Paving the Bare Spots

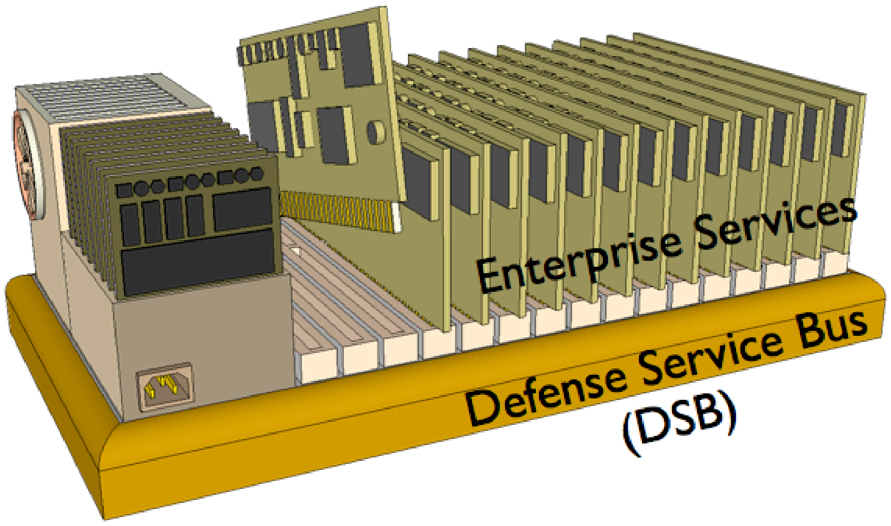
Building the Enterprise-wide Defense Service Bus

A White Paper by Brad J. Cox, Ph.D.
Binary Consulting; Bethesda MD
bcox@binary-consulting.com

*T*his paper describes how Department of Defense (DOD) CIOs and policy groups responsible for net-centricity, interoperability, and transformation can facilitate the creation of a service bus that works for the whole enterprise instead of just within project stovepipes. Modeled after standards development bodies like OASIS and open source development groups like The Apache Foundation, the approach is broadly applicable.

What is the Defense Service Bus (DSB)?

A service oriented architecture consists of more than just services and applications. These are only the bricks of a SOA architecture. They are embedded within and supported by mortar; the infrastructure that supports interactions between services. We call this mortar the Defense Service Bus (DSB)[1].



DOD has championed standard interfaces since early in the industrial revolution[2]. A stringent acquisition process now requires every project to report the standards it uses in its DODAF TV-1 table which are checked against the list of acceptable (mandated) standards in the DISR Online repository. The Net-Centric Operations and Warfare Reference Model (NCOW RM) describes the activities,

[1] I coined this term to avoid the usual term, Enterprise Service Bus (ESB). This is to avoid distracting debates over what a "true" ESB is. For example, whether or not the DSB supports non-transport features such as mediation and orchestration is up to the DSB Foundation, not ESB marketeers.

[2] Planning the Software Industrial Revolution; Brad J. Cox; IEEE Software Magazine; Software Technologies of the 1990's. http://virtualschool.edu/cox/pub/PSIR/

services, technologies, and concepts that enable a DOD-wide net-centric enterprise information environment. Compliance with the NCOW RM is one of the Net-Ready Key Performance Parameters (NR-KPP), which describes net-ready attributes for the exchange of information and the end-to-end operational effectiveness of that exchange. The NR-KPP incorporates net-centric concepts for achieving Information Technology (IT) and National Security Systems (NSS) interoperability and supportability. These resources help program managers, testers, and milestone decision authorities in assessing and evaluating IT and NSS interoperability.

Yet in spite of the emphasis on standards, interoperability remains the exception more than the rule. The Army's Future Combat System (FCS) project is based on its own bus, the System of Systems Common Operating Environment (SOSCOE). Net-centric Enterprise Systems (NCES) is based on another bus, the Service Oriented Architecture Foundation (SOAF). Until recently[3], there was little coordination between these projects on crucial interoperability questions such as how message-level security should be handled. The interoperability plan was to use adapters, mediators and gateways to bridge the two systems. But Rube Goldberg showed how reliability and performance can suffer if adapters replace intentional design. Financial and other resources are wasted on adapters that would have been better spent on functionality.

But what else can projects do when they have unique requirements that cannot be meet with standard solutions. Policy only imposes more requirements without providing solutions for meeting them. Without a central enterprise space in which projects can collaborate on enterprise-wide infrastructural issues like security and interoperability, projects can only address their own needs while leaving interoperability as somebody else's problem.

[3] Cooperation improved with the signing of a memorandum of agreement in early 2006. Initial interoperability meetings are scheduled to begin this summer.

Paving the Bare Spots

When my college wanted to stop students from taking shortcuts to class that resulted in bare spots in the lawn, it tried two entirely different approaches. The first was erecting “Keep off the Grass” signs and rope barriers and then punishing those who violated them. When that failed, they tried the “pave the bare spots” approach. They delayed building sidewalks until bare spots appeared and paved sidewalks over them. This solved the problem permanently and painlessly since the sidewalks were now exactly where students needed to go... the carrot instead of the stick.

The DSB is the SOA sidewalk; the path that enterprise traffic traverses. Like “Keep off the grass” signs, interoperability policy requirements are followed when convenient and otherwise ignored. Paving the bare spots goes further than just imposing requirements. It also provides a solution; a reference implementation that projects can pick up and use. This eliminates any incentive to circumvent the requirements since a fully compliant solution is available at no cost. But being a reference implementation, projects with special needs are free to build their own if they must. In that case, they bear the burden of demonstrating that their solution complies with interoperability requirements.

The trick, of course, is defining a standard that meets most project’s requirements. How this can be done is the subject of this paper.

What is the DSB “Foundation”?

The approach to be described here is modeled after the governance models of the Organization for the Advancement of Structured Information Standards (OASIS[4]) and The Apache Foundation (ASF[5]). To avoid premature specificity as to who might play this role within DOD, the paper will call it the Defense Service Bus Foundation (DSBF).

[4] OASIS Policies and Procedures; http://www.oasis-open.org/who/policies_procedures.php

[5] The Apache Foundation; http://www.apache.org/

The foundation is responsible for defining, developing and releasing the internal DSB standard and its reference implementation as a centrally managed, policy- and standards-compliant whole. The foundation manages a web site as the access point for internal and external communications and decides key policy issues such as who may participate and under what conditions. Typically, but not necessarily, it is the legal owner of property contributed by its members.

Although this is no simple task, it is simpler than building infrastructures in independent projects that must interoperate when they are combined. By managing the DSB within enterprise space instead of within each project's space, the DSB can be upgraded and new versions released via conventional binary release procedures with minimal impact on each project that uses it.

None of this implies that DOD should "get into the business of building software". Off the shelf solutions should be used when they are suitable to DOD requirements. The proposed approach should only be used to fill gaps between DOD needs and what off-the-shelf solutions can provide.

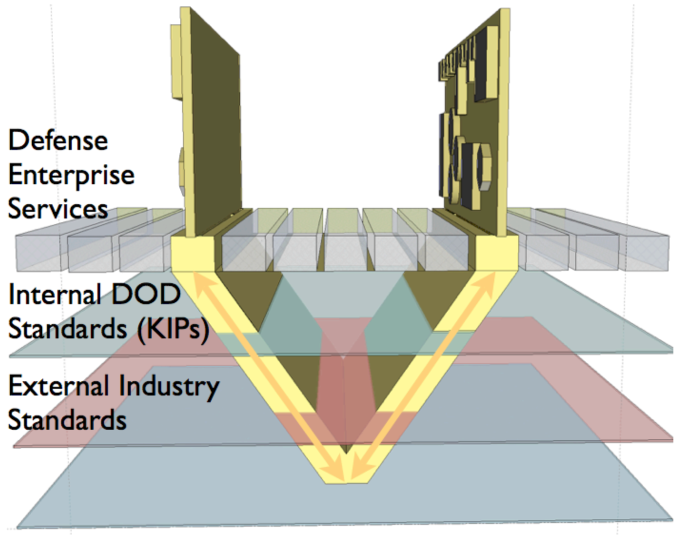
Nearly all of the lower SOA support layers are already available as off the shelf solutions. These provide the substrate for adding whatever is missing. For example, synchronous messaging is supported by Apache's Axis project (among others) and a standard API for asynchronous messaging is defined by Sun's JMS specification. By adopting these (or similar) as the foundation layers, DOD can focus on bridging remaining gaps. Some examples of gaps are, while Axis provides hooks (message handlers) that can support almost any security standard, it provides no specific message-level security model suitable to DOD. Similarly JMS does not define a wire standard for asynchronous messages, so applications based on different vendor implementations will not interoperate. And there is no off the shelf solution for enclaves; sites with intermittent or degraded connections to the rest of the enterprise.

None of this means that everyone agree on a single protocol capable of handling all interactions between all DOD systems. "The" DSB only means that an internal standard has been adopted for the enterprise that all parties accept as the standard. To paraphrase Einstein, this standard must be as simple as possible, but no simpler. If the consensus is that different communication pathways are needed between parts of the DOD System of Systems, then the DSB must provide the required pathways. The foundation only provides an enterprise-wide space within which parties collaborate on finding "simple as possible but not simpler" solutions for the enterprise as a whole.

This is not primarily a technical problem. The hard part is arriving at a consensus on a sound approach that meets legitimate project requirements. We'll return to the hard part once I describe the DSB as a technical artifact.

Technical Characteristics of the DSB

Service oriented architecture is the latest in a long line of integration technologies. Each adds a new level of integration to earlier layers such as high-level languages, structured programming, object oriented programming, client-server, etc. The new layer is important because it is the first widely adopted layer that supports enterprise-wide integration; the ability to deploy services enterprise-wide.


Defense Enterprise Services
Internal DOD Standards (KIPs)
External Industry Standards

From a service developer's viewpoint, the DSB is the interface between their application and everything else in the enterprise. From this external perspective, the DSB is the largest and most complex part of any SOA, more so than any service. But internally, the DSB is just the topmost of a stack of standards-compliant layers that work together to support interactions between services. The lowest layer (transport) contains the GIG and the hardware and operating systems that the DSB supports. Middle layers support the operations that services use as the message moves through the sending and receiving stacks to its destination.

The integration technologies of the past haven't been discarded. They remain available for when they are needed. That is, the DSB does not just support SOA. Only the highest layers do that. The lower layers are still accessible, as always governed by policy restrictions on their use. Internally, the DSB is a layered collection of libraries, operating system calls, and hardware. The lower layers are accessible for applications to use if they must in order to meet throughput or other requirements.

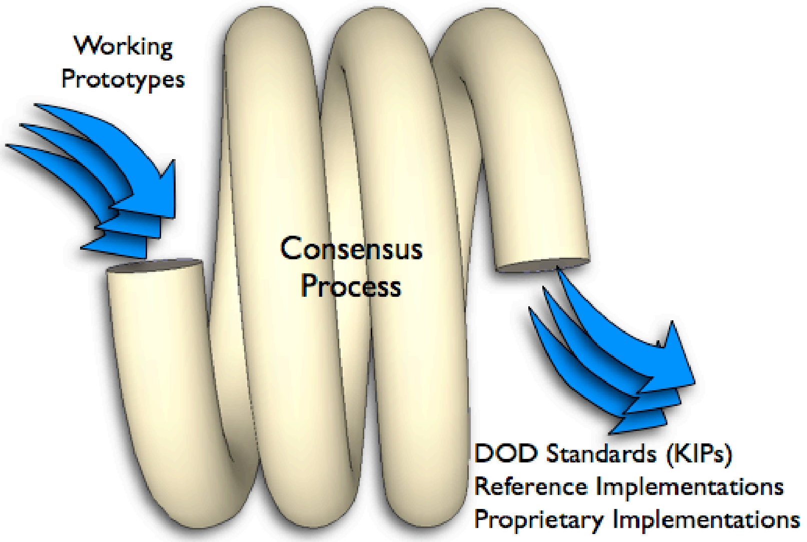
In practice, the Foundation decides where to draw the line between DSB- and service-provided functionality. That said, the top layer is unlikely to be standard SOAP. DOD security policies require *secure* messaging (among other things), and this is not supported by DISR-mandated standards. SOAP is nonetheless available in lower layers for when less than secure messaging can be used. Defining an enterprise-wide standard for message-level security, and building a reference implementation of that standard, is the obvious place for the foundation to begin. NCES's SOAF and FCS's SOSCOE are two sources of prototype implementations. A process for combining working prototypes to arrive at an internal consensus standard is described in the next section.

The DSB implements the interface between enterprise-wide applications so it must be strictly governed by standards. Standards govern interfaces, not how clients (services) work internally. By encapsulating standards withiin a concrete reference implementation, projects don't have to understand, interpret and implement the standard. This is a considerable simplification because standards documents are often voluminous, easily misinterpreted, and maddeningly vague. Clients must only understand how to use the DSB to interact with the service they're calling, not the stack of standards upon which the DSB is based.

The DSB is just a reference implementation. This means projects can use it if it meets their requirements but can develop their own if it doesn't. In that case, they bear the burden of complying with DOD interoperability (and other) requirements. The reference implementation is distributed as source code to serve as an executable demonstration of one way of meeting the standard's intent.

How the DSB "Foundation" Works

This paper has described the DSB's technical characteristics and the advantages of managing infrastructures by the enterprise instead by each project. That was the easy part. We turn now to the essential problem; how to get from a bus that works within stovepipes to one that works enterprise-wise.



Obviously, this is a issue that requires consensus and this can be a problem unto itself. Nonetheless, standards organizations and open source development groups have developed cooperative models that have demonstrated sufficient success to warrant consideration within DOD. Descriptions of these models, and comparisons between them, are available in the footnotes[6] so this paper concentrates on how the model applies within DOD.

Policy groups have political power but lack the technical expertise and local knowledge of project requirements to just impose their will unilaterally. A consensus process is needed mobilize the local knowledge that is distributed between the programs, their contractors, and industry as a whole. A similar distribution of power exists between Object Management Group (OMG), OASIS, or Liberty Alliance and member companies such as Sun, IBM, Microsoft, etc. Standards organizations are therefore a reasonable model for how internal standards and reference implementations could be negotiated within DOD.

Although the proposed governance process is similar to those of standards bodies, there are several differences. The biggest difference is that the proposed DOD-internal process operates downstream of and separately from external standards processes. Its scope is internal, influencing how external standards are applied within DOD rather than influencing industry as a whole. In that respect, the process is similar to the existing Key Interface Profile (KIP) process, which also specifies how existing external standards apply at critical (key) internal interfaces. It differs in that the KIP process only produces KIP documents. It does not produce reference implementations of the KIPs; actual reference implementations that projects can pick up and use, assured that the implementation complies with internal and external standards.

In The Rise and Fall or CORBA[7], Mitchi Henning uses OMG's experiences with the CORBA Component Model (CCM) to emphasize the importance of reference implementations for holding complexity in check. OMG focuses on standards, leaving implementations up to its members. Users vote to issue RFPs for specifications, members submit draft specifications in response, and the members vote on which draft to accept as the standard. Since working reference implementations are not part of the process, design by Powerpoint can sometimes dominate sound technical work. Complexity grows if changes are accepted that can't be efficiently implemented. And if there is no free reference implementation, users must buy them from OMG members. Henning claims that these issues caused CCM to be displaced by EJB (Enterprise Java Beans) and then SOA.

In learning from to this example, there should be no separation between defining a standard and building its reference implementation. In fact, the standard emerges late in the process. The process begins when groups develop solutions

[6] Web Services and Service-Oriented Architectures; http://www.service-architecture.com/

[7] The Rise and Fall of CORBA; Michi Henning; ACM Queue vol. 4, no. 5 - June 2006; http://www.acmqueue.com/modules.php?name=Content&pa=showpage&pid=396

to needs that only they perceive, typically without knowledge of other groups' activities or needs. Interest in turning these into standards emerges as overlapping solutions are discovered. At this point a foundation emerges to consolidate them. The process proceeds as the contributing members critique partial solutions to decide on a common one, not primarily through top down abstract design. A reference implementation is developed to serve as a concrete harness for testing the abstract ideas expressed in the standard. Often commercial implementations develop simultaneously within commercial members' home projects.

The open source community uses a number of tools to support such work. The approach described here is based on the Apache Software Foundation (ASF)[8]. The ASF was founded to support the Apache web server but expanded its scope to include SOA infrastructure (Xerces for XML, Axis for SOAP, etc) and other less recognizable but important efforts. The following summary shows how DOD could use a similar model for those unfamiliar with open source development tools and procedures.

- **Budget**: The budget can be small even in absolute terms but certainly compared to today's approach. Foundation management is drawn from existing management already responsible for interoperability within DOD. Technical staff is drawn from those already working on infrastructural components within project stovepipes. Instead of confining infrastructural work to their home project, they publish it to the foundation's change management system. Several such systems are available such as Subversion[9]. The consensus process occurs within the change management tool as project members decide how to merge differences into the next release, supplemented as required by tools such as chat and email. Face-to-face meetings are seldom, if ever, required.
- **Membership Classes**: The foundation's board decides such membership issues as who can contribute to and read from the repository and how to balance the interests of policy-makers, project managers, consultants, commercial partners and technical contributors ("committers"). Most open source bodies recommend the notion of "meritocracy" which concentrates power in the hands of contributers with the technical knowledge to make far-reaching technical decisions. Change requests are submitted by the members who lack committer privileges via web-based tools designed for this purpose, such as Bugzilla.
- **Requirement/Bug Reporting:** DSB users submit bug reports and feature requests via any of several web based systems. These are automatically distributed to the vol-

[8] How the Apache Software Foundation works; http://www.apache.org/foundation/how-it-works.html

[9] The goal of the Subversion project is to build a version control system that is a compelling replacement for CVS in the open source community; http://subversion.tigris.org.

unteer committer community via mail or web based bug- and requirements-handling systems.

- **Change Process**: Changes are submitted electronically by anyone with commit privileges via the change management system. Privileges are typically granted by the committer community based on applicant's prior contributions. Such changes don't impact the current release, but are maintained separately in the submitters' "branch". These may be accepted into the release branch, the "trunk", during a merge process during which all changes are scrutinized and critiqued by the community as a whole. Accepted changes are merged with other changes, compiled, tested, and released as the next DSB release. All steps are digitally mediated, fully automated, and incur little cost.

Where do we go from here?

This has summarized the advantages of distributing and managing a reference implementation of the DOD-internal enterprise infrastructure standard. It described cost-effective ways to define a consensus-based internal standard, and reference implementations of the same, under the leadership of a foundation that currently does not exist. The approach is modeled after the governance structures of standards bodies such as OASIS and open source development groups such as The Apache Foundation.

Notice that the proposed approach is considerably different from the memorandum of agreement followed by meetings that the FCS and NCES projects seem to have embarked on. The crucial distinction is the creation of a new space, managed by the foundation, within which projects collaborate on infrastructural issues affecting the enterprise as a whole. A few unmistakable signs of the recommended approach in action are:

- A foundation is defined to manage enterprise space and to accept long-term responsibility for its contents. The contents are originally incompatible prototypes contributed by the members, but evolve through a consensus-building process that culminate in the definition of the internal DOD standard and reference implementation we've called the DSB.
- The foundation installs a change management system (and supporting tools) to support cross-project collaboration within enterprise space.
- Projects contribute internally developed infrastructures (SOSCOE and SOAF, for example) to enterprise space and remove them from the originating project's space.
- Personnel previously assigned to developing these infrastructures in project space are encouraged by their management to work on them within enterprise space in collaboration with other projects.

In other words, a formal distinction between enterprise space and project space is the crucial sign of the proposed approach in action. Space simply means a change management system and supporting tools that is managed by the enterprise rather than by specific projects. Enterprise space is owned and operated by the DSB foundation to represent the interests of the enterprise as a whole.

This paper has concentrated on the advantages of this approach for brevity. Although open source processes have made remarkable achievements, consensus-making is an inherently political process that defies concise description and has no guarantee of success. Although the SOAP, WSDL and UDDI standards converged rather quickly, similar efforts have failed.

Nonetheless, managing infrastructure that should interoperate in each project stovepipe is exactly like expecting every homeowner to build their own roads. Isn’t it time to try a new approach within DOD?