plus approach is also integrated into your cost/schedule status report (C/SSR) system. For example, a periodic schedule estimate at completion (EAC) is accomplished along with the cost EAC. [SPAT92] Ideally, you will not state any specific schedule requirement in your RFP, but allow offerors to propose a realistic schedule which you will evaluate for realism during source selection.

8.3 Developing the RFP

The contract vehicle must be designed to clearly express a vision of final product goals and development effort requirements. Thus, the development of the Request for Proposal (RFP) is your first step towards bringing Government and industry together as a cohesive, high-performance team. The RFP also marks the culmination of the strategic planning process and represents the formal means for communicating government requirements to industry. Too often, the RFP is viewed as an administrative (rather than a technical) document. Its administrative function must be secondary to its technical function. The RFP must contain clear and sufficient technical guidance so the contractor has a definite picture of how the system is envisioned to perform once delivered. It is also important that a technical functional description of software requirements is included and that those requirements are clearly scoped. Inconsistencies, insufficient detail, and inappropriate software requirements will result in an inadequate response from industry to government needs.

NOTE: Because every program has unique requirements, it is beyond the scope of these Guidelines to provide specific information on what is important for every source selection. However, if you consider the suggestions listed herein, you will be well on your way to a successful acquisition.

Considerable time and effort is required to form a comprehensive software development RFP. Your program office *must work with the future user of the system* (your customer) to establish requirements, expectations, schedules, and support needs. Both the program office and the user must remember that well-defined, performance specifications yield good contracts and the better scoped the requirement — the better response from industry. [MARCINIAK90]

Early industry involvement in acquisition planning often improves the quality of the RFP and fosters a sense of government/industry partnership. Early involvement is an iterative and interactive communication process. You are required to integrate early industry involvement in your acquisition planning for all competitive procurements that are estimated to exceed \$24 million. Additionally, early communications with potential sources on any intended acquisition is plain common sense. Figure 8-3 illustrates the major RFP preparation process components.

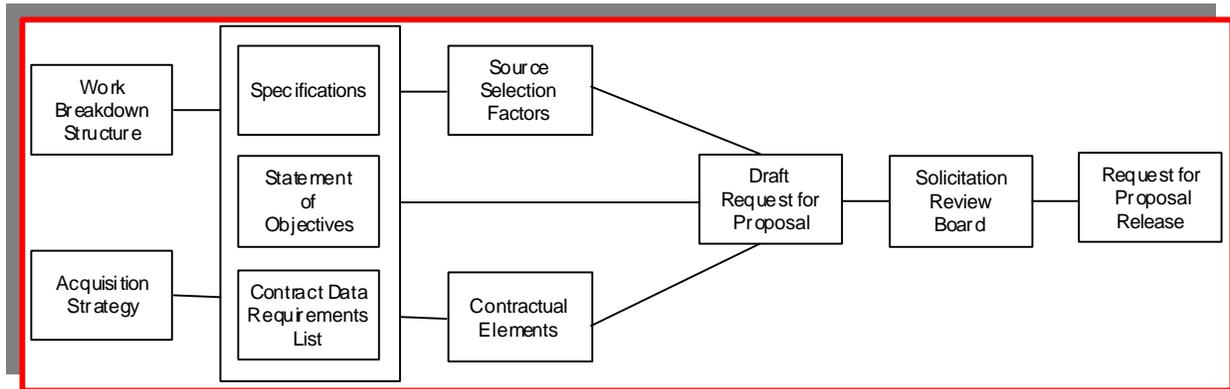


Figure 8-3. RFP Preparation Process

The following chapters contain discussions on software engineering subjects that must be addressed in the RFP.

- Chapter 6, *Risk Management*: Addressing Risk in the RFP,
- Chapter 10, *Software Development Maturity*: Addressing Software Development Maturity in the RFP,
- Chapter 12, *Software Support*: Addressing Software Support in the RFP, and
- Chapter 13, *Software Estimation, Measurement, and Metrics*: Addressing Measurement and Metrics in the RFP,
- Volume 2, Appendix S, *Software Source Selection*.

8.3.1 RFP Development Team Building

It takes highly qualified, in-house personnel to develop the RFP. Competent people are also indispensable to judge product deliveries and keep the program headed for success once the contract is awarded. Assembling a qualified acquisition team to bring on the last team member is another opportunity to practice your team building skills. Your acquisition team must include personnel from the supporting and using agencies, as well as software, contracting, and cost analysis experts. Team members creating the portion of the Statement of Objectives (SOO) related to software development must be knowledgeable in software engineering and management. Program office personnel serve as team leads and must make sure the user and supporting organizations' needs, concerns, and desires are fully addressed.

The RFP team develops the mission capability, proposal risk, performance confidence assessment, and cost/price evaluation factors for the acquisition. The software specification must be thorough and source selection evaluation factors rigorous. Your evaluation factors should only, however, measure those items that are valid discriminators and directly traceable to requirements. The

RFP must require that each proposal submitted contains sufficient information for a thorough assessment of each offeror's software development experience, tool availability, product assurance, team skills and experience, software support, and program management capabilities. The offeror's proposal should describe how their product and process will satisfy required and desired functionality. The RFP should include:

- Specific tasks (quantifiable, measurable, and testable),
- Specifications and standards to be used for the program (relying on commercial standards and practices, whenever possible),
- Planned use of government-furnished equipment (GFE), government-furnished information (GFI), and/or government-furnished software (GFS),
- Requirements for the contractor to provide a comprehensive layout of program schedules (internal reviews, formal peer inspections, testing, technical meetings, etc.),
- Requirements for relevant and pertinent domain experience,
- Requirements for a thorough Software Development Plan (SDP) and plans for its implementation and updates, to include a proposed test process plan,
- Requirements to describe use of the chosen programming language, commercial-of-the-shelf (COTS), and reuse,
- Requirements for appropriate software documentation,
- Requirements for an open systems architecture and architecture performance verification,
- Requirements for resources growth/margin verification,
- Requirements for a total life cycle/total systems perspective,
- Requirements for prototyping and/or demonstrations (ideally, a demonstration of an executable architecture as part of the proposal),
- Requirements for a progress, process, and quality measurement program, including a specific Metrics Usage Plan,
- Requirements for a software quality assurance (SQA) program,
- Requirements for supportability planning,
- Requirements for a Risk Management Plan and its implementation,
- Requirements for a Process Improvement Plan and its implementation,
- Requirements for a process control mechanism [e.g. software development tools, management tools, etc.],
- Requirements for developing interface software with other system software and/or hardware,
- Requirements for assessing software development maturity/capability,
- Planned communications with any Independent Verification & Validation (IV&V) contractors or agencies, and
- Requirements for delivery of the life cycle support environment.

NOTE: The first item “clear, concise statement of specific tasks” is particularly important. For example, a requirement to respond to user requests in “real-time” is ambiguous because there is currently no standard definition of the term “real-time.” It is much better to provide a numerical value (such as “within one microsecond”) for such a requirement.

8.3.2 Statement of Objectives (SOO)

The SOO is the primary document for translating performance requirements into contractual tasks. The SOO must be consistent with the summary work breakdown structure (WBS) and contain sufficient information for the offeror to prepare a detailed contract WBS [discussed in Chapter 7, *Acquisition Planning*]. It must also contain tasking information for contract data requirements lists (CDRLs). While the SOO states the specific tasks to be performed, *it must not tell the offeror how to do the required work*. Do not spell out specific qualitative and quantitative technical requirements in the SOO. Instead, offerors must be solicited to propose their solution to your stated need

8.3.3 Contractual Data Requirements List

The RFP's CDRL is the primary vehicle for acquiring documentation from contractors. It lists all data item descriptions (DIDs) that apply to the development program. DIDs describe the data the contractor is required to provide, along with delivery instructions (such as media and format). Each CDRL entry contains the DID identifier, title, requesting organization, distribution, referenced SOW task(s), and a remarks section where information (such as tailoring) is included. *Nearly all DIDs require tailoring for appropriate application to a contract.*

8.3.4 Subcontracting

On major DoD software-intensive system acquisitions, seldom is all the work performed by one contractor. Instead, the winning contractor is the prime who in turn arranges with other contractors, subcontractors, to complete some portion of the effort. Currently, *over 60% of total weapon systems development and production efforts are subcontracted*. [BAKER92] This includes most of the software for these systems. Note that the government has no direct involvement in the relationship between the prime and subcontractor. The following is for information purposes only.

Parallel hardware /software efforts involving prime and subcontractors have resulted in adversarial relationships and turf battles between the prime (hardware) and the subcontractor (software) — often impeding software development efforts. Although the prime is responsible for product quality, the prime must include in their subcontracts all contractual requirements necessary to ensure that software products are developed in accordance with prime contract requirements. Subcontracts also generally state that the Government must be given access to subcontractor facilities to review software products and activities required by the prime contract.

Inappropriate contract types are often a subcontractor problem because software developers seldom have solid requirements. A firm-fixed-price (FFP) contract is risky because the subcontractor views all requirements changes as *out-of-scope* changes, whereas the prime views them as *in-scope* with the System/Segment Specification (SSS). Since the subcontractor must develop the Software Requirements Specification (SRS) from the SSS, many problems occur when the SRS does not accurately reflect the performance requirements desired by the Government. Other problems occur when too many specifications are left up to the discretion of the subcontractor.

These contrasting views of responsibility are illustrated in Figure 8-4. When software programs experience difficulties, primes and subcontractors often resort to blaming each other. When schedules are not met, it is always the other company's fault!

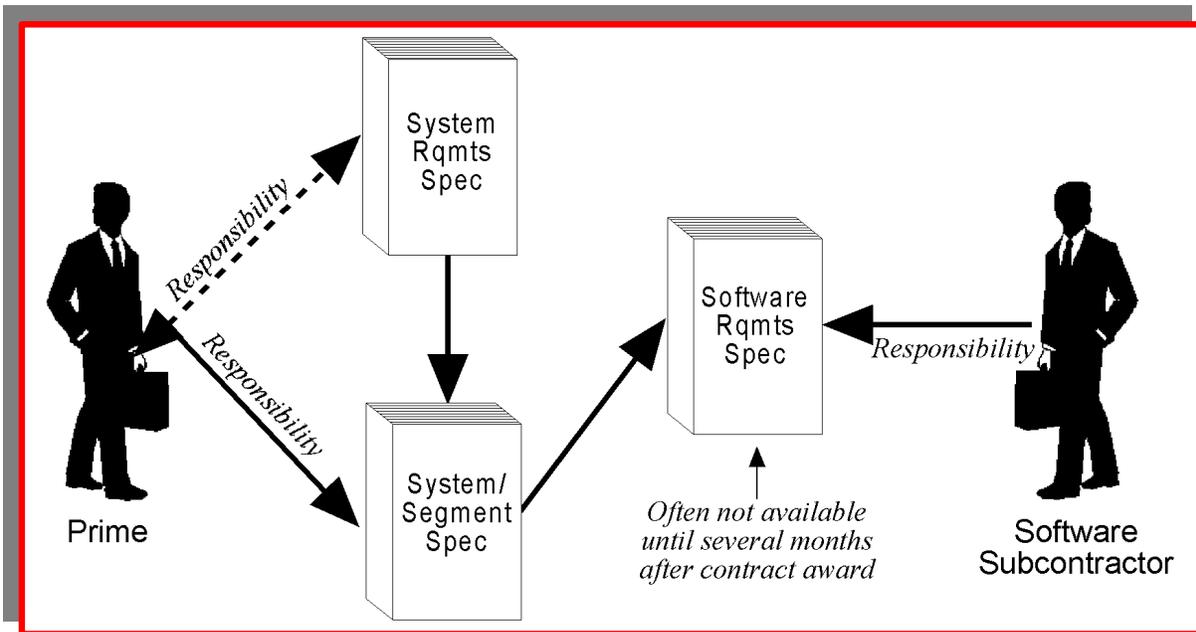


Figure 8-4. Prime/Software Subcontractor Development Responsibility [MOORE]

In addition to differing development effort views, primes and subcontractors frequently have differing business perspectives and goals. Software subcontractors are often hired only for the development effort, whereas the prime makes a profit throughout the life cycle up to deployment, as illustrated in Figure 8-5. [MOORE]

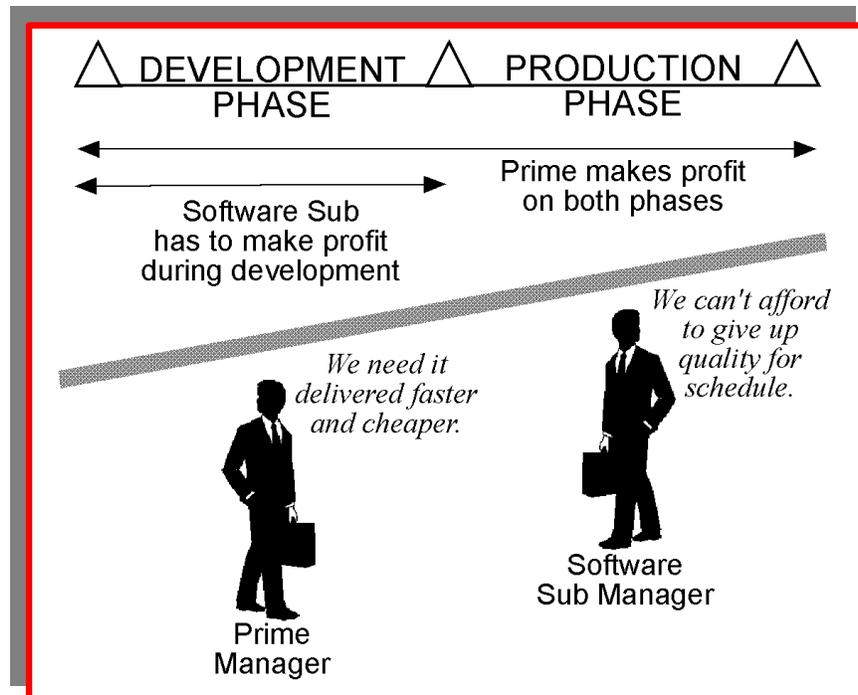


Figure 8-5. Prime-Sub Different Business Perspectives [MOORE]

Subcontracting can also be enigmatic because DoD managers have no direct control over subcontractors. You can only direct your prime contractor. You should also remember that you manage the contract, not the contractor. It's up to the prime to direct their subcontractors. Figure 8-6 illustrates the communications distance between you and your software developer and between the user and the software developer. In the RFP, you must ensure that the prime contractor provides the needed direction the Government requires of the software developer. One approach is to suggest that the prime contractor develop and follow a Subcontractor Management Plan that ensures government visibility and participation in the management process. After contract award, the program office, with the consent of the prime contractor, should be allowed to make periodic escorted visits to critical subcontractors. The plan should also make provisions for a joint team (user/SPO/prime/subcontractor) software development effort.

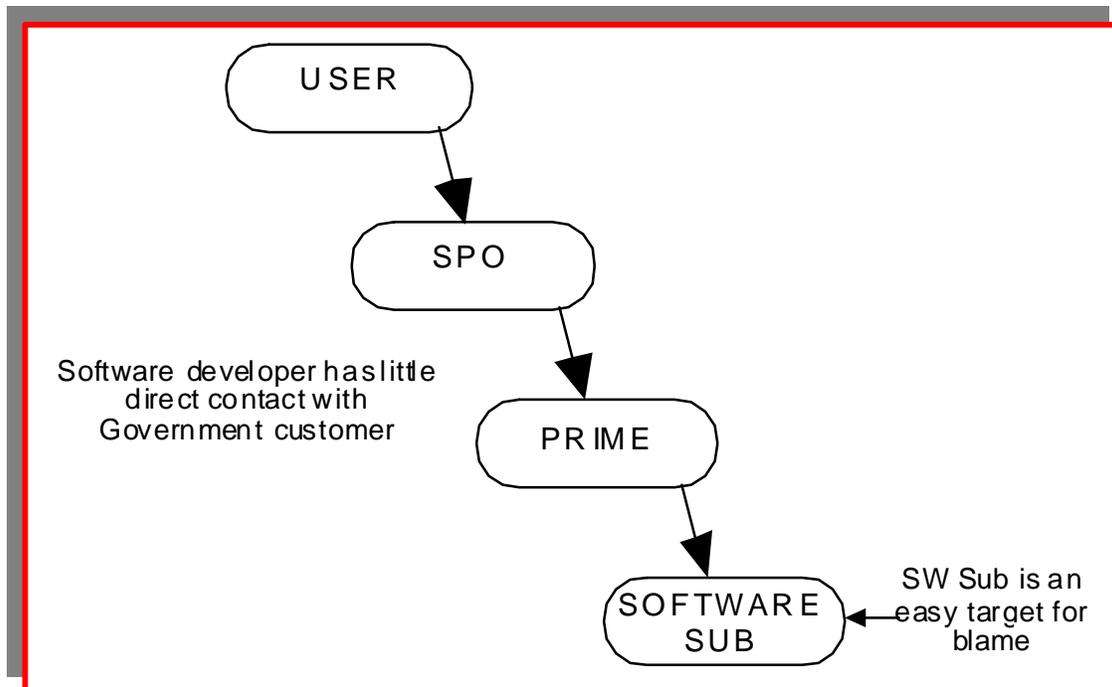


Figure 8-6. Chain of Government-Subcontractor Communications [MOORE]

8.3.5 Joint Venture Partnerships

To avoid difficulties associated with subcontracting, some firms enter into joint ventures. In a joint venture, two or more prime contractors create a single corporate entity for a specific program. An example is the Boeing-Lockheed-General Dynamics joint venture for developing the F-22. This arrangement has provided a low-risk business approach for all joint team members and the Government. The contractual agreement among the three companies calls for equal sharing of any cost overruns. If one team member encounters a problem, the other two members must help solve it — or they all suffer. The degree to which each company has subordinated self-interests to achieve team goals for the F-22 is another first in defense contracting lore. [VELOCCI91]

Joint ventures place much of the management burden on the contractors. Because the joint venture establishes the team members as equal team partners, it enhances cooperation and avoids many problems associated with subcontracting. If one member is lax, the other member(s) can exercise considerable leverage. Joint venture also removes much of the technology transfer requirements from the government program office. When evaluating joint venture proposals, source selection officials must ascertain whether a joint venture team is a viable competitor. However, the contractual arrangement details between the joint venture team members are not the Government's concern. [KRATZ84]

While joint venture partnerships eliminate some subcontracting issues, they present other management challenges associated with contractor teaming arrangements. This complex agreement between companies requires the formation of a new corporate entity, election of officers and a board of directors, assignment of personnel, and establishment of accounting and administrative procedures. Large responsibility is placed on parent members and on what is, in

essence, a start-up company. Therefore, joint ventures should not be construed as the end-all panacea for subcontracting quandaries.

The subcontracting dilemma may not be altogether eliminated either, since joint venture members may not be precluded from subcontracting out a portion of their individual tasks, of which software will always be a target. This can place an additional management layer between you and your software developer. Another potential disadvantage to joint ventures is that while equal partners are established, they may be geographically separated. This leaves you with three (or more) entities with which to interface at multiple locations, further complicating program management. [KRATZ84]

8.4 Special Software RFP Considerations

As you learned in Chapter 4, *DoD Software Acquisition Environment*, new acquisition streamlining initiatives state that essential government needs should be met with a minimum SOO. Mills expands on the concept of *teamwork* to one of government/industry “*partnership*,” where both team members share the responsibility for success. By transforming the Government’s contract monitoring role from the older documentation-driven review it, approve it, and baseline it paradigm, the new partnership role places the contractor solidly in charge of the process and the emerging product.

To meet DoD’s requirement for an overall manageable procurement, five key elements are essential in an RFP for a major software-intensive system. A risk management path for each element entails a minimum SOO with reduced proposal instructions. Offerors are then left with defining their actual approach. There is a strong synergistic relationship among the software RFP elements. For example, by allowing contractor control of baselines until very late in development, without the need for 100% correctness and completeness of documentation, it is possible to reduce former resistance to open sharing of technical information. *This promotes partnership*. At the same time, it is essential to clearly agree upon technical milestone points and formal reviews, and to establish a framework to ensure contract progress is achieved. Confidence in the offeror’s software development process permits greater trust in the offerors ability to achieve contract milestones. The five key software RFP elements are:

1. Software development process,
 2. Contractor documentation and formats,
 3. Contractor control of baselines,
 4. Direct technical visibility, and
 5. Proactive risk management.
- **Software development process.** A mature contractor process helps ensure that they will produce supportable, quality software on schedule in a predictable, consistent manner. The contractor’s practices must also be documented, maintained current by the development team, and be available for Government review. This supports the need for continuous verification of process maturity and effectiveness.
 - **Contractor documentation and formats.** Documentation deliverables should maximize the use of information in the form and format used to develop the software.

- **Contractor control of baselines.** Allowing the contractor to retain configuration and engineering control of baselines until they are stable, frees the developer from the government review and approval cycle which also supports partnering.
- **Direct technical visibility.** This may be implemented with the following requirements:
 - The contractor must plan and implement a means for sharing software development information with the Government. The contractor should be required to provide access to current working documentation in the language and format normally used for software development. This includes Government access to software engineering tools and databases.
 - Documentation, where possible, should reside in electronic format in the automated software engineering environment.
 - The contractor must plan the information sharing mechanism so that little or no contractor assistance is required for government personnel to access information. The information can be used as a basis for formal government recommendations to the contractor, and whenever practical, should be used to simplify the formal technical review process. Thus, you need not provide formal approval of shared information on a day-to-day basis.
- **Proactive risk management.** In the past, risk was reduced by requiring the delivery of a series of documents. Each deliverable was typically reviewed and approved by the Government to ensure quality and to independently verify contractor adherence to schedule. In principle, this document-driven contract monitoring was an efficient way to manage software development risk and perform program oversight. In practice, the oversight role progressively removed the developer from responsibility for design as each new document was approved. Since the Government performed the review and found the errors, the contractor only had to deliver a product on schedule and correct any errors the Government found. This approach too often led to increased reliance on testing and diffused the responsibility for quality problems, which often remained hidden until system delivery.

Providing Government access to software development information promotes partnership by removing disincentives for information sharing. Technical concerns are, for the most part, transferred from the schedule enforcement aspects of oversight. The monitoring role includes a technical function where you participate in reviewing emerging products and gain visibility into program progress and product quality. Knowing that you are monitoring product quality prior to delivery, the contractor is more inclined to build quality into your product. You also have the opportunity to directly verify process effectiveness and program metrics.

To ensure industry participation in contract monitoring, your RFP should request that offerors address the important aspects of program management and monitoring in their proposals. These requirements must be brief and concise in the SOO, thus requiring expansion and clarification in the offeror's proposal. The offeror's commitment to partnership and an improved contract monitoring role should be an important software source selection factor [*discussed below*]. This not only provides a performance incentive, but also rewards offerors who are committed to quality and process improvement. This monitoring role also must be strongly supported by proactive risk management.

When preparing your RFP, it is often difficult to communicate the need for a comprehensive response from industry. Because software is just one part of a system, the detailed proposal instructions necessary to obtain a risk-based proposal can easily be confused with non-value added requirements. When upper management is told that source selection requires more time

and attention, your program schedule can often become the constraining factor. Despite these obstacles, *reducing software risk during source selection is one of the most crucial management activities you will perform*. Unfortunately, the acquisition planning and RFP preparation process is lengthy and challenging. We must do more with less and we are all anxious for a stable program which can be smoothly budgeted. Schedule and availability of program funds are often the our main constraints. However, at the heart of the challenge is the requirement to meet your particular program's needs by selecting the contractor of whom you are assured represents the least risk with the greatest potential for success. [MILLS95]

8.4.1 Commercial-off-the-Shelf (COTS) Software

Our procurement cycle for major software-intensive systems is usually 10-plus years from inception to IOC. This acquisition cycle contrasts with the commercial software life cycle in which a product is enhanced in 6-9 months, a new product is developed in 12-18 months, and a product is obsolete in 36-48 months. [FAA94] *Hence, DoD is often procuring systems which are technologically obsolete by the time they are fielded*. Strict regulations and the long acquisition cycle translate into commensurately exorbitant costs for custom developed software. Software costs are also high because DoD has borne the entire financial burden for software maintenance (either contracted or in-house).

An alternative to developing and maintaining our own custom software is the use of software that has already been developed. It may have been developed for general use by industry and business, or for specific use by a specific group. Software which has already been developed and is available for use by any customer is known as commercial-off-the-shelf (COTS) software. It can save a development effort a great deal of time and cost. It can also single-handedly ruin a program. Both advantages and disadvantages of using COTS are discussed here. Make sure you study both!

CAUTION! COTS is a double-edged sword. Properly used, it can give the developer a great advantage. However, when it is not the right tool for the job, it can prove disastrous. Know when and how to use it, and when and how not to use it.

8.4.1.1 COTS Advantage

The Pentagon's long-range budget plans show future funding will be devoted to technological modernization after several years of steep declines. To modernize our software technology, cut costs, increase quality, and indeed, to maintain our superpower status, *we must become software users, instead of custom software developers*. By removing requirements for government-unique accounting standards, product specifications, and processes, DoD's purchasing system must become more compatible with that of the commercial marketplace. In addition, preference for the use of commercial standards and processes (established through the June 1994 Perry Memo), protection of technical data rights for commercial items, and a broadened exemption from cost data requirements is essential. [SULLIVAN94]

The 1994 report of the Defense Science Board identified an urgent need to integrate major parts of our defense-industrial base with our commercial-industrial base. It concluded that this integration allows DoD access to those technologies, products, and processes dominated by the

commercial sector that are far more advanced than military technology (e.g., electronics, software, computer hardware, robotics, telecommunications, etc.). [HERMANN94] The advantages to purchasing COTS software include:

- Intense competition leading to commodity-like pricing and alternative sourcing,
- Hundreds, to millions, of product users leading to low defect latency,
- Market pressures to rapidly innovate, leading to better products,
- Some degree of standardization leading to interoperable components between otherwise competing software manufacturers,
- Ease of migration to often revolutionary, future technologies,
- Built-in compliance with standards (although these are often as *de facto* as they are *de jour*),
- Exploitation of lower cost, quicker evolutionary development processes, and
- The use of commercially-developed tools and software engineering environments for increased automated development. [FAA94]

By exploiting these advantages in our acquisitions, we can achieve lower costs, faster developments, and more flexible maintenance with the ability to phase in new requirements throughout the software life cycle. We can gain greater capability by using COTS [and government-of-the-shelf (GOTS) and non-developmental item (NDI) (i.e., reuse)] instead of relying solely on new developments to meet our needs. COTS increase productivity by decreasing the lines-of-code to be developed, and improve quality by using code that is already tested and proven. In fact, COTS products can sometimes entirely preclude development, because they are often cheaper and more readily available than developed software. A well-used COTS application has been refined through updates (or versions) and corrected for latent defects — making it more reliable than newly developed, untried code. Additionally, vendor support of COTS is usually available. In the DoD environment, general-use applications such as word processors, spreadsheets, and cost models should generally be acquired as COTS.

8.4.1.2 COTS Integration

The Technical Architecture Framework for Information Management (TAFIM) is commonly known as a standards-based architecture (SBA). The integration of COTS with an SBA involves the partitioning of system capabilities into well-bound modules. An example of this integration is the mapping of modules from the TAFIM to traditional client/server components (presentation management, application function, data management.) Encapsulation refers to the classification and isolation of each system capability into appropriate client/server components. Further encapsulation within a component is sometimes necessary to ensure greater flexibility and ability to interchange COTS with other system components. This controls maintenance costs by allowing the developer to incorporate new technology rapidly with minimal impact to the other partitioned components.

Shrink-wrapped COTS products require little more than installation and configuration management. The use of a development language is not required to integrate software capability. Some shrink-wrapped examples include:

- Embedded communication software that contains all the logic necessary to enable the communications function, with only connectivity and configuration parameters required.
- Relational database management systems, from simple personal computer products to distributed solutions that provide centralized access to one or more disparate data sources. Some of these data access products are referred to as *middleware*.
- Vendor applications that perform standard commercial functions required by DoD, such as financial data tracking and administration, which provide common industry calculations and formula manipulation.
- Advanced industry scanning and image manipulation technology, such as Electronic Data Interchange (EDI) translators and wireless communications products.

Advantages of using shrink-wrap COTS are the incorporation of the latest technologies, automating manual processes, low initial cost relative to a new development effort, reduction of maintenance costs, and timely solutions to changing requirements. Often called the “*golden handcuffs*,” the disadvantages of shrink-wrap COTS include: proprietary development language (i.e., 4GLs), difficult customization to unique DoD requirements, inconsistent support for established standards, lack of real-time support, limited support for a centralized DoD data repository, and changing vendor relationships and can result in tight integration between two products for one version and loose (or no) integration for another version. Hillier explains, the keys to removing the “*golden handcuffs*” of reliance on vendor proprietary solutions include:

- Implementing the application component in a standard 3GL, independent of any proprietary COTS solution;
- Limiting the use of *de facto* vendor standards and seamless integrated COTS solutions to environments with no migration or cross functional requirements; and
- Selection of “*shrink-wrapped*” COTS and development environment COTS based upon product superiority within each selected partitioned component.

Factors to consider when selecting COTS products include:

- **Security.** Does the COTS solution provide two-way authentication, authorization, access control, privacy, and integrity?
- **Middleware supplier independence.** Does the solution truly provide greater independence from computer and network suppliers or does it simply shift to different ones?
- **Server supplier independence.** Does the solution employ general purpose server-independent middleware, to reduce vendor reliance and avoid affecting the client interface to the middleware layer?
- **Standards and interoperability certification.** Does the vendor supply verification of standards compliance (when appropriate)?
- **Training curve.** Does the product require considerable training and does the vendor provide it?
- **Costs.**
 - **Commodity pricing.** Is the COTS based on plug-compatible standards to promote lower licensing costs?
 - **COTS product and prerequisite support products.** Are there any hidden costs?
 - **New requirements.** How mutable is the COTS product to changing requirements? Do simple to moderate changes induce a ripple effect across the rest of the software?
 - **COTS version upgrades.** Are major changes planned in the COTS product to maintain a competitive edge with rivals?

- **Performance.** Does the COTS product provide (or enable the development of) the capabilities needed to satisfy mission needs?
- **Time to develop.** Will the use of the COTS product decrease (or at least not extend) the time to market for the system?
- **Memory, storage, and processing power.** Processing capability is cheap, but some solutions take lots and lots of memory which eventually increases cost at the PC and work station level. With what degree of efficiency does the software suite employ memory with reserves for future growth? For COTS, NDI, and software reuse, criteria should assess the complexity of integration. How transportable is the software and standard hardware architecture?

The level of COTS integration achievable depends on the amount of COTS software that can meet mission needs, of which there have been 100% COTS solutions. This situation, however, is not the norm. Most military systems require the addition of mission-specific logic. Some even have unique performance requirements that almost entirely preclude the use of COTS. These classes of systems can be categorized as:

- Predominantly COTS,
- Integration of COTS, new 3GL, and reusable 3GL assets, and
- Predominantly 3GL.

CAUTION! COTS products derive their quality (identification of latent defects) by an extensive body of users who identify the defects for fix in subsequent releases over a prolonged period of time. Quality is not necessarily designed-in as would be the case in a Cleanroom-based development. Accordingly, where safety or other critical requirements would suffer if the COTS contains hidden bugs, custom development may be preferable.

8.4.1.3 COTS Integration Lessons-Learned

The following are lessons-learned on the Loral Service Layer ReARC COTS software integration program.

- The customer's willingness to pay to gain the benefits of COTS must be understood. There must be a willingness to: abandon/modify some requirements if COTS cannot meet them; deal with sometimes disparate user interfaces; and accept the lack of control that COTS brings to the change process (slower reaction to problems, less desire to incorporate changes).
- Metrics for COTS integration are difficult to extract. *[The high degree of integration between COTS and ReARC developed code made correct allocation of efforts difficult to determine. Late selection of COTS product baseline (and even which requirements would be met with COTS) made the allocation of funds difficult between COTS and developed code.]*
- While COTS are less expensive than developed code for large, general applications (e.g., operating systems, DBMS), experience does not indicate this is true for smaller, niche products. Savings are anticipated in formal qualification tests (FQT) and operations and management (O&M). However, the customer is not always willing to be flexible enough to achieve reduced costs from COTS.
- Non-technical COTS selection criteria are as important as the technical. These include: vendor stability; product cycle; availability of support for back-level releases; and willingness of the vendor to work with the contractor.

- Managing product licenses can be a big headache. Therefore, get as flexible a set of licensing terms as possible — a site license is the best; *node-locked* licenses are the worst. Remember that every hardware baseline change can impact your COTS product baseline causing days of down time in the development lab.
- Do not believe what you read — *fly before you buy!* Early prototyping and integration with developed code (before CDR) is the only way to ensure that the product performs as advertised.
- Understand prerequisites and dependencies between products, especially when planning upgrades.
- Make sure procurement processes are in place to expedite COTS delivery.
- Establish hardware and COTS product baselines early. *[ReARC changes from a heterogeneous hardware environment (Sun and RISC) to a RISC-only environment was positive, but came too late and drove up costs by having to find replacement COTS products. Slow response from vendors can best be dealt with if problems with COTS are found early.]*
- Work with the customer to negotiate modifications to documentation requirements for COTS products.
- The cost of integrating COTS is much more front-end loaded than the cost of developing code. *[The developed code algorithm used on ReARC allocated 50% of cost to Preliminary Design Review (PDR) and Critical Design Review (CDR) phases, and 50% to Code and Unit Test, Software Integration, and CSCI Testing. They estimated that 75% of COTS integration costs come in the PDR and CDR phases. The potential exists for substantial savings in FQT, if agreements regarding verification of requirements satisfied by COTS are established early.]*
- Spend as little time as possible on paper trade studies. Get products to the lab for prototyping and integration. [RAND95]

8.4.1.4 More Cautions About COTS

The advantages of COTS (availability, cost, reliability) are evident in that all major DoD software-intensive systems are progressively increasing the use of COTS hardware and software. Aside from the advantages, there are, however, also some downsides to the use of COTS. Most COTS software is proprietary and the supporting agency cannot make changes to it. Therefore, COTS is not a good choice for weapon systems where the software must be continuously enhanced in response to changing mission requirements. Also, the Navy Seawolf BSY-2 program taught us that it is sometimes quicker and cheaper to simply build. You may be hard pressed to meet highly demanding volume and reliability requirements with commercial products not designed to perform under large-scale military conditions.

There are some fundamental differences between commercial software products and developmental software, as the final report of the 1991 Joint Command COTS Supportability Working Group concluded. [COTS91] Although cheaper than developing it yourself, be aware it is often difficult to integrate all the COTS applications (especially for weapons systems) needed to provide the required functionality. Even if your integration is successful, (for example, with 26% COTS combined with 74% developmental software) you can encounter configuration control problems. With new versions of each vendor's product being delivered at varying times, taking full advantage of the enhancements of each new product through changes in interfaces and interoperability with existing software can be like trying to catch a leprechaun to get his pot of gold. Every time you are about to grab him, he pops up somewhere else.

To alleviate this burden, traditional support approaches must be tailored to accommodate the COTS difference. Your approach should be commercially oriented to tap into the support infrastructure vendors establish to support their products. The 1991 report claims *the goal is not to become locked into sole-source contractor support for COTS*, but to take advantage of the competition inherent within the commercial sector which keeps prices low and quality high. [COTS91] To survive in a competitive market, vendors are intensely attuned to their customers' needs. Changes in your requirements will either show up in the next release of their software — or in that of their competitors.

NOTE: COTS products must be fully compatible with an open systems environment and must be accompanied by an assurance that they will be maintained by the COTS supplier for the life of the system.

Another concern with COTS is the chance for introducing a commercial virus into large, interconnected military software systems. Therefore, be very careful with software acquired from electronic bulletin boards, the public domain, or shareware sources, as they may contain hidden defects (and/or viruses) that can result in system failures, loss of critical data, or compromised security. That is not to say that these same problems do not exist in commercially acquired COTS, but that the incidence is lower. An additional hazard with integrating different software packages (not originally designed to work together) is that sometimes you get unexpected responses (or *side-effects*) under stressed operational conditions. Thus, for COTS to be effective, they must meet *de jour* or *de facto* interface standards. Your RFP must emphasize COTS compliance with controlled interfaces (e.g., those developed by the IEEE, ANSI, ISO, or NIST) that allow for evolutionary software exchanges. The benefit in using controlled interfaces is that software can be swapped out to improve reliability, gain performance, or better address changing requirements without impacting the entire system. Because hardware and software components must perform as integrated units, NIST has identified four key MIS interface categories which help in hardware/software partitioning:

- Application program interface (API),
- Human computer interface (HCI),
- Information storage and retrieval interface (ISRI), and
- Communications interface (CI). [FAA94]

To mitigate interoperability risk, you must evaluate each offeror's ability to judiciously select the proper standards for each interface category. NIST has developed an Applications Portability Profile (APP) model to ensure interoperability on a systems-wide basis, and to help make informed decisions on which interface standards to include. This model is useful for intelligently applying interface standards to the specific software system being procured.

The approach for achieving robust, easily upgradeable software varies widely among domains and should be approached differently. A rule of thumb for DoD software domains is:

- For a desktop environment, use 100% COTS;
- For MIS and some command and control (C2) systems, expect to base 80% of the system on COTS; and
- For weapons systems and real-time command, control, communications, and computers (C4) and C3I systems, maximize the use of COTS, GOTS, and NDI to allow technology upgrades of computer platforms and software components (such as display drivers, operating systems, database management, and communications) as technologies evolve. [FAA94]

The remaining unique components must be controlled at the interface level to allow for future upgrades with minimum impact to the system. When procuring COTS, consider the following:

- Require that COTS deliverables be included in the contractor's Logistics Support Analysis (LSA). The LSA must address contract and organic support for COTS throughout the system life cycle;
- Ensure sufficient documentation is provided for COTS software for life cycle operation and support;

NOTE: COTS documentation need not adhere to requirements which may be levied on developmental software.

- Support COTS software at the vendor's current revision level, unless upgrades will adversely impact operational capability;
- Use competitive commercial practices to the maximum extent when supporting COTS software. For COTS software in systems with a life cycle greater than five years, consider recompetition of logistics support contracts; and
- Obtain locally purchased COTS software from your requirements contract if it is designated as a mandatory purchase item.

8.4.1.5 Cautions About Modifying COTS

COTS software consists of unmodified software applications (except as intended by the vendor). *Commercial markets, independent contractors, and vendors control the design configuration and support (i.e., enhancements, modifications, and upgrades) of COTS software — not DoD.* There is a basic reason why we do not want to engage in the modification of COTS. If you change even a small portion of a COTS product, when the next version comes out your software will no longer be compatible or upgradeable to it. You will be faced with patching old technology, while advances in state-of-the-art pass you by. Thus, *do not purchase COTS software with the intention of modifying it!* By so doing, you negate its benefits and create a very substantial source of program risk.

When making your COTS decision, remember that DoD has learned the hard way why modification of COTS software is not a sound decision. Rationalizations for modifying COTS have included: some, but not all, of the users requirements were met with the COTS package; or the COTS package did not comply with Public Law, DoD regulations, Service regulations, policies, procedures, or current systems interface requirements. Although these sound like good reasons to do some tweaking of a COTS package, the consequences of doing so have been costly and often regrettable.

WARNING! Modifying COTS software can be hazardous to the success of your software development and downright deadly for follow-on support.

The Depot Maintenance Management Information System (DMMIS) is an example of a program that made the *change-the-COTS* decision. The DMMIS is based on the CINCOM MRPII COTS package. The team's original intent was to utilize this package unmodified. But after further analysis, they determined that the CINCOM package would have to be customized to satisfy the user. The decision to proceed with the COTS customization was based on the following issues:

1. The COTS package did not meet the original users' requirements;
2. The differences between CINCOM's manufacturing-based software and the Air Force's re-manufacturing-based processes; and
3. The need to interface with legacy software systems.

Although the DMMIS team believes they delivered a product superior to the original COTS package in less time than if developed from scratch, they learned some valuable lessons on why modifying proprietary software is not recommended. The team summarized the consequences of their COTS modification and lessons-learned as follows:

- They had to pay individual COTS charges to license each application running on each mainframe.
- They had to pay maintenance on the customized COTS packages.
- They had to pay an integrator to maintain their customized COTS packages.
- Difficulties surfaced when they wanted to take advantage of new releases of the original COTS. Now they no longer have the luxury of simply download a new version because their customized COTS has to be retrofitted to accommodate any new versions — costing down time and money.
- Customizing the COTS was time/money consuming because they had to understand what was needed, understand how and why the COTS was coded the way it was, and then figure out how to change it to meet their specialized needs.
- The combination of having to purchase the original COTS, having to employ an integrator on a FFP contract (in addition to having unstable requirements) led to more difficulties. The FFP contract forced the integrator to make short-term, quick fixes, the cheapest way possible. This, they know, will end up costing more in the long-run. It also required frequent open discussions between the developer and the user to reach an understanding — causing requirements creep.
- They learned that good requirements definition is crucial. This should be the first step before deciding on whether to modify COTS or develop software.

Recommendations the DMMIS team made for other programs considering a COTS modification include:

- Have the vendor make the modifications (not always possible).
- Instead of modifying the COTS package to match policies and procedures, the policies and procedures should be modified to match the COTS. This requires high-level management intervention and oversight.
- When software is available that meets the users' needs, COTS makes sense. When nothing is available, it may be necessary to develop software.

- If there is no other choice but to modify a COTS package:
 - The COTS package should meet most “*core*” requirements. Additional customization should be modularized, taking advantage of CASE tools (both in the COTS package and in the modifications), as much as possible. Make modifications *outside* the COTS package (i.e., put a shell around it) by modifying the COTS inputs and/or outputs, or by providing additional manipulations or enhanced data analysis so that new versions of the COTS package can *plug-n-play* with minimal changes by the integrator.
 - Use a cost-plus-fixed-fee (CPFF) or a firm-fixed-price-incentive-fee (FFPIF) contract so the contractor proposes the best long-term solution.

REMEMBER: You can surround COTS with interim functional layers that modify their inputs/outputs, but DO NOT MODIFY COTS!

8.4.2 Data Rights

Computer software data rights are of great importance to both the Government and the contractor. According to the FAR, the term “*data*” simply means *recorded information*, including software. “*Computer software*” means computer programs, computer data bases, and the documentation thereof. Policies governing the rights to these data are found in FAR Part 27.4, DFARS 227.72, and DFARS 252.227-7014.

NOTE: Purchasing COTS software should be your acquisition strategy only for those components that do not require change or maintenance by the Government. Thus, data rights for COTS should not be required.

Software data is divided into to categories: commercial computer software or commercial computer software documentation and noncommercial computer software and noncommercial computer software documentation. Regarding commercial computer software or commercial computer software documentation,

- (a) The Government shall have only the rights specified in the license under which the commercial computer software or commercial computer software documentation was obtained.
- (b) If the Government has a need for rights not conveyed under the license customarily provided to the public, the Government must negotiate with the contractor to determine if there are acceptable terms for transferring such rights. The specific rights granted to the Government shall be enumerated in the contract license agreement or an addendum thereto. [DFARS 227.7202-3]

Noncommercial computer software and noncommercial computer software documentation has data rights specified in three categories:

- **Unlimited rights** means rights to use, modify, reproduce, release, perform, display, or disclose computer software or computer software documentation in whole or in part, in any manner and for any purpose whatsoever, and to have or authorize others to do so. [DFARS 252.277-7014(a)(15)]

Unlimited data rights are usually associated with computer software developed exclusively with Government funds. For other situations that may apply, see DFARS 227.7203-5.

- **Government Purpose rights** means the rights to-
 - (i) Use, modify, reproduce, release, perform, display, or disclose computer software or computer software documentation within the Government without restriction; and
 - (ii) Release or disclose computer software or computer software documentation outside the Government and authorize persons to whom release or disclosure has been made to use, modify, reproduce, release, perform, display, or disclose the software or documentation for United States government purposes. [DFARS 252.227-7014(a)(11)]

Government purpose rights usually apply to software developed with mixed (Government and contractor) funding.

- **Restricted data rights** apply only to noncommercial computer software and mean the Government's rights to-
 - (i) Use a computer program with one computer at one time. The program may not be accessed by more than one terminal or central processing unit or time shared unless otherwise permitted by this contract;
 - (ii) Transfer a computer program to another Government agency without the further permission of the Contractor if the transferor destroys all copies of the program and related computer software documentation in its possession and notifies the licensor of the transfer. Transferred programs remain subject to the provisions of this clause;
 - (iii) Make the minimum number of copies of the computer software required for safekeeping (archive), backup, or modification purposes;
 - (iv) Modify computer software provided that the Government may-
 - (A) Use the modified software only as provided in paragraphs (a)(14)(i) and (iii) of this clause; and
 - (B) Not release or disclose the modified software except as provided in paragraphs (a)(14)(ii), (v) and (vi) of this clause;
 - (v) Permit contractors or subcontractors performing service contracts (see 37.101 of the Federal Acquisition Regulation) in support of this or a related contract to use computer software to diagnose and correct deficiencies in a computer program, to modify computer software to enable a computer program to be combined with, adapted to, or merged with other computer programs or when necessary to respond to urgent tactical situations, provided that-
 - (A) The Government notifies the party which has granted restricted rights that a release or disclosure to particular contractors or subcontractors was made;
 - (B) Such contractors or subcontractors are subject to the use and non-disclosure agreement at 227.7103-7 of the Defense Federal Acquisition Regulation Supplement (DFARS) or are Government contractors receiving access to the software for performance of a Government contract that contains the clause at DFARS 252.227-7025, Limitations on the Use or Disclosure of Government-Furnished Information Marked with Restrictive Legends;
 - (C) The Government shall not permit the recipient to decompile, disassemble, or reverse engineer the software, or use software decompiled, disassembled, or reverse engineered by the Government pursuant to paragraph (a)(14)(iv) of this clause, for any other purpose; and
 - (D) Such use is subject to the limitation in paragraph (a)(14)(i) of this clause; and

- (vi) Permit contractors or subcontractors performing emergency repairs or overhaul of items or components of items procured under this or a related contract to use the computer software when necessary to perform the repairs or overhaul, or to modify the computer software to reflect the repairs or overhaul made, provided that-
 - (A) The intended recipient is subject to the use and non-disclosure agreement at DFARS 227.7103-7 or is a Government contractor receiving access to the software for performance of a Government contract that contains the clause at DFARS 252.227-7025, Limitations on the Use or Disclosure of Government-Furnished Information Marked with Restrictive Legends; and
 - (B) The Government shall not permit the recipient to decompile, disassemble, or reverse engineer the software, or use software decompiled, disassembled, or reverse engineered by the Government pursuant to paragraph (a)(14)(iv) of this clause, for any other purpose. [DFARS 252.227-7014(a)(14)]

Restricted rights apply to computer software developed exclusively at private expense.

CAUTION! The entire issue of data rights is very esoteric. This discussion is intentionally general, as data rights are a sticky, controversial subject. It is recommended that you consult your contracting officer and/or legal advisor about your specific acquisition to flush out all data rights issues and alternatives. The primary concern should be the capability to maintain the software during its government-use life cycle at a reasonable cost (i.e., data rights per se are not the issue). The objective is life cycle supportability.

8.5 Source Selection Factors

Source selection factors for Air Force acquisitions are grouped into four areas: cost or price, past performance, mission capability (technical), and proposal risk. Factors and subfactors are to be limited to those that are real discriminators. Evaluation factors, subfactors, and elements:

- (i) Are to be written in enough depth to communicate the measures of merit used to determine how the proposal will be evaluated and rating determined;
- (ii) Should include only those specific program characteristics that are significant enough to have an impact on the source selection decision, such as those identified through program risk analysis;
- (iii) Are to be set forth in Section M of the draft and final RFPs, Evaluation Factors for Award. In addition, the relative importance of all factors, subfactors, and elements shall be specified in Section M of the RFP; and
- (iv) May be quantitative, qualitative, or a combination of both.

The Mission Capability factor should be limited to six subfactors, unless additional subfactors are justified, documented in the Source Selection Plan, and approved by the Source Selection Authority. Proposal risk is to be assessed at the Mission Capability subfactor level. Subfactors are not normally used for Past Performance and Cost or price.

See FAR Part 15.101-1, 15.101-2 and AF FAR Supplement 5315.304 for additional guidance regarding evaluation factors and subfactors.

Two categories that should always be considered as mission capability factors are first, does the proposed product or service satisfy the objectives of the SOO (product issues), and second, has the offeror described a development process that has the planning and maturity required to develop the product or service described within the resources estimated (process issues)?

8.6 Source Selection

Selecting a development contractor you can respect, trust, and with whom you can communicate and cooperate, is a major milestone in achieving program success and software excellence.

NOTES:

- 1. The GSA has published two reports: *Improving Industry/Government Communications in Major Information Technology Acquisitions and Communications Between Government and Industry: A Reference Guide for Federal Information Processing (FIP) Resources Acquisitions*. [See Volume 2, Appendix A for information on how to obtain them.]**
- 2. See FAR Part 15 and AFFARS Part 5315 for detailed descriptions of source selection requirements and activities.**

Consideration of these issues does not imply that contractor organizations are disrespectful, dishonest, hard-to-work-with cutthroats, or that their employees are contemptuous, lazy, thieving cheats. Cooperation, communication, respect, and trust are fragile commodities that must be established slowly and carefully and can evaporate quickly. The necessary conditions required for these high-performing team attributes to occur include the following:

- **Contracts history.** The offeror has never done anything on any other contract to make you distrust them, or if they have, their most recent accomplishments are sufficiently successful to supplant their earlier shortcomings. They should furnish historical data (size/time/effort) on 3-4 completed programs in the same domain and of comparable size and complexity as your proposed program. These data should then be used to assess bidder productivity.
- **Understanding of the problem.** The offeror knows what you want them to do, and has convinced you that they know how to do it.
- **Awareness.** You know their plan for meeting the requirement and are aware of their progress. Furthermore, they make sure you are aware of what they have accomplished, what the risks and problems are, how they are going to mitigate risks and solve problems, and where they are going. [HUMPHREY90]

The selection of a source(s), and subsequent award of a contract(s), follows a structured process which is designed to ensure impartiality while identifying the best-value for the Government. To solicit the best response from industry, offerors must know what discriminators you intend to use when they make their decision on whether to bid. The source selection process is illustrated in Figure 8-7. As with every other step in the software development process, *planning is the key to contracting success*.

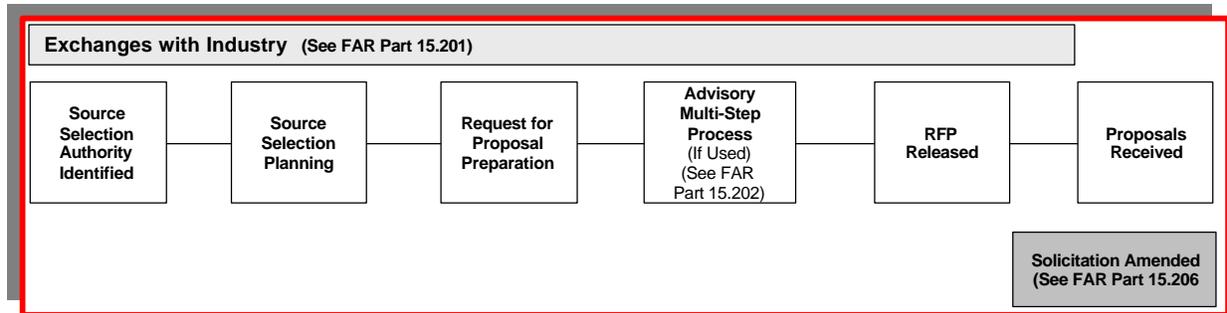


Figure 8-7. Source Selection Preparation Process

8.6.1 Source Selection Team Building

A key requirement for source selection is for the Source Selection Evaluation Team (SSET) chairperson to assemble a qualified team of evaluators. The SSET includes software experts in specialties such as software engineering, software architectural engineering, operating systems, compilers, software quality management and measurement, and database management systems and applications. SSET membership should also include functional user representatives and the Software Support Activity (and other Computer Resources Working Group (CRWG) members). However, remember the goal is to have a well qualified team that can address all stakeholder issues, not to have a large team with minimal qualifications who are only there to protect their parochial interests.

8.6.2 Source Selection Planning

Source selection planning is normally worked as an integral part of the RFP preparation process. It provides policy and procedures for developing the source selection plan and proposal evaluation. Government software acquisition source selection should include a judgment on the validity of each bidder's proposal based on a comparison with the Government's baseline plan and the bidder's demonstrated performance (i.e., historical corporate software development success).

8.6.3 Advisory Multi-Step Process

The advisory multi-step process is used by and agency to advise prospective offerors about their potential to be viable competitors. The agency may publish a presolicitation notice that provides a general description of the scope or purpose of the acquisition. The presolicitation notice should identify the information that must be submitted and the criteria that will be used in making the initial evaluation. Information sought may be limited to a statement of qualifications and other appropriate information (e.g., proposed technical concept, past performance, and limited pricing information). At a minimum, the notice shall contain sufficient information to permit a potential offeror to make an informed decision about whether to participate in the acquisition. This process should not be used for multi-step acquisitions where it would result in offerors being required to submit identical information in response to the notice and in response to the initial step of the acquisition.

The agency evaluates all responses in accordance with the criteria stated in the notice, and advises each respondent in writing either that it will be invited to participate in the resultant acquisition or, based on the information submitted, that it is unlikely to be a viable competitor. The agency advises respondents considered not to be viable competitors of the general basis for that opinion. The agency informs all respondents that, notwithstanding the advice provided by the Government in response to their submissions, they may participate in the resultant acquisition. [FAR 15.202]

8.7 Proposal Evaluation and Contract Award

Once proposals have been received, they must be evaluated against the source selection factors. This evaluation results in a ranking of the offerors. This ranking along with other findings are presented to the Source Selection Authority (SSA) for final decision. After consideration, the SSA awards the contract to the offer the SSA considers to be the best. See Figure 8.8 for the process flow.

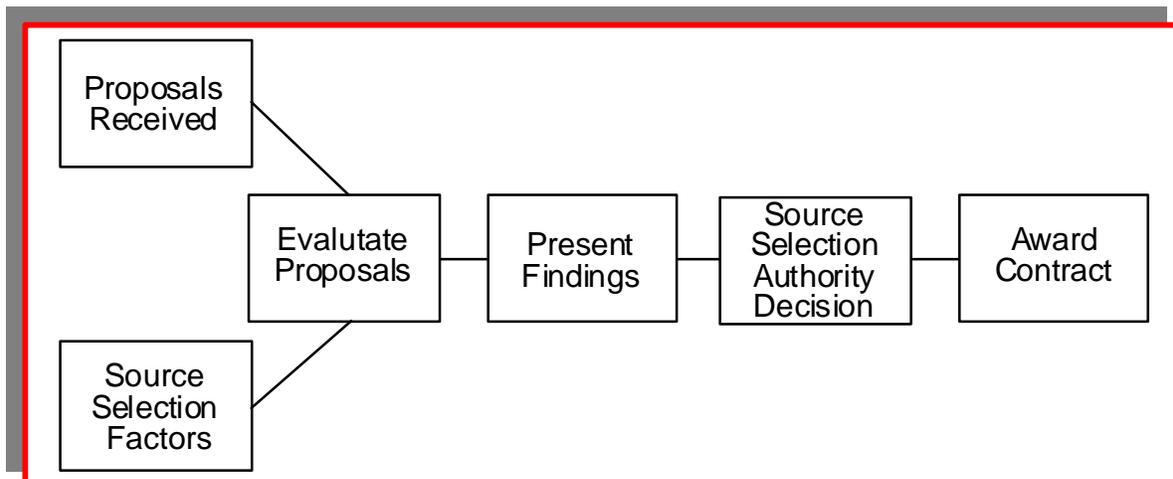


Figure 8-8. Proposal Evaluation and Contract Award Process

NOTE: Source selection factors must be approved by the SAA before RFP release.

8.7.1 Proposal Evaluation

The source selection evaluation team is responsible for the cost or price, past performance, mission capability (technical), and proposal risk evaluation of each proposal. (See FAR Part 15.305, 15.404, 15.407 and AFFARS 5315.305 and 5315.404) The evaluation must be rigorous, especially for system software, and alternative proposals must be evaluated for realism and value. Remember, the only evaluation criteria allowed are the evaluation factors and subfactors established during source selection planning. This is why it is so important that the evaluation factors and subfactors selected provide the capability to discriminate between offerors. Those offerors chosen for FPR should be required to submit to a software capability evaluation performed by the source selection team (and DoD personnel skilled in these evaluations). An SCE below Level 3 presents high risk and should be color-coded downward.

After completing technical, managerial, and cost evaluations, the program office sometimes conducts oral discussions with potential contractors to clarify any ambiguities in their proposals. If the offeror's approach is acceptable, it must then be made part of the contract.

Since peak manpower and staffing profiles so profoundly affect software development schedule, quality, and reliability, the *lower limits* on these two variables should be negotiated into all contracts.

CAUTION! To avoid too many nuisance changes to the contract and to preclude the contractor from using the inflexibility of “contractually specified processes” as an excuse for not meeting schedules, overrunning cost, or not meeting performance requirements, only the critical, top-level portions of the offeror’s proposal should be made part of the contract.

8.7.2 Best-Value versus the Cost of Poor Quality

When it comes to source selection, cost has always been a major consideration. Too often, it has been *the driving factor*. If you consider all the selection criteria and discriminators discussed herein, and believe you have identified the *best* developer who can deliver the highest quality product on a predictable schedule, the “*value of predictability and quality*” should influence your decision. *Remember, the Government is as interested in buying a sound and predictable process as an attractively packaged and well-described product.*

- **Best-value.** Best-value, as defined in the Federal Acquisition Regulation (FAR), “means the expected outcome of an acquisition that, in the Government’s estimation, provides the greatest overall benefit in response to the requirement.” It reemphasizes the concept that contracts should be awarded on the basis of “cost and other factors considered.” Too often, cost has been used as the only evaluation factor. FAR 15.101 describes the best-value continuum.

“An agency can obtain best value in negotiated acquisitions by using any one or a combination of source selection approaches. In different types of acquisitions, the relative importance of cost or price may vary. For example, in acquisitions where the requirement is clearly definable and the risk of unsuccessful contract performance is minimal, cost or price may play a dominant role in source selection. 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.”

- **Cost of poor quality.** Hungry for business in a shrinking defense market, companies are often prone to “*buying-in*” on programs for which they have little ability to deliver. Be leery of those organizations who promise low costs, but do not meet even rudimentary SCE maturity levels or your other source selection discriminators. The *cost of poor quality* can be significant (in terms of scrap and rework expense) when a contractor has to perform a process more than once to complete the work correctly. Even worse, you might have to write off \$100 million or more when an unsuccessful program is terminated, as happened recently with several programs. The challenge is to quantify, for “*best-value*” purposes, the surety that comes from selecting a contractor with a solid process, even if he is not the lowest bidder. It can be accomplished. It is helpful to identify, in your RFP, the names of one or more consulting firms who will assist you in calculating this value. At the same time, you must be assured that the best-value proposal carries a reasonable price tag. Be aware, *quality does not cost — it pays!*

Be aware, with best-value comes an added responsibility on the part of the Government to inform offerors in a clear, unambiguous RFP how you will evaluate them equally — in exactly the same precise manner. Otherwise, from the loser’s perspective, you are simply choosing the contractor you want regardless of any other considerations. You must clearly state whether you are buying the least cost, minimally compliant, or best-value. If best-value, you must state that technical solutions are more important than the costs associated with the program, while still working within an established budget. (SEE FAR Part 15.101-1) Remember, *valid source selection decisions must be based on life cycle costs — not only up front costs*. If your available funding profile is a constraint, it should be so identified to all offerors.

WARNING! The final bidding and pricing process. To remain competitive, offerors often reduce quality assurance manhours to shave costs. It is imperative that the SSET tracks any changes at FPR and change technical/management evaluations as required. While the quality assurance requirement remains part of the contract, performance in this area may be jeopardized. [MARCINIAK90]

8.7.3 Present Findings to the SSA

After evaluating each proposal against the evaluation factors and determining merit as compared to the factors, the SSET leader presents the results to the SSA. For the Air Force, the format for reporting evaluation findings can be found in AFFARS 5315.305.

8.7.4 SSA Decision and Contract Award

The SSA is the contract award authority. The SSA’s decision must be based on a comparative assessment of proposals against all source selection criteria in the solicitation. While the SSA may use reports and analyses prepared by others, the source selection decision must represent the SSA’s independent judgment. The source selection decision is documented, and the documentation includes the rationale for any business judgments and tradeoffs made or relied on by the SSA, including benefits associated with additional costs. Although the rationale for the selection decision must be documented (See AFFARS 5315.308-90 for a list of required documents), that documentation need not quantify the tradeoffs that led to the decision. [FAR Part 15.308 and AFFARS 15315.308]

ATTENTION MANAGERS! Success requires a contractor with top-down commitment to software engineering practices, concepts, technology, training, planning, people, and the willingness to commit corporate resources to achieve quality goals. The best-value contractor you select must be able to provide you with a superior technical solution at a reasonable cost.

8.8 Protests

To avoid protests, you must be careful when crafting your best-value RFP. You must articulate precisely what you mean by “value.” Vendors need to know about your program, its mission, goals, and objectives. They must understand how this software purchase plays in the accomplishment of DoD’s mission and in what user environment it will operate. You must state your needs in functional terms as much as possible. Without this background, offerors are likely to be in the dark about what you value most, and therefore, about how you will evaluate their proposals. [PETRILLO93] Clear, explicit detail must be included in your RFP about your definition of “value” and how proposals will be rated. The technical and cost relationships in award criteria and how they will be applied to the selection must be specified. Cost must be given relative weight along with technical criteria so there is not the impression of indiscreet flexibility (and thus subjectivity) in your evaluation.

NOTE: Also make provisions in your RFP, Section M, for assignment of value to product features or development approaches not anticipated.

Make sure your technical evaluators have ensured that every mandatory solicitation requirement has been met by the proposed awardee. A surprising number of protests are sustained because the awardee did not meet a mandatory requirement. Conversely, limit the number of mandatory requirements in the solicitation to those that are absolutely necessary. A situation in which the solicitation calls for more than 500 mandatories is a recipe for protest.

Unfortunate experiences have shown that DoD cannot always rely on the integrity of even well-known vendors. Best-value procurements are particularly prone to error. In three recent major acquisitions, vendors have displayed the misconception that an agency can make whatever choice they wish in best-value procurements. This is only true to the extent that the decision is well-reasoned and documented. It is also essential that an agency follow its own rules. The General Services Board of Contract Appeals (GSBCA) sustained protests where the solicitation states that an award would be based on a particular weighting of cost and technical factors, and where the agency did not follow that formula. [See “Centel Systems versus Department of the Navy,” GSBCA No. 12011-P, 1992 BPD, paragraph 359.]

If sophisticated cost-technical tradeoffs are being made, it is essential that those making them have the necessary skills to perform the appropriate judgments and to do the quantifications required. This is especially true if the choice is the technically high-scored, high-cost solution. The case law does not require the source selection team to close the price gap between a high-tech, high-priced solution and a low-tech, low-priced solution. However, as a practical matter, it is very difficult to explain why you chose a \$150 million *blue-blue* solution over a \$100 million *green-green* solution unless you close the price gap in quantifiable terms. Sometimes this is

accomplished through productivity studies and sometimes by evidence of other types of savings. There is no particular formula for quantifying a best-value price-gap closing, *but it must be defensible*.

A mere list of desirable features along with the statement that these features are worth the extra money does not normally do the job. On one recent Internal Revenue Service (IRS) award, for example, the source selection decision lacked documentation. There was no detailed narrative statement explaining why the superior aspects of Company A's proposal were worth the large gap in price between its proposal and the other offers — from \$500 to \$700 million dollars. The only documented support for closing the price differential was something the IRS called “*Methods A & B*.” The agency's “*A & B*” approach attempted to quantify the differences between the relative power of the multi-user systems offered. The IRS attempted to express this difference in dollar terms, using what is often called a “*normalization process*.” Company A's mid-level systems were 6 times more powerful than Company B's systems. Because of this, the IRS analysts assumed that they would have to buy 6 times fewer systems from Company A as from Company B and discounted Company A's price accordingly. This simplistic analysis was rejected by the GSBCA because the IRS did not take into account the benefits the Government would receive from the different offers. The offers were made on the basis of the IRS' estimated quantities, and Company B did not offer 6 times as many systems as Company A. The second time around the IRS did a much more sophisticated job of measuring the comparative benefits of the different offers and the GSBCA, and ultimately the Federal Circuit Court, sustained their award.

Another recent case illustrates the problems of using a crude *normalization process*. It also illustrates the problems in relying on one approach in general. It is better to use several methods and to look at each one of them carefully. Often a price-gap closing will require a source selection team with a combination of information resource management and accounting skills of a high order. Institutional or personal arrogance in recognizing this requirement is often punished later on. The theory that “*we do not need help to do our own procurements*” is fine until the protest is decided adversely for DoD. Then no excuses are accepted. Do not assume that your in-house people are as knowledgeable as the extremely expensive experts who will be hired by disappointed vendors.

Be prepared to “*fight fire with fire*,” and do not be hesitant to bring on your own big guns. *It is suggested that serious consideration be given to hiring an outside expert to “red team” the award decision before it is signed.* This expert may wind up testifying for the Government. His or her insights will be extremely helpful in anticipating future challenges. SAF/GCP will help you secure such an expert if desired. In anticipation of the need to employ experts, you must identify your intentions in your RFP, including the identity of the experts, or firm of experts, you intend to employ.

- **Industry involvement.** Keeping industry involved throughout the acquisition planning phases is also essential to a successful procurement. Up until the RFP is released, you should maintain an open, public dialogue with industry through industry briefings, by providing open reading libraries, allowing industry to brief you on potential solutions, and by releasing draft RFPs. One or more draft RFPs ensure that requirements are better defined and understood. Industry can better respond to draft RFPs than amendments to RFPs that cost time and money, or even worse, multiple RFPs. [WAYS94] You can require that offerors submit a cost and technical tradeoff analysis based on their understanding of your system’s mission. You can also have them submit alternative proposals offering different solutions with low-cost/low-performance and high-cost/high-performance alternatives. [BRENDLER93] This will enable the SSET to consider the implementability, applicability, and validity of each offeror’s proposal.

Can an exponentially lower price be credible? Perhaps. In one recent acquisition a small, innovative company using Cleanroom engineering bid \$20 million. The next lowest bids were in the \$120 million range. The lowest cost bid, which the source selection authority acknowledged had the best technical proposal, was not accepted because the cost was considered too low (and risky). In this instance, it may have been in the Government’s best interest to make two awards — one for the \$20 million proposal and another for a more conventional proposal. This would have served the dual purpose of reducing risk, and, if the Cleanroom solution had resulted in a successful product, demonstrating that revolutionary improvement in software productivity, quality, and cost is feasible.

8.9 Navy Seawolf Lessons-Learned

The largest Ada software development in the US Navy, the AN/BSY-2 software for the SSN 21 Seawolf submarine, was estimated to cost \$1.4 billion, was to be made up of 4.6 million source lines-of-code (SLOCs) (of which 3.0 million will be new and 1.6 million reused and COTS), and was to be built by over 600 software personnel. The AN/BSY-2 software component has come under GAO scrutiny “*as being exposed to unnecessarily high risk by rushing production to meet milestones.*” [JENKS92] Lessons-learned from a program of this size and complexity are invaluable for software RFP preparation. Recommendations from Seawolf lessons-learned include:

- Review subcontractor agreements against the prime contract for contractual consistency.
- Require early identification of all commercial products and associated licensing agreements.
- Require that the contractor establishes a system performance model upfront to be maintained throughout the program.
- Expand database design documentation requirements to include a logical, as well as, a relational and physical design.
- Clearly define firmware documentation deliverables.
- Require standardized software Style Guides for technical documentation produced across multiple developer sites. Make the prime contractor establish a single point-of-contact as the Style Guide distributor.
- Include electronic format as an optional contract delivery medium.
- Require that the contractor provides analyses of metrics data with respect to the SDP.
- Require installation of secure links between software developers and prime contractor sites.

- Require the prime contractor to establish controls for the management of common code and the document control process in software developers' SDPs and documentation.
- Require the use of a standard set of support tools across development teams (e.g., compiler, code counter, document generator).
- Require strict version control of support tools throughout development to ensure all delivered software and firmware is compiled or assembled under the baselined version.
- Require the prime contractor to implement a shared problem reporting system across all developers early in the program.
- Evaluate hidden risks associated with the use of COTS tools (e.g., limited life cycle vendor support, upward incompatibilities, and schedule impacts associated with porting tools).
- Require the establishment of common databases to track all prime item development specification (PIDS) requirements for flow-down, and to ensure consistency across interfaces during integration.
- Make sure the contractor allocates additional time and resources for tool modification and for resolution of COTS interface/performance problems.
- Encourage the use of a standards checking tool to ensure high quality, maintainable code.

8.10 References

- [ALIC92] Alic, John A., et al., Beyond Spinoff: Military and Commercial Technologies in a Changing World, Harvard Business School Press, Boston, Massachusetts, 1992
- [BAKER92] Baker, Emanuel R., "TQM in Mission Critical Software Development," G. Gordon Schulmeyer and James I. McManus, eds., Total Quality Management for Software, Van Nostrand Reinhold, New York, 1992
- [BRENDLER93] Brendler, Beau, "Best-Value Unclear: Procurement Philosophy Explored at Meeting," *Washington Technology*, October 7, 1993
- [COTS91] *Joint Command Commercial-off-the-Shelf (COTS) Supportability Working Group (CSWG) Final Report*, June 1991
- [DuPICQ80] du Picq, Col Charles Ardnant, Battle Studies, 1880
- [FAA94] "Report of the 'Open System Development' Subcommittee," briefing prepared by the Federal Aviation Administration, Research, Engineering, and Development Advisory Committee, Headquarters, Washington, D.C., 1994
- [HERMANN94] Hermann, Robert, "Defense Science Board Task Force on Acquisition Reform," *Army Research, Development, and Acquisition Bulletin*, January-February 1994
- [HUMPHREY90] Humphrey, Watts S., Managing the Software Process, Addison-Wesley, Reading, Massachusetts, 1990
- [JENKS92] Jenks, Andrew, "DoD to Change Purchasing," *Washington Technology*, August 27, 1992
- [KRATZ84] Kratz, L. A., Drinnon, J. W., and Hiller, J. R., Establishing Competitive Production Sources: A Handbook for Program Managers, Defense Systems Management College, Fort Belvoir, Virginia, August 1984
- [MARCINIAK90] Marciniak, John J., and Reifer, Donald J., Software Acquisition Management: Managing the Acquisition of Custom Software Systems, John Wiley & Sons, Inc., New York, 1990.
- [MILLS95] Mills, Andy, "Software Acquisition Improvement: Streamlining Plus Risk Management," paper presented to the Seventh Annual Software Technology Conference, Salt Lake City, Utah, April 1995
- [MOSEMANN94] Mosemann, Lloyd K., II, comments provided to AFPAM 63-115, May 1994
- [PETRILLO93] Petrillo, Joseph J., "If Agencies Won't Articulate Needs, Can There Be Best-Value?" *Government Computer News*, October 25, 1993
- [RAND95] Rand, Linda M., "ReARC COTS Software Integration Lessons-learned," briefing presented on July 24, 1995
- [SPAT92] "Software Process Action Team, Process Improvement for Systems/Software Acquisition," Air Force Systems Command, Final Report, June 20, 1992
- [SULLIVAN94] Sullivan, Bruce E., "The Section 800 Report: Streamlining Defense Acquisition Law," *Army Research, Development, and Acquisition Bulletin*, January-February 1994
- [VELOCCI91] Velocci, Anthony L., Jr., "ATF Development Program's Risk Light for Lockheed, Financial Officer Says," *Aviation Week & Space Technology*, May 13, 1991
- [WAYS94] Ways, John P., "Best-Value Not the Best Approach: Procurement Strategy is Well-intentioned But Hard to Implement," *Washington Technology*, February 10, 1994