SYM-AM-17-048

Proceedings of the Fourteenth Annual Acquisition Research Symposium

Wednesday Sessions Volume I

Acquisition Research: Creating Synergy for Informed Change

April 26–27, 2017

Published March 31, 2017

Applying the Fundamentals of Quality to Software Acquisition

Steve Bygren—The MITRE Corporation

Greg Carrier—The MITRE Corporation

Tom Maher—The MITRE Corporation

Patrick Maurer—The MITRE Corporation

David Smiley—The MITRE Corporation

Rick Spiewak—The MITRE Corporation

Christine Sweed—The MITRE Corporation

Abstract

Historically, software developed under government contracts often does not stand up under real-world use, and defects frequently result in cost and schedule overruns. While proposed development activities from contractors commonly list measures to improve quality, these descriptions cannot be used to select a winning bidder if they are not part of the evaluation criteria. By making software quality requirements explicit at the proposal stage, contractor selection can be influenced by criteria based on best practices in software development.

If we want to improve the quality of our software, a "Quality in Depth" approach is needed—introducing quality-related measures at every stage of software acquisition. In a previous article, one of the authors provided recommendations for improving software quality at the construction phase (Spiewak & McRitchie, 2008). This article discusses how to apply these same principles to the source selection process.

In order to find a way to include software practices as selection criteria, the authors set out to identify and recommend changes to Sections L and M of a government RFP (Request for Proposal) or IFPP (Instructions for Proposal Preparation) and EC (Evaluation Criteria) in an attempt to improve software and system quality. These changes will enable selection teams to identify contractors whose software development processes and compliance with software quality standards are more likely to produce the desired results.

Background

What Is Software Quality?

Quality is often thought of as an absence of defects. With many software products however, "defect" does not adequately describe the range of phenomena that affects software quality as perceived by the customers, end users, and other stakeholders. Using Crosby's philosophy, we define the term "software quality" to mean conformance to the requirements of the software product's users and other stakeholders (Crosby, 1979). The more closely a software product conforms to these requirements, the higher its quality.

We are particularly interested in software quality as it affects the acquisition process for defense-related software. While end user requirements are of prime importance, poor software development and quality monitoring practices in early- and mid-stage acquisition can result in failure to provide the desired results. These failures range from unwanted or missing features to cost and schedule overruns to critical flaws in system security or reliability.

How Do You Measure Software Quality?

Software quality as an outcome is best measured by the number of defects encountered after development is complete as the numerator, divided by the "size" of the software as the denominator. One could also argue that if two different products were to be compared, some sort of "difficulty factor" could be applied, as well as references to the software language used or development environment employed (e.g., assembly code versus high order languages, or object-oriented versus functional languages, etc.).

Metrics exist which can be used to estimate the potential defects in code. These are based on the use of function points as the measure of "size." Function points can also be (loosely) correlated with the commonly used measurement SLOC, or Source Lines of Code.

Approach

This article is the outcome of a study the authors conducted at MITRE. Our approach was to gather information from subject matter experts (SMEs), contracting officers, and acquisition experts for recommendations for additions to proposal documents. Part of this study was conducted through interviews and SME email group lists. Reference materials from the Air Force and Navy were found which provided recommendations from prior work (USAF, 2008; Office of the Assistant Secretary of the Navy [ASN(RD&A)], 2008). We then adapted the suggestions to Sections L and M to more thoroughly describe software quality related criteria for source selection. Some of these criteria are aimed at the technical evaluation team, while some can be used by cost evaluators and past performance evaluators, as well as the technical team.

Recommendations for Section L (Instructions for Proposal Preparation)

1. The offeror's proposal shall include a proposed Software Development Plan (SDP) which describes their approach to software development, including the tools, techniques, and standards to be used for development; unit testing and component testing; integration tools and techniques (including configuration management) used to ensure the integrity of system builds; the number and type of reviews that are part of the development process; and the methods and tools used to manage defect reports and analysis, including root cause analysis as necessary. The proposed SDP will form the basis for a completed SDP to be available after contract award as a CDRL (Contract Deliverable Requirements List) item, subject to government review and approval.
2. The offeror shall describe their plan for effective code reuse in order to minimize the amount of new code to be developed. Reused code can come from any origin, including previous efforts by the offeror or as provided by the government in the bidders' library.
3. The offeror shall provide a Basis of Estimate (BOE) describing the rationale for the proposed staffing. The detail of the BOE shall include labor hours for each labor category (e.g., system engineering staff versus software engineering staff) for the identified tasks in the Work Breakdown Structure (WBS) as it relates to the Statement of Work (SOW).
4. The offeror shall describe the process for orientation and training for all project employees (e.g., certification and training in software best practices including Information Assurance [IA] and risk management).
5. The offeror shall describe related systems experience, including a description of previous experience developing software of the same nature, and a

description of the extent to which personnel who contributed to these previous efforts will be supporting this effort.

6. The offeror shall describe proposed development practices. For example, if spiral / incremental development, they shall describe the number, duration, and scope of spirals, as well as how the use of your approach would result in improved product quality and user satisfaction over time.[1]
7. The offeror shall provide an Integrated Master Schedule (IMS) and accompanying narrative that describes all significant program activities that are aligned with the proposed program staffing profile. Include a timeline for completion of each activity identified in the proposed program. Provide details that clearly describe the purpose for and importance of key activities. Identify all critical path elements and key dependencies.

Recommendations for Section M (Evaluation Criteria)

The proposed SDP shall show a complete and comprehensive software development process, which incorporates best practices as well as standards such as IEEE 12207-2008. The contractor will be evaluated based on how their processes, as described in the SDP, incorporate the use of software best practices.

Evaluation criteria related to the SDP include the following:

- The number and type of peer reviews
- The use of automated unit testing including test coverage requirements
- The use of automated syntax analysis tools and adherence to the rules incorporated by them (Jones, 2010)
- The comprehensiveness of integration and test methods, including continuous integration tools if used
- The use of readiness requirements such as unit test and syntax analysis for code check-in
- Configuration management and source code control tools and techniques
- The extent to which root cause analysis of defects is part of the development process
- The selection of software source code to be reused, replaced, or rewritten from previous implementations or other origins, including a description of how it will be ensured that reused code meets or is brought up to the same standards as newly developed code. Risks associated with reused software shall also be discussed. Such software shall include government rights to the source code.

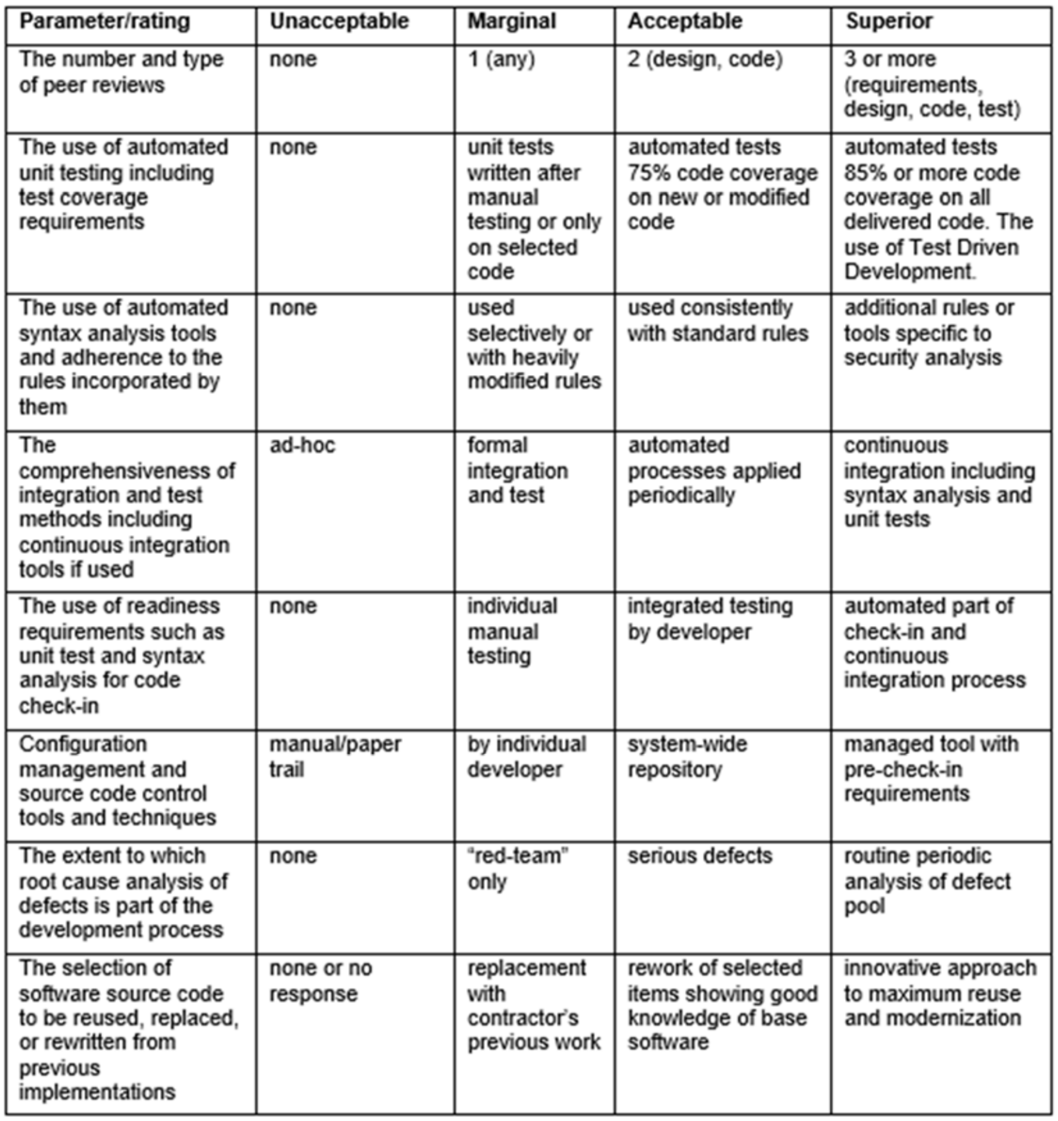
The IMS and accompanying narrative will be evaluated for level of detail and relevance of significant program activities, degree of alignment, the proposed program staffing profile, and integration of the proposed SDP into the IMS. Additionally, critical path

[1] Note that while not part of the technical evaluation, the government evaluation team will examine Contractor Performance Assessment Reports [CPARs] for relevant performance by the respondent on other contracts.

elements and key dependencies will be assessed for relevance, completeness, and the manner and level of risk containment.

Table 1 delineates a sample rating scale for SDP evaluation criteria.

Table 1. Sample Rating Scale for SDP Evaluation Criteria



Parameter/rating	Unacceptable	Marginal	Acceptable	Superior
The number and type of peer reviews	none	1 (any)	2 (design, code)	3 or more (requirements, design, code, test)
The use of automated unit testing including test coverage requirements	none	unit tests written after manual testing or only on selected code	automated tests 75% code coverage on new or modified code	automated tests 85% or more code coverage on all delivered code. The use of Test Driven Development.
The use of automated syntax analysis tools and adherence to the rules incorporated by them	none	used selectively or with heavily modified rules	used consistently with standard rules	additional rules or tools specific to security analysis
The comprehensiveness of integration and test methods including continuous integration tools if used	ad-hoc	formal integration and test	automated processes applied periodically	continuous integration including syntax analysis and unit tests
The use of readiness requirements such as unit test and syntax analysis for code check-in	none	individual manual testing	integrated testing by developer	automated part of check-in and continuous integration process
Configuration management and source code control tools and techniques	manual/paper trail	by individual developer	system-wide repository	managed tool with pre-check-in requirements
The extent to which root cause analysis of defects is part of the development process	none	"red-team" only	serious defects	routine periodic analysis of defect pool
The selection of software source code to be reused, replaced, or rewritten from previous implementations	none or no response	replacement with contractor's previous work	rework of selected items showing good knowledge of base software	innovative approach to maximum reuse and modernization

Note. The categories provided in Table 1 were suggested in a conversation with Jeff Pattee, Chief, Product Definition, Airspace Mission Planning Division, Electronic Systems Center, USAF.

Incorporating Software Quality Measures in Contracts

The contract development process includes several steps at which information can be gathered and requirements set to include software quality as a measure of vendor performance.

Sections L & M or equivalent from the RFP

- Add software quality measures as a discriminating factor in selecting the contractor

- Enumerate expectations in this area:
    - Types of methods used
    - Evidence to be provided

TRD (Technical Requirements Document), SOO (Statement of Objectives), and SOW (Statement of Work)

Add requirements in the form of deliverable items—as CDRLs or DAL (Data Accession List) items as appropriate. Examples include the following:

- Output of automated unit tests showing code coverage at or above required minimum
- Output of automated syntax analysis showing conformance to pre-determined rules
- Evidence of accomplishing required peer reviews
- Itemized list of tools with version numbers used to produce output from each source module
- Programmer's reference manual with examples
- Interface definitions
- List of all software components with the following information:
    - Purpose and function
    - Interfaces provided
    - Language/version for each module
    - Complete source code
- Source from architectural design tool where available
- Use cases (text and diagrams)
- Class diagrams where applicable
- Complete list of any third-party components with version numbers
- Contact information for any outside dependencies
- Build procedures, including documentation for building all software components from source code
- Test procedures—including any automated unit tests with source code, test scripts

Rationale for Incorporating Recommended RFP Language

The recommended RFP language was derived by the authors from a variety of sources including MITRE acquisition subject matter experts, existing guidance documents from the Navy and Air Force, and also from the authors' experience. We've tried to provide a succinct rationale as to why the language asks for specific information from the contractor in the RFP:

- The Software Development Plan (SDP) is a maturity indicator of the bidder's development process. By evaluating this, and then putting its provisions under contract, it becomes possible to select a contractor on the basis of development methodology and then obligate them to perform as proposed.
    - Automated unit tests & comprehensive peer reviews are widely used best practices. Capers Jones (2008a) has noted that these are among the required steps to achieve effective defect removal.

  - Continuous Integration (CI) often includes the automated invocation of tests and code analysis during the build process. CI and static analysis expose problems sooner in the development process. The sooner problems are discovered, the lower the cost to resolve them.
  - Root cause analysis prevents the introduction of defects and is a recognized best practice in all approaches to process improvement. It is a Capability Maturity Model Integration (CMMI) Level 5 practice area. Prevention is more cost effective than detecting and fixing defects after they are introduced.
- The Basis of Estimate (BoE) helps the evaluator understand the bidder's cost to compare against industry averages and government cost models. By examining proposed labor categories, this can be checked against predicted labor distributions from government cost models as well.
- The Integrated Master Schedule (IMS) can be checked for alignment with required milestone dates, and it supports an independent estimate.

Guidance for Evaluation Team Experience

The government's evaluation team must have relevant software engineering experience. The experience should cover the full life cycle of software development from design to development, integration, testing, and delivery. If the proposal is seeking a particular style of development methodology (e.g., waterfall, spiral/incremental, agile), then the evaluation team should have experience in that methodology in order to evaluate the RFP response.

Since a significant portion of the suggested contract language relates to software quality monitoring, the evaluators should be familiar with unit testing, peer reviews, continuous integration (CI), static code analysis, and metrics. Finally, evaluators should have some knowledge of various practices and approaches of applying these techniques, for example, when it comes to test-driven development.

The field of software engineering is diverse. It is insufficient to simply have general software engineering experience on the evaluation team without further having experience in the applicable domain(s). Examples of these domains include real-time/embedded, kernel/operating systems, numerical/digital signal processing, web applications, SOA, information retrieval/search, security, and human-computer interface.

Finally, the evaluation team should have an understanding of the CMMI process and rating criteria.

Guidance for Evaluating Technical Responses

The recommended contract language in this article includes Section M of the RFP, also appearing as Evaluation Criteria. The language is not very specific so as to elicit responses that are more original than simply claiming the ability to do a long list of things that the government requires. In this section, we discuss more specific guidance for the evaluation team in evaluating the responses.

In advance, the team should define objectives that are sought after and then define measurable criteria. The more objective the criteria, the better, though it is recognized that coming up with this criteria can be a challenge. After defining criteria, they are prioritized and then weighted in a scheme the team deems appropriate.

The following are some general evaluation tips:

- If key staff are identified in the proposal, how likely are they to be available during contract execution?
- In reference to quality assurance processes, does the proposal language favor or at least mention "empowerment" of the QA team over engineering processes?
- Regarding the contractor's approach to Automated Unit Testing, does the contractor require that unit tests be passed and cover a reasonable percentage of code before code can be checked in? Does the contractor use Test Driven Development?
- Regarding the contractor's approach to automated syntax analysis, does the contractor require that syntax analysis be performed and that all required rules are followed before code can be checked in?
- Regarding development build and integration, does the contractor use an automated build process which incorporates syntax analysis and automated unit testing?

You can expect that the response is going to claim appraisal at a specific CMMI maturity level (commonly at least Level 3). This can be verified with the Appraisal Disclosure Statement (ADS) document. Another source is the Standard CMMI Appraisal Method for Process Improvement (SCAMPI). For the larger contractors, particularly when work is further sub-contracted out, look for further CMMI level compliance information on the specific division/unit and sub-contractor(s) as applicable.

Development Process

If the proposal declares that a development process will be used that will involve multiple iterations/spirals/increments (which is standard practice), then the evaluation team should look for further details on the process including the following:

- What is the duration and scope of each increment?
- Are lessons and obstacles from one increment reviewed for improvement to a subsequent increment?
- Is user (customer) feedback interaction only up front or do most increments incorporate this? And how is that feedback prioritized?
- Are multiple increments planned in sufficient detail, or are only the present and possibly next increment planned?

Software Engineering

One key thing to look for in a proposal is the degree to which the contractor has experience in the technology the RFP calls for them to deliver. The more complex the system, the more important applicable contractor experience is.

Many DoD systems have a degree of interoperability and integration required of them. For integration with particular systems, verify if the contractor has experience with that system or has relationships with third parties with integration capabilities that will be used. The contractor should also participate in applicable Communities of Interest (COIs).

Testing processes and technologies that support them are important. Look for information on a test plan or strategy. If the proposal is serious about continuous integration and use of supporting tools, then listing the software to be used for this is a promising sign.

Information on how the tools are used (e.g., by exception and/or monitored on a periodic basis—and what period) is also telling. If the proposal includes information on the proposed system design, then the evaluators could look to see how "testable" the design is, particularly as it is incrementally built.

Conclusions

While it is important to implement quality measures in software construction, this is undertaken after a contractor has been selected. The authors recommend an in-depth approach, beginning with the process of selecting the contractor. It can be easy to overlook the importance of including specific language in the proposal documents in order to be able to select the right contractor from those responding to a Request for Proposal. In order to accomplish this goal, it is critical to specify the instructions in Section L (or the IFPP) and the evaluation criteria in Section M (or the EC) so that these can be used to assign strengths or weaknesses appropriately. This is an early, but often neglected, piece of the puzzle involved in building quality software products for defense applications.

Reference List

Crosby, P. B. (1979). *Quality is free: The art of making quality certain*. New York, NY: McGraw-Hill.

DoD. (2010, August 5). *Defense acquisition guidebook*. Retrieved from http://at.dod.mil/docs/DefenseAcquisitionGuidebook.pdf; current version available at https://dag.dau.mil/Pages/Default.aspx

Jones, C. (2008a, June). Measuring defect potentials and defect removal efficiency. CrossTalk. Retrieved from http://www.crosstalkonline.org/storage/issue-archives/2008/200806/200806-Jones.pdf

Jones, C. (2008b, August). *Software quality in 2010: A survey of the state of the art.* Retrieved from http://www.spr.com

Jones, C. (2010). *Software engineering best practices*. New York, NY: McGraw-Hill.

Office of the Assistant Secretary of the Navy (ASN[RD&A]). (2008, September). *Guidebook for acquisition of Naval software intensive systems*. Retrieved from https://acquisition.navy.mil/rda/content/download/5657/25845/version/1/file/Guidebook+for+Acquisition+of+Naval+Software+Intensive+SystemsSEP08.pdf

Spiewak, R., & McRitchie, K. (2008, December). Using software quality methods to reduce cost and prevent defects. CrossTalk. Retrieved from http://www.crosstalkonline.org/storage/issue-archives/2008/200812/200812-Spiewak.pdf

United States Air Force (USAF). (2008, August 15). *Weapon systems software management guidebook*. Retrieved from https://acc.dau.mil/adl/en-US/24374/file/49721/USAF%20WSSMG%20%20ABRIDGED.pdf