Software Standards: Their Evolution and Current State

Reed Sorensen
TRW

This article touches on the last 30 years of software standards development in the Department of Defense, the purpose of and the difference between formal and de facto standards, the Ada experience, some current thinking regarding the documentation monster, and an update on J-STD-016 status.

"Stuff Happens" – so do Standards

If you sit down and try to categorize all the types of standards, the list gets really long really fast [1]. Under each type you start to come up with multiple entries. Standards seem to proliferate and infiltrate any aspect of human endeavor that involves any degree of complexity. Standards are not inherently good or bad, they just seem to inherently "be" — meaning that whether they are developed formally by a big, slow-moving international committee or whether they grow spontaneously from an innovative free market, standards are not going away.

While standards as a life-form will survive, evolving a specific species can be (and probably always is) difficult. Standards have to be cared for, they have to be fed, they have to evolve or they become extinct. Two examples:

1. Something as basic to the military as the ability of joint task forces to share standardized location data, just "ain't" that easy [2];
2. J-STD-016 has been going through the revision and ballot process for more than a year and should be undergoing re-balloting as you read this, but delays in this process have threatened its existence.

Software Standards Evolved Over 30 Years

Table 1 is a representative list of standards. Common school of thought has software standards evolving from the black ooze of hardware standards of the 1970s to early '80s. Some hardware standards began including words, sometimes in the form of appendices, to deal with software, e.g. MIL-STD-483 [3] and MIL-STD-490 [4]. Later, software-specific standards evolved that talked like software animals using software terminology, but walked liked hardware animals because they were direct hardware descendants, e.g. DOD-STD-2167 [5]. Today, the software community has troubled over the differences between software and hardware enough that software standards (EIA/IEEE J-STD-016 [6], IEEE/EIA 12207 [7]) are genuine software standards rather than feathered lizards.

Both MIL-STD-498 and J-STD-016 promote a more flexible approach to defining the software process and collaboration among all stakeholders, e.g. with joint technical and management reviews. Compared to older standards, MIL-STD-498 and J-STD-016 provide:

- a more complete life cycle perspective with links to system engineering
- better consistency with modern development models (incremental, evolutionary, reuse/re-engineering)
- improved accommodation of nonhierarchical design methods
- more flexibility with documentation formats and media
- attention to reusability, security, safety, and risk management
- improved references to metrics and indicators
- more responsive in-process evaluation and review activities
- significantly improved transition to software support and sustainment activities

Meanwhile, IEEE/EIA 12207 provides a higher-level view, with fewer specific requirements, and spans not only development, but acquisition, supply, operation, and maintenance.

Table 1. *Chronological list of some major software and software related standards.*

Date	Designator	Topic
1968	MIL-STD-490	Specification Practices
1970	MIL-STD-483	Configuration Management Practices
1978	DOD-STD-480A	Configuration Control
1979	MIL-S-52779A	Quality
1983	MIL-STD-1815A	Ada83 Language Reference Manual
1983	DOD-STD-7935	Documentation Standards
1983	DOD-STD-1679A	Software Development
1984	DOD-STD-1644B	Trainer System Software Development
1985	MIL-STD-490A	Specification Practices
1985	MIL-STD-1521B	Technical Reviews and Audits
1985	DOD-STD-2167	Defense System Software Development
1988	DOD-STD-7935A	Documentation Standards
1988	DOD-STD-2167A	Defense System Software Development
1988	DOD-STD-2168	Defense System Software Quality
1992	MIL-STD-973	Configuration Management
1994	MIL SPEC REFORM	
1994	MIL-STD-498	Software Development and Documentation
1995	J-STD-016-1995	Software Life Cycle Processes, Software Development, Acquirer-Supplier Agreement
1998	IEEE/EIA 12207	Software Life Cycle Processes
1998	EIA-649	Configuration Management
1999	J-STD-016	Software Life Cycle Processes, Software Development

Standards Enable Communication

Standards are about communication — communication between organizations, e.g. between the acquirer and developer, among developers, between the developer and the maintainer, communication between two commercial-off-the-shelf software products or two software units, and communication between a fax machine built by Canon and another built by Sharp.
The revolution in knowledge that accompanied the printing press was a communication revolution. The printing press provided the means to build upon the experience of those who came before[1]. A standard is either a specialized document that captures the experience of the many for the use of those who follow, or it is a groundswell that, by virtue of inertia, establishes default experience. In the first case, many may meet in committee and have a balloting process to gather experience from the broadest base of useful knowledge. In the second, a company may flood the marketplace with its solution to a problem in hopes that it will become a de facto standard — not that the solution is inherently the best but if the solution becomes omnipresent it will provide the most effective mechanism for most people to deal with most others (Microsoft Windows 3.1 for example).

Formal Standards Influence Software Development

Entire cottage industries pop up around formal standards. There are a lot of consulting organizations offering CMM® and ISO 9000[2] expertise. Seminars are available and tools are marketed to support "498" [8] and "12207." All of these kinds of free market ventures do their best to influence software development.

But not everyone buys into the concept of formal standards. Formal standards tend to be imposed by management rather than because the programmers sign a petition demanding that they want to be CMM Level 3, though life for the programmer is often better at Level 3 than Level 1.

The Interplay of Formal Standards, De Facto Standards, and Best Practices

Alex Polack of ATA, a vendor of software documentation technology, sees standards as a way to canonize best practices. Whether canonized or not, some advocates for best practices see them as a more natural and useful approach to improving software development than are formal standards. In *Rise and Resurrection* [9], Edward Yourdon suggests:

"If the standards/methods procedures group wants to rename itself as the best practices group, then it should realize that its job is to act like social scientists visiting an alien race deep in the forest: observing behavior, documenting existing practices, and offering advice on which practices have been observed to produce good results, e.g. higher levels of quality."

In concert with evolving best practices, the more recent high-level commercial standards, such as IEEE/EIA 12207, take a process-oriented approach and expect a software development organization to define and continuously improve its software process, then customize that process for individual development projects. The Department of Defense (DoD) Single Process Initiative further encourages developers to define and utilize its own processes.

A grassroots approach to standards is observed in the development tools, platforms, and graphical user interfaces that proliferate in the marketplace. Resulting de facto standards obviously influence development. A software development organization may be a "Windows shop" or a "Unix shop" or "Linux shop."

The Influence of the Ada Experiment

The Ada experiment caused some to step back and consider the purpose of requirements and design. You do not just sit down and start coding in Ada. It takes considerable work to get to a piece of Ada code that can be compiled, so you end up thinking a lot about the design. This thinking is indispensable in large systems.

Systems developed in Ada are arguably easier to maintain than had they been developed in C or another language less verbose than Ada. Because Ada is verbose it is easier for a maintainer to read than many languages, such as C, which are designed to be easy to write rather than easy to read. Ada is designed to be easy to read.

Did Ada influence the object-oriented community? While it is hard to measure such an influence, clearly Ada was the first language that enforced rather than simply permitted encapsulation and abstraction. You can argue that Java was influenced by Ada by observing that Java is C++ syntax with Ada semantics. Ada's contributions to software engineering aside, it was not a public relations success for the standards community. The 1986 mandate that Ada be used in new defense systems left a bad taste for many software development organizations, an aftertaste that remained even when free Ada compilers, trained Ada programmers, and Ada development tools became plentiful sometime later.

Data Item Descriptions and the "D" Word

With the announcement in 1994 by Secretary of Defense William Perry on Mil Spec Reform [10], the pendulum of combined thought had reached the zenith of its swing regarding the enforcement of formal standards. To a large extent, DoD got out of the expensive business of caring for and feeding standards. As the pendulum has swung from rigid enforcement of standards to total lack of enforcement[3], Data Item Descriptions (DIDs) often disappear from the contractual setting. This leaves many developers confused about how the data should be presented. Acquirer and developer find themselves reinventing the wheel. It is a little like discarding all known languages and developing your own, even though both parties speak English fluently. Like English, the DIDs provided a basis for communication. Time need not be used reinventing nor rediscovering, if acquirer and developer agree from the start to follow the J-

The Capability Maturity Model and CMM are registered in the U.S. Patent and Trademark office to Carnegie Mellon University.

STD-016 DIDs (called Software Product Descriptions or SPDs).

Often, the misapplication of 2167A resulted in documentation — the "D" word — that took on a life of its own. Whereas the software standards evolved from the primordial goo at the fringe of the hardware disciplines, the emergence of the associated software documentation was seemingly from a lab in Dr. Frankenstein's basement. It escaped to terrorize the DoD village, consuming lots of dollars, and causing general havoc among the defense populace. In an attempt to drive a wooden stake through the documentation monster's heart, no DIDs appear in IEEE/EIA 12207[4].

The apparent lack of DIDs has been/is troublesome for acquirers and developers who were accustomed to the detailed guidance provided by DoD-STD-2167A and even by MIL-STD-498. Much like a communist society without an iron curtain, the software acquisition/development community is confused by the new-found freedom of no DIDs. The developer is directed to provide "data," the popular euphemism for the "D" word. But what is this data to look like? Developers often want an example so they do not have to start from scratch. While IEEE/EIA 12207 does not leave one to start from scratch, some developers almost feel that way. Tables 3 and 4 show samples of the level of guidance provided by IEEE/EIA 12207. But the solution is available; one can use the materials referenced by IEEE/EIA 12207.1-1997 "Table 1 - Information item matrix," which includes the J-STD-016 SPDs and other IEEE standards or one can use J-STD-016, as shown in Table 2. Further, while MIL-STD-498 was cancelled in May 1998, the DIDs from MIL-STD-498 have not been cancelled, although they may be cancelled when the full use J-STD-016 is available.

The latest thinking on documentation is to use the natural work products of the development process for today's complex systems that are likely to need continuing evolution. Rather than requiring the developer to add extra steps to the development processes to transform software development products into a prescribed standard format, the developer may generate the required information directly from the development environment — as long as the information is usable by all stakeholders during initial and continued development and sustainment. Thus, documentation may be in the form of a requirements database, Computer-Aided Software Engineering (CASE) tool data, software development folders, and other artifacts produced during software development. For these reasons, EIA/IEEE J-STD-016 and MIL-STD-498 relaxed the requirements for documentation form to concentrate on usable content, i.e. capturing engineering and planning information in the form used for project management and development activities.

Useable Replacements	Obsolete DIDS		
J-STD-016 & MIL-STD-498 22 Product Descriptions Titles & MIL-STD-498 DID#s	**DOD-STD-2167A 17 DIDs Titles & DID#s**	**DOD-STD-7935A 11 DIDs Titles & DID#s**	**DIDs Superceded by MIL-498 DIDs**
Computer Programming Manual (CPM) DI-IPSC-81447	Software Programmer's Manual DI-MCCR-80021A		MCCR-80021A
Data Base Design Description (DBDD) DI-IPSC-81437		Database Specification DI-IPSC-80692	IPSC-80692; MCCR-80305
Interface Design Description (IDD) DI-IPSC-81436	IDD DI-MCCR-80027A	interface design from Unit Specification (US) DI-IPSC-80691	MCCR-80027A
Interface Requirements Specification (IRS) DI-IPSC-81434	IRS DI-MCCR-80026A	interface requirements from US DI-IPSC-80691	MCCR-80026A, 80303
Operational Concept Description (OCD) DI-IPSC-81430	systems concept from SSDD	systems concept from Functional Description (FD) DI-IPSC-80689	IPSC-80689
Software Design Description (SDD) DI-IPSC-81435	SDD DI-MCCR-80012A	design from US, MM	MCCR-80012A, 80304, 80306; IPSC-80691
Software Development Plan (SDP) DI-IPSC-81427	SDP DI-MCCR-80030A	section 7 of FD	MCCR-80030A, 80297, 80298, 80299, 80300, 80319
Software Product Specification (SPS) DI-IPSC-81441	SPS DI-MCCR-80029A plus maintenance programmers' procedures from CRISD	Maintenance Manual (MM) DI-IPSC-80696	MCCR-80029A, 80317; IPSC-80696
Software Requirements Specification (SRS) DI-IPSC-81433	SRS DI-MCCR-80025A	requirements from Unit Specification DI-IPSC-80691	MCCR-80025A, 80301
System/Subsystem Design Description (SSDD) DI-IPSC-81432	S/Segment DD DI-CMAN-80534 (without OCD data)	design data from FD and System/Subsystem Spec (SS) DI-IPSC-80690	CMAN-80534; MCCR-80302
Software Transition Plan (STrP) DI-IPSC-81429	Computer Resources Integrated Support Document (CRISD) DI-MCCR-80024A (without maintenance programmers' procedures)	MM planning data	MCCR-80024A
System/Subsystem Specification (SSS) DI-IPSC-81431	S/Segment S DI-CMAN-80008A	requirements from FD, SS	CMAN-80008A; IPSC-80690
Software Test Description (STD) DI-IPSC-81439	STD DI-MCCR-80015A	detailed portion of Test Plan (PT) DI-IPSC-80697	MCCR-80015A, 80310
Software Test Plan (STP) DI-IPSC-81438	STP DI-MCCR-80014A	high level portion of PT	IPSC-80697; MCCR-80014A, 80307, 80308, 80309
Software Test Report (STR) DI-IPSC-81440	STR DI-MCCR-80017A	Test Analysis Report DI-IPSC-80698	MCCR-80017A, 80311; IPSC-80698
Software User Manual (SUM) DI-IPSC-81443	SUM DI-MCCR-80019A	End User Manual DI-IPSC-80694	MCCR-80019A, 80313, 80314, 80315; IPSC-80694
Software Version Description (SVD) DI-IPSC-81442	Version Description Document DI-MCCR-80013A		MCCR-80013A, 80312

Table 2. *A partial list of obsolete DIDs and usable replacements, excerpted from* Army Communications-Electronics Command Common Equivalent (ECOM) Software Engineering Center Request for Proposal Guidebook (www.sed. monmouth.army.mil/se), *Operations, Strategic Planning & Policy, RFP Guide.*

Are Current Software Standards Used?

Yes. Gary Hebert, a civilian electronics engineer at Hill AFB, says he used IEEE/EIA 12207 to understand the current thinking on documentation. He had come from a 2167A background. He found that IEEE 12207 provided flexibility allowing the use of electronic documents rather than hardcopy. Based on this direction, his organization uses Ives Development's "Team Studio Analyzer" to capture the design of a Lotus Notes database system. In this case, the tool generates a record describing the attributes for each of the Lotus Notes components, e.g. forms, views, subforms, agents, scripts or subroutines. These records are part of a documentation database. According to Nigel Cheshire, CEO for Ives Development, one of the advantages of a database over a traditional collection of documents is the ability to generate reports against the database to verify that the software design complies with an organization's standards. For example, you could generate a report to verify that each user-editable field has help text attached as per your internal corporate standard.

And no. In speaking with some Air Force personnel, I find that early this year they were using MIL-STD-498 as a reference document to recommend a series of software development documents and also to develop a concept of operations using the Operational Concept Description Data Item Description (DID). They were just barely aware of the commercial version of the standard (J-STD-016) and were unaware of IEEE/EIA 12207.

An Update on J-STD-016

While MIL-STD-498 was cancelled in May 1998, the J-STD-016 is still alive and well and is being updated. Those interested can get the trial-use J-STD-016-1995 now, and will be able to get the updated J-STD-016 when it is available. Both the trial-use version and the full-use version will provide that same reference functionality they were getting from MIL-STD-498, but with the benefit of the latest thinking.

With the DoD prohibition of putting standards on contract, the provisions in J-STD-016 for contractual use will be changed to provide for use as a basis of agreement on the activities and work products of the development process, where the agreement may take any form, from a handshake to a formal contract, and may be within an organization or between organizations. The full-use standard will be intended for project-specific application, on legacy systems and in organizations that have legacy processes (i.e., processes based on the old DoD standards), not as a basis of defining an organizational life cycle process. The expectation is that defense and commercial developers and acquirers will use international and professional standards such as IEEE/EIA 12207 as a basis for their long-term organizational process definition.

The conversion of J-STD-016 to a full-use standard was

Purpose: Describe a planned or actual function, design, performance, or process.

A description should include:

a) date of issue and status
b) scope
c) issuing organization
d) references
e) context
f) notation for description
g) body
h) summary
i) glossary
j) change history

Table 3. *Description — generic content guideline from IEEE/EIA 12207.*

Table 4. *Description — software interface design from IEEE/EIA 12207.*

Purpose: Describe the interface characteristics of one or more system, subsystem, hardware item, software item, manual operation, or other system component. May describe any number of interfaces.

IEEE/EIA 12207.0 reference 5.3.5.2, 5.3.6.2

Content: The software item interface design description should include:

a) generic description information (See Table 3)
b) external interface identification
c) software component identification
d) software unit identification
e) external-software item interface definition (e.g., source language, diagrams)
f) software item-software item interface definition (e.g. source language, diagrams)
g) software component-software component interface definition (e.g., source language, diagrams)

Characteristics: The software item interface design description should:

a) support the life cycle data characteristics from annex H of IEEE/EIA 12207.0 (see 4.2 of this guide)
b) define types of errors not specified in the software requirements and the handling of those errors

delayed in Environmental Impact Analysis (EIA) balloting and has been further delayed in combined EIA and IEEE ballot resolution due to a lack of resources to complete the resultant revisions. Hopefully, by the time this article is published, re-balloting will be in progress and the schedule for its completion will be available from the EIA and the IEEE.

EIA/IEEE J-STD-016 and MIL-STD-498 DIDs vs. DOD-STD-2167A and DOD-STD-7935A

Table 2 provides usable product descriptions from J-STD-016 and DIDs from MIL-STD-498. The DIDs from DOD-STD-2167A and DOD-STD-7935A were harmonized to produce the DIDs of MIL-STD-498. The product descriptions in the annexes of J-STD-016 are the commercial equivalents of the DIDs in MIL-STD-498, and have the same names.

For each obsolete DID associated with DOD-STD-2167A and 7935A, the table shows the corresponding usable J-STD-016 product description and MIL-STD-498 DID. The table also shows the DIDs that each MIL-STD-498 DID officially superceded; some were from DOD-STD-2167A or 7935A and some were from other MIL-STDs.

(Note that 498 has been cancelled, but as of this writing, the DIDs have not been cancelled.)

Acknowledgements

I want to thank Marilyn Ginsberg-Finner for providing her comments, the J-STD-016 updates, and Table 2. Les Dupaix, Dr. David Cook, Gary Hebert, and Alex Polack, also contributed to this article.

About the Author

Reed Sorensen has more than 20 years experience with TRW developing and maintaining software and documentation of embedded and management information systems; providing systems engineering and technical assistance to program offices, and consulting with many DoD organizations regarding their software configuration management and documentation needs. He provides configuration management, software control, interface control, and systems requirements analysis in support of intercontinental ballistic missile sustainment, modifications, and replacement programs. Sorensen has published several articles in *CrossTalk* on various software related subjects.

TRW ICBM Systems
Attn: Reed Sorensen N14GC
888 South 2000 East
Clearfield, Utah 84015-6216
Voice: 801-525-3357
Fax: 801-525-3355
E-mail: Reed.Sorensen@siinet.trw.com

References

1. Some sources of lists — http://www-library.itsi.disa.mil./, http://standards.ieee.org/catalog/index.html, http://global.ihs.com/cgi-bin/litmus_test.cgi?FRITTER=134547&NODE=PC
2. Polydys, M.L., "Operation Data Storm: Winning the Interoperability War through Data Element Standardization," *CrossTalk*, July 1999.
3. MIL-STD-483, Configuration Management Practices, 1970.
4. MIL-STD-490, Specification Practices, 1968.
5. DOD-STD-2167, Defense System Software Development, 1985.
6. J-STD-016, Software Lifecycle Processes, 1999.
7. IEEE/EIA 12207, Industry Implementation of International Standard ISO/IEC 12207:1995 (ISO/IEC) Standard for Information Technology, 1998.
8. MIL-STD-498, Software Development and Documentation, 1994.
9. Yourdon, Edward, *Rise and Resurrection of the American Programmer*, Yourdon Press Computing Series, page 128.
10. Perry,William J., Secretary of Defense, DoD Policy on the Future of MILSPEC, *CrossTalk* September 1994.

Notes

1. This came when the advent of written language by the press made it available on a scale of a magnitude greater than handwriting provided.
2. A series of international standards on quality.
3. Alex Polack notes that from the mid-1980s to the mid-'90s, there was a tendency to use standards as a club to beat up or monitor developers. Marilyn Ginsberg-Finner notes that within current DoD acquisition reform policy, standards are primarily guidance to developers in defining their software process and standards are still essential as a basis for acquirers to evaluate the processes, activities, and work products that an offeror proposes and provides.
4. IEEE/EIA 12207.1-1997 "Table 1 - Information item matrix" references standards, including J-STD-016 that have DID-like guidance.