The DII COE: An Enterprise Framework

Dr. Gregory Frazier
SAIC

The Defense Information Infrastructure (DII) Common Operating Environment (COE) is a framework for the creation of a set of cooperating computing enterprises. Its goals include the elimination of "stove-pipe" systems and cost reduction via software reuse, reduced need for system administration, and simplified system integration. This article describes in brief the salient features of the DII COE. It describes some of the challenges related to migrating systems to a DII-compliant environment, and concludes by arguing that the key to the successful use of the DII COE is for all participants to be aware of and work toward building and maintaining an extensible, general-purpose environment.

In 1994, the Defense Information Systems Agency (DISA) began development of the Global Command and Control System (GCCS). The goal of GCCS was to link the commander-in-chief (CINC) sites into a single planning and logistics network. It utilized an architecture that was developed in the Navy Joint Maritime Command Information System (JMCIS) program for integrating software modules developed by multiple contractors. In 1995, GCCS was operational and DISA had begun work on the Defense Information Infrastructure (DII). In 1998, GCCS version 2.0 was the first Department of Defense (DoD) system to utilize the DII Common Operating Environment (COE) as the basis for its implementation, being deployed on top of the DII COE Version 3.1.

Since then, the DII COE has been or is being used as the framework for more than 100 DoD computing systems (see Figure 1). The term "COE" has moved into common parlance for corporate Chief Information Officers, and foreign governments are looking to construct common operating environments for their computing enterprises. Although the DII COE has been in existence for more than three years and has enjoyed remarkable success, it is not widely understood. This article offers insight into the DII COE architecture and the rational for its design.

An Enterprise Focus

The DII COE is a framework for the development of enterprise computing systems. In order to understand the decisions made regarding its design, it is important to first understand what an enterprise computing system is, and what it means to be a framework.

An enterprise is a venture – the act of an organization working toward a common goal. Enterprise computing is the establishment and use of a computing system that supports this venture by implementing the business processes of an organization. The DoD is a grand-scale organization with approximately 11 million military and civilian participants. Its purpose is equally grand: the defense of the United States of America. This is, of course, too large an organization and too complex a venture to be understood as a single enterprise. Thus the DoD is partitioned into many smaller organizations, each executing their own enterprise. However, these are collaborative enterprises – the organizations work together to support common goals. The DoD enterprise computing system is a system of systems in which information and services are shared across enterprise boundaries.

A framework is a software package that supports the creation of architecture by guiding developers toward a particular set of design patterns. If the developers conform to the framework and utilize its services, then the resulting system will reflect the desired architecture. It is important to note that a framework is not itself a system. The DII COE is a framework for the construction of modular, scalable, distributed Command, Control, Computer, Communications, Intelligence, Surveillance and Reconnaissance (C4ISR) computer systems. It is a collection of tools for the creation of these systems; it is a set of software modules that can be (re-)used to construct these systems. Or, to quote the DII COE Integration and Runtime Specification [1]:

"The DII COE emphasizes both software reuse and data reuse, and interoperability for both data and software. But its principles are more far-reaching and innovative. The COE concept encompasses:
- An architecture and approach for building interoperable systems.
- A minimal but extensible security architecture and a set of security services.
- An environment for sharing data between applications and systems.
- An infrastructure for supporting mission-area applications.
- A rigorous definition of the runtime execution environment.
- A reference implementation on which systems can be built.
- A collection of reusable software components and data.
- A rigorous set of requirements for achieving DII compliance.

Figure 1: *Military Systems Operational or Being Developed Using the DII COE*[1]

Army		Navy		Air Force	Marine Corps
AALPS	DTSS	AADC	NSPF	AFWeather	AFDI
ACGS	FATDS	CADRT	NSS	AMC BDM	ARTDF
AMBISS	FAAD C21	COMDAC	NSSN	AMS	GALE
AMDWS	FIRESTORM	CCS	PTW	AWACS	GCCS
AMDPCS	GCCS-A	CUB	REDS	A2IPB	GCCS-I3
AMPS	GCSS-A	CV/TSC	SFMPL	CSEL	GCSS
ANGSC-52	IBDAS	GCCS-M	SH60 MPS	DCAPES	JCACTD
ASAS	IMETS	IUSS	SRMT	Defense IEMATS	JCALS
ASD	ISYSCON	JMPS UPCs	SCCS	FORTE	JDISS
ATCS	LW	KSQ-1	TACLOGS	GBS	JDP
ATLAS	MCCCC	LAMPS	TAU	GCCS-AF AETC	JMCSID
BCTP	MCS	MEDAL	TEAMS	GCSS-AF IF	JMPS
BMC3	MFCS	METOC	TERPES	IMDS	JSCGS
BSM	PEGEM	MPA	TCAC	IMOM	JTAT
CINC CSA	RCAS	MPAS	TDSS	ISC2S	Joint Tactical Term
CNCMS	SAS	MSBL	TTWCS	MAMS	Joint Targeting Tool
CNPS	TAIS	NAVSSI	VTC	Rosetta	JWARN
CR/HMS	TCAIMS	NFCS		SBMCS	MEPED
CSCE	THAADBMC3I			STRATCAT	MIDB
CSSCS	TPSOPS			TBMCS	TNP
CTIS	TSIU				JMNS
C4IJM	UAV				
DCARS	WARSIM				

- An automated tool set for enforcing COE principles and measuring DII compliance.
- An automated process for softtware integration.
- An approach and methodology for software and data re-use.
- A set of application program interfaces (APIs) for accessing COE components."

Fighting Stovepipe Systems

A key driver for the DII COE is the fact that the DoD is composed of multiple cooperating enterprises. While each of these enterprises is dedicated to accomplishing a specific goal, the data products of one enterprise are consumed by other enterprises as they work together to accomplish the overall goal of defense. However, most computing systems built for the DoD are [historically] stovepipe systems; systems that operate in total isolation from the rest of the computing environment. This is in contrast to a *system of systems* model, where data flows seamlessly from one business process to another.

Stovepipe systems do not support abstract goals such as extensibility or interoperability that are central to the DoD's ability to field cooperating enterprises. A computing architecture that is intended to unify the DoD must be capable of adopting the configuration appropriate for a particular enterprise's mission while maintaining a commonality that will allow that enterprise to interoperate with other DoD computing enterprises.

The DII COE is, first and foremost, a framework intended to prevent the creation of stovepipe systems and promote cooperative enterprises.

Other DII COE Goals

While the principal goal of the DII COE is to eliminate stovepipe systems, there are a number of important drivers for the design of the DII COE architecture:

- Increased software reuse. While each enterprise within the DoD has a unique mission and a computing system to support that mission, there is inevitably some functionality that the enterprise will have in common with others.
- Reduced need for computer system administrators. The DoD personnel using computer systems should be warfighters whose mission is supported by the computing system, not system administrators whose mission is to support the computing system.
- Improved technology insertion. The enterprises that comprise the DoD are long-lived enterprises. Their life span dwarfs the technology cycle, and thus there must be an avenue for technology insertion in DII COE's computing architecture.
- Simplified system integration. Given the distributed development environment in which multiple contractors contribute modules to one or more integrators, and each integrator delivers the resulting system to multiple organizations to field, it becomes important to push as much integration responsibility as possible to the developers and the deployers.

The following section will describe the DII COE and discuss how it achieves these goals.

The DII COE Framework

The first DII concept dealt with by a system integrator, an administrator, a developer, or even as a user is the segment. A segment is a unit of software or data that has been packaged such that it can be installed on a DII-compliant computer using the software installation tool of the DII COE. All software that is to be installed on a DII computer must first be put into segment format.

What is typically done for government off-the-shelf software (GOTS) is that software developers segment it before delivery to a program engineering office. For commercial off-the-shelf (COTS) software, the program office typically purchases the product and then hires contractors to segment it. However, the model envisioned for the DII COE was for commercial vendors to put their software into segment format.

Either way, an engineering office receives software in segment format, accompanied by a set of documents that include installation procedures, a users manual, test plans and test results, version description, and specification of the segment's compliance level. The compliance level is the degree with which the segmented software is in compliance with the DII Integration and Runtime Specification (I&RTS – discussed in the next section). The engineering office tests the segment, and then makes that segment part of its build list. It is now available to be delivered to the field, either as part of a standard build or as an optional segment.

Segmentation is controversial for several reasons. First, it is an additional cost. Program offices dislike having to spend the money to have software segmented on top of the cost of developing or purchasing said software. This is particularly resented with COTS software, which generally arrives in an installation package. Second, there are not extensive tutorials on segmentation, and the software used to support segmentation can be a bit cranky. The learning curve can be very steep for newcomers to segmentation. There are, however, a number of reasons to utilize a single software-packaging standard for the DII:

- Segments facilitate configuration management. All software in segment format shares a standard versioning system, a standard naming system, and has a standard set of documents associated with it. With a single program (the COEInstaller) you can see which segments are installed on a given machine, what resources a segment requires, and what other segments a given segment depends upon.
- The same installation software is used to install every segment on every computer. System administrators are not forced to deal with the vagaries of how different developers decide to have their software installed.
- Segments facilitate software reuse. Segmentation forces the developer to prepare the software for integration. Once in segment format, a software package can easily be transferred between program offices and used in a variety of contexts.

When developing a segment, the most important fact to keep in mind is that a segment should be of general utility. It is possible to segment software in such a way that it is useful only for a very specific purpose. This defeats the reuse goal of segmentation and ensures that the same software will have to be re-segmented. A better approach is to segment software in as general a manner as possible, and then provide a separate configuration segment that modifies the installation for a specific need. This is the approach taken for infrastructure services in the COE such as the iPlanet Enterprise Server and the Oracle Database Management System (DBMS).

The Common Operating Environment

The common operating environment is just that; an operating environment that is common across computing systems and problem domains. There is a common misconception that all DII segments are part of the COE, or that by segmenting an application one is making it part of the COE. This is not the case. Segmenting software makes it available to the DII, but not all software is common. The I&RTS refers to segments that are not part of the

COE as mission applications.

Mission applications are software packages that are intended to provide functionality that corresponds to a specific problem. While it is important for mission applications to comply with the I&RTS and conform to the conventions of the DII COE, these applications will not be widely used and thus not part of the COE.

The software that comprises the DII COE is placed into three categories: the kernel, the common support applications, and the infrastructure services. The kernel is that portion of the DII COE that is installed upon every computer. It is the bootstrap portion of the COE, containing such functionality as the segment installer, the file permission and disk partition configurations, etc. The common support applications are desktop applications that are deemed to be of general use: a word processor, an email client, a Web client, and others. The infrastructure services are services that are deemed to be of general use: various DBMS products, a Web server, a directory server, and others.

Every computer that is to be a *DII computer* must have the kernel installed upon it. The rest of the COE – the segments that comprise the common support applications and the infrastructure services – is available for installation, but is not required to be present on every machine. What is required is that they be available to any system. In other words, a segment developer or system designer can make the assumption that the segments that are in the COE are available for use and can incorporate those segments into his/her design without knowing anything else about the deployment environment.

The Integration and Runtime Specification

The COE and segmentation are specified in the I&RTS; it describes how a computer is configured such that it is DII compliant, and how to build software segments that are DII compliant. It describes the various types of segments, including software segments, COTS segments, data segments, database segments, Web segments and others. In its own words:

> "This document is an engineering specification that describes how modules must interact in the target system. System architects and software developers retain freedom in building the system, but runtime environmental conflicts and data conflicts are identified and resolved through automated tools that enforce COE principles [1]."

It is required reading for anybody who aspires to build, test, or integrate segments, or anybody who wishes simply to understand the DII COE. It is available at <http://dod-ead.mont.disa.mil/cm/geneal.html>.

Compliance Levels

The commonality of the COE rests on the fact that mission applications use the COE to provide common functionality. If segments provide their own implementations of the COE functionality, then the benefits described above are not observed: no cost savings through reuse, no cost savings due to reduced system maintenance, no facilitation of technology insertion, and no interoperability enabled via shared infrastructure services. Instead, we would see a retreat to stovepipe systems delivered as segments.

The purpose of compliance levels is to provide a metric of how well a segment makes use of the COE. This provides a quantitative measure both of how well a segment is integrated with the COE and the progress that a segment makes over successive deliveries. The I&RTS specifies eight levels of compliance. At lower levels, we are dealing with the basic structure of segmented software:

Level 3

Platform Compliance: The application is well behaved in the platform context. It runs on a version of the standard operating system that is supported by the DII. It does not utilize hard-coded port assignments. It does not bypass the standard GUI APIs for the platform. It can safely execute alongside DII-compliant applications, etc.

Level 4

Bootstrap Compliance: The application is packaged in segment format (i.e. it can be installed and deinstalled using the COEInstaller).

Level 5

Minimal DII Compliance: The segment conforms to the I&RTS to the extent that it does not represent a security risk and does not negatively impact system configuration when installed.

In levels 6-8, the compliance levels measure how well the segment makes use of the COE. Appendix B of the I&RTS provides questionnaires that allow developers to evaluate the compliance level of their segments.

DII Adoption Challenges

In this section, barriers to DII adoption are discussed. These barriers are organized in three categories: cost, platform availability and knowledge.

Cost

One of the major obstacles to adoption of the DII is the expense of migrating a given system to use of the DII COE and/or achieving a given compliance level. There are two types of systems that experience significant cost penalties when pursuing DII integration: legacy and leading edge.

Legacy systems experience high costs due to the need to reengineer portions of the system to achieve a given level of DII compliance. This cost can be prohibitive, particularly if it means replacing a constituent product (e.g., replacing the database). When dealing with legacy systems, it is important to understand when to enforce the DII compliance directives and when to relax them. These directives do not make sense for many legacy systems, particularly those that are at the middle or toward the end of their life cycles and will never observe the maintenance cost reductions potentially available from the COE.

Leading-edge systems experience high integration costs due to the lack of support provided by the COE. The DII realizes cost savings via the reuse that segmentation promotes. A system that is utilizing a leading-edge technology may be ahead of the COE and will bear the brunt of the design and segmentation costs without receiving the benefits of reuse. For a small program that anticipated obtaining the majority of its functionality from a COTS product, the relative cost of segmenting that product can be prohibitive.

Worse, a leading-edge system that makes use of a not-yet-adopted technology runs the risk that the DII COE will at some point incorporate the technology but in a way that is incompatible with how the system used it. This risk can be alleviated by active participation in the DII COE Technical Working Group (TWG) process[2] – but the program must then dedicate manpower to track or participate in the TWG and [possibly] experience schedule delays awaiting clear direction from the TWG and/or Architecture Oversight Group (AOG). Also, program management offices that are new to DII may be unaware of the TWGs' existence or not know how to contact their service or agency AOG representative[3]. Leading-edge programs should be blazing the trail, showing DISA how (or how not) to integrate new technologies. Their participation in the DII COE TWGs must be encouraged and supported, in order to reduce costs both for those programs and for the DII COE.

Platform Availability

The DII COE (Version 4.x) currently supports three platforms: Microsoft Windows, Sun Solaris running on Sun workstations, and Hewlett-Packard HP-UX. All other platforms are unavailable if one is required to be DII compliant. There is a Kernel Platform Compliance (KPC) Program[4], but only two computing systems have successfully gone through the KPC, and both did so on version 3.3 P1 of the DII COE kernel[5]. Since the version 4.0 kernel had an almost entirely different kernel code baseline from the 3.x version, neither platform has been recertified for the current version of the kernel.

Even among the three core COE platforms, the kernel is not the same. The paucity of platforms supporting the DII COE and the variance among the DII COE platforms are the result of the kernel being specified by its source code. To quote the Defense Information Infrastructure (DII) Common Operating Environment (COE) Kernel Platform Compliance (KPC) Program Document for DII COE Kernel Version 4.200 [2], the KPC Program certifies that a platform "executes the Government Supplied Kernel Source code with the same behavior as the current 'Reference Platforms'." In other words, a KPC-certified computer is one that runs the kernel source code that it receives from the DII COE Engineering Office. This has three ramifications.

First, for platforms that cannot share the code base (e.g. Windows NT and Unix/Motif), the kernels differ in unspecified ways. Second, for platforms that desire to support DII, their only recourse is to port each version of the kernel to their platform, which is logistically untenable. Third, there is no mechanism for non-traditional platforms (non-uniform multi-computers, for example) to achieve DII certification via the KPC Program. While the KPC Program does include a test suite to evaluate the kernel port, this test suite does not comprise a specification of the kernel. It is the reference implementation source code that acts as the specification[6], so any platform that cannot compile that source code cannot be in the DII.

Getting the Word Out

The final and most significant impediment to DII COE adoption is the general lack of knowledge regarding what the DII is, what the COE is, and how a program can leverage the DII to facilitate the development of their system. The cost and platform support issues discussed in the previous sections can be addressed by the thoughtful requesting and granting of waivers to allow systems to conform to the intent of the I&RTS without being hamstrung by its rules. To do this, however, requires that both the program management office and the contractors executing the contract have a thorough understanding of the DII.

While there is a great deal of data available regarding DII and the DII COE, it can be challenging for newcomers to discover the information that they need. The principal source of information is the I&RTS. This is required reading for anybody who intends to develop an application for use in DII, integrate a DII system, manage a DII-related project, or deploy a DII system. However, the I&RTS is not a how-to manual or a tutorial – it is a specification of the DII runtime. The DISA and DII COE Web sites also provide a great deal of information. It is incumbent upon the DII participant to find the information that they require from these sources.

Conclusion

The ongoing success of the DII and its COE depends upon a number of factors. First, the DII must maintain its relevance. This means continuing to support technology insertion efforts. It also means that the organizations that are deploying DII-compliant computing systems must understand not only the letter of the I&RTS, but also its intent. They must execute their system development such that the result fits with the spirit of the DII. This can be captured in words such as *extensible*, *open*, and *reusable*. Wherever possible, avoid one-of-kind solutions.

If a program requires a common service established with a particular configuration, deliver two segments: one that contains the common service, and a separate segment that configures that service. Do not view the I&RTS segment compliance criteria as roadblocks, but rather as guidelines. And never think in terms of a final delivery. ◆

References

1. Defense Information Infrastructure (DII) Common Operating Environment (COE) Integration and Runtime Specification (I&RTS), Version 4.1, 3 Oct. 2000,<http://dod-ead.mont.disa.mil/cm/general.html>.
2. Defense Information Infrastructure (DII) Common Operating Environment (COE) Kernel Platform Compliance (KPC) Program Document for DII COE Kernel Version 4.2, p. 4, 31 May 2001, Version 1.1 (draft),<http://diicoe.disa.mil/coe/kpc/kpc_program_doc.doc>.

Notes

1. Information provided by the DII COE Engineering Office.
2. See <http://diicoe.disa.mil/coe/aog_twg /aog_twg_page.html> for a listing of TWGs and their home pages.
3. See <http://diicoe.disa.mil/coe/aog_twg/aog/members.htm> for a listing of AOG members.
4. See <http://diicoe.disa.mil/coe/kpc/KernelPlatformProgram.htm>.
5. The Compaq certificate is at <http://diicoe.disa.mil/coe/kpc/Compaq/19990831certNoSig.png> and the SGI certificate is at <http://diicoe.disa.mil/coe/kpc/sgi_cert.jpg>.
6. There is the dilemma that if any of the kernel source code was modified in order to get it to compile on the platform, then the specification has been modified and the platform is no longer running the government supplied source code.

About the Author

Gregory Frazier, Ph.D., has worked at Science Applications International Corporation (SAIC) since December of 1984. He was the integrator for Internet applications (the Web, newsgroups, Internet relay chat) for the Global Command and Control System and spent a year as the chief engineer as the DII Integration Contract for SAIC. For the last several years he has focused on research of enterprise computing for SAIC and is currently the software architect for the Joint Network Management System. He received a bachelor's degree in computer science and engineering from Massachusetts Institute of Technology and a doctorate in computer science from the University of California at Los Angeles.

8301 Greensboro
Mail Stop E-2-5
McLean, VA 22102
Phone: (703) 676-6459
Fax: (703) 676-7123
E-mail: gregory.frazier@saic.com