2004 Command and Control Research and Technology Symposium
June 15-17, 2004

Model-Driven Development of Command and Control Capabilities For Joint and Coalition Warfare

Robert W. Jacobs

Computer Systems Center Inc.
6225 Brandon Ave.
Springfield, VA 22150
bjacobs@csci-va.com

in support of
Joint Single Integrated Air Picture System Engineering Organization
1931 Jefferson Davis Highway, Suite 1100
Arlington, VA 22202

Robert Jacobs is a senior systems engineer working within the Joint Single Integrated Air Picture System Engineering Organization. He recently retired from the United States Navy after 27 years of commissioned active service as a Naval Flight Officer in A-6 aircraft and later as an Aeronautical Engineering Duty Officer and major program manager in the Naval Air Systems Command. Mr. Jacobs graduated from the U.S. Naval Academy in 1971, and received MSEE and Electrical Engineer Degrees from the U.S. Naval Postgraduate School. He also received a MS in Systems Engineering at George Mason University, where his concentration was in command, control, communications, and intelligence systems.

Model-Driven Development of Command and Control Capabilities For Joint and Coalition Warfare

Robert W. Jacobs
Computer Systems Center Inc.
6225 Brandon Ave.
Springfield, VA 22150

ABSTRACT

Warfighter operational requirements for Joint tactical battle management and command and control (BM/C2) now reflect warfare *capabilities* required of networked interoperable systems, as opposed to the previous single-system orientation. However, network-centric warfare won't realize full combat potential until the Department of Defense (DoD) acquisition processes are geared toward developing system-of-systems materiel responses. DoD acquisition processes for solving interoperability must be transformed from a system-system interface basis, which leads to an unworkably complex N^2 problem, to unprecedented (for DoD) holistic networked-system approaches. Initial such thrusts at the DoD-enterprise level, e.g., architecture frameworks, generalized technical architectures, Net Centric Enterprise Services, and the Net Centric Operations and Warfare Reference Model, are not sufficient to address the real time and complex adaptive system engineering challenges of Joint tactical BM/C2. The Joint Single Integrated Air Picture System Engineering Organization (JSSEO) is pioneering the technical and organizational processes needed for engineering development of information technology systems of systems, and prototyping DoD materiel solutions for network-centric operations. This paper discusses the JSSEO Model Driven Architecture™ approach to developing Joint tactical aerospace BM/C2, and points out the implications of broader application to Joint and Coalition command, control, communications, computer, intelligence, surveillance, and reconnaissance system-of-systems network-centric transformation.

INTRODUCTION

The nature of Joint tactical battle management and command and control (BM/C2) warfare requires closely coordinated interoperation of heterogeneous distributed C2 systems, which must appear to the warfighter to be a virtual system of systems (SoS). Tactical system networks are highly dynamic with mobile nodes, *ad hoc* membership, unreliable wireless connections, and are subject to deliberate information-based and physical attacks. The interoperating systems also must function as independent or stand-alone entities. These virtual SoS are appropriately labeled by Maier and Eberhardt as collaborative SoS, a term that connotes important unique qualities for both system operations and acquisition management [Maier]. Collaborative SoS operations are characterized by loosely coupled, independently controlled nodes sharing collaboration rules. Interoperability occurs not from centralized orchestration but rather from nodes operating according to centralized guidance and common objectives to modulate

individual self-interest. While military C2 SoS will never be without centralized control authority, the DoD vision of transformational BM/C2 capabilities aligns with the collaborative class of SoS. The network-centric development challenge for tactical BM/C2 is especially stressful in that unreliable networking and real-time data requirements rule out centralized data-services solutions, i.e., nodes must be fully capable collaborating peers.

Regarding acquisition management, collaborative SoS are characterized by multiple, quite independent system acquisition programs. Systems are planned, programmed, budgeted, and built as individual systems, notwithstanding the recent revamping of the DoD requirements generation system to address capabilities across systems. The nature of collaborative SoS acquisition management mirrors its operation, i.e., it involves collaboration of highly independent programs managed according to centralized acquisition guidance and common objectives (i.e., capability and interoperability requirements) to modulate program self-interest. Military C2 systems generally fit this mold. Adding to complexity, the life cycle of a complex C2 SoS is one of continual evolution as legacy systems are updated or retired, and new systems are added. Collaborative SoS are sometimes termed federations of systems (FoS).

Successful transformation of C2 SoS to meet DoD's network-centric vision requires transformed DoD technical processes that deal with these collaborative operational and acquisition qualities. The Joint Single Integrated Air Picture (SIAP) Systems Engineering Organization (JSSEO) is pioneering the necessary DoD enterprise-scale engineering processes that will evolve legacy systems into collaborative C2 SoS for network-centric operations in Joint and Coalition tactical aerospace BM/C2. Other network-centric transformation initiatives should consider taking a similar approach, learning from the JSSEO example.

JSSEO is applying a Model-Driven Architecture (MDA™) approach toward the development of aerospace C2 capabilities, for which a single integrated air picture is foundational. JSSEO will produce a common-kernel computer program that subsequently will then be tailored for, and implemented in, the majority of military front-line tactical air sensor, weapon, and C2 systems. MDA™ is defined by a set of specifications developed by the Object Management Group™ (OMG™) to improve computer software interoperability and reusability. The data-driven nature of C2 SoS means that the powerful MDA™ concepts adapt well to the collaborative SoS challenges outlined above. All aspects of the JSSEO engineering approach are model-centric, relying on an extension of the OMG™ Unified Model Language for an executable modeling formalism. JSSEO and the many federated-system acquisition programs, known as JSSEO partners, must collaborate to produce fielded computer program implementations that produce consistent data on every system without reliance on centralized solutions.

JSSEO believes executable modeling and MDA™ to be essential to engineering of collaborative SoS. Current DoD enterprise-level approaches for managing SoS interoperability, like the Net Centric Operations and Warfare Reference Model, DoD

Architecture Framework, and Joint Technical Architecture, simply do not have the technical strength to deal with the extremely complex engineering challenges. Other evolving network-centric solutions, especially Net Centric Enterprise Services, are not sufficiently robust in the challenging tactical real-time wireless-connection environment. MDA™, as implemented by industry and adapted by JSSEO, does have the requisite technical power, but requires innovative engineering practices and organizational structures with DoD-wide ramifications. This paper overviews how the JSSEO is using executable modeling as the backbone of processes and integrated teams for requirements specification, analysis, computer-program design, integration on host systems, testing, and logistical planning. Specific recommendations are made on how the JSSEO approach can be applied on a DoD enterprise scale.

BACKGROUND

The following background provides context needed to understand the importance and implications of the JSSEO engineering approach to DoD collaborative and evolutionary SoS development.

JSSEO and SIAP

The Joint Single Integrated Air Picture System Engineering Organization (JSSEO), operating under Joint Forces Command (JFCOM) oversight, is working to fulfill JROC-validated requirements to develop a Single Integrated Air Picture capability for the purpose of allocating resources and ordnance to perform network-centric warfare.

The Single Integrated Air Picture (SIAP) is a state of mutual consistency of data about aerospace objects within the network of peer tactical nodes. The edge devices in the network, and the radio-frequency communication capability that connects these edge devices, constitute the "first tactical mile" of the Global Information Grid (GIG). Edge devices in this network consist of aerospace sensors, tactical battle management and command and control (BM/C2) systems, and weapons. These edge devices are connected today to more remote facilities and data by tactical data links and fixed communication paths. In the near future, these edge devices will be connected by more flexible, higher throughput communication capabilities (i.e., Transformational Communications Architecture).[1]

JSSEO is developing common computer program components that, when integrated into warfighting units, will perform Joint and Coalition tactical aerospace BM/C2 functions. These components will operate as peers in weakly connected mobile *ad hoc* networks. While the peer-to-peer networking will be Internet Protocol (IP) based, legacy tactical datalinks will continue to exist for the time-being, if not indefinitely, and contribute to the complexity of the engineering challenge. For brevity, tactical data link aspects will not be addressed in this paper.

[1] Attributed to CAPT J.W. Wilson, USN, Technical Director of JSSEO

The primary JSSEO product is an executable model, named Integrated Architecture Behavior Model (IABM), specifying Joint and Coalition tactical aerospace BM/C2 functionality. The IABM will be implemented, i.e., tailored and integrated by partners, into fielded computer program components of their weapon, sensor, and command and control systems. "IABM implementations" thus become computer applications in numerous GIG edge devices, and as such will be a part of the GIG.

JSSEO views the notional IABM operating context as depicted in figure 1. The nodes in figure 1 represent tactical role-defined entities having one or more embedded IABM implementations that interact as peers. In this regard, a SIAP "peer" is defined as an entity with an IABM implementation. Nodes performing tactical support roles may or may not have embedded IABM implementations, depending on the nature of tactical support being provided. For example, a theater intel center could be a peer in the SIAP sense, or could provide support to the networked peers via peer-interfaced command and control (C2) systems like Joint Command and Control (JC2) intel servers. Note that a warfighting unit, such as a reconnaissance-surveillance-targeting aircraft, may perform multiple roles, and could contain multiple IABM implementations.

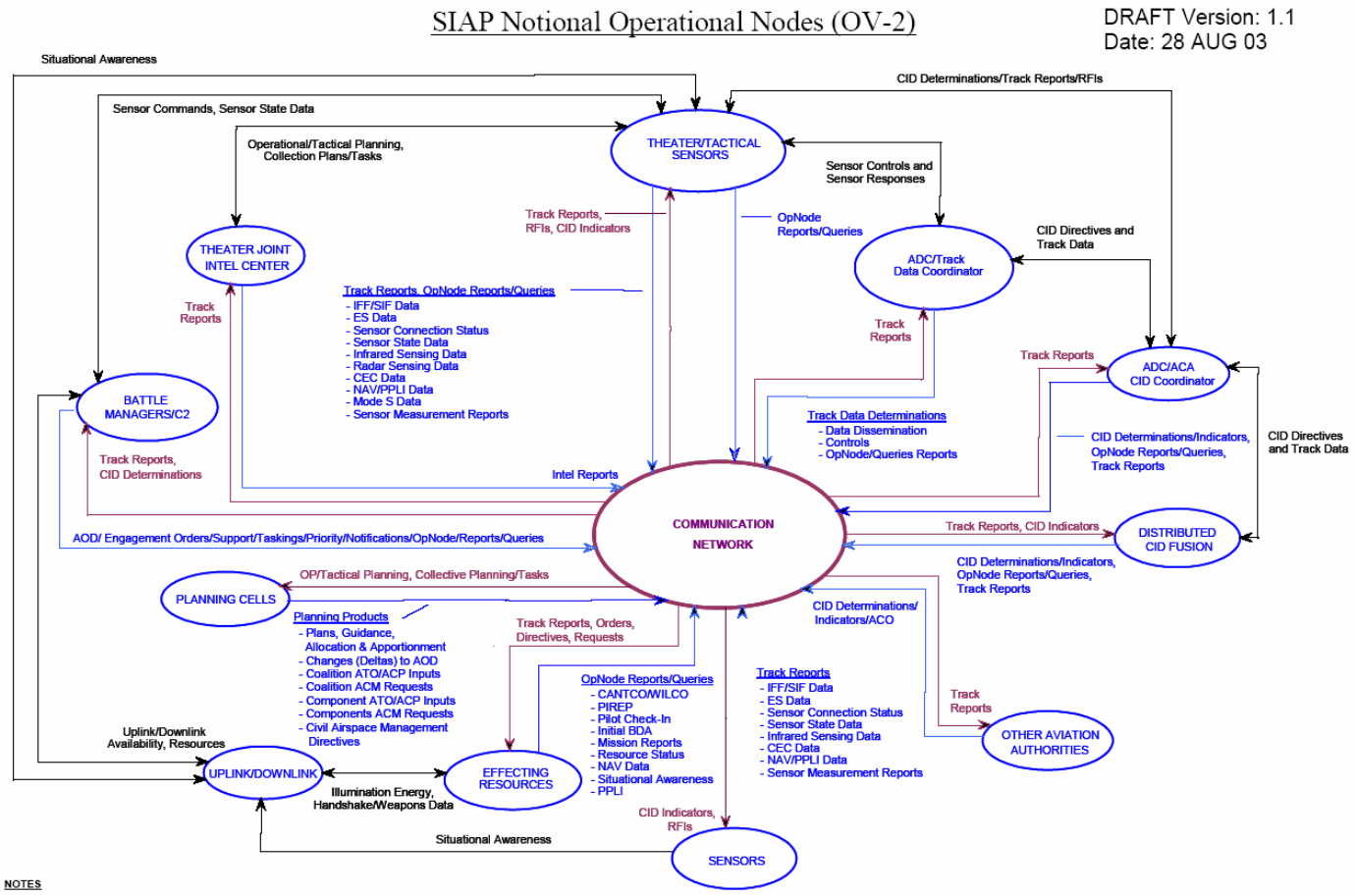

Figure 1. Generalized SIAP Context (JSSEO Integrated Architecture)

IABM ensemble behavior reflects the strategic intent of DoD network-centric transformation. All the transformational architectural tenets [Stenbit] apply:

- Post before processing; avoid unnecessary processing delays
- Users pull the data they need
- Collaborate to make sense of data
- Only handle information once
- Communicate over reliable and assured networks

As an embedded component in tactical aerospace systems that generate or use data on the edge of the GIG, the IABM is a critical implementer of transformation for tactical real-time BM/C2 warfare. The operational scope of IABM distributed system functionality covers not only determining and rapidly posting (according to user pull) the location and identity of aerospace objects, but also collaborating to manage the distributed resources that comprise the infrastructure underlying SIAP. For example, a set of sensors networked across a theater via their embedded IABM implementations will be controllable as one integrated sensor, optimally tasked for aerospace warfare situational awareness, engagement support, target-acquisition cueing, and other operator-specified actions. This must happen with reliability and assurance over wireless networks.

OMG™ Model Driven Architecture™

The software industry formed the Object Management Group™ (OMG™) in 1989 to address software interoperability, portability, and reusability using object-oriented technologies. OMG™ became an industry leader in computer component interoperability using middleware, the Common Object Request Broker Architecture (CORBA™) family of standards. In the 1990's, it became clear that middleware alone could not attain component interoperability and reusability goals. The standards development and approval process was longer than the technology evolution cycle time, so interface-type standards could not meet industry need. OMG™ recognized technology dependence to be the root cause problem and subsequently adopted the Model Driven Architecture™ (MDA™) standards framework. The most significant of the standards are the Unified Modeling Language (UML), an object-oriented language that OMG™ adopted when multiple languages and methods coalesced, and the OMG™ Meta-Object Facility (MOF). MDA™ depends for implementation on both technologies, which are briefly explained in the next section.

MDA™ embodies a profound yet fundamentally simple programming concept: build models with rigorous separation of implementation technology from business concerns. A model of the stakeholder business data, operations, and rules, one that does not embed computing technology dependence, is known as a (computing-host) platform-independent model (PIM). Platform is a relative term depending on where the separation of business and technology is defined, e.g., at the middleware layer, the operating system boundary (and it could be a distributed operating system) or possibly a lower systems layer in a SoS architecture. Use of "platform" as a DoD jargon term for a warfighting unit is specifically avoided here. A rigorously independent PIM would not be seriously affected when inevitable technology upset occurs, unless the new technology fundamentally

affects the business model. Implementation technology is addressed subsequently in the model when the PIM is mapped onto the platform, or platforms in the case of distributed computing, leading to a platform-specific model (PSM). Mapping is a model translation that occurs either by marking the PIM for an automated mapping tool, mapping by hand, or something in between. The platform-specific implementation process then renders the final code artifact that runs on the platform.

Figure 2 depicts these concepts. For example, platform A could be a radar sensor processor and platform B could be a C2 system processor. The PIM can be implemented on multiple target technologies. A PSM is shown in the figure, but it can be an optional intermediate artifact. If the platform technology also has been modeled, then it is possible to interpret the PIM directly into the implementation using auto-generation tools, e.g., using a model compiler to render a technology platform-specific (e.g., Web-services) implementation.

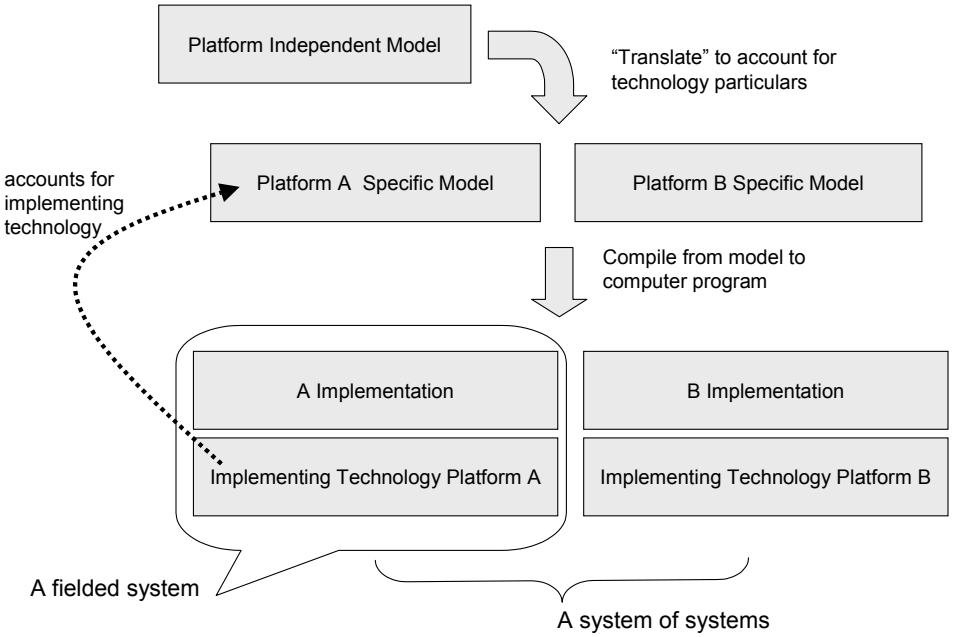

Figure 2. MDA™ Concepts

A Joint and Coalition BM/C2 collaborative SoS is an extremely complex system. However, if the complexity can be distilled to the business essence, as MDA™ promotes, then the engineering challenge becomes manageable. Rather than exhausting resources trying to control technology developments and component interfaces as in the past (and imprudently coupling technology to capabilities too), MDA™ permits large scale IT systems engineering teams to focus intellectual power on the more invariant business rules and object model architecture. Since models can be compiled to computer program components, interoperability theoretically can be implemented and tested at the model level, requiring no further intellectual processes (with their potential disruption to interoperability) to obtain the development end item, the integrated computer program.

MDA™ promotes models that can be interpreted by machine. To achieve the promise of MDA™ as leverage against SoS systems engineering challenges, JSSEO believes for practical reasons the PIM must an executable model in UML. An executable model is one that will compile and run in simulation. SoS evolution is so complex that model correctness requires the development process rigor that model execution demands. Under MDA™, executable UML becomes effectively a new level in the hierarchy of languages for computer program development. This is another in a series of natural progressions in abstraction from machine coding, to assembly language, to higher order programming language. Maier defines abstraction as "representation in terms of presumed essentials, with a corresponding suppression of the non-essential" [Maier, p. 293]. As computer programming languages have evolved in the past, each level farther from the platform saw the technology particulars replaced by more abstract forms. Under MDA™, the technology dimension theoretically can be completely abstracted away[2].

Two MDA™ Standards: UML and MOF

To help understand why MDA™ opens a path to successful evolution of SoS, one not available with traditional systems engineering practices, a brief acquaintance with the main building blocks of MDA™ is required. These are UML, its executable profile, its meta-model, and the Meta Object Facility (MOF).

In the 1990's, a number of object-oriented software modeling notations were developed independently. As the result of collaboration among leading exponents, a single unified notation, UML, became standardized under the OMG™. UML incorporated the best ideas from its predecessors, and thus gained wide industry acceptance. However, collaboration led to high complexity. The current UML specification, version 1.5[3], has 736 pages [OMG™, 2003]. The complexity makes it adaptable, but proper usage can require in-depth knowledge. UML is extensible, and generally, extensions for specific applications are called "profiles."

A point about the systems engineering value of UML must be made. INCOSE[4] and other groups are working with the OMG™ to adapt UML to systems engineering and, as a result, UML deficiencies in this regard are being addressed. Some major gaps noted in UML version 1.4 for modeling systems involve deficient expressions for continuous time behavior, input/output flow, hierarchical modeling, non-behavioral characteristics, properties, physical interfaces, and requirements constructs [OMG™, 2002]. Version 1.5 added action semantics that support executable models. The pending UML version 2.0 addresses many of the system engineering gaps, to a varied degree. However, since an ensemble of IABMs implements a collaborative information-technology (IT) SoS where

[2]In practice the defining of a platform as an implementing technology is only relative to the context. For example, a business model that has been translated for a Web-services technology platform would be considered by the business stakeholder to be a PSM specific to middleware, but to the Web-services designer it is a PIM to be translated for a specific hosting platform.

[3] UML 2.0 finalization is underway.

[4] International Council on Systems Engineering

the primary issues are about data and computer applications, the system engineering shortfalls of UML have not been important. For most collaborative IT SoS, the key component issues are computer-program-oriented and yield well to UML expression today. While adapting UML for systems engineering should be fruitful, Maier captures the real value of UML with the insight:

> "The primary importance of UML is that it may lead to more broadly accepted standardization of software and systems engineering notations" [Maier, p. 214].

That statement is consistent with the JSSEO model-based systems engineering experience.

JSSEO modeling uses a profile for UML known as executable UML [Mellor]. Relying mostly on class diagrams, state charts, and the recent UML action specification standard, UML complexity has been pared to the subset of UML that precisely and unambiguously specifies structure, dynamics, and constraints needed for executability. Model executability enforces precise specification. Executable UML is not yet a standard, but it has gained acceptance from several tool developers, which is evidence of the practical nature of Maier's insight. Executable UML tools also provide the simulation environment needed to compile and run the executable on a UML virtual machine.

Meta-Level	Description	Example Elements
M3	MOF, i.e. the set of constructs used to define metamodels	MOF Class, MOF Attribute, MOF Association, etc.
M2	Metamodels, consisting of instances of MOF constructs	UML Class, UML Association, UML Attribute, UML State, UML Activity, (& other-modeling-language elements)
M1	Models, consisting of instances of M2 metamodel constructs	Class "Customer," Attribute "Account#" (& types modeled in other languages)
M0	Objects and data, i.e., instances of M1 model constructs	Customer "Joe Doaks", Account# "6543" (& other instances of types)

Figure 3: MOF Meta-Levels [adapted from Frankel]

An inspired early decision by OMG™ to keep MDA™ language-independent permits application of MDA™ going beyond software engineering, leading to powerful model and tool interoperability. MDA™ language independence relies on meta-models, the technical heart of MDA™. Models built in accordance with UML are, by definition,

compliant with the UML meta-model. For example, a class "track" in a UML model must conform to the semantics and syntax rules of a UML "class." In order to relate UML to other languages, OMG™ has created a universal modeling language parent. This parent, the higher level meta-model, closely resembles UML. This model has a meta-meta-model relationship to our "track." The OMG™ specification for the Meta-Object Facility (MOF) created this parent language, the self-defining "MOF model" or simply "MOF." Figure 3 depicts the meta-model levels defined by OMG™ in the MOF architecture [Frankel, p.105]. Using MOF, a formal model of each M2 meta-model (i.e., each modeling language) can be defined. UML is MOF-compliant, and through the UML MOF ancestry, model elements in UML can precisely relate to elements of other MOF-compliant M2 meta-models. It is not necessary for these MOF-compliant languages to be object-oriented. MOF-compliant UML models can be exchanged, stored in MOF repositories, and even made to interact in a run-time environment. To enable this, OMG™ developed a standard for mapping MOF to the Extensible Markup Language (XML) known as XML Metadata Interchange (XMI). Using XMI, tools can support heterogeneous model interactions. PIM components can be retained in a MOF repository for reuse by any other MOF-compliant PIM. PIM representations of business operations and rules, which typically have a longer life than technology, are good candidates for retained model components. This capability for model interaction opens the path to auto-generation of development artifacts from higher-order modeling. For example, a translator/compiler reflecting a model of the technology platform could interoperate with a PIM to yield most or all of a computer program implementation.

Without elaborating in detail how UML, MOF, and XMI theoretically work, the point to note is that OMG™, by creating a modeling environment designed to build and exchange models, has spawned a rapidly growing environment for practical engineering of IT SoS. The technology-neutral meta-data integration approach that relates UML, data model languages, middleware languages (e.g., CORBA™), etc., has already led to powerful vendor products for UML-based executable modeling. One of these products, iUML™ from Kennedy Carter LTD[5], forms the executable modeling environment used by JSSEO.

JSSEO TECHNICAL APPROACH

The next section addresses the executable model, the distributed system architecture, and the agile process that is producing the model.

Integrated Architecture Behavior Model

JSSEO is performing object-oriented computer program development. The primary JSSEO artifact is the Integrated Architecture Behavior Model (IABM), an executable object model that is a Platform Independent Model (PIM). The IABM functional scope covers all upper level[6] Joint and Coalition tactical aerospace battle management and command and control (BM/C2) capabilities needed in operational military systems.

[5] http://www.kc.com/press/press_siap.html
[6] I.e., functions above the transport layer of the Open Systems Interconnection (OSI) reference model.

While a single IABM can be executed for model development purposes using the development tool (iUML™), designing and testing the behavior of an ensemble of IABMs is a more difficult engineering challenge. For this reason, JSSEO is developing a modeling and simulation environment complaint with the High Level Architecture[7] (HLA). Multiple IABM instances can be created, connected into a model federation, and stimulated according to Common Reference Scenarios that have been approved by Joint requirements authorities.

The object model is partitioned into domains:

> "Each domain is an autonomous world inhabited by conceptual entities. The conceptual entities in one domain require the existence of other conceptual entities in the same domain, but they do not require the existence of identified conceptual entities in other domains." [Mellor, p.14]

Domains permit separation of the object model into distinct subject matters. This partitioning has profound SoS development implications. Discussion of management implications is deferred to the next section. Properly encapsulated domains become reuse primitives, which will grow greatly in importance over the life cycle. An IABM is constituted when domains are bridged together. Not all IABM implementations need to be instantiated using all domains. Each implementation only contains the domains relevant to the warfighting unit's mission. However, all warfighting units with the same mission processing requirements must use the same domains.

To manage the IABM implementation domain content, the IABM domains are characterized in three types: core, optional, and adaptation; depicted in figure 4. All implementations receive all core domains and those optional domains that apply to the mission. Adaptation domains are interface domains specific to particular sensor, weapon, or command and control systems. JSSEO is constructing generic adaptation domains, e.g., a 3-dimensional phased array radar interface, to facilitate final domain construction by the partners. One might ask why the adaptation domain is in the PIM since it appears to depend on technology. In this case, the radar technology defines the IABM business rules (e.g., data exchange protocols) independent of the radar operating system and processor technology, which has been abstracted away.

[7] IEEE Std 1516 series

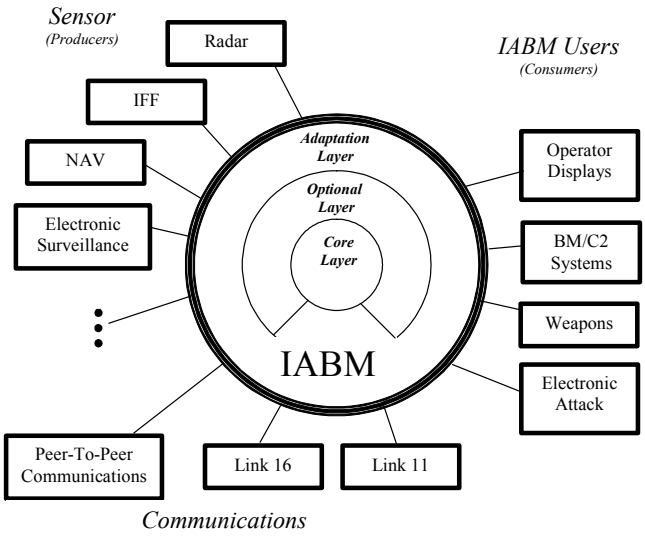

Figure 4. IABM Notional Configuration (JSSEO)

When the IABM has passed acceptance testing, IABM implementation generation will be performed by individual partners. Partners will translate the IABM PIM into an IABM Platform Specific Models (PSM) by selecting the proper domains and tailoring them to operate on their host system computer processing resources. The PSM will then be compiled into the end item code (i.e., the particular IABM implementations). Ideally, the compiled code will not require further modification or extension for integration. However, the target hosts are nearly all legacy systems. The "openness" of the host will determine the ease and extent to which unaltered model-compiled code can be integrated. Manipulation at the host level greatly increases risk to interoperable and predictable behavior of IABM implementations, and must be aggressively controlled if not avoidable. The host may be a distributed set of processors. For example, a sensor adaptation domain could be hosted in a radar processor, while the core and optional domains are hosted in a mission computer. Domain integrity and coupling vulnerabilities are critical design issues, since the interdomain communication mechanisms can vary greatly. Domain bridges, part of the IABM, account for these issues.

Interoperability Architecture

The fundamental IABM solution for interoperability is predicated on common processing at network nodes. By managing internodal data distribution on an object basis, common processing of common inputs provides nodes with common results, in the form of distributed, replicated (on an object basis according to need) aerospace track data sets at each node. This is the SIAP state of mutual data consistency.

The common processing architecture is considered to be the most viable architecture in view of peer-ensemble-behavior complexity[8]. The collaborative SoS necessary for

[8] The utility of MDA™ and executable architecture modeling for collaborative SoS is not limited to a common processing architecture. For example, a PIM for the collaborative SoS could be partitioned so that

Joint tactical aerospace BM/C2 exhibits the characteristics of dynamic complexity as defined by Calvano and John, mainly that systems composed of relatively simple components can exhibit complex and unpredictable emergent behavior. Designing for predictable BM/C2 behavior over a large scale deployment of heterogeneous systems is extremely challenging. In nature, complex behavior typically involves very large numbers of identical primitives, as with avalanches in a dune of sand. In the natural case, there appears to be some predictability in behavior[9] [Calvano]. Similarly, as depicted in figure 5, IABM implementations in each interacting system are networked common-processing components that are identical at the domain level.. Thus is achieved logical homogeneity across the collaborative SoS so complexity can be understood and controlled.

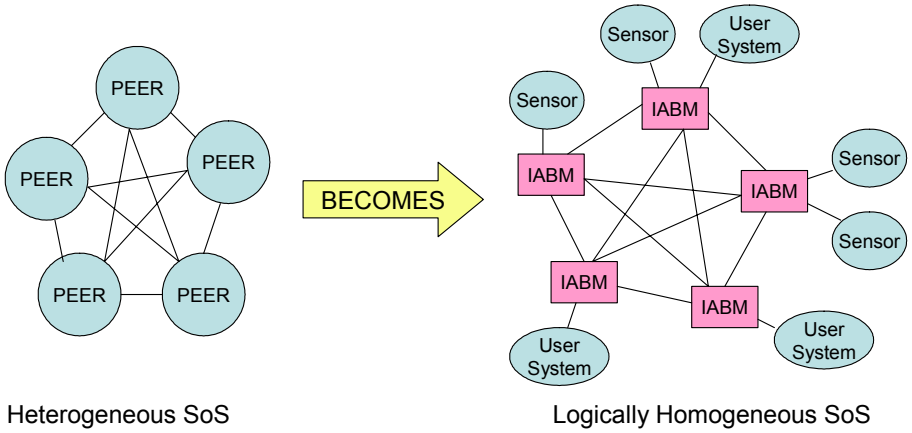

Figure 5. Architecture Solution For Complexity

The primary IABM operation will be the fusion of sensor measurements. Peers with sensors will produce and distribute the sensor measurements to other peers. Before distributing the measurement, the IABM in the peer will look for an association to an existing track[10]. If found, the measurement will be distributed as an associated measurement report (AMR). During networked operation, AMRs will constitute the vast majority of data transferred between peers. Considering the large number of legacy sensors, each with its own sensor-processor peculiarities and report syntax and semantics, trying to accomplish global interoperability an interface at a time (as implied in the current DoD enterprise architecture approach[11]) is daunting and almost certainly

---

each component system is allocated only its share of the processing, with all partitions having nothing in common.

[9] A power law relationship between event magnitude and frequency of occurrence has been observed.

[10] Track, as used here, is a perceived location and identity state of an aerospace object, to which other data can be associated

[11] The currently mandated DoD enterprise interoperability engineering environment depends on architecture frameworks [DoD CIO, 2004], monolithic data models, technical reference models and technical architectures to guide acquisition offices in system to system interface definition and disclosure. Movement from interface reliance has started with efforts such as the Net-Centric Data Strategy, but much more of a transformation is needed.

unmanageable. However, with an IABM to adapt sensor data for common core processing, interoperability challenges are greatly reduced.

The main interoperability challenge for the IABM involves real-time data distribution (over unreliable channels) to keep the distributed replicated peer databases consistent. This is accomplished mainly by rigorous control of inputs to the common domains, along with background processing to find and fix replication errors. Algorithms to distribute data among ensemble peers (transport layer services are assumed to exist at each peer) must handle three distinct types of data exchange: sensor-sourced state updates, decision data, and dataset transfers. SIAP data does not include audio and video streams, although this is not ruled out in the future. AMRs comprise the main state-update data exchange; in this case a *track*-state update for an aerospace object. Since sensor-sourced state updates will be relatively frequent, the effect of lost or erroneous peer-to-peer messages will wash out rapidly. Therefore, distributed IABM data stores that are observed-state-based are convergent or stable, and lower level quality-of-service (QoS) message transport generally can be tolerated. Decision event and certain dataset transfers require delivery guarantees. An example decision event is a decision by an IABM to promote into a track a set of unassociated measurements from a peer's sensor. Decisions at one peer require action by other peers to replicate the decision in the appropriate IABM data store. To continue the example, the decision by an IABM to initiate a track must be announced to other affected peers, who will then initiate their own tracks. Once the track with an initial state is established, the peers are all ready to receive AMRs when the original peer sends updates. If a decision message is somehow missed, peer data stores instantly become divergent. Finally, when network topology changes, such as connection dropouts, reconnections, network startup, or detection of a new node, then network peer data store consistency must be established or reestablished. In performing real-time data distribution, IABM domains must operate in concert with the peer-to-peer network-operations functions being developed for the GIG to, e.g., obtain knowledge of network state, route messages according to QoS requirements, limit messaging when network loading exceeds capacity, and reserve path resources for situations like engagements.

Using the Information Age Warfare language of Alberts, *et. al*., the IABM in conjunction with other peer systems performs operations in the Information and Cognitive Domains [Alberts, 2001]. The IABM operates in the Information Domain by distributing sensor observations and putting data into meaningful context using *a priori* knowledge. An example of the latter would be comparing a tracked aerospace object's behavior to known patterns to formulate or increase confidence in a combat identification. The IABM also operates to support the Cognitive Domain by developing data fusion products to promote situational understanding, awareness, assessment, and decision-making. Using intermediate fusion products such as resource pictures and threat assessments, the IABM can present decision recommendations to an operator, or if authorized, issue orders for action. Uncertainty associated with data, fusion products, and decision recommendations is determined by understanding and accounting for the basic uncertainty in sensor measurements and identity declarations. While Cognitive Domain BM/C2 capability will not be in the first IABM implementation, the architecture will be in place.

Insight to the nature of Cognitive Domain operations of the IABM ensemble can be gained by examining command and control of a network of peers. Understanding C2 of IABMs also illuminates the operational implications of the common processing architecture. Alberts observes the need for common perception of command intent, and the need to be able to reconcile different perceptions [Alberts 2002]. This observation is precisely reflected in the IABM architecture. The ensemble of IABMs must operate using common C2 rules, and therefore must maintain a common C2 Rule Set, one of the IABM datasets. The need for rigorous consistency management of distributed data stores should now become clear. The ability to deliberately plan, and dynamically replan the C2 Rule Set are important collaboration requirements on the IABM. As peers process AMRs and develop the track data store plus other fusion products, at some point C2 Rules will lead to a decision for action. For example, by periodically assessing SIAP quality, and sensors' capability and location, IABM distributed resource management C2 logic may call for a recommendation to change a sensor's parameters (e.g., field of regard, power) for improved coverage. An IABM could have operational authority assigned by the C2 Rule Set (as planned by the local commander) to so task its local sensor. It is envisioned that each IABM will internally develop a distributed, replicated Common Task Set that allows each peer to keep track of the resource tasking across the network. Similar requirements and solutions exist for distributed weapons control.

Agile Development Process

JSSEO employs a modified agile development process. Agile development principles[12] have been found to be extremely important to JSSEO in the development of collaborative SoS. The most germane are:

- Early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development.
- Deliver working software frequently.
- Subject-matter experts and developers must work together throughout the project.
- Working software is the primary measure of progress.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

IABM builds, known as Time Boxes, are produced every four weeks, with an eight week lifecycle for each build. Each Time Box goes through successive phases of domain requirements specification, domain modeling, IABM integration and testing, and Time Box assessment. Other JSSEO technical teams operate with less agility outside the Time Box process to accomplish architecture definition, definition of system requirements for IABM releases (currently expected on a two year cycle), and verification, validation, and acceptance testing. The latter activity includes the development of an HLA-based testbed and other venues for live testing of IABM implementations. The first IABM release is

[12] The selected principles are borrowed or adapted from http://agilemanifesto.org/principles.html

planned for September 2005. Partner system development teams will follow up with several[13] years of adaptation domain and IABM implementation development, integration, and testing before fielding.

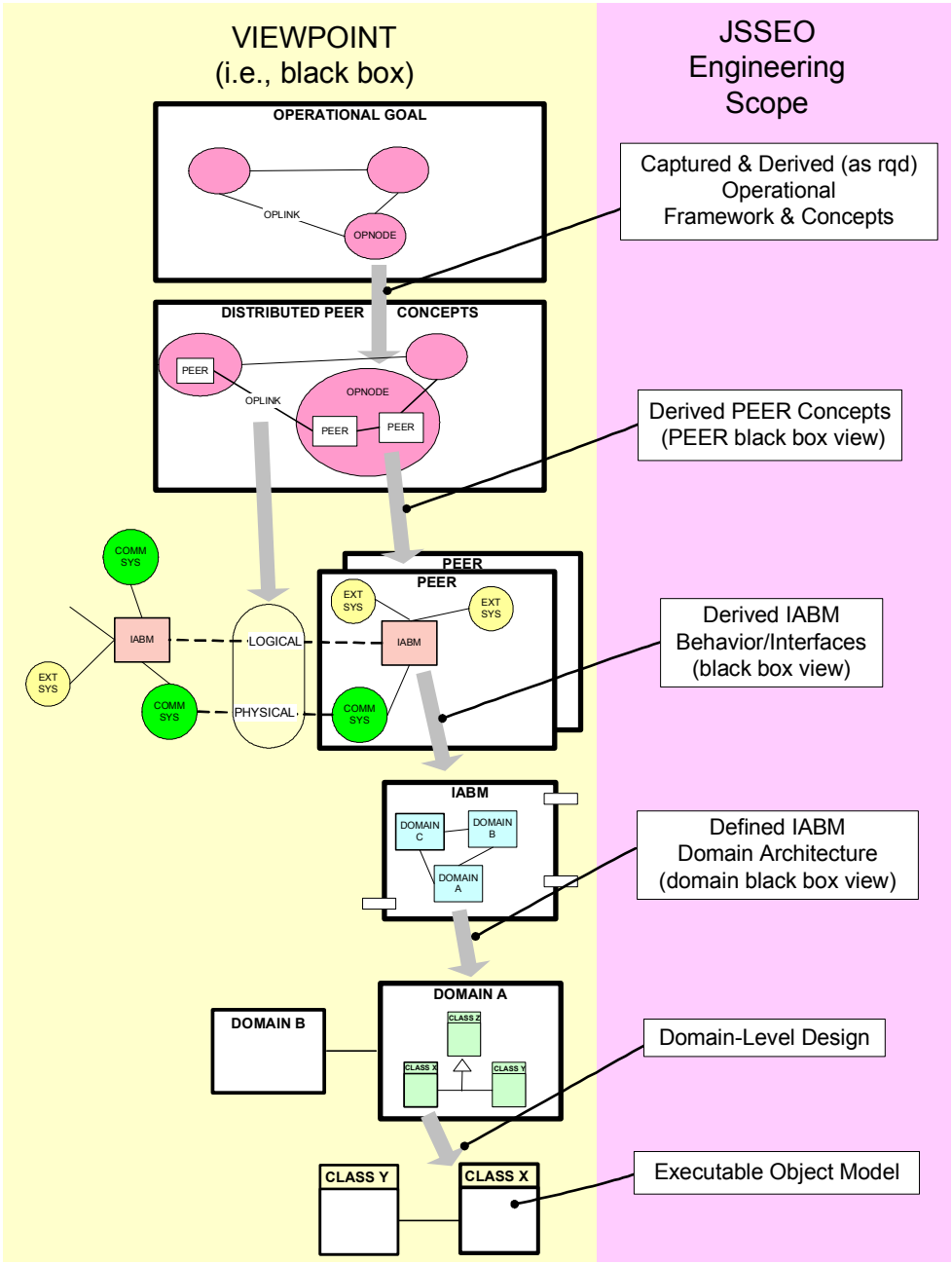

Figure 6. JSSEO Engineering Dimensions

It is important to understand the hierarchical dimensions of a collaborative SoS and appreciate that all dimensions must be managed. The JSSEO process addresses six

[13] The number of years is host-system dependent. A minimum of two years is expected.

architectural dimensions, all of which have special process requirements. Figure 6 shows the dimensions and the engineering scope that JSSEO must account for. Each dimension has its own level of requirements and operating concepts, product (configuration item) development, associated development teams, and traceable relationships. Lower levels are dependent on higher ones for requirements definition and products. All dimensions must be appropriately covered, or critical design disconnects will exist at the lower dimensions.

Time Boxes and executable modeling currently apply at the IABM and lower dimensions. At higher levels, product release intervals are longer and executable modeling has not been applied. Use of executable distributed IABM system requirements and architecture computation-independent models are being considered with the expectation that agile (although somewhat less so than previously described) development principles could be applied at the upper dimension levels too.

Alberts contemplates the difficulty in developing a SoS, and concludes that DoD needs an engineering cultural change that relies less on requirements specification and more on process that promotes prototyping and rapid co-evolution of mission capability packages (MCP)[14]. His expectation evidently is that experimentation rather than product-oriented engineering is more important at this time, as DoD progresses through the innovation phase of transforming for network-centric warfare. From the materiel viewpoint, the JSSEO agile development process reflects his conclusion, although not as he apparently expected. He was skeptical that direct engineering of SoS was realistic in the Information Age [Alberts, 2002, p.16]. Yet this is precisely what JSSEO is doing with MDA™ under an agile development process.

JSSEO ENGINEERING MANAGEMENT APPROACH

Through close examination of a few successful though reduced-scale FoS, Krygiel learned the importance of a collocated task force, which is the JSSEO approach. "The single facility and common environment can bring cohesiveness to build the twin citizenship necessary for federations to succeed" [Krygiel]. JSSEO employs a large single-site team of engineers performing or preparing for all phases of the systems engineering lifecycle. Operational representatives from Joint Forces Command and the Joint Air and Missile Defense Organization are present as well. The military branches are participating by populating the team with engineers, and acquisition managers from the partner programs are starting to develop integration teams as extensions to the task force. As the IABM maturity grows, the partner system integration teams will become integrally involved with the main task force to perform adaptation domain developments, PSM development, and preparations for legacy system integration of the IABM implementation.

[14] Defined as "…concepts of operations, command and force structures, the corresponding doctrine, training and education, technology, and systems with a support infrastructure designed and tailored to accomplish specific missions." [Alberts, 1995]

Although the task force approach is essential to developing collaborative SoS, it is IABM model structure itself that defines the product organization of JSSEO engineering. Recall that model domains reflect highly coherent and loosely coupled subject matter capsules. Working under an evolving architectural definition, very small teams of subject matter experts and modelers work collaboratively on individual domains. Responding to just-in-time Time Box requirements, these domain developer teams incrementally repeat a design, build, test cycle for IABM domains. Domains are then integrated and tested as a system, generating architecture and requirements feedback for subsequent Time Boxes. Refactoring of domains is a critical normal practice. The learning feedback also goes to technical teams addressing the distributed system complexity issues, SoS architecture and requirements, and testers developing the test processes and infrastructure.

Domains define the primary JSSEO engineering management requirements. In essence, the model domain is the systems engineering primitive-level component, i.e., it is the lowest level configuration item. The JSSEO task force is organized to build and integrate domains. The task force composition is fluid because domains are in varying states of initiation and completion. Domains that require technical issue resolution, such as determining the appropriate track-building algorithms, are supported by focus teams populated with nationally recognized engineers or scientists, and with JSSEO experts to lead them. Once the algorithm or technical approach is defined, the focus team disbands. As the domain stabilizes, iUML™ supports requirements identification on an entity basis, so traceability linkages can be chained from an operational requirement down to particular classes in the object model. Domain by domain using tool features, the IABM configuration is baselined, controlled, documented, and verified and validated.

By continually integrating results, the intense development activity at the domain level does not become chaotic. Thus a large group of engineers with limited knowledge of the total context of the IABM is kept productively and concurrently employed producing a single computer program. This is done despite the fact that a fully developed architecture does not exist; it too is being evolved as the agile process moves forward.

As partners become active in IABM development for the adaptation domains, the same principles will apply except the development effort may move outside the central task force. However, partners will be using generic adaptation domains built expressly for tailoring to specific system types. Domain subject-matter encapsulation allows the coherent and orderly expansion of the task force technical activity to multiple development locations. The domain-tailoring approach demonstrates the more general point that domains represent the working unit of reusable computer code, resused as IABMs are evolved to address more systems and expand collaborative SoS capabilities in the future.

IMPLICATIONS AND RECOMMENDATIONS

The JSSEO common processing architecture is only one solution path for SoS interoperability, and the points made here (both in the preceding and in the following) apply generally to any interoperability engineering approach, unless noted.

Federation of Object Models

Command and control (C2) of Joint and Coalition warfare requires *ad hoc* federations of systems, i.e., federations that can be reliably created on the battlefield without prior knowledge of membership. This paper describes how the Joint and Coalition aerospace segment of this federation capability is being created. The IABM object model is the vehicle for designing the capability to structure collaborating systems into *ad hoc* federations. However, network and major sensor, weapon, and BM/C2 resources are typically not allocated just for single-mission C2. Network-centric warfare requires distributed resource management across all mission areas, extending the complexity issues well beyond what JSSEO is grappling with. What is required is the ability to federate on an *ad hoc* basis all Joint and Coalition battle management and command and control (BM/C2) systems. The logical solution to this challenge lies in developing and then federating object models from all major mission areas. This BM/C2 federated object model can then be used in a macro scale version of the JSSEO approach for understanding, designing, integrating, and testing capability.

Recommendation: DoD acquisition leadership should initiate engineering task forces to address particular major mission areas, partitioned so as to cover all required network-centric operational capabilities. Leverage the JSSEO task force approach as an acquisition-community prototype for developing collaborative SoS, with the specific goal for each task force to develop a mission-specific executable object model compliant with the OMG™ Model Driven Architecture™ (MDA™).

Platform Independent Model Implementation

Handling of the Platform Independent Model (PIM) to accomplish implementation greatly increases risk to interoperability. Before implementation, the PIM is closely controlled and exhaustively tested for predictable and emergent behaviors. In contrast, subsequent processes for translation of the PIM to a Platform Specific Model (PSM), compilation to an implementation, and integration in the host platform can vary in repeatability, being anywhere from totally automated to totally manual. A legacy host with an open architecture, and platform-specific tools for PIM translation and compilation is in a much better position to control risks in PIM development. Life cycle management of Joint aerospace BM/C2 is much more manageable in this case. Changes in platform technology can be reflected in the platform-specific tools without affecting federation capability. Changes in the PIM to reflect increased capability or changes in the business of BM/C2 can be readily fielded as computer program component changes to the federating systems. However, in the case of a coordinated upgrade in hosts of multiple types, the fielding schedule may be dependent on the type with the least process automation.

Recommendation: Examine Joint and Coalition system federation requirements. Incentivize or initiate federate-system preparations for PIM implementation by developing open architectures and host-platform object models (in executable UML) that support model-compiler development and PIM-implementation integration. Perform this

activity in parallel with or prior to PIM development to promote learning on the part of the system partners, help task force requirements discovery, and reduce overall development time.

Executable Models of Operational Requirements

Transformational network-centric operations and warfare concepts generally are not well articulated and indicate a lack understanding of implications of coming technologies. Existing expressions rely on operational architecture views that typically are little more than unpartitioned lists of functions. For example, it is clear that the IABM ensemble must use a common C2 Rule Set for policy management of IABM behaviors. This is consistent with published concepts on policy-driven operations. However, what do these rules need to cover, and how does the warfighter desire to control them? The conceptual behavior and business rules of Joint and Coalition BM/C2 could be well expressed in a computation-independent executable requirements model, i.e., an executable operational architecture, maintained by appropriate warfighter representatives. These models, on their own, could be used as formal specifications of operational requirements, while "agile" excursions could be developed to gain better understanding.

Recommendation: DoD provide a cadre of system engineers trained in expressing operational concepts and requirements in executable operational architecture models. Similar to the JSSEO approach, form small teams of operational subject matter experts and modelers. Incorporate executable operational architectures as requirements specifications into Mission Capability Package capability definition processes.

Common Processing Implications

This section applies only to architectures designed for common processing when implemented in heterogeneous systems. In this case, some issues arise that call for enterprise-level policies.

- Despite the recognition that acquisition processes must be more oriented to network-centric capability and less platform-centric, and whether for political or technical reasons, the reality is that some processes will continue to be platform-centric. This is certainly true for operational testing, and interoperability and security certifications. If the core processing is highly tested and common in all systems intended to be federates, how much less investment is needed in testing or certification of each individual system? What operational testing and certifications can be accomplished on a model-driven basis at the PIM level? It is clear that models consisting of end item computer code have a different utility in testing than the more typical abstract models of an end item. Can the difference be leveraged in simulation-based acquisition to reduce platform-centric process redundancies?
- The Global Information Grid (GIG) communications programs and Network-centric Enterprise Services (NCES) are currently under development, with little apparent understanding of who is at the edge of the network. While this "build it and they will come" transport and below approach works for the phone company, Joint BM/C2 interoperability must be developed at the higher data and application levels as well. Fortunately, a common processing approach makes this problem

infinitely easier. The higher level interoperability issues, most of which are shared by multiple mission-related GIG-edge devices, are best addressed with enterprise-common business-level solutions using MDA™ approaches like those employed by JSSEO. Specific issue areas involve network C2, information assurance and security, mission-specific data tagging, control of transport service quality, and NCES service invocation.

Recommendation: DoD should address GIG policies and development processes from the point of view that mission capability models of the GIG-edge devices will exist and can be leveraged for total GIG development, especially for the specific issues and areas above. Additionally, examine operational testing, interoperability certification, and security certification processes to leverage common processing architectures in major-mission classes of edge devices.

Model-Driven Network-Centric Development Environment

Developing C2 capabilities for network-centric operations requires an engineering environment in DoD that is equipped to handle the complexities of collaborative SoS. Architecture frameworks, static data models, reference models, and catalogs of admissible technical standards are artifacts of a deliberately weak standards-based approach, and have not been successful. Current net-centric transformation initiatives above the transport layer are technology-dependent, following an Internet approach emphasizing metadata[15], Extensible Markup Language (XML), and Web-services[16] standards. While this approach mimics the Internet solution to building an extremely large collaborative SoS, in essence it relies on communication middleware technologies that can be slow and unreliable to the warfighter BM/C2 needs, and susceptible to technology churn. It also invites acquisition programs to take a data-centric approach to interoperability without attention to the behavior of distributed applications in a FoS, which reflects the real net-centric operational capability. While these measures will greatly improve network-centric operations, they beg to repeat the lesson OMG™ learned with CORBA™, i.e., that middleware is not enough. The vision of Joint and Coalition BM/C2 interoperability requires a more robust collaborative SoS development approach. As discussed previously, the complexities of a collaborative SoS practically dictate a development approach using an executable object model, or federation of object models, and MDA™ principles. The OMG™ Meta Object Facility (MOF) and the MOF meta-level modeling approach exemplify what is needed for a DoD enterprise information technology (IT) engineering environment. The existing OMG™ infrastructure permits model interoperability, tool interoperability, and may well be adaptable for collaborative IT SoS in DoD. Also needed is a development and testing simulation environment where federate-system models can interact using a realistic GIG transport layer testbed.

[15] The DoD Net-Centric Data Strategy [DoD CIO, 2003] calls for systems to register their metadata, and use metadata to advertise, publish, subscribe, and exercise "smart pull."

[16] NCES primarily depends on the Web-services standards: Universal Description, Discovery and Integration (UDDI), Simple Object Access Protocol (SOAP), and Web-Services Description Language (WSDL).

Recommendation: DoD should employ MDA™ and executable object modeling in the development of NCES. Also, examine the OMG™ MDA™ infrastructure as the foundation for an enterprise engineering environment that addresses the full scope of complexity issues, not just aspects like data interoperability. Define a robust environment at all meta-levels that serves all network-centric development needs. Prepare to support federation of large major-mission object models, of which the IABM and its implementations are only one type. Provide a network-centric simulation environment, with executable object model of the GIG transport layer interfaces so that GIG transport services for GIG-edge devices can be included in simulation federations.

CONCLUSIONS

This paper has discussed an ongoing model-driven development of tactical aerospace battle management and command and control (BM/C2) capability for Joint and Coalition warfare. JSSEO is employing the OMG™ Model Driven Architecture™ (MDA™) to build an Integrated Architecture Behavior Model (IABM), a Platform Independent Model. The IABM is an executable UML model of the business of Joint and Coalition tactical BM/C2 as it is to be performed by collaborative federations of systems (FoS). The IABM, a set of tightly coherent domains loosely coupled using domain bridges, is a highly abstracted form of computer program that will be transformed and integrated into host federates. As a PIM, the IABM is essentially agnostic to the underlying computing technology, and therefore is strongly immune to rapid evolution that is characteristic of commercial technology. Using MDA™, JSSEO is pursuing an architecture that is predicated on common processing in every federate. The common processing architecture reduces the complexity in federating heterogeneous systems to a more controllable problem, namely understanding the distributed behavior of an ensemble of homogeneous IABM implementations having a common object model. Developing the object model using executable UML allows JSSEO to employ an agile process, continually delivering running code that can rigorously verify and validate the architecture and requirements. In performing this work, JSSEO collaborates with partner acquisition program offices for the federating systems, using a task force organization. JSSEO systems engineering team structure and processes leverage the executable object model's domain structure to manage the collaborative development environment and control development risk. Against the extremely difficult problem of mission-wide net-centric transformation, the domain-level encapsulation of subject matter decomposes the management challenges as it does the technical complexity.

While the first IABM will not be complete until September 2005, enough has been accomplished to demonstrate that JSSEO is successfully performing network-centric transformation for Joint and Coalition warfare. However, JSSEO is only covering the aerospace BM/C2 mission area, a subset of the network-centric transformation scope. Moreover, while current and evolving DoD network-centric transformation policies will improve interoperability, they are not sufficient for successful transformation of major FoS. This paper asserts that the JSSEO approach to collaboration should be viewed as the prototype of a new DoD network-centric acquisition approach, one that uses MDA™ and executable UML modeling. Proceeding in this way both defines the end-item business-level computer program to be integrated in component systems of a FoS, and

frames the collaboration activity of the task force. The paper goes on to point out a number of positive implications in this approach for network-centric transformation, and makes implementing recommendations. In summary form, these are

- Initiate task forces for other mission areas and leverage JSSEO learning.
- Prepare candidate systems for federation by developing open architectures and executable UML object models of their applicable components.
- Help warfighters perform executable requirements modeling as a new form of mission capability requirements specification.
- Examine the implications of MDA™ and federated common-processing architectures with regard to established DoD development, testing, and certification processes. Take advantage of the modeling to reduce redundant platform-centric work.
- Go beyond current metadata and middleware technology environment initiatives Mimic the OMG™ MDA™ example, plus provide transport simulation for an engineering environment to address the full scope of network-centric transformation.

REFERENCES

Alberts, David S., *Mission Capability Packages*, January 1995, http://www.dodccrp.org/MissCap.htm.

Alberts, David S., Gartska, John J., Hayes, Richard E., Signori, David A., Understanding Information Age Warfare, DoD Command and Control Research Program (CCRP), August 2001.

Alberts, David S., Information Age Transformation, DoD CCRP, June 2002.

Calvano, Charles N., and John, Phillip, *Systems Engineering in an Age of Complexity*, Systems Engineering, Vol. 7, No.1, 2004.

DoD Chief Information Officer (CIO), *DoD Architecture Framework, Version 1.0*, 9 February 2004.

DoD Chief Information Officer (CIO), *DoD Net Centric Data Strategy*, 9 May 2003.

Frankel, D. S.; Model Driven Architecture™, Applying MDA™ to Enterprise Computing; Wiley Publishing, Inc., Indianapolis, IN, 2003.

Krygiel, Annette J., Behind The Wizard's Curtain: An Integration Environment For a System of Systems, DoD CCRP, July 1999.

Maier, Mark W. and Rechtin, Eberhardt; The Art Of Systems Architecting; Second Edition, CRC Press, 2000.

Mellor, Stephen J., and Balcer, Marc J.; Executable UML, A Foundation for Model-Driven Architecture; Addison-Wesley; 2002.

Object Management Group™, Inc. (OMG™); *Requirements Analysis For UML for Systems Engineering (SE), Draft Version 0.4*; OMG Document # syseng/2003-02-01 ; November 12, 2002.

Object Management Group™, Inc. (OMG™); OMG™ Unified Modeling Language Specification, Version 1.5, March 2003.

Stenbit, John P., *Statement of John P. Stenbit, DoD Chief Information Officer, before the Subcommittee on Terrorism, Unconventional Threats and Capabilities, House Armed Services Committee*, United States House of Representatives, April 3, 2003.