

# A Conceptual Model for Interoperable Command and Control Acquisition

**John A. Hamilton, Jr.**

Department of Computer Science  
and Software Engineering  
Auburn University  
Auburn, AL 36849  
*hamilton@eng.auburn.edu*

Interoperability is a hard problem and there has been no shortage of searches for quick fixes. Comparatively easy problems are fixed as “proofs of concept,” which then wither when unable to scale up to the huge size of the DoD enterprise. In order to perform a meaningful analysis of individual interoperability tools, we developed a methodology (the Rosen - Parenti Model) and analyzed the domain against which the tools were to be evaluated. We next mapped the Joint Capabilities Integration and Development System against that domain.

The results were and are instructive. The research team determined that in many cases, the tools themselves were driving processes. In our view, tools should be developed to support processes, not drive processes. First, we will describe the methodology developed and then move into the domain analysis. Although we will briefly report the results of our tool analysis, we stress that our findings indicate that the tools themselves are not terribly important. What is important is the domain model itself. The domain model provides an effective means of developing useful DoD Architecture Framework-compliant architectures. Most importantly, the domain model addresses the very real differences in perspective when implementing a DoDAF-compliant architecture.

**Keywords:** Command and control, interoperability, operational architecture, system architecture, mission architecture, JCIDS, DoDAF

## 1. Introduction

Interoperability is a hard problem, and there has been no shortage of searches for quick fixes. Interoperability tools are often proposed as quick fixes. Comparatively easy problems are solved as “proofs of concept,” which then wither when unable to scale up to the huge size of the DoD enterprise. Recognizing this trend, the Under Secretary of Defense for Acquisition, Technology, and Logistics (USD AT&L) tasked the author and his staff to study interoperability tools [1]. Interoperability tools were loosely defined as tools that somehow support or assess interoperability. A major motivation for this study was the increasing realization that scale issues are a dominant challenge in DoD interoperability. Many vendors claimed that their interoperability tools could solve this scale problem. What we determined

was that no tool was likely to solve scale issues, and in fact the domain itself was responsible for many of the scale issues. Succinctly, too many organizations are procuring and fielding too many systems.

A simple survey of available tools revealed the following. There were and are many so-called interoperability “tools” on the market; some are actually useful. Most do not scale well. That is, they perform well for small demonstrations then choke when confronted with communication architectures of the magnitude typical of a U.S. Combatant Command.

In order to perform a meaningful analysis of individual interoperability tools, Captain David Rosen and Major Jennifer Parenti developed a methodology and analyzed the domain against which the tools were to be evaluated. The resulting definitions of the life-cycle architectures: role-centric, system-centric, organization-centric, and mission-centric architectures are important because each reflects the differing perspectives of the varying concepts of operations

(CONOPS) in the domain. Identifying and defining the domain model is the major contribution of this paper. This model does not assume or imply duplicative DoDAF architecture products for each domain. Those with implementation experience with the DoDAF will recognize that the perspective of the architect can result in some very different architectural representations. One failure of architecture tools is that by their very nature those tools limit the representations in a much more rigid manner than specified in the DoDAF itself. This paper makes no attempt to address various combatant commands, service or agency guidance that in some cases imposes additional limits on architect methods and formats.

The results were and are instructive. The research team determined that in many cases, the tools themselves were driving processes. In our view, tools should be developed to support processes, not drive processes. The rather startling result of this study was that the tools themselves are largely irrelevant and that an understanding of the underlying conceptual model of the DoD interoperability domain determines the applicability and usefulness of any interoperability tool. First, we will describe the methodology developed and then move into the domain analysis.

## 2. Goals, Assumptions, and Approach

Based on the broad guidance received from Under Secretary of Defense (AT&L), the research group established three goals for the study:

- 1) Ensure tools are developed to address required functions;
- 2) Avoid redundant development efforts;
- 3) Provide the *right* connections between tools.

Achieving these goals required a rational framework for evaluation and a respect for potential users throughout the department. In other words, the tools had to meet or exceed a rational set of criteria and should be reasonably usable for the DoD workforce. Early experiences with some interoperability tools with horrific user interfaces sensitized the team to the importance of a user-friendly front end.

In order to attain these goals, we applied the process depicted in Figure 1. The two upper blocks show a set of “one-time” conceptual work that was required to reach truly general conclusions. By defining the domain over which interoperability is applied in the DoD, we have laid out the battle space so that we could position tools within it. In order to support this placement, we developed a functional model of the tasks within this domain, against which we could evaluate tools.

Finally, we recognized that you couldn’t evaluate

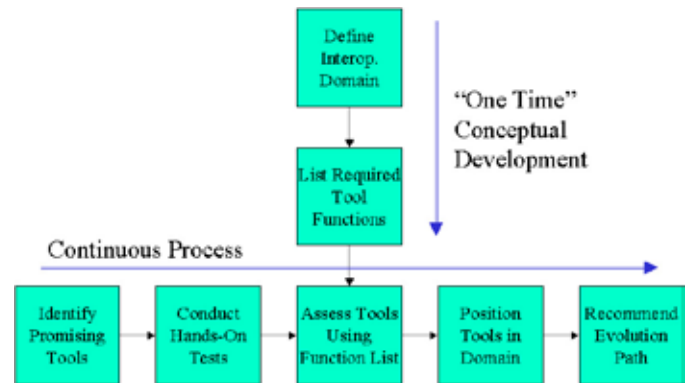


Figure 1. Interoperability tool study process [2]

tools without using them. We also observed that from a tool perspective (as well as other perspectives), architecture and interoperability are strongly related.

The context for evaluation recognized that different users have different needs. For example, tools useful to a staff officer in a service acquisition command may not be useful to an operational planner serving on a Combatant Command staff. Tools and tool users often have agendas, and the research group worked to avoid a “not invented here” syndrome.

Hands-on testing of the three tools was conducted during the development of an as-is command and control architecture for U.S. Alaskan Command [3]. The three tools evaluated were InterPRO, the LISI Inspector Tool and a prototype of JCAPS. JCAPS, the Joint C4ISR Architecture Planning System, was a Windows-based database designed to capture the architecture products from C4ISR Architecture Framework-compliant architectures. (The C4ISR Architecture Framework was the predecessor of today’s DoD Architecture Framework.) Our team implemented the first complete command and control architecture for a joint headquarters. To this author’s knowledge, we have published the *only* detailed cost data for architecture implementation [3]. Documenting 100 command and control systems took just under one engineer man-year (2100 man-hours). Less than one-third of that time was spent using the JCAPS tool. The significance of this was that even with a “perfect tool” which required zero time to load, we still required more than 1400 man-hours to complete the architecture. What we found using the JCAPS prototype was typical of the shortcomings found in most DoDAF architecture tools. The tool forced a particular methodology. Worse, in JCAPS, we could not properly query the data we entered because the database design was flawed and the interface and export features were completely proprietary. The Office of the Assistant Secretary of Defense Command, Control, Communications, and Intelligence (ASD C3I) provided JCAPS to the team.

We also looked at the LISI Inspector Tool. LISI stands for Levels of Information Systems Interoperability [4]. The LISI Inspector Tool should not be confused with the LISI scoring system. Although beyond the scope of this paper, it should be noted that the LISI measurement scale is fundamentally flawed in at least two major respects: first, LISI makes no allowance for requirements. The underlying model assumes (incorrectly) that “more interoperability” is better and therefore a higher LISI score is desirable. Second, the computation of levels is based strictly on an arbitrary scoring system that attempts to measure things that are not mutually commensurate. In spite of these deficiencies, the LISI rating score is being mandated by the JCIDS process [5].

The LISI Inspector Tool is a simple Microsoft Access-based database that is loaded by filling out an online questionnaire about the technical specifications of the communications device. We found the questionnaire captured useful interoperability attributes and stored them in a standard format. Unfortunately, the LISI Inspector Tool is not freely available and requires an exorbitantly expensive contract with the developer.

InterPro is also proprietary, and our team was not able to use it except for a canned demonstration at the Joint Interoperability Test Command (JITC) at Fort Huachuca, Arizona. As noted in [6], InterPro is a Theater Air and Missile Defense-oriented tool. In this regard, we agreed with the JITC’s assessment that InterPro did apply requirements to its mapping of interoperability attributes. However, it was not clear to us that InterPro could successfully be expanded to other mission areas.

The three tools were evaluated through use, but the larger question(s) remained: where and how did these tools fit into the domain of interoperability. A conceptual model was necessary to determine the requirements against which to evaluate the tools.

### 3. Interoperability Domain in the DoD

The Rosen-Parenti model, depicted in Figure 2, is divided into four quadrants. The upper half of the model represents the domain of the planning and acquisition communities. The lower half of the model represents the operational community. The left half of Figure 2 represents the war-fighting Combatant Commands and Joint Task Force (JTF) commanders. Activities that fall in the Joint war-fighting arena will be shown there. The right half of Figure 2 represents the Services and the Defense Agencies. Activities that fall in their areas will be shown there.

Brigadier Tim McKenna, Ph.D., Royal Australian Artillery, observed at the 2001 Pacific Architectures Conference that Americans duplicated joint and



Figure 2. The Rosen-Parenti model domain of interoperability [7]

service staff functions [8]. This observation was obvious to us during the domain analysis. Recognizing that function duplication does exist, the research team used observed that the clear trend since the 1986 Goldwater-Nichols Act has been for the Combatant Commands (COCOMs) to assume more and more of the war-fighting tasks while the Service Staffs focus on providing and supporting forces to the Combatant Commands.

Further confusing matters, there are a large number of players engaged in acquisition activities. Many of the worst interoperability problems come from “rogue” acquisition efforts; that is, home grown systems developed independently from various commands and agencies. For the purposes of this domain analysis, we opted to follow the officially prescribed acquisition process that protects the oft-quoted but rarely understood Title 10 United States Code responsibilities of the services. Among a great many other things, Title 10 empowers the services to man, equip, and train operational forces. System requirements and system funding still originate in the services while increasingly, operational forces are employed in joint commands. *It should be noted that while the Rosen-Parenti model is focused on DoD processes, major acquisitions are heavily influenced by a number of external players from the executive and legislative branches and assorted lobbying organizations. For them most part, these external forces influence the acquisition quadrant, particularly resource allocation. However, external political influences can also affect requirements definition.*

Many problems and most interoperability problems start in requirements. We next map the DoD Joint Capabilities Integration and Development System (JCIDS) process as specified in [5], against

the previously defined domain in Figure 3. A full discussion on the merits of the CJCSI 3170, The Joint Capabilities Integration and Development System, is beyond the scope of this paper.

### 3.1 From ICDs to System Program Offices

We start in the upper left-hand quadrant. This quadrant, the intersection between the Combatant Commands and the Planning and Acquisitions Agencies, represents the Joint Planning Community, which includes the Joint Staff, U.S. Joint Forces Command, the Joint Requirements Oversight Committee (JROC), and other functionally oriented agencies such as the Missile Defense Agency. Here, the requirements process begins with the formulation of capability-based requirements in the form of Joint Capabilities Documents (JCDs) or Initial Capabilities Documents (ICDs). JCDs identify the scenarios against which capabilities and attributes are assessed [5]. Multiple ICDs may be derived from a JCD or the process for a single capability may begin with an ICD. JCDs are used to develop families of related capabilities and therefore have a “one-to-many” relationship with ICDs. One JCD can spawn many ICDs. The ICD defines the capability gap in terms of the functional area(s), the relevant range of military operations, time, obstacles to overcome, and key attributes with appropriate outcome measures of effectiveness, e.g., distance, effect (including scale),

etc. [5]. As ICDs are designed to focus on capabilities instead of specific platforms or services, they easily fall into the realm of joint cooperation, and are rarely service-centric.

Capability requirements flow from the ICDs to the Capability Development Documents (CDDs). CDDs develop the materiel solution, including key performance parameters, for each incremental capability. As such, they straddle the line between the Combatant Commands and the services. A CDD must trace its requirements back to the ICD that it supports. The Capability Production Document (CPD) addresses the production attributes and quantities specific to a single increment of an acquisition program [5]. The sponsor who funds the program—usually one of the services, completes the CPD.

Requirements flow from the CDDs down to the Program Management Offices (PMOs) and System Program Offices (SPOs) for implementation. SPOs are most often organized by system, although there is a recent move to make them more functionally oriented. Likewise, these activities most often fall in the realms of the services, even in the acquisition of joint systems, which usually have a lead service for acquisition. However, we recognize there are exceptions to this model.

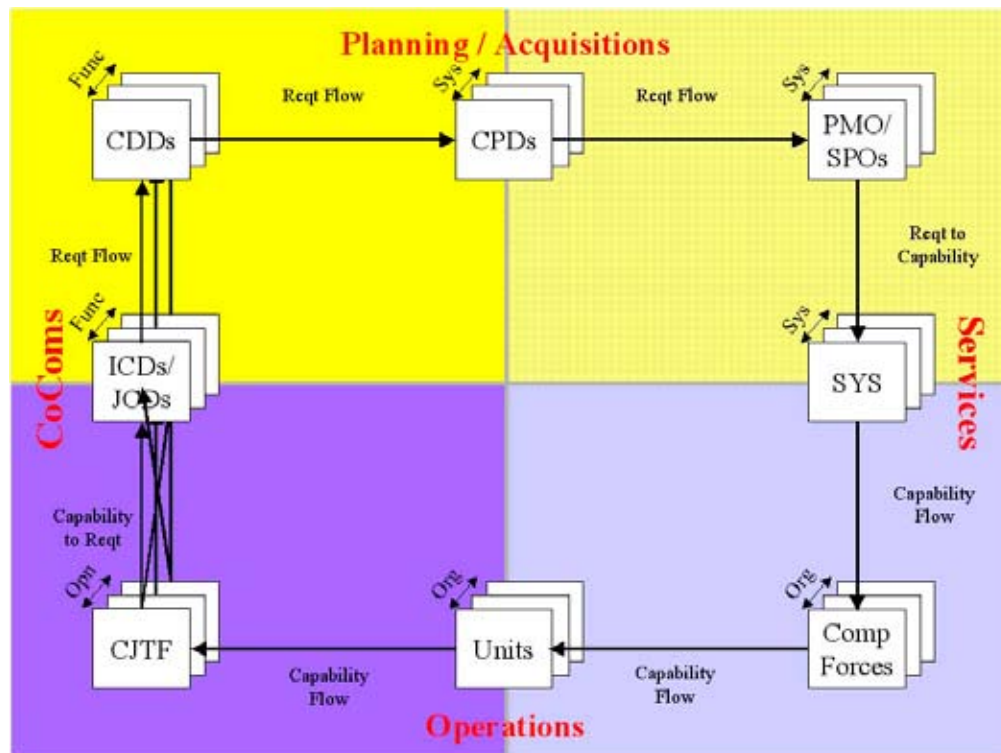


Figure 3. Players and documents mapped to the Rosen-Parenti model domain [7]

### 3.2 From SPOs to Component Forces

The systems that are produced by the SPOs represent the transition from requirement to capability. It also represents the line between the acquisition community and the operational community. As systems are fielded, capability flows from the SPOs and acquisition agencies to our component commanders and forces. Forces are hierarchically organized by command structures, or, rather, along organizational boundaries. This is another “many-to-many” relationship, as systems are integrated into multiple organizations, and each organization will use multiple systems.

### 3.3 From Service Components to Combatant Commands

The transition from our home-based and service-organized forces to our deployed Joint Task Forces occurs through the deployment of individual units, organizationally structured, which come together to form a joint war-fighting force. For example, Battery F, 7<sup>th</sup> Field Artillery, is an Army component unit assigned to the 25<sup>th</sup> Infantry Division (Light) at Schofield Barracks, Hawaii. Battery F could be assigned to either a standing Joint Task Force (most likely) under U.S. Pacific Command or assigned to a specially created JTF to support a specific joint or combined operation.

## 4. Completing the Circle

JTF Commanders can turn capabilities back into requirements through the formation of Joint Capabilities Documents (JCDs), Functional Area Analyses, Functional Needs Analyses, and Functional Solutions Analyses. These, in turn, feed back into the creation of the ICDs. In theory, the flow of capabilities and requirements start and stop in the Combatant Commands. In practice, the cycle is a long one with many delays and a cycle time of many years. A technology-dependent military force cannot keep up with the accelerating rate of technology change.

Ideally, it is the concept of operations (CONOPS) that ties all the functional areas together, from requirements generation to deployment. But there are really many different CONOPS—some that are functionally oriented, some that are system-oriented, some that are organizationally oriented, and some that are oriented toward a particular operation or engagement of force. The relationships between the various concepts of operation are very complex. They must all be consistent with one another—and the intersections between them govern their relationships. We can think of each of them as slices through a “master” CONOPS that covers the entire domain.

In some respects, the operational architecture is the embodiment and implementation of a CONOPS. We will examine how system architecture, in tandem with operational architecture, is used throughout the process to support interoperability. The role of CONOPS and the “slices” of CONOPS will become clearer when we discuss mission-centric and role-centric architectures.

## 5. Architecture in the Rosen-Parenti Domain

This study looked at architectures in terms of the Rosen-Parenti domain and the users of the architecture. Simplistically, we can say that the operational and system views establish *what* systems must connect, and the system and technical views establish *how* systems must connect. For this reason, the study evaluated system and operational views in terms of each of the four quadrants in the domain as shown in Figure 4.

Three architectural views were defined in the early versions of the Army Technical Architecture [9] and later the Joint Technical Architecture [10]. A more detailed treatment of the operational, system, and technical views was set forth in the C4ISR Architecture Framework Document [11] and later the DoD Architecture Framework [12]. A more detailed treatment of these three views is available online in *Joint Command and Control Interoperability: Cutting the Gordian Knot* [3].

### 5.1 Role-Centric Architectures

In the role-centric operational architecture (Figure 5), a functional area is defined in terms of the roles required to perform it and the activities allocated to those roles. Since these activities are the ones that we *plan* to be performed with the roles, we refer to them as

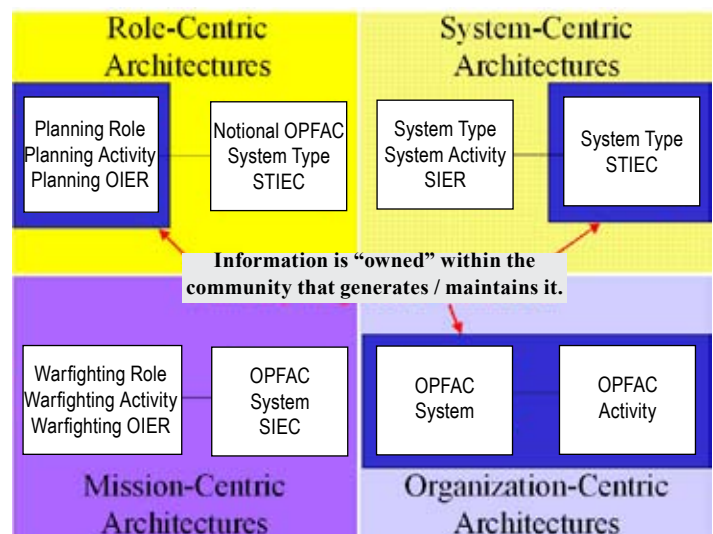


Figure 4. Architectures aligned with quadrants

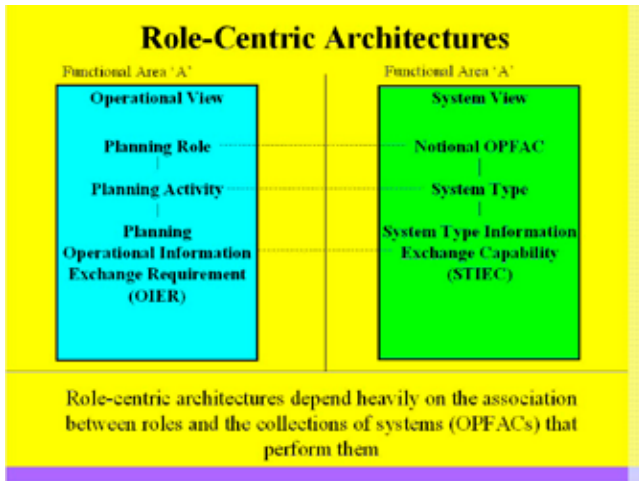


Figure 5. Role-centric architectures [13]

planning activities. Operational Information Exchange Requirements (OIERs) document the required interactions between planning activities. Again, since these are planned exchanges, we refer to them as planning OIERs.

In the role-centric system architecture, notional Operational Facilities (OPFACs), which are manned system-of-systems, are defined within the context of their ability to meet the roles defined in the corresponding role-centric operational architecture (corresponding by functional area). The system types that are part of a notional OPFAC have the ability to exchange information with other systems. We refer to the ability of one system type to exchange a particular piece of information with another system type as a system type information exchange capability (STIEC). This capability is exactly analogous to an information exchange requirement (IER) in that it includes the same information, but represents a *can* rather than a *must*. (It is worth observing, at this point, that in a real implementation, we would expect the STIEC to be associated with information about *how* the information is exchanged, such as the network or protocol used. We omit these details from this discussion for simplicity.)

Notional OPFACs are assigned to the roles that they can fulfill from the operational architecture. Each system type within the OPFAC is bound to the planning activities it will perform. Finally, STIECs can be associated with the planning OIERs they implement in the operational architecture. In the role-centric case, the system architecture is meaningless without the context of the operational architecture. (The purpose of the system architecture is its ability to define systems that can meet the objectives of the operational architecture.)

The role-centric architecture can be viewed as a set of associated bindings between roles and OPFACs

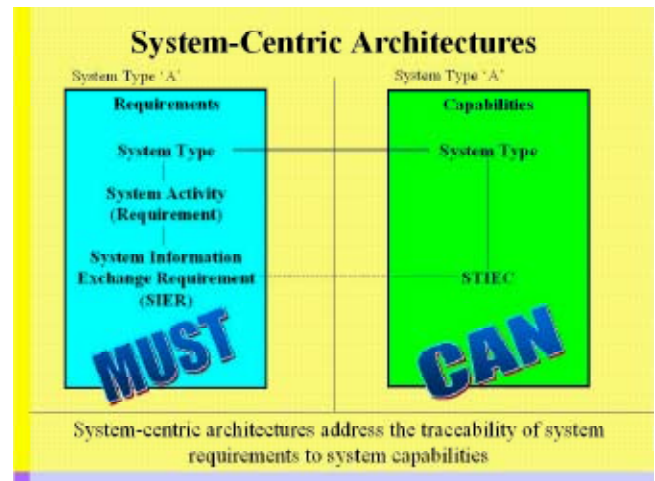


Figure 6. System-centric architectures [13]

(i.e., the operational and system architectures are not really separable; but are rather two views of a larger, integrated architecture).

## 5.2 System-Centric Architectures

Clearly, the system-centric architecture is closely related to the role-centric architecture. The planning OIERs specified in the role-centric architecture need to be allocated to the system types that will perform those roles. Similarly, the system capabilities, which are the responsibility of the service acquisition community, must be used by the Combatant Command planning community in the development of role-centric system architecture shown in Figure 6.

In more detail, we see the requirements architecture showing what a system is required to do, by the parameters of the role-based architecture. The system type has a set of system activities that it is required to perform. Between sets of these activities are system information exchange requirements. These are similar to the planning OIERs, but they are phrased in terms of the need for *system types* to exchange data, whereas in the role-based architectures, the IERs (OIERs) were between *roles*. This implies that functional allocation has occurred, and this is part of the process of developing requirements for system types within the overall system-of-systems context specified at the ICD level.

The capabilities architecture shows what a system type can do, based on the capabilities of the system. It does so in exactly the same way as the role-centric architecture—using system type information exchange capabilities (STIECs). It also provides traceability to the requirements tree represented by the operational architecture.

Combining the requirement and capability views,

we can think of the system-centric architecture as a set of associations between system requirements and capabilities. The system-centric operational architecture defines, to a PMO/SPO, how their system is being employed (or will be expected to be employed) in the field.

The system-centric system architectures, showing system capabilities, must be kept up-to-date—so that planners creating the role-centric architectures are using the best information to assign systems to roles. These architectures, like the role-based architectures, must factor time into account as well. It is important both to define the capabilities of today’s systems and to project the capability expected to be achieved by the systems of tomorrow.

### 5.3 Organization-Centric Architectures

Within the services, organization-centric architectures focus on the specific capabilities and systems within an organizational construct. This can be accomplished at many levels, from the Combatant Commands down to the individual units. The organization-centric architectures capture the capabilities and systems that each organization can contribute to an operation as shown in Figure 7.

Organization-centric architectures focus on the specific capabilities and systems within an organizational construct. Operational Facilities (OPFACs) are identified within the command structure as locations where activities take place. In contrast to the notional OPFACs in the role-centric architectures, these OPFACs represent specific, actual OPFACs.

The operational architecture will identify the activities each OPFAC is capable of performing. Typically, these activities can be mapped to some higher-level UJTL (Uniform Joint Task List) or JMETL

(Joint Mission Essential Task List) task. Since these are activities a unit can perform, we refer to them as unit activities.

The system architecture will identify all the systems located within that OPFAC. The capabilities of those systems will be known, as they are instantiations of (particular instances of) system types in the system-centric and role-centric architectures. From the point of view of interoperability, it is vitally important that these architectures are kept up-to-date, so the JTF planners know a unit’s capabilities when the unit is added to the JTF.

### 5.4 Mission-Centric Architectures

This brings us to the mission-centric architecture, the responsibility of the JTF Commander within the Combatant Commands. Mission-centric architectures focus in on a very specific mission. Choosing functional areas from the role-centric architectures forms them. Then mapping the requirements of these functional areas against OPFACs form the organization-centric architectures. This process of allocating units to functions is clearly the core of JTF formation.

Looking more closely, the mission-centric operational architecture is a compilation of roles, activities, and required information exchanges determined necessary to meet the specific mission at hand as shown in Figure 8. In this case, the roles are specific instances of the generic roles described in the role-centric architecture. To show this difference, we refer to them as mission roles. In the same way, the activities are specific activities required to perform a mission, so we refer to them as mission activities. Similarly, we refer to the OIERS as mission OIERS.

The mission-centric system architecture is a compilation of OPFACs and systems that are available

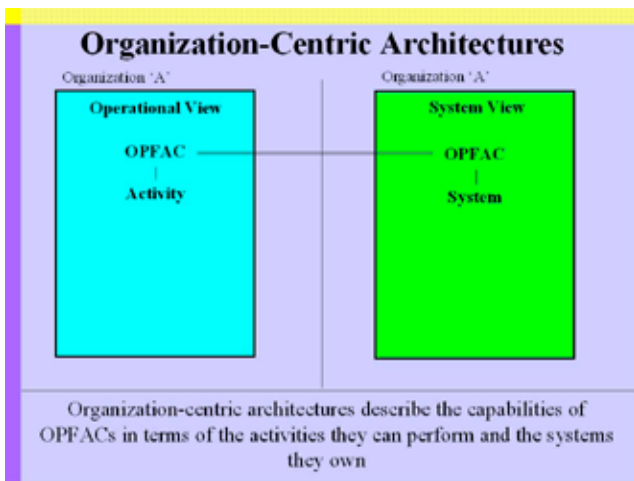


Figure 7. Organization-centric architectures [7]

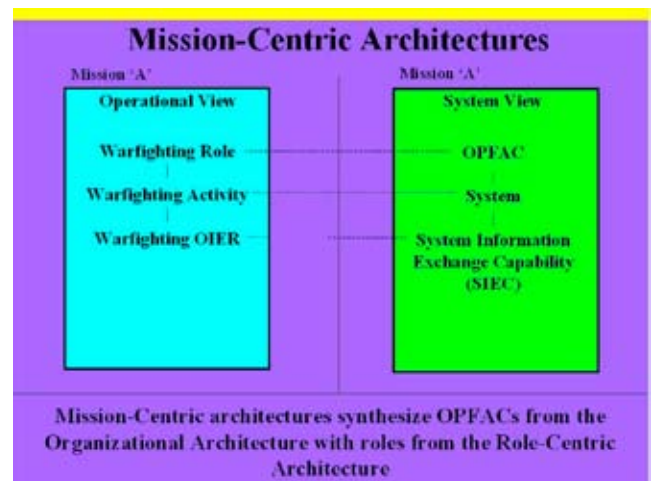


Figure 8. Mission-centric architectures [7]

to fulfill the roles identified in the operational architecture. Based on the system types of the systems, we can identify the system information exchange capabilities (SIECs) as necessary from the STIECs in the role- and system-centric architectures.

The operational and system sides of this architecture are explicitly linked, as a change in one must force a change in the other in order to fulfill the mission. Any disparities between the two must be abated either by adding organizations or systems, or by eliminating

activities. Otherwise, there will be activities and functions assigned on the operational side that cannot be supported on the system side.

The list of system types capable of performing the assigned roles (from the role-centric system architecture) must be validated against the systems owned by the forces (from the organization-centric system architecture) to ensure that these units are properly equipped to fulfill these missions. (An example of a unit that might be capable of fulfilling an activity, but not have the proper systems: a unit capable of providing Close Air Support may be able to fulfill the activity of CAS, but may be inappropriate to a given scenario if it is not equipped with, e.g., the ability to talk to NATO forces.)

If there are any discrepancies in either validation loop (as shown in Figure 9), the mission-centric architecture must be modified to accommodate whatever capabilities are available—this may mean a quick deployment of systems from other units (or acquisition centers), the deployment of different forces from other areas, or may require a change in the mission altogether, if it is determined that we just do not have the capability to support that mission at that given moment.

We can now look across the entire process and see the importance of architecture throughout the system's lifecycle, in all parts of the DoD (Figure 10). We see how architecture is used to determine the interoperability requirements at the functional level in the role-centric

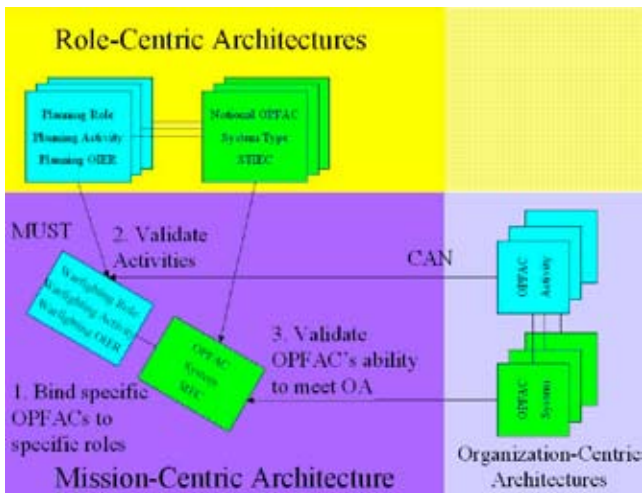


Figure 9. Validating mission-centric architectures [7]

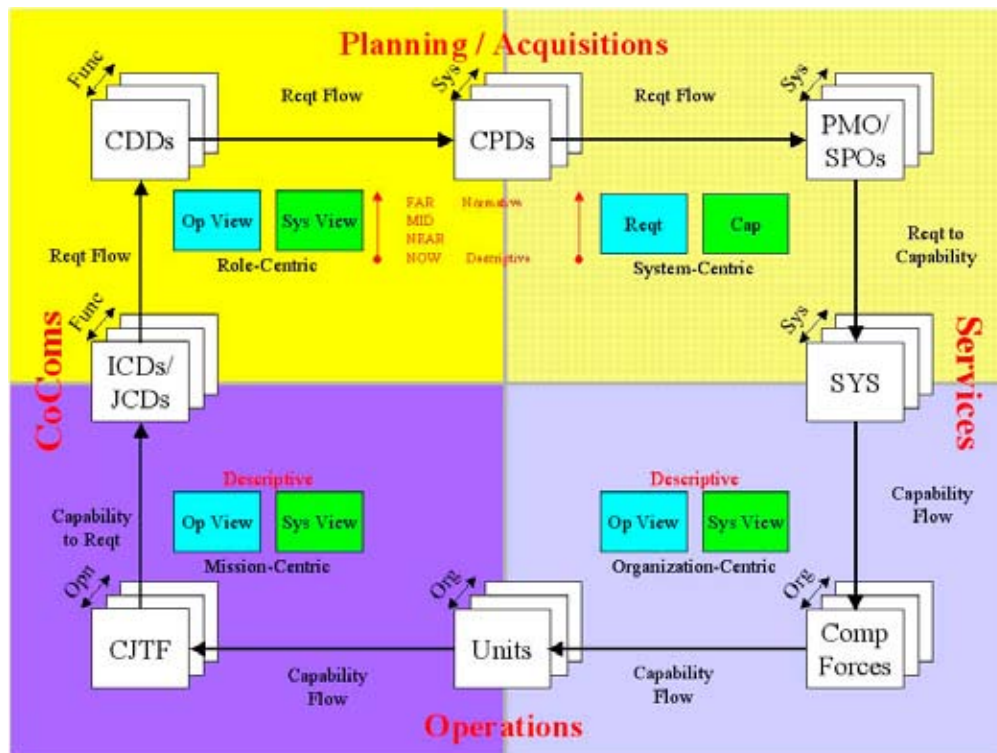


Figure 10. Lifecycle architectures across the Rosen-Parenti domain [7]

architecture; how these requirements are translated into system requirements and implemented in the system-centric architecture; how the locations of the systems throughout the force are maintained in the organization-centric architecture; and how the Combined JTF commander draws on the role-centric and organization-centric architectures to match force with function in a way that is supported by the systems.

## 6. Mapping the Conceptual Model to the DoD Architecture Framework

The DoD Architecture Framework is mandated for use by the DoD [5]. One of the strengths of the DoDAF is that while it prescribes twenty-two product views, the specific design and implementation of those views is not mandated. In actually implementing a DoDAF-compliant architecture, the architects soon recognize the different viewpoints encapsulated in the lifecycle architectures summarized in Figure 10. We can examine the Alaskan Command Architecture effort against each lifecycle architecture.

The DoDAF is composed of twenty-two architecture products in four categories: two all-view products, two technical view products, eleven system view products, and seven operational view products [14]. Succinctly, the all-view products provide an architecture summary and the technical view products provide a listing of standards used to implement each system such as TCP/IP, etc. The role of the technical views and all-views in the lifecycle architectures are consistent and need not be discussed further. The key to successfully applying the lifecycle architectures to a DoDAF implementation is in the system views and the operational views. System views (SVs) can be visualized as the wiring diagrams showing how systems interconnect. Each of the lifecycle architectures can be represented using the prescribed views of the DoDAF. The difference is the viewpoint in which the architecture is rooted. This difference in viewpoint does not imply duplicative architecture products for each viewpoint.

### 6.1 Role-Centric Architecture Applied to the DoDAF

The role-centric architecture begins in the capabilities-oriented joint world as defined in the upper left quadrant of the Rosen-Parenti model. The role-centric architecture emphasizes roles and activities and the logical DoDAF starting point is the OV-5 Operational Activity Model, the OV-4 Organizational Relationships Chart and the OV-2 Operational Node Connectivity Description. Required activities, capabilities really, are first listed and then their relationships defined

in an OV-5 Activity Model. Typically a functional decomposition or flow model is used to represent the relationships between activities. Activities are directly tied to nodes in the OV-2 Operational Node Connectivity Description. A node is a representation of an element of architecture that produces, consumes, or processes data [14]. The OV-2 is the initial crossover point to the system views. Every node in an OV-2 must have a corresponding node in an SV-1, Systems Interface Description.

### 6.2 System-Centric Architecture Applied to the DoDAF

The starting point for a system-centric architecture is the equipment authorization for the entities being modeled such as a Table of Organization and Equipment (TO&E). The system-centric architecture attempts to establish “ground truth” with regard to the systems actually deployed. For a DoDAF-compliant architecture, the starting point is the SV-1 System Interface Description. The SV-1 maps all of the systems and their interfaces to each other. Somewhat problematic is the question of how to deal with unauthorized equipment that is deployed. While this unauthorized equipment should also be mapped, this may be a potential source of confusion when the system views are compared with the operational views. The remaining system views can be built out from the SV-1. In particular, the SV-2, Systems Communications Description; the SV-3, Systems to Systems Matrix; the SV-5, Operational Activity to Systems Traceability Matrix; and the SV-6, Systems Data Exchange Matrix. The system views are aligned with the operational views as shown in Table 1 below.

**Table 1.** System view—operational view crossovers

SV-1 Systems Interface Description	OV-2 Operational Node Connectivity Description
SV-5 Operational Activity to Systems Function Traceability Matrix	OV-5 Operational Activity Model
SV-6 Systems Data Exchange Matrix	OV-3 Operational Information Exchange Matrix

Simply speaking, a DoDAF architecture needs to be internally consistent. The operational views and system views listed in Table 1 are paired products that are directly comparable with each other. While these are not the only places to check for internal consistency, these views are the logical starting point.

### 6.3 Organization-Centric Architecture Applied to the DoDAF

As might be expected, the basis for the development of an organization-centric architecture begins with an Operational View 4 (OV-4), Organizational Relationships Chart. The OV-4 should be built on OPFACs that provide a direct tie to the systems enumerated in the Systems View 1 (SV-1), Systems Interface Description. In an organization-centric architecture, the nodes that provide the needed capabilities of the organization are identified first. The OV-2, Operational Node Connectivity Diagram, is based on the level of detail required and the OV-4. If the organization is U.S. Pacific Command, with nearly 300,000 personnel from all four armed services assigned, the OV-4 will be very complex and most likely, multiple nodes in the OV-2 will be needed to represent each organizational element. As before, the crossover point to the system views occurs in the mapping of the OV-2 to the SV-1 Systems Interface Description.

### 6.4 Mission-Centric Architecture Applied to the DoDAF

The concept of a mission-centric architecture is well established. The DoDAF Deskbook [15] examples specifically use the U.S. Central Command close air support architectures; other examples are based on mission threads such as precision engagement and theater air and missile defense [16]. The mission-centric architecture is in the domain of the joint task force, the joint headquarters that are fielded to accomplish specific tasks and missions. Consider a joint task force organized to provide humanitarian assistance in the Aleutians and a joint task force organized to conduct theater warfare in the Northeast Asia. Both joint task forces may have the same organizations and equipment, but have very different required capabilities. The OV-1 High-Level Operational Concept Graphic can capture the high level mission. The mission-centric architecture starts with the OV-5 Operational Activity Model and the OV-2 Operational Node Connectivity. From there, the development of the remaining views proceeds normally. Two issues to be considered: first, the implication of a mission-centric architecture is that there will be multiple mission-centric architectures for each organization. Second, while many architectural components can be reused in the mission-centric architecture, constructing multiple mission-centric architectures only the significant scalability issues associated with the DoDAF.

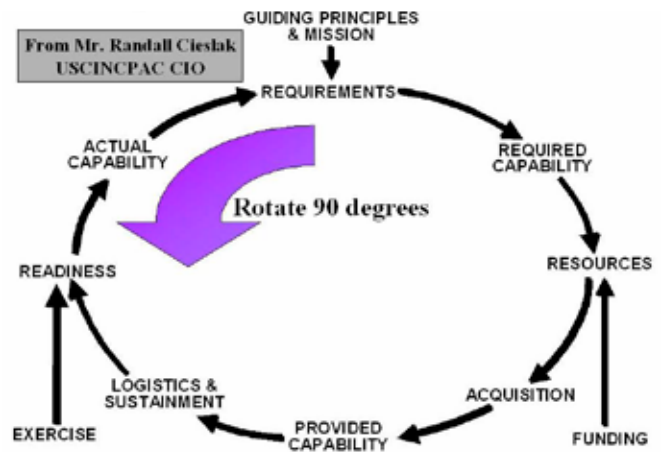


Figure 11. Capabilities information base [17]

### 7. U.S. Pacific Command's Capabilities Information Base

One interesting result of this research project was that the Rosen-Parenti domain was developed independently of some similar work being conducted in the U.S. Pacific Command [17] as shown in Figure 11. This model is remarkably similar to Rosen and Parenti's depiction; the mapping can be found by rotating Mr. Cieslak's graphic 90 degrees counterclockwise and superimposing it on the Rosen-Parenti domain as shown in Figure 12.

As can be seen, the two models are virtually identical. Mr. Cieslak's external arcs are:

- Funding, which feeds into resources, now at the top of the model;
- Exercise, which feeds into readiness, now at the bottom of the model; and
- Guiding principles and mission, which feeds into requirements, now at the left of the model.

### 8. Architecture as a Basis for Simulation

The DoDAF is a useful modeling methodology. The Rosen-Parenti model is a useful model to define the context of a DoDAF or other modeling methodology. However, the DoDAF is essentially a static lay-down that does not easily address performance issues. Real performance analysis of a proposed system is best done using traditional simulation methods. In the DoDAF, reference is made to "executable architectures." An "executable architecture" is defined as the use of dynamic simulation software to evaluate architecture models [14]. The system attributes from a DoDAF-compliant architecture can be used to directly load a network simulation tool thus producing an executable

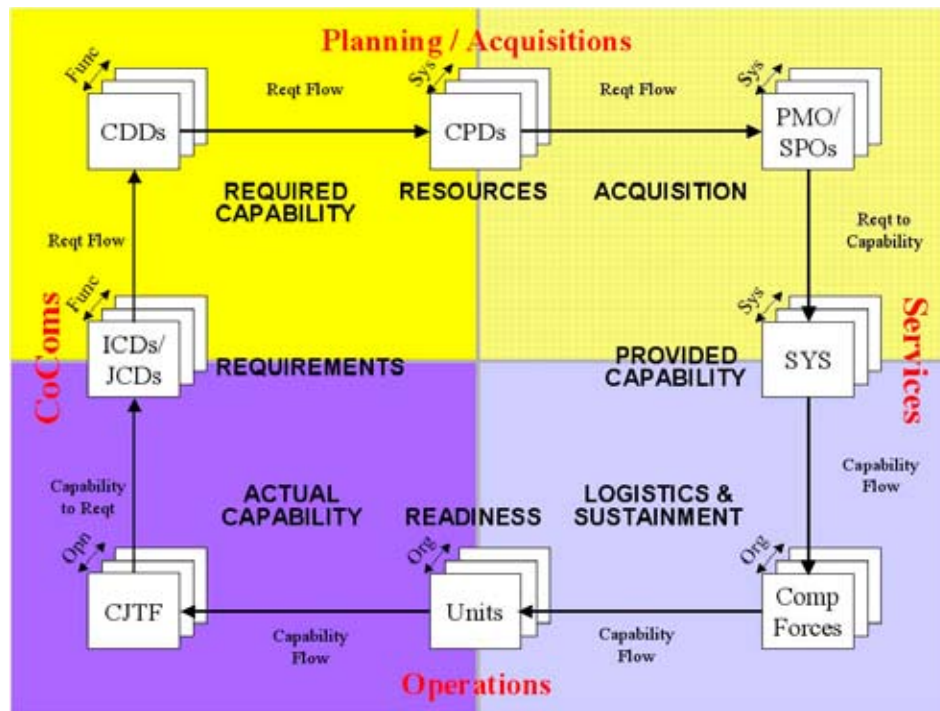


Figure 12. Cieslak's capabilities information base mapped to the Rosen-Parenti model domain [2]

architecture. Such an executable architecture can be used to validate the operational and system views and check the internal self-consistency of the DoDAF-compliant architecture.

A network simulation derived from a DoDAF-compliant architecture has multiple uses. Operational concepts (CONOPS) can be evaluated via the simulation. More importantly, such an executable architecture can address throughput and utilization issues not directly covered in static DoDAF system and operational views.

Not only can DoDAF-compliant executable architectures support operational planning, executable architectures provide the critical third leg needed to validate a DoDAF architecture.

### 9. Validating a DoDAF-Compliant Architecture Through Network Simulation

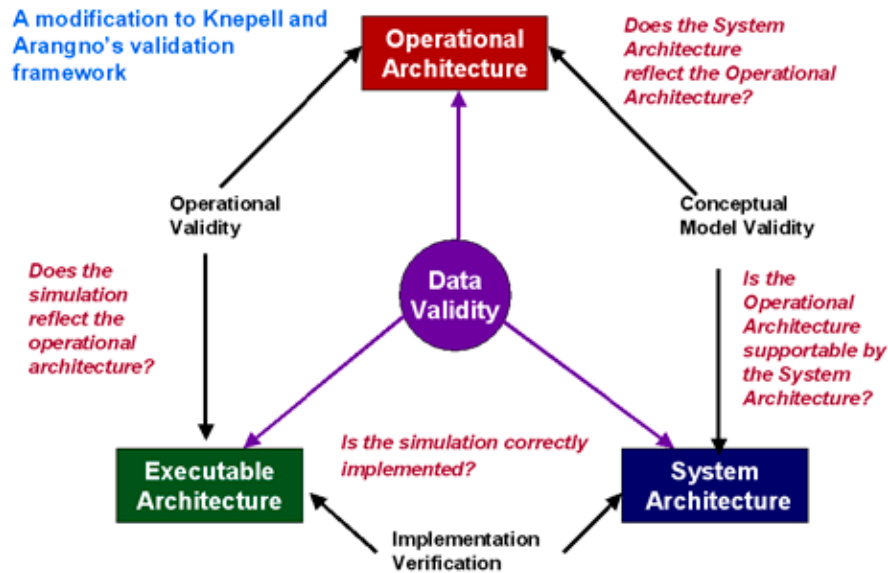
Since the current Defense Acquisition System [18] mandates the use of the DoDAF, developers have strong motivation to demonstrate that their architectures are valid and internally consistent. A model is valid if it can produce the outputs that are equivalent to the ones that would be observed given the same inputs in the environment being modeled. In other words, the model is capable of predicting the behavior of the system being modeled within a specified tolerance. The precise meaning of equivalent inputs and outputs will vary depending upon the simulation.

The term *verification* describes the activity of insuring that a particular implementation faithfully satisfies the requirements of a specification over a given range of inputs. If it is known a priori that a model is valid, then historical data may be used with the understanding that demonstrating a correspondence between program outputs and the physical phenomena being modeled implies the verification of the simulation program in question.

There is a direct relationship between operational views and system views. Figure 13 is a variation of Knepell and Arangno's validation structure [19] adapted for application to the DoDAF.

An operational concept is not valid if it cannot be supported by the systems available in theater. So in this sense, the systems views validate the conceptual model of the operational view as shown in Figure 13. Conversely, the validity of a system architecture can be evaluated against how well it supports the requirements documented in the operational architecture.

The employment of an executable architecture adds a new and needed dimension to the verification and validation of a DoDAF-compliant architecture. Executable architectures can assess the validity of an operational concept. While the system views may provide the needed connectivity to support the operational concept described in the operational views, the system views alone do not give sufficient insight into meeting operational performance and capacity needs. It can be argued that required performance



**Figure 13.** Validating a DoDAF architecture

can be extrapolated from the system views, but an executable architecture can provide a much more dynamic and flexible means of evaluation.

Executable architectures can validate the operational concepts in the operational views by exercising the system architecture with realistic traffic loads. It is straightforward to create traffic scripts based on traffic data collected from similar exercises conducted on the many instrumented ranges in the DoD.

Finally, an executable architecture can verify the implementation of the system views. It is common in the network simulation domain to use the simulation to debug its underlying model and vice versa.

## 10. Conclusions

As we looked more closely at these tools, we realized that the tools could not be considered outside of the context in which they were used. Since interoperability issues are driven by large amounts of data, one had to consider what data was available within a tool, in addition to the functionality of the tool, in order to consider its utility. When looking at long-term utility, it was important to consider how the data was managed and maintained—and, in fact, whether it was likely to be managed and maintained. This could only be done by looking at the way the tool was embedded in the business practices of the tool-using community. Thus, the environment in which the tool is used is as important, perhaps even more important, than the tool itself.

A major finding of this research was the recognition and definition of the four distinct user communities for command and control interoperability tools. Ultimately, to achieve interoperability goals across the DoD, these communities must interact using a process driven by exchange of architectural information, rather than by an exchange of static documents. Although providing tools for this type of collaboration will help, better tools are not enough to achieve this revolution in business affairs. What must be recognized is that all architecture should be based on the needs of the user community [20]. The major contribution of this research is to identify the DoD user communities in the context of the DoD acquisition system—the Rosen-Parenti model.

As with most serious research projects, the “tools study” took many different directions. The long-lasting impact of this study is not the fleeting snapshot of high cost proprietary tools that could be fitted into one of the domain quadrants. Rather, it was the definition of a conceptual model that defines the requirements for interoperability.

First and foremost, serious, unbiased analysis of the domain space demonstrated that lack of tools was not the problem. Rather, lack of a defined domain space was the problem. The Rosen-Parenti model remains a useful means of illustrating the acquisition versus operational and joint versus service push-pulls that make any sort of large-scale interoperability solution very difficult. This project pioneered the concept of user-oriented architectures, recognizing that different organizations and staff use architecture for different purposes. Noting the lack of serious discussion of

modeling and simulation in the DoDAF, we have described a practical means to use architecture to support modeling and simulation and using simulation to verify and validate and architecture. Finally, this paper outlines how different user communities can use a common underlying architecture structure, the DoDAF, for lifecycle architectures relevant to their community.

## 11. References

- [1] Gansler JS. Decision Memorandum and Meeting Summary – Joint Command and Control Integration/Interoperability Group (JC2I2G) In Process Review (IPR) of 22 November 1999, Washington, D.C.; 2000 Jan 3.
- [2] Rosen JD, Parenti JL. Aligning Interoperability Tools within the DoD Battlespace; 2000 Pacific Architectures Conference; 2000 Jun 12–15; Honolulu, Hawaii; 2000.
- [3] Hamilton JA Jr. Command and Control Interoperability: Cutting the Gordian Knot. San Diego, CA: SCS Press; 2004; 33–57.
- [4] C4ISR Architecture Working Group. Levels of Information Systems Interoperability; 1998 Mar 30; Washington, D.C.
- [5] Chairman of the Joint Chiefs of Staff Instruction 2005 May 11. Joint Capabilities Integration and Development System, CJCSI 3170.01E; Washington, D.C.
- [6] Douglas G, Tran P. Joint Interoperability Certification. Program Manager, Sept-Oct 1999; 24 – 27.
- [7] Rosen JD. Aligning Interoperability Tools within the DoD Battlespace (revised); Presented to Dr. V. Garber, Director Interoperability, Under Secretary of Defense (Acquisition, Technology, & Logistics); 2000 Nov 17.
- [8] McKenna T. Architecture, Politics, and the Australian Defense Enterprise. 2001 Pacific Architectures Conference; Pearl Harbor, HI; 2001 Jun 18–22.
- [9] Department of the Army. Joint Technical Architecture – Army Version 5.0; 1997 Sep 11; Washington, D.C.
- [10] JTA Development Group. Department of Defense Joint Technical Architecture Version 3.0; 1999 Nov 15; Washington, D.C.
- [11] C4ISR Architecture Working Group. C4ISR Architecture Framework Version 2.0, 1997 Dec 18, Washington, D. C.
- [12] DoD Architecture Framework Working Group 2004. DoD Architecture Framework Version 1.0; Volume 2 Production Definitions; 2004 Feb 9; Washington, D.C.
- [13] Parenti JL. Engineering Software for Interoperability Through Use of Enterprise Architecture Techniques. DoD Software Technology Conference; 2002 Apr 29 – May 2; Salt Lake City, UT.
- [14] DoD Architecture Framework Working Group 2004. DoD Architecture Framework Version 1.0; Volume 1 Definitions and Guidelines; 2004 Feb 9; Washington, D.C.
- [15] DoD Architecture Framework Working Group 2004. DoD Architecture Framework Version 1.0 Deskbook; 2004 Feb 9; Washington, D.C.
- [16] Dickerson CE, Soules SM, Sabins MR, Charles PH. Using Architectures for Research Development and Acquisition. Office of the Assistant Secretary of Defense, Network Integration and Interoperability; 2004; Washington, D.C.
- [17] Cieslak RC. U.S. Pacific Command Information Capabilities Framework (ICF); 2002 Feb 25; Camp H.M. Smith, Honolulu, HI.
- [18] DoD Instruction 5000.2. Operation of the Defense Acquisition System; 2003 May 12; Washington, D.C.

- [19] Kneppell PL, Arango DC. Simulation Validation. Los Alamitos, CA: IEEE Computer Society Press; 1993.
- [20] Giammarco K, Carlomusto M, Lock JD. Development and Analysis of Integrated C4ISR Architectures. Acquisition Review Journal. Aug – Nov 2005; Defense Acquisition University, Fort Belvoir, VA; 305–317.

## 12. Acronyms

ASD – Assistant Secretary of Defense  
 AT&L – Acquisition, Technology and Logistics  
 C3I – Command, Control, Communications, Intelligence  
 C4ISR – Command, Control, Communications, Computers, Intelligence, Surveillance, Reconnaissance  
 CAS – Close Air Support  
 CDD – Capability Development Document  
 CJCSI – Chairman, Joint Chiefs of Staff Instruction  
 COCOM – Combatant Command  
 CONOPS – Concept of Operations  
 CPD – Capability Production Document  
 DoD – Department of Defense  
 DoDAF – Department of Defense Architecture Framework  
 ICD – Initial Capabilities Document  
 IER – Information Exchange Requirement  
 JCAPS – Joint Capabilities Architecture Planning System  
 JCD – Joint Capabilities Document  
 JCIDS – Joint Capabilities Integration and Development System  
 JITC – Joint Interoperability Test Command  
 JMETL – Joint Mission Essential Task List  
 JROC – Joint Requirements Oversight Council  
 JTF – Joint Task Force  
 LISI – Levels of Information System Interoperability  
 NATO – North Atlantic Treaty Organization  
 OIER – Operational Information Exchange Requirement  
 OPFAC – Operational Facility  
 OV – Operational View  
 PMO – Program Management Office  
 SIEC – System Information Exchange Capabilities  
 SPO – System Program Office  
 STIEC – System Type Information Exchange Requirement  
 SV – System Views  
 UJTTL – Uniform Joint Task List  
 USD – Under Secretary of Defense

## Acknowledgement

The Rosen-Parenti model and graphical illustrations were developed by Captain J. David Rosen, U.S. Air Force Reserve, and Major Jennifer L. Parenti, U.S. Air Force, under the direction of the author as part of a study directed by the Under Secretary of Defense for Acquisition, Technology, and Logistics (USD AT&L). The author fully credits Captain Rosen and Major Parenti for their model development and other intellectual contributions. However, the responsibility for any errors or omissions is the author's alone. The author also thanks the anonymous referees for their very helpful improving comments.

## Author Biography

**John A. "Drew" Hamilton, Jr., Ph.D.**, is an associate professor of computer science and software engineering at Auburn University and is currently president of SCS. He has a B.A. in Journalism from Texas Tech University, an M.S. in Systems Management from the University of Southern California, an M.S. in Computer Science from Vanderbilt University, and a Ph.D. in Computer Science from Texas A&M University. Prior to his retirement from the U.S. Army, he served as the first Director of the Joint Forces Program Office, Director of the Ada Joint Program Office and on the Electrical Engineering & Computer Science Faculty of the United States Military Academy. CRC Press publishes his book, *Distributed Simulation*, written with Lieutenant Colonel David A. Nash and Dr. U. W. Pooch.

The views expressed in this paper are the opinions of the author, and do not reflect the official opinions of any U.S. government agency.