DOD-STD-2167, *Defense System Software Development:* Point, Counterpoint, and Revision A

Peter Coad Jr.
TELOS Software Methods Training

TELOS Aerospace Systems
320 N. Halstead
Pasadena, California 91107
(818) 351-2341

DOD-STD-2167, *Defense System Software Development,* is a product and process standard for mission-critical software development (it may and has been applied on other software developments as well). The standard is now in wide use in the U.S. and abroad.

Yet the application of any standard is challenging. Problems and rework can be avoided by examining DOD-STD-2167 from a point/counterpoint view, and then taking action accordingly.

DOD-STD-2167 Revision A will be released in December 1987. Its potential impact is evaluated too.

Background

Software cost overruns and schedule setbacks have long been a problem for many organizations, including the U. S. Government.

An investigation into this trend prompted the Joint Logistics Commanders (JLC) of the U.S. Government to sponsor a series of initiatives, and then a software standardization program, to address three major problems:

(1) Lack of a consistent standard for software development and documentation
(2) Inadequate visibility and control of software development efforts
(3) Lack of a uniform approach to acquiring software.

The earliest workshops which led to DOD-STD-2167 (referred to as 2167) were conducted in 1979. From 1983 to 1985, both the Government and three major industry associations were actively involved in putting together 2167. Some projects put the standard into effect as early as August 1984 (the author was a software methods consultant on such a project). The official release date of 2167 was June 4, 1985.

Even at the time of publication, certain issues were tabled for further consideration. People knew that the standard would need to be changed, to address these issues and to respond to feedback from those applying the standard. The intent of DOD-STD-2167 and its future versions is to capture the state of good practice, rather than the state of the art.

The next version of 2167, DOD-STD-2167A (referred to as 2167A) is expected to be issued in December 1987. 2167A has major changes directly reflecting lessons learned from applying 2167:

- Keeping language independence (with no obstacles to Ada®)
- Including systems engineering
- Tailoring the standard for specific projects
- Reducing the documentation requirements
- Adding flexibility to promote documentation automation

An overview diagram of 2167A is included at the end of this paper.

2167: Point, Counterpoint, and Revision A

To examine 2167 from both point and counterpoint perspectives, we will examine 2167 from practical experience--since November 1984, the author has consulted and trained professionals on the tailoring of 2167 and the application of modern software methods in the context of 2167.

To examine the potential impact of 2167A, we will examine the revised standard itself. [The September 11, 1987 draft is used for this article; the conference presentation will cover the latest 2167A documentation set available at that time.]

The key issues examined in this section are the following:

Visibility and Control
Consistent Framework
Defaults and Tailoring
Potential Cost Savings
Systems Engineering
Reviews.

Visibility and Control

• Point

DOD-STD-2167 provides the contracting agency with visibility and control.

Software cost overruns and schedule setbacks are painful. The Government's primary intention with 2167 is to get contractors to have more control over the software development process.

• Counterpoint

Admittedly, cost overruns and schedule setbacks are undesirable. And the Government's intentions are admirable. But 2167 goes too far in required visibility and control---too much time and money must be spent on the contractually required documentation and the multitude of minute details called for by 2167.

• Revision A

The new version of the standard is simpler and shorter (about 50 pages, down from the previous 70 or so). Fewer documents are required (18 distinct documents are established, rather than the previous 24). 2167A allows for much more flexibility; it is simply less confining. In addition, 2167A has been reformatted for enhanced readability.

Consistent Framework

• Point

2167 provides a consistent framework for software development. It is a consistent process and product standard. Moreover, 2167 places heavy emphasis on requirements definition, preliminary design, and detailed design.

• Counterpoint

Consistency is a wonderful thing. And emphasizing requirements definition and design is fine too. But once again, 2167 goes too far. The consistent approach is incompatible with alternate development processes, notably object-oriented development, evolutionary development, rapid prototyping, and multiple build (design a little / code a little / test a little) approaches.

• Revision A

2167A goes to great [almost extreme] lengths to avoid expressing any defaults. For example, software development "phases" are called "activities", to avoid any process bias. So a contractor can now readily define and utilize alternate development processes, including object-oriented development, evolutionary development, rapid prototyping, and multiple build (design a little / code a little / test a little) approaches.

Defaults and Tailoring

• Point

2167 specifies a default software development process and corresponding default software methods, capturing the state of good practice in 1985. In addition, 2167 makes room for contractors to propose alternate approaches and methods.

• Counterpoint

Yes, 2167 does include defaults. And it is interesting that the Government went so far as to define a state of good practice for software processes and methods. But no process or method is appropriate for all development projects in all organizations. Such defaults are totally out of place in such a standard.

Moreover, these defaults have inhibited the introduction of the latest (and possibly proprietary) software methods which are different than the defaults. To be safe, the defaults are taken as law; both the contracting agencies and the contractors bid the job, do the work, and review the results according to the book---every method, every example table, every document, every "jot and tittle."

Producing every document according to the letter of the law has significant economic and engineering effects. For projects in which a subset of the documents would be appropriate, cost savings are missed. For larger projects, the burden of documentation is so high that the entire engineering organization becomes document-driven. More effort is expended to produce the documents than to engineer the system under consideration. Furthermore, when customer reviews focus on format compliance, rather than on engineering content, the dilemma is only amplified.

The data requirements and default tables sometimes call for descriptive attributes sooner in the development cycle than necessary. At times, to be compliant, attributes are added with little meaning or engineering value, simply to fulfill a data requirement.

• Revision A

No defaults are included in 2167A. It is painstakingly process, method, and language independent.

Even the example tables are very clearly identified as just that---examples.

Reducing and eliminating unneeded paragraphs and documentation is encouraged by 2167A. And format modification to support automated tool generation of the documentation is also encouraged.

The documentation burden has been eased but remains an issue, as with any software development standard.

Data attributes are required "as appropriate," which still is a problem for those who want it all, even when inappropriate.

Potential Cost Savings

• Point

2167 represents a potential cost savings by replacing its inconsistent, incomplete, and hard-to-comprehend predecessors with one uniform standard across multiple Government agencies. This has potential cost savings. Moreover, a consistent standard makes data collection for better metrics possible.

• Counterpoint

Yes, 2167 represents a potential cost savings by replacing its inconsistent, incomplete, and hard-to-comprehend predecessors with one uniform standard across multiple Government agencies. But this assumes that the agencies will use the same basic standard. Yet different Government agencies have already developed their own tailoring and customization of 2167. And as for the documentation burden of 2167, time will tell whether the added up-front expense pays off in the development cycle in building systems which better meet customer needs, and during the lengthy maintenance phase which follows.

• Revision A

For 2167A and its cost effectiveness, time will tell.

Systems Engineering

• Point

2167 addresses the software engineering process and its products. It does not need to cover systems engineering issues.

• Counterpoint

It is silly to isolate software engineering from the overall systems engineering perspective. To be useful in practice, systems engineering should be addressed.

• Revision A

2167A includes systems engineering activities and documentation as an integral part of the software development standard.

Reviews

• Point

2167 establishes many internal review and formal reviews.

• Counterpoint

Internal reviews (best known as inspections, a formal application of walkthroughs) are excellent. And periodic reviews to communicate with the contracting organization is essential. However, 2167's rigid phase-by-phase development approach has adverse cost and schedule impacts on software development. As a major review approaches, the contractor begins to focus its key technical personnel on preparing an award-winning presentation. Documentation is frozen well in advance for the presentations. This usually means that what is presented at the review is inconsistent with the up-to-the-minute presentations made by the key technical staff (who have been busy enhancing the development concepts while preparing for the review).

Periodic customer reviews, to give visibility as to where the project is at a given point in time, would be much more effective than the giant review at the end of each phase espoused by 2167.

• Revision A

Formal reviews in 2167A remain a potential problem. But this may be alleviated for a given project, by specially defining reviews as periodic "in process" checks with the customer, rather defaulting to the more customary major theatrical productions.

DOD-STD-2167A: A Recap

DOD-STD-2167A, *Defense System Software Development*, is due for release in December 1987.

The new version of the standard is simpler and shorter (about 50 pages, down from the previous 70 or so). Fewer documents are required (18 distinct documents are established, rather than the previous 24). 2167A allows for much more flexibility; it is simply less confining.

2167A goes to great [almost extreme] lengths to avoid expressing any defaults.
No defaults are included in 2167A. It is painstakingly process, method, and language independent.

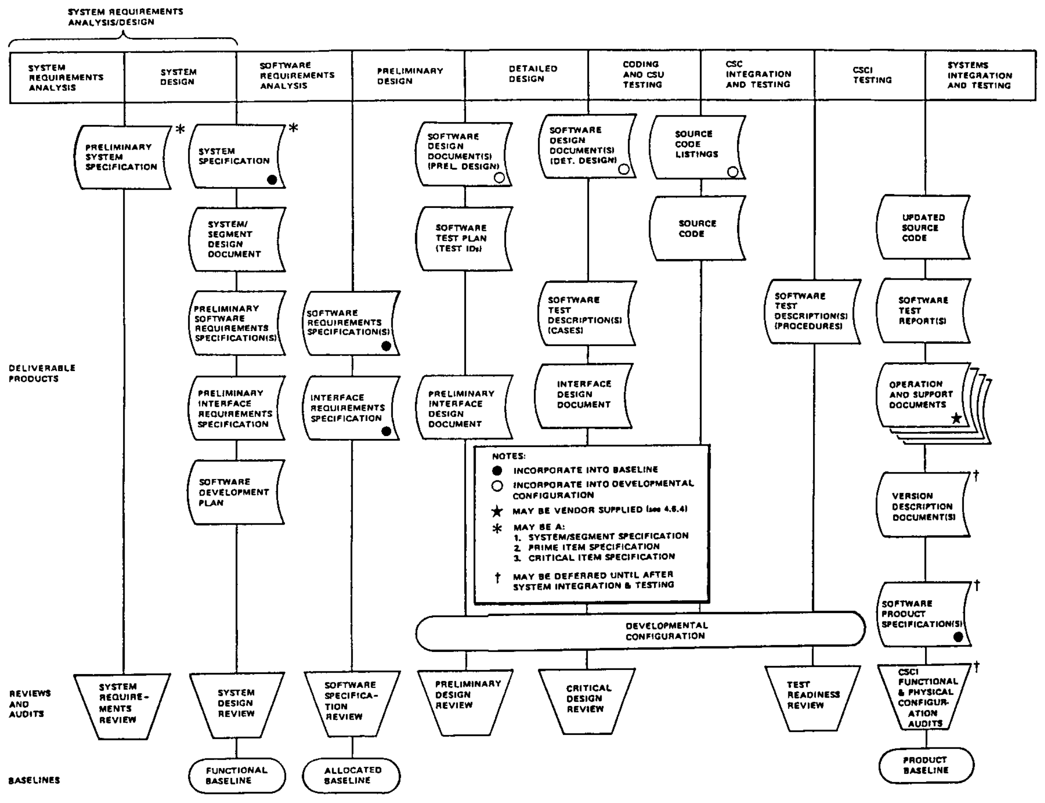
Reducing and eliminating unneeded paragraphs and documentation is encouraged by 2167A. And format modification to support automated tool generation of the documentation is also encouraged. The documentation burden has been eased but remains an issue, as with any software development standard.

Data attributes are required "as appropriate," which still is a problem for those who want it all, even when inappropriate.

2167A represents a potential cost savings. Time will tell. Moreover, a consistent standard makes data collection for better metrics possible.

2167A includes systems engineering activities and documentation as an integral part of the software development standard.

Formal reviews in 2167A remain a potential problem. But this may be alleviated by specially defining the reviews as periodic "in process" checks with the customer, rather than the more customary major productions.


SYSTEM REQUIREMENTS ANALYSIS/DESIGN
SYSTEM REQUIREMENTS ANALYSIS
SYSTEM DESIGN
SOFTWARE REQUIREMENTS ANALYSIS
PRELIMINARY DESIGN
DETAILED DESIGN
CODING AND CSU TESTING
CSC INTEGRATION AND TESTING
CSCI TESTING
SYSTEMS INTEGRATION AND TESTING
DELIVERABLE PRODUCTS
PRELIMINARY SYSTEM SPECIFICATION *
SYSTEM SPECIFICATION ● *
SYSTEM/ SEGMENT DESIGN DOCUMENT
PRELIMINARY SOFTWARE REQUIREMENTS SPECIFICATION(S)
PRELIMINARY INTERFACE REQUIREMENTS SPECIFICATION
SOFTWARE DEVELOPMENT PLAN
SOFTWARE REQUIREMENTS SPECIFICATION(S) ●
INTERFACE REQUIREMENTS SPECIFICATION ●
SOFTWARE DESIGN DOCUMENT(S) (PREL. DESIGN) ○
SOFTWARE TEST PLAN (TEST IDs)
PRELIMINARY INTERFACE DESIGN DOCUMENT
SOFTWARE DESIGN DOCUMENT(S) (DET. DESIGN) ○
SOFTWARE TEST DESCRIPTION(S) (CASES)
INTERFACE DESIGN DOCUMENT
SOURCE CODE LISTINGS ○
SOURCE CODE
SOFTWARE TEST DESCRIPTION(S) (PROCEDURES)
UPDATED SOURCE CODE
SOFTWARE TEST REPORT(S)
OPERATION AND SUPPORT DOCUMENTS ★
VERSION DESCRIPTION DOCUMENT(S) †
SOFTWARE PRODUCT SPECIFICATION(S) ● †
NOTES:
● INCORPORATE INTO BASELINE
○ INCORPORATE INTO DEVELOPMENTAL CONFIGURATION
★ MAY BE VENDOR SUPPLIED (see 4.6.4)
* MAY BE A:
1. SYSTEM/SEGMENT SPECIFICATION
2. PRIME ITEM SPECIFICATION
3. CRITICAL ITEM SPECIFICATION
† MAY BE DEFERRED UNTIL AFTER SYSTEM INTEGRATION & TESTING
DEVELOPMENTAL CONFIGURATION
REVIEWS AND AUDITS
SYSTEM REQUIRE-MENTS REVIEW
SYSTEM DESIGN REVIEW
SOFTWARE SPECIFICA-TION REVIEW
PRELIMINARY DESIGN REVIEW
CRITICAL DESIGN REVIEW
TEST READINESS REVIEW
CSCI FUNCTIONAL & PHYSICAL CONFIGUR-ATION AUDITS †
BASELINES
FUNCTIONAL BASELINE
ALLOCATED BASELINE
PRODUCT BASELINE