Ada* AND DOD-STD-2167A: STATUS AND IMPLICATIONS

presented by

Donald G. Firesmith

Chairman SIGAda SDSAWG

Magnavox Electronic Systems Company
M/S 10-C-3 Dept. 566
1313 Production Road
Fort Wayne, IN 46808

(219) 429-4327

* Ada is a registered trademark of the U.S. Government (AJPO).

Topics

- DOD-STD-2167
- DOD-STD-2167A (Initial Draft)
- DOD-STD-2167A (Second Draft)
- DOD-STD-2167A (Third Draft)
- Ada as the Default
- Process Standard Issues
- Method-specific Omissions
- Implications
- Future direction of SDSAWG

DEFENSE SYSTEM SOFTWARE DEVELOPMENT (DOD-STD-2167)

- Approved for use by all departments and agencies of the DoD on 4 June 1985.
- Supersedes Navy-specific MIL-STD-1679 and a great many DIDs.
- Contains requirements for the development of Mission Critical Computer Resource (MCCR) software.
- Establishes a uniform software development process.
- Intended to be language and method independent.

DOD-STD-2167A (Initial Draft)

- Initial draft of Revision A released for formal review on 1 September 1986.
- Ada Community review coordinated by the SIGAda Software Development Standards and Ada Working Group (SDSAWG).
- Several hundred comments received from Ada Community.
- SDSAWG comments together with the SDSAWG Issues and Subissues Report submitted to the JLC/CSM Subgroup for incorporation.

DOD-STD-2167A (Second Draft)

- Second draft of Revision A released for formal review 1 April 1987.
- Reviewed by Ada Community and SDSAWG membership and subgroups.
- Several hundred more comments received from Ada Community.
- Some SDSAWG Issues and Concerns appear addressed.
- SDSAWG comments submitted to JLC/CSM Subgroup via CODSIA SDS Task Group and AJPO.
- AJPO submitted comments on the standard, but not on the DIDs.

DOD-STD-2167A (Third Draft)

- Third draft of Revision A presented to a small group of industry and government representatives at JLC/CSM Comment Resolution Meeting on 20-24 July 1987.
- Third draft (and other documentation) to be distributed to the SDSAWG membership and subgroups.
- More SDSAWG Issues and Concerns appear addressed.
- Many AJPO comments postponed to DOD-HDBK-287.
- SDSAWG Issues and Subissues Report to be updated.
- DOD-STD-2167A to be released as approved standard sometime in October, 1987.

Ada as the Default

- Why?
  - Consistency with DOD Directives
  - Promote Ada Usage
  - Same Application Area
  - Precedents in DOD-STD-1777 and DOD-STD-1778
- Specific Issues:
  - Definition of Unit
  - Coding Standards
  - Ada in the DIDs

Definition of Unit

- Concerns included:
  - Ada Units vs. Computer Software Units
  - Functional Units vs. Data Units
  - Logical vs. Physical Units
  - Unit Testing
    - Lack of Definition
    - Subprogram Testing vs. Package Testing
  - Nested Units
  - Visibility between Units
  - Distinction between LLCSCs and Units

- According to DOD-STD-2167:

  "Unit.

  The smallest logical entity specified in the detailed design which completely describes a single function in sufficient detail to allow implementing code to be produced and tested independently of other Units. Units are the actual physical entities implemented in code."

- According to the third draft of Revision A:

  "Computer Software Unit (CSU).

  The smallest logical entity specified in the design of a [Computer Software Component] and the actual physical entity in code that implements a testable aspect of the requirements."

Coding Standards

- Language-Independent Standard
  - Logic constructs
  - Precompiler requirement
  - Fixed in draft Revision A
- Ada Coding Standard
  - Default, Example, or Minimal
  - Order of Coding Standards
    (Fixed in second draft of Revision A)
  - Default Nature of Prolog
  - Reusability and Portability
  - Ada-specific Comments

Ada in the DIDs

- The DIDs do not cantain any examples of the use of Ada to present design information.
- A precedent for the use of Ada exists in MIL-STD-1777 and MIL-STD-1778.
- The required information is not always relevant and often involves compiler-dependent details down at the bits and hardware level.
- Use Ada code to document Ada design.
- Although multiprocessors and tasks are now an issue, there is no place to document concurrency below the CSCI level. Concurrency paragraphs are needed for TLCSCs and LLCSCs in the STLDD and for CSUs in the SDDD.

- Use of Ada in the DIDs would:
  - Promote the use of Ada
  - Promote use of Ada as a Design Language
  - Ensure consistency between requirements, design, and code
  - Promote readability of identifiers
  - Ease of document update
- Ada Requirements and Design DIDs (like NASA)? Status unknown.

Process Standard Issues

- Why?
  - Process Standards inhibit Innovation.
  - Best Software Development Process is Application, Language, Methodology, and Time dependent.
  - Contain "how-to" constraints that are in direct violation of the Acquisition Streamlining Directive (DODD 5000.43).
  - Acquisition vs. Development standards.
  - SDP loophole is viewed as insufficient.

- Specific Issues:
  - Waterfall Life-Cycle
  - Static Hierarchy
  - Top-Down Default
  - PDL Default/Requirement
  - Functional Orientation
  - DoD Acquisition Process
  - Formal Reviews

Waterfall Life-Cycle

- According to DOD-STD-2167:

  "4.1 Software development cycle. The contractor shall implement a software development cycle that includes the following six phases: ..."

- Many other models exist.

- Incremental reviews and overlap are insufficient.

- Incompatible with recursive methods, such as Object-Oriented Development (OOD).

- PDR and CDR may need to be redefined.

- Violates DODD 5000.43.

- According to the third draft of Revision A:

  "4.1.1 Software development process. The contractor shall implement a software development process for managing the development of the deliverable software. The contractor's software development process for each CSCI shall be compatible with the contract schedule for formal reviews and audits. The software development process shall include the following major activities, which may overlap and may be applied iteratively or recursively: ..."

Static Hierarchy

(Software Organization)

- According to DOD-STD-2167:

  "4.2 Computer software organization. Computer software developed in accordance with this standard shall be organized as one or more CSCIs or other types of software (see 1.2.1). Each CSCI is part of a system, segment, or prime item and shall consist of one or more Top Level Computer Software Components (TLCSCs). Each TLCSC shall consist of Lower-Level Computer Software Components (LLCSCs) or Units. LLCSCs may consist of other LLCSCs or Units. TLCSCs and LLCSCs are logical groupings. Units are the smallest logical entities, and the actual physical entities implemented in code. The static structure of the CSCIs, TLCSCs, LLCSCs, and Units shall form a hierarchical structure as illustrated in Figure 3. The hierarchical structure shall uniquely identify all CSCIs, TLCSCs, LLCSCs, and Units."

- According to the third draft of Revision A:

  "4.2.6 Computer software organization. The contractor shall partition and organize CSCIs into Computer Software Components (CSCs) and Computer Software Units (CSUs) based on system, operational, and support requirements. The contractor shall allocate requirements from a CSCI to one or more Top-Level CSCs (TLCSCs) and then to Lower-Level CSCs (LLCSCs) and CSUs, as appropriate (see Figure 3)."

- Note that TLCSCs and LLCSCs are no longer explicitly described as "logical groupings" only.

- Mapping to Ada structure is both difficult and controversial.

- The software organization:
  - Is fundamental to the DOD-STD-2167 design process and the DIDs.
  - Inhibits reuse and certain design methods by imposing a hierarchical decomposition method.
  - May only be useful for Configuration Management.
  - Is a "how-to" design constraint that violates DODD 5000.43.

Top-Down Default

- According to DOD-STD-2167:

  "4.8 Development methodologies. The contractor shall use a top-down approach to design, code, integrate, and test all CSCIs, unless specific alternate methodologies have been proposed in either the SSPM or SDP (see Appendix D) and received contracting agency approval (see 6.2)."

- Not all methods are top-down (e.g., recursive and bottom-up approaches are often superior). DOD-STD-2167 itself incorporates bottom-up testing and integration.

- The optimum approach impacts the ease of reuse and is impacted by Ada compilation order restrictions.

- The top-down default violates DODD 5000.43.

- According to the updated third draft of Revision A:

  - "[Foreward] 2. This standard is not intended to specify or discourage the use of any particular software development methodology. The contractor is responsible for selecting software development methodologies that best support the achievement of contract requirements."

  - "4.2.2 [Software] development methods. The contractor shall use systematic and well documented software development methods to perform software requirements analysis, design, coding, integration, and testing of the deliverable software. The contractor shall describe in the Software Development Plan the development methods and how the methods will be implemented in support of the formal reviews and audits."

Program Design Language

Default/Requirement

- Default during Preliminary Design.
- Requirement during Detailed Design.
- PDL may not be always appropriate for Preliminary Design.
- PDL may not be useful for Detailed Design.
- PDL's purpose changing.
- Violates DODD 5000.43.
- Fixed in second draft of Revision A.

Functional Orientation

- By promoting functional decomposition methods (a "how-to" design constraint), DOD-STD-2167 violated DODD 5000.43.
- Software Requirements Specification content and format were in terms of functions.
  - Fixed in second draft of Revision A.
- Original definitions of Static Hierarchy entities were in terms of functions.
  - Unit definition fixed in second draft of Revision A.
  - CSC definitions to be fixed in final Revision A.

DoD Acquisition Process

- DOD-STD-2167 is driven by the DoD Acquision Process, Directives, and system-level standards.
- Acquisition Process is system/hardware oriented.
- Appropriate for modern Software Development Methods?
- Software First.
- May be very difficult to fix.

Formal Reviews

- The formal reviews should be methodology dependent.
- The linear "bottle-neck" nature of the formal reviews and the evolving nature of the Critical Design Review (CDR) imply "how-to" manage and design constraints that violate DODD 5000.43.
- Due to improvements in Revision A, this is more of a problem with MIL-STD-1521 than with DOD-STD-2167.

Method-Specific Omissions

- Why?
  - DOD-STD-2167 is largely silent.
  - Ada designed to promote prototyping and reuse.
- Specific Issues:
  - Prototyping
  - Reuse
  - Automation

Prototyping

- Prototyping impacts life-cycle.
- MIL-STD-1521 formal reviews (Hardware vs. Software).
- No detailed design prior to PDR (MIL-STD-1521).

Reuse

- No definition of reuse or reusable provided.
- Assumes everything built from scratch.
- Inhibits reuse (contracting agency approval required to reuse).
- Not appropriate for building reuse libraries.
- Incompatible with top-down default.
- Compromise reached at JLC/CSM Comment Resolution Meeting.

Automation

- More than just Document generation.
- Impact on Life-Cycle.

Implications

- Some major problems probably remain:
  - Static Hierarchy
  - Ada vs. language-independent design DIDs
  - Relationship to MIL-STD-1521
- Because each version of Revision A has been an improvement, use the latest version and tailor for Ada (if possible).

Future direction of SDSAWG

- Ada DIDs for DOD-STD-2167
- Tailoring Guidelines for Ada Projects
- DOD-HDBK-287
- MIL-STD-1521
- DOD-STD-2167B
- NASA standard and DIDs
- NSA standard and DIDs