12th ICCRTS

Coalition Command and Control in the Networked Era

Axiomatic Design Approach for Designing Re-Configurable C4ISR Systems

K. Nyamekye, Integrated Activity-Based Simulation Research, Incorporated, Rolla MO 65401

POC: Dr. Kofi Nyamekye, President & CEO
Integrated Activity-Based Simulation Research, Incorporated
Rolla, MO
Tel: 573-647-1702, Fax (US): 1-866-738-7353, Fax (Outside US): 1-573-234-6380

Email: kofinsoyameye@earthlink.net

Topics: C2 Concepts, Theory, And Policy; Modeling and Simulation; C2 Technologies and Systems

ABSTRACT

The paper presents a generic model for designing re-configurable C4ISR systems. The paper recognizes that despite the detailed guidelines provided by the Department of Defense Architecture Framework (DODAF) for designing systems-of-systems (SOS), DODAF may not be appropriate for creating adaptive C4ISR systems. Most importantly, DODAF lacks any theoretical foundation for designing not only C4ISR systems, but also for creating any SOS such as the Department of Defense's (DOD) visionary Global Information Grid (GIG) that requires integration of coalition partners. Using axiomatic theory, the scientific concepts for creating Integrated Manufacturing Production Systems (IMPSs), the value system model, and borrowing from Martin's work [Martin Book II 1990] for creating information-based enterprises, we have constructed the generic model that could serve as a template for designing an adaptive C4ISR. More importantly, the paper recognizes that future DOD systems-of-systems must follow the ***Power to the Edge*** principles, which the DOD's Net-Centric Checklist supports for creating the GIG to operate within the Service-Oriented Architecture (SOA) environment. Such an architectural requirement is critical for achieving the DOD's visionary GIG, which fulfils an adaptive battlefield ecosystem on demand.

INTRODUCTION

In recent years, attention has been focused on designing a next generation fighting force that will be flexible, re-configurable, and lethal, using information systems as the major competitive advantage over an enemy force [Nyamekye et al. June 2004]. According to the Department of Defense (DOD), such architecture is the next generation of integrated Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance (C4ISR) system for the wired battlefield. While such an idea is extremely powerful, especially when fighting a less organized enemy force in an urban warfare environment, several challenges exist for realizing the design and implementation of such an information system grid. Designing such architecture for the wired battlefield requires first understanding the fundamental requirements of the architecture. For example, will the architecture serve the war-fighting operations only, business operations or both? How will it address coalition partners operations during combat and non-combat operations, such as humanitarian operations? Of particular importance is the architecture flexibility, meaning that the architecture must be re-configurable to adapt to dynamically changing battlefield scenarios, especially in urban warfare environment [Nyamekye et al. June 2004; Nyamekye et al. July 2006].

Such a concept is similar to the concepts for designing an ***on demand business***, an idea originally pioneered by International Business Machines (IBM) [IBM May 2003]. According to IBM, on demand business is: "An enterprise whose business processes – integrated end-to-end across the company and with key partners, suppliers and customers – can respond with flexibility and speed to any customer demand, market opportunity or external threat." In this case, the company may be equivalent to the central command center on the battlefield, while the key partners, suppliers, and customers will represent the tactical units, coalition partners, and other non-governmental organizations (NGO) such the news media groups distributed across the battlefield. As a flexible integrated distributed system, on demand business must adapt itself to any external threat such as a terrorist attack on any enterprise in the system. Despite the significant attention that such a system has recently received, the concepts for designing it are still not mature. More importantly, compared to the battlefield, on demand business operates in a less hostile environment. Consequently, current concepts for designing the architecture for on demand businesses are insufficient for designing the architecture for the C4ISR.

Compounding the C4ISR design problems is the issue with the U.S. Department of Defense Architecture Framework (DODAF) [DODAF Version 1, Volume I February 2004; DODAF Version 1, Volume II February 2004]. The DODAF was established to replace the C4ISR and more importantly to address the integration issues associated with many different DOD systems. According to Strassmann [Anthes November 27 2006], the DOD currently has 3,700 systems (excluding intelligence and war-fighting systems), including 174 supporting human resources in the U.S. Navy alone. Strassmann points out that most of those systems were built one at a time and most have their own communications and access methods, interfaces, data definitions and so on. Thus, integrating these separate systems is an enormous challenge. More importantly, these disparate systems are potential sources of cyber attacks – 3,700 points of vulnerability [Anthes November 27 2006]. Previous similar issues gave birth to DODAF. Despite its detailed discussions of the framework that a system's designer must follow to ensure that the designed sub-system would integrate with other DOD systems-of-systems (SOS), DODAF lacks the theoretical basis for any system design. Most importantly, when coupling exists among the functional requirements during the design and operation of C4ISR, DODAF cannot reveal such coupling among the functional requirements. Coupling means that as the design tries to satisfy one functional requirement, other functional requirements are changed also. For example, suppose the goals of designing the logistical support system for the battlefield are: 1) to maximize the service level for the frontline soldiers 2) to minimize the total operational cost. Meeting the first goal of the design should not affect the second goal. If coupling occurs, the design is flawed. The old design must be discarded and a new design created. We will discuss coupling in axiomatic theory later. Consider the following example. Suppose a system designer adopts DODAF for creating an agile sub-system. That is, a sub-system that can adapt to different war-fighting scenarios. By adaptability, we mean that the sub-system could scale-up or scale-down, not just its reusability alone. That is, it can accommodate new operational activities to satisfy new missions if needed. We will discuss operational activities shortly. For each version of the sub-system, the framework requires that the designer must create a separate high-level operational concept describing the business processes or missions, high-level operations, organizations, and geographical distribution of assets. The DODAF refers to such a high-level operational concept, Operational View-1 (OV-1). To describe the operational threads that the sub-system must execute to achieve a war-fighting mission or a business goal, the designer must construct an Operational Activity Model, which DODAF calls the Operational View-5 (OV-5). Furthermore, DODAF requires that each new version of OV-5 traces to its associated OV-1 [DODAF Version 1, Volume II February 2004]. Therefore, if the designer wants to compose a new OV-5 to achieve a new mission, he or she must change the old OV-1 in DODAF's database in order for the OV-1 to agree with the new OV-5. That is, without a new OV-1 coupling will exist between the new OV-5 and the old OV-1. The implication of this deficiency in DODAF is that in today's battlefield environment, especially in an urban setting, the enemy is constantly changing its tactics. Thus, in order for the war-fighter to adapt to the enemy's tactics, he or she must have the flexibility to compose new OV-5 (change the operational thread on the fly) to defeat the enemy, without having to construct a new OV-1 – without having to worry that the new thread will violate the status quo. Because DODAF requires that an OV-5 be specified from an existing OV-1, creating a new OV-1 to match a new mission capability (through a new OV-5) would in turn lead to islands of systems-of-systems. In fact, this was the main issue behind Strassmann's point that the DOD eliminates its stove-piped systems by using Service-Oriented Architecture principles to redesign its SOS. Thus, DODAF may not be an appropriate framework for designing integrated adaptive systems-of-systems for future combat operations. Please note that OV-5 is the key ***product*** for describing a mission's capabilities and relating those capabilities to a mission's accomplishment [DODAF Version 1, Volume II February 2004]. We will explain the term ***product*** in DODAF later.

The importance of designing an agile C4ISR cannot be overemphasized. Alberts et al. [Alberts et al. 2003] have emphasized that the ***Power to the Edge*** principles be used to create new Command and Control (C2) value system. We will define the value system and discuss a brief overview of the ***Power to the Edge*** [Alberts et al. 2003] later. The new C2 value system in turn requires appropriate capabilities from C4ISR systems, Figure 1. (Appendix A shows the list of figures.) Thus, to realize the full potential of the new C2 ecosystem, we must not only construct C4ISR that eliminates stove-piped systems but also we must create the C4ISR that can change as conditions on the battlefield change. More importantly, it must integrate with coalition partners or non-governmental organizations (NGOs) systems-of-systems. Consequently, a new approach is needed for the design of C4ISR. This paper fulfils such a need.

In the subsequent sections, we will discuss the literature survey of the previous approaches for the C4ISR design, an overview of DODAF, a discussion of axiomatic theory, and other design architectures for systems-of-systems (SOS) design. Furthermore, we will discuss the Service-Oriented Architecture (SOA), as the new DOD's vision for creating an ecosystem that can achieve interoperability and agility, such as the DOD's Global Information Grid (GIG). A new generic design approach for constructing an adaptive C4ISR, using axiomatic design and SOA concepts will be provided. We will borrow from Martin's work [Martin Book II 1990] on designing information-based enterprises, partitioning of sets, and Integrated Manufacturing Production Systems (IMPSs) concepts for the C4ISR design. Conclusions will then follow.

LITERATURE REVIEW

Extensive literature review has unearthed few technical publications on C4ISR systems design. Barsoum et al. [Barsoum et al. 2002] have provided the most comprehensive discussion on C4ISR. In a technical publication, the authors discussed the basic architectural design principles that a C4ISR designer must incorporate into designing C4ISR systems. They looked at three domains, namely: C2, ISR (Including Target Acquisition), and Communication. Though they identified and discussed the specific tenets, such as, the Army Joint Tactical Architecture (JTA), Technology Insertion, and System Management Future Tactical Army Systems (FTAS) for each domain and other tenets for the three domains, their focus was only on the detailed discussion of many types of Future Tactical Army Systems (FTAS) C2 tenets. Among them, were: ***C2 on the Move***, ***Graceful Degradation***, ***Robust Information Grid***, and so on. For example, ***C2 on the Move*** implies that the FTAS supports the integrated Units of Action [Barsoum et al. 2002] on the battlefield. That is, the Units of Action may be distributed across the battlefield; yet, they must be integrated into other units on the battlefield. Furthermore, during combat operations, the units must quickly adapt to the enemy's tactics as they engage the enemy at high tempo. Though the authors' work is particularly useful in emphasizing the important tenets for designing the C4ISR, they did not show any scientific basis for creating the C4ISR. Without such a theoretical framework, a designer may not achieve a robust C4ISR design based on the authors' architectural tenets.

As noted before, the DOD constructed DODAF to replace C4ISR [DODAF Version 1, Volume I February 2004; DODAF Version 1, Volume II February 2004], and most importantly to provide the framework for designing an integrated systems-of-systems (SOS). Some background on DODAF is essential before subsequent discussions of DODAF. Three views exist for the framework: Operational View (OV), Systems View (SV), and Technical Standards View (TV), Figure 2. The DODAF consists of twenty-six architecture products, Figure 3. An architecture product is a graphical, textual, and tabular item that describes a given architecture and the characteristics pertinent to that architecture, Figure 3. We should emphasize that the term, *integrated architecture* as specified in DODAF, refers to an architecture description that has integrated OVs, SVs, and TVs. That is, common points of reference exist to link OV and SV and the SV and TV, respectively. For example, the Operational Activity to Systems Functionality Trace-ability Matrix (SV-5) relates operational activities from the Operational Activity Model (OV-5) to system functions from the Systems Functionality Description (SV-4); the SV-4 system functions relate to systems in the Systems Interface Description (SV-1); thus bridging the OV and SV. Thus, an architecture is defined to be an *integrated architecture* when products and their constituent architecture data elements are developed such that architecture data elements defined in one view are the same (i.e., same names, definitions, and values) as architecture data elements referenced in another view [DODAF Version 1, Volume II February 2004]. For simplicity, DODAF uses the term architecture to represent *integrated architecture*. Figure 4 shows the relationship among the various products [DODAF Version 1 February 2004]. Each architectural view is discussed briefly.

The OV describes the tasks and activities, operational elements, and information exchanges required for accomplishing the DOD missions [DODAF Version 1, Volume II February 2004]. Such missions include the war-fighting missions and business processes [DODAF Version 1, Volume II February 2004]. The OV contains graphical and textual products. These products consist of identification of the operational nodes (OV-2, Figures 3 and 4) and elements, assigned tasks and activities, and information flows required between nodes. An operational node is an element, of the operational architecture, for example a Logistics Node, that produces, consumes, or processes information [DODAF Version 1, Volume II February 2004]. In addition, the OV defines the types of information exchanged, the frequency of exchange, which tasks and activities are supported by the information exchanges, and the nature of information exchanges [DODAF Version 1, Volume II February 2004].

The SV describes the physical systems (for example servers, tanks, etc.) and interconnections providing for, or supporting, the DOD functions [DODAF Version 1, Volume II February 2004]. As noted previously, the DOD functions include both war-fighting and business functions. The SV contains graphical and textual products, Figure 3. It associates system resources with OV. These system resources support the operational activities and facilitate the exchange of information among the operational nodes [DODAF Version 1, Volume II February 2004].

The TV is the minimal set rules of governing the arrangement, interaction, and interdependence of system parts or elements [DODAF Version 1, Volume II February 2004], Figure 3. It ensures that a system satisfies a specified set of operational requirements [DODAF Version 1, Volume II February 2004]. The DODAF shows the JTA as an example of TV [DODAF Version 1, Volume II February 2004]. The TV provides the technical systems implementation guidelines upon which engineering specifications are based, common building blocks are established, and product lines are developed [DODAF Version 1, Volume II February 2004]. The TV includes a collection of the technical standards, implementation conventions, standards options, rules, and criteria organized into profile(s) that govern systems and system elements for a given architecture [DODAF Version 1, Volume II February 2004]. It is interesting to note that the SOA addresses all the requirements specified for the JTA.

Of particular importance is how DODAF addresses the Mission Capability Package (MCP), which includes Doctrine, Organization, Training, Materiel, Leadership and Education, Personnel, and Facilities (DOTMLPF). Its integrated architectures with DOTMLPF provide a structured and organized approach for defining the capabilities (through OV-5) and understanding the underlying requirements for achieving those capabilities [DODAF Version 1, Volume II February 2004]. Please note that the MCP describes and discusses how the DOD conducts its war-fighting and business operations [Alberts et al. 1999]. Though DODAF provides very useful detailed guidelines for SOS design, it may not be appropriate for constructing integrated adaptive systems as noted earlier. In fact the DOD's Net-Centric Checklist (NCC) [Net-Centric Checklist May 12 2004] does not even require that DODAF guidelines be used as the blueprint for constructing the Global Information Grid (GIG) [Net-Centric Checklist May 12 2004]. Rather, the NCC suggests that the Service-Oriented Architecture (SOA) principles be used for designing the GIG, with DODAF serving as a reference source [Net-Centric Checklist May 12 2004]. A direct quotation from the NCC attests to this [Net-Centric Checklist May 12 2004]: "*(t)he purpose of the Net-Centric Checklist is to assist program managers in understanding the net-centric attributes that their programs need to implement to move into the net-centric environment as part of a Service-Oriented Architecture in the Global Information Grid. A Service-Oriented Architecture is a design style for building flexible, adaptable distributed-computing environments for the Department of Defense (DOD)*. It is quite interesting to note that the NCC uses the ***Power to the Edge*** logo for the title page of the NCC's document. The ***Power to the Edge*** logo may reinforce the fact that SOA is the main architectural blue print for constructing the GIG that would meet the needs of an ***edge*** organization – ***interoperability*** and ***agility*** [Alberts et al. 2003]. Most importantly, DODAF lacks any theoretical foundation.

Using DODAF, MITRE and Lockheed-Martin [Wisnosky et al. November 2004] have proposed a visionary architecture to create information-based distributed enterprise systems for the Network Centric Warfare (NCW). They call it Activity-Based Methodology (ABM). Despite the deficiencies in DODAF, these investigators remarkably demonstrated that the dynamic Operational Activity Model (OV-5) is essential in showing the detailed dynamic aspects of the individual mission paths of each soldier on the battlefield. Their work strengthens the idea that the success of a future war-fighter's mission on the battlefield could be enhanced if the soldier could autonomously make a decision on which operational leaf activities to execute for him or her to adapt to the changing enemy's tactics. Indeed such a concept is already being practiced on the battlefield [Knowledge Wharton May 17 2006]. An excerpt from Knowledge Wharton puts it this way [Knowledge Wharton May 17 2006]: *"(t)he commander of the Third Army, Lieutenant General R. Steven Whitcomb, summed up the new approach to military discipline: "We are teaching our soldiers how to think rather than what to think." By way of illustration, he noted, his commissioned officers must memorize fewer operating procedures now than in past decades, giving them far greater leeway to improvise and innovate as field conditions evolve and dictate. Lieutenant Colonel Christopher L. Ballard is chief of training at the Udairi base in northern Kuwait that is responsible for preparing those entering Iraq. He said, "We stress decision making by every soldier.""*

As noted before the ***Power to the Edge*** is the most comprehensive visionary concept to date on integrated agile battlefield ecosystem. Alberts et al., the architects of the concept, briefly discuss it as follows [Alberts et al. 2003]: ***"(p)ower to the Edge*** *is about changing the way individuals, organizations, and systems relate to one another and work. Power to the Edge involves the empowerment of individuals at the edge of an organization (where organization interacts with its operating environment to have an impact or effect on that environment) or, in the case of systems, edge devices. Empowerment involves expanding access to information and the elimination of unnecessary constraints. For example, empowerment involves providing access to available information and expertise and the elimination of procedural constraints previously needed to de-conflict elements of the force in the absence of quality information."* They point out that the ***Power to the Edge*** is the yardstick for responding to increased uncertainty, volatility, and complexity in military operations, especially in urban warfare operations. This point is very intriguing. For years, designers of manufacturing and retail value systems have known that to achieve competitive advantage, their value systems must adapt to the changing customer requirements. Thus, if the U.S. manufacturing and retail value systems can use ***Power to the Edge*** principles to achieve integrated systems-of-systems, they can dynamically respond quickly to the customer needs and at low cost, which in turn could give them superior competitive advantage. Indeed, Alberts et al. have emphasized such a requirement for such SOS [Alberts et al. 2003]. Much has been published about adopting the adaptive value system concepts, practiced by some global enterprises, for creating the DOD Network Centric Warfare [Alberts et al. 1999]. In fact, Alberts et al. [Alberts et al. 1999] were the early proponents of such a concept. Other studies too have cited Wal-Mart and Toyota Production System as examples of such adaptive ecosystems [AMR Research April 25 2005; SAP February 9 2007]. Most importantly, the concept of "sense-and-respond or pull system" [Handfield et al. 2002], an idea originally proposed by Bradley et al. [Bradley et al. 1998], at Harvard University, and noted by Alberts et al. [Alberts et al. 1999] as an essential concept for NCW, has been well practiced by Wal-Mart and Toyota Production System for decades. Thus, it is fitting to draw a comparison between military operations and global value systems such as Wal-Mart and Toyota Production System when discussing the ***Power to the Edge***. Figure 1, which shows the value chain model for C2, reinforces this assertion. Despite significant contribution of the ***edge*** concepts to understanding the importance of integrated agile SOS, the authors did not discuss how to design re-configurable C4ISR value systems. A brief discussion of the value system or a supply chain is essential before further discussions.

A supply chain is a collection of many different enterprises (from the raw material supplier to the customer) that collectively work to together to produce and deliver a product to a customer [National Research Council 2000]. The different enterprises are equivalent to the systems-of-systems in GIG. The customer is equivalent to the soldier on the battlefield, and the network is equivalent to the communication and information management among the members in the supply chain. Recently Nyamekye [Nyamekye April 2006] has proposed a new value chain model for predicting the total cost of ownership for a product, process, systems-of-systems based on the four-stage lifecycle model of a product or service [Martin Book II 1990; Smith September 15 2005]. Figure depicts 5 depicts Nyamekye's value chain model [Nyamekye April 2006]. Though his value chain model is more appropriate than Porter's value chain model [Porter 1985] for designing major DOD weapon system and systems-of-systems, and most importantly it is in agreement with the lifecycle value chain model proposed by DODAF [DODAF Version 1, Volume I February 2004] and the recent DOD's emphasis on total cost of ownership for major weapon systems and systems-of-systems [Performance-Based Logistics], Nyamekye's model is an evolving concept. Thus, in this paper we will use Porter's value chain model to describe each enterprise, Figure 6. Future publications will be updated for Nyamekye's model. An enterprise value chain is thus a collection of individual activities that increase the market form or function of the enterprise product or service [Wisconsin Manufacturing Extension Partnership]. The integrated value chains, is the supply chain and it is sometimes called the value system, Figures 7a and 7b. Figure 7a is the value system of a firm or enterprise that participates in a single industry. An industry is a market in which similar or closely related products are sold to buyers [Porter 1985]. The cocoa industry is an example of a single industry -- only cocoa products are sold to buyers. Hershey's value system, for example, participates only in the cocoa industry [Nyamekye July 2006]. Figure 7b shows the value system of a firm or enterprise that participates in many industries. A diversified industry involves a variety of products (not similar) sold to buyers. Cargill's value systems, for example, participate in diversified industries – cocoa industry, soybean industry, and so on [Nyamekye July 2006]. In this paper, we will use the term "supply chain" interchangeably with "value system". Thus, the technical and scientific challenges in designing a re-configurable C4ISR are similar to the technical and scientific challenges in designing an adaptive value system.

The NATO SAS-050 conceptual model (equivalent to the C4ISR, Figure 1) shows an enterprise as a combination of policies and investments that create and distribute the assets available in order to the appropriate capabilities for the C2 [NATO SAS-050 January 2006]. The model does not show the scientific model for designing the enterprise to support the C2.

Zachman [Zachman 1997] has proposed architecture for designing the C4ISR system. Zachman calls the architecture the Framework for Enterprise Architecture. The Framework, as it applies to Enterprises, is simply a logical structure for classifying and organizing the descriptive representations of an Enterprise. According to Zachman, the descriptive representations of an Enterprise are significant to the management of the Enterprise as well as to the development of the Enterprise's systems. The Framework was derived from analogous structures that are found in the older disciplines of Architecture/Construction and Engineering/Manufacturing (ACAEM). The ACAEM classifies and organizes the design artifacts created over the process of designing and producing complex physical products (e.g. buildings or airplanes.) However, Zachman's Framework lacks sound scientific concepts for addressing the complex issues in designing the architecture for the C4ISR system. In addition, Zachman's Framework does not address how to design an integrated adaptive systems-of-systems.

Recognizing that for centuries design has been treated as an art, the National Science Foundation (NSF) funded a research program at the Massachusetts Institute of Technology (MIT) in the early 1980s to establish the scientific basis for design [Suh 1990].

Under a major grant from NSF, Suh and his coworkers conducted a major research program that led to the establishment of axiomatic design theory [Suh 1990]. According to Suh, "design involves a continuous interplay between what we want to achieve and how we want to achieve it." What we want to achieve is the goal of our design, and how we want to achieve it is our physical solution, Figure 8. Suh further explains that we must state the goals of a design in the functional domain or functional space, and generate the physical solution in the physical domain or physical space, Figure 8. The design procedure then involves interlinking these two domains at every hierarchical level of the design process. The two domains are independent of each other. What relates these two domains is the design.

To begin any design, we must determine the design's objectives by defining it in terms of specific requirements, called the functional requirements (FRs). Then, to satisfy these functional requirements, we must create the design solution in terms of design parameters (DPs). The design process involves relating these FRs of the functional domain to the DPs of the physical domain, Figure 8. Suh established two fundamental axioms that form the scientific basis of the axiomatic approach to design. They are:

AXIOM 1: In a good design, the independence of functional requirements (FRs) is maintained.

AXIOM 2: The design that has the minimum information content is the optimal design.

AXIOM 1 simply states that in designing any system, we must meet the goals (strategic or tactical requirements) of the system independently. For example, suppose the goals of designing an information visualization system are: 1) maximize the information benefits per unit cost and 2) minimize the total operational cost. According to AXIOM 1, the final design must satisfy both goals independently. Meeting the first goal should not affect the second goal. AXIOM 2 says that among the different designs that will meet both goals, the design that will require the least amount of information to describe it or will achieve the highest reliability of the system will be the best design. AXIOM 2 establishes the theoretical foundation for an optimum design of a product, process or a system, for example software, organization and so on. We should note that classical optimization models, from operation research field, do not generally yield optimum results when more than one criterion for which the system must be optimized, exist [Nakazawa and Suh 1984]. For example, when the goals of designing logistics system, are both maximizing customer service and minimizing the distribution costs, classical optimization models do not achieve optimum results. Consequently, axiomatic approach is superior to the traditional optimization techniques when the design must meet more than one goal, concurrently. Furthermore, we can use axiomatic design to evaluate an existing design for improvements. We will discuss more details about axiomatic theory for creating the C4ISR later.

In addition to the functional requirements, a set of constraints may also exist. Constraints are factors that establish the boundary on acceptable design solutions. For example, some designers treat cost as a constraint. On the battlefield, how much collateral damage, and how many casualties are "acceptable" in a theater operation, could represent the constraints [Alberts et al. 2003]. Constraints are very similar to functional requirements in character and attributes except that the independence of constraints is not required in a good design.

Using axiomatic design for Design for Six Sigma (DFSS), Nyamekye has also recently noted that an enterprise could achieve a superior competitive advantage if it could use distributed enterprise simulation model to optimize the design and operation of its value system before committing substantial dollars to build the systems-of-systems [Nyamekye February 6-7 2006]. When combined with the previous work of Nyamekye on Activity-Based Simulation (ABS) [Nyamekye March 2000] for designing supply chains, the DFSS could be useful for designing re-configurable C4ISR. In fact, Nyamekye's emphasis on activity-model, as the basic construct of ABS, is in agreement with MITRE and Lockheed-Martin Activity-Based Methodology [Wisnosky et al. November 2004] that also used the leaf operational activity-based model for the ABM.

Figure 9 shows the Future Combat Systems (FCS) "V" model proposed by Krone [Krone May 2004]. Figure 9 is the framework for designing the re-configurable service-based simulation architectures, for the Boeing FCS. The acronym SDD represents System Development and Demonstration in Figure 9. Though Krone's work is useful for discussing the design of systems-of-systems such as the Army Future Combat Systems (FCS), he did not provide any theoretical background for his work.

Enterprise engineering models have been proposed for designing system architectures [National Research Council 2002]. Enterprise engineering is a set of activities that deal with designing and redesigning business entities, for example industrial systems, administrative systems, or service systems [Vernadat 1996]. According to the National Research Council, enterprise engineering goes through the following stages: business entity identification, business entity conceptualization and definition; requirements definition; design specification; implementation description; building and testing; and release of the system to operation. The National Research Council further points out that during operation, the business entity must go through performance evaluation, change management, and continuous process improvement. Using enterprise-engineering concepts, the enterprise integration community has established the GERAM (Generalized Enterprise Reference Architecture and Methodology) as the framework for enterprise integration.

The GERAM uses the CIMOSA (computer-integrated manufacturing open system architecture). The National Research Council emphasizes that CIMOSA can be used for understanding the interrelationships among enterprise activities, application domains, and industries, and more importantly, it can be used as the framework/architecture for integrating different simulation models in different domains. The GERAM and CIMOSA lack theoretical foundation. More importantly, GERAM continuous process improvement model uses the business process reengineering architecture by Hammer et al. [Hammer et al. 2001], which also lacks any scientific basis especially for designing systems-of-systems.

The CMMI (Capability Maturity Model Integrator), developed by Carnegie Mellon University, can be used to improve and appraise the performance development of organizations [CMMI 2003]. No scientific basis exists for it.

The most comprehensive work to date on designing distributed enterprise systems is by National Institute of Standards and Technology (NIST) [IMS MISSION]. In recognition of importance of global supply chains on the U.S. economy, NIST formed an International Consortium of Research Scientists called the IMS MISSION in 1999 [IMS MISSION] to study the manufacturing supply chains. In fact, the author of this paper was a member of the IMS MISSION. The goal of the MISSION was to integrate and utilize new, knowledge-aware, technologies of distributed persistent data management, as well as conventional methods and tools in various enterprise domains, to meet the needs of globally distributed modeling and simulation. Because a supply chain is a distributed enterprise system, NIST chose it as a test bed for the research. Using DOD's High Level Architecture and Runtime Infrastructure (HLA RTI), NIST designed distributed enterprise model and simulation. Each member of the supply chain was modeled in a PC, using a different simulating platform. For example, PROMODEL, a simulation platform, modeled the activities of a logistics supplier. Another simulation platform modeled an enterprise channel value chain. Though NIST did not optimize the value system, their study has provided much understanding of the complexities in designing and operating a value system. Furthermore, NIST's work did not decompose the simulation processes into leaf activities (or atomic operational activities) for creating the stateless or uncoupled services, an essential ingredient for the Service-Oriented Architecture (SOA) [Arsanjani November 9 2004; Zimmermann et al. June 2 2004] which the DOD has recommended for designing SOS. This issue is important for creating complex systems from composition (choreography or orchestration) of leaf or fined-grained services or atomic activities. As noted before, a leaf activity forms the basic activity-model of MITRE and Lockheed-Martin Activity-Based Methodology [Wisnosky et al. November 2004] and Nyamekye's Activity-Based Simulation [Nyamekye March 2000] effort, respectively.

Though the previous studies have provided some insights into the design of value systems, none has established the scientific concepts for expressing the design of value systems as concurrent interactive subsystems (value chains), which may evolve and die dynamically, as they adapt to their environments. As noted before, such a concept is known as complex adaptive systems [Nyamekye November 2006]. Activities in the value systems interact concurrently and in certain order of sequences, and may evolve and die dynamically, in response to the dynamic changes of their environment. For example, enterprises continuously mix and match their strategic activities that may run sequentially or concurrently throughout the value system, in response to the changing market conditions [Porter 1985]. That is, they look for ways to choose activities for creating linkages to consistently achieve superior performance [Porter 1985]. On the battlefield, such a scenario would be equivalent to achieving superior competitive advantage over the enemy force. Thus, activities may move around (e.g., moving logistics activities around on the battlefield) and may die (e.g., end of the life cycle of activity instance) entirely after they achieve their desired overall system (integrated systems-of-systems) goals, under certain dynamic battlefield or market conditions. Expressing such concurrent systems dynamically during design of SOS is extremely challenging. Until the birth of mobile computing (or mobile activities), the theoretical concepts for expressing the design of concurrent systems never existed. Thanks to the work of Milner [Milner 2004], at Cambridge University, we can now take any combination of activities and create linkages, "on the fly", to achieve the desired performance. We can theoretically express the dynamic behavior of instances of strategic activities as they evolve and die. Using information science, Milner has established the theoretical concepts known as pi-calculus for expressing how activities could be combined to interact concurrently and in certain sequences in any systems-of-systems. With such a theoretical framework, a commander on the battlefield could choose the strategic value activities and link them together in a certain order either sequentially or concurrently with other strategic value activities in the value system to determine which combination of strategic activities could achieve the best performance—defeat the enemy force. Should the enemy change its tactics, the commander on the battlefield could choose different strategic value activities to achieve the desired performance. Note that a business process or process is a sequence of two or more activities that serve a purpose or goal for an enterprise [Havey 2005]. A business process or simply process is sometimes known as compound activities [Havey 2005]. Milner's work has also provided the theoretical basis for expressing the design of complex adaptive systems. Though Milner's work provides the theoretical basis for concurrent interactive systems, his work did not show how to actually design the systems-of-systems.

Wil van der Aalst et al. [Wil van der Aalst et al. 2004], at Technical University of Netherlands, have established the design patterns for actually designing concurrent systems. Though their effort augments Milner's work, Wil van der Aalst et al.'s work was only confined to simple systems. That is, they did not address systems-of-systems. An interesting point to note here is that in both efforts, activity was the basic construct of their work. However, to be useful in systems-of-systems design, Milner's work and Wil van der Aalst et al. design patterns must compliment axiomatic design. Today *process theory* [Fokkink

2000; Havey 2005] owes its scientific basis from Milner and Wil van der Aalst et al. efforts, respectively. More importantly, the proponents of Service-Oriented Architecture (SOA) argue that Milner's work was the scientific basis for the SOA [Havey 2005]. We will discuss SOA later.

Agents modeling have received much attention by many computer scientists as the theoretical framework to build agile information-based systems-of-systems, similar in concept to Milner's work. Wooldridge [Wooldridge 2003] defines an agent as follows: *"(an) agent is computer-based system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives*." Because many value systems-of-systems are global, they use system's infrastructure such as networks, servers, an Infosphere (information management system for SOS) [Nyamekye et al. July 2006] to integrate all the various value chains. Each sub-system in the infrastructure is prone to attacks. Thus, we need an intelligent agent as a sensor in each sub-system to gather inputs from the sub-system's environment, and produce as output actions that would enable the sub-system to achieve its local goals. Of particular importance is distributed enterprise simulation, where different sub-systems must be simulated. When any sub-system fails during a simulation run, un-predictable results will occur. Thus, we must ensure that each sub-system has agent(s) that could remedy any failures. Thus, agents modeling must be part of any distributed enterprise simulation model. More importantly, many global value chains that are using SOA as their standard architecture propose intelligent systems for an SOS to achieve a high reliability [Murch 2004]. However, potential problems exist in managing and coordinating the communication among the mobile agents in complex adaptive systems [Wooldridge 2003; Kleinrock July 4 2004]. As Kleinrock, at University of California-Los Angeles, puts it [Anthes July 4 2005]: *"(w)e once arrogantly thought that any man-made system could be completely understood, because we created it. But we have reached the point where we can't predict how the systems we design will perform, and it's inhibiting our ability to do some interesting system designs. We are allowing distributed control and intelligent agents to govern the way these systems behave. But that has its own dangers; there are cascading failures and dependencies we don't understand in these automatic protective mechanisms."* Thus, we must use creative ways to integrate the design of mobile agents infrastructure within the C4ISR.

As discussed earlier, the DOD has adopted the Service-Oriented Architecture (SOA) as the key architecture for designing the Global Information Grid, Net-Centric Checklist (NCC) [Net-Centric Checklist May 12 2004]. Thus, a discussion of the SOA is essential, not only for constructing GIG but also for creating re-configurable C4ISR. Despite extensive publications that have appeared on SOA in recent years, few have neatly focused on discussing it from the viewpoint of modeling a large-scale SOS. Among them are the notable publications by Arsanjani [Arsanjani November 9 2004] and Zimmermann et al. [Zimmermann et al. June 2 2004]. This paper follows the concepts recommended by both authors. Arsanjani shows the conceptual model of SOA as follows, Figure 10 [Arsanjani November 9 2004]:

- *A service provider, who publishes a service description and provides the implementation for the service,*
- *A service consumer, who can either use the uniform resource identifier (URI) for the service description directly or can find the service description in a service registry and bind and invoke the service, and*
- *A service broker, who provides and maintains the service registry.*

The basic purpose of SOA is to allow an enterprise to be adaptive *on demand* to dynamic market conditions, thereby achieving business goals through the composition of reusable services [Zimmermann et al. June 2 2004]. Furthermore, it permits integration of heterogeneous applications, operating systems, and so on, regardless of the platforms or language [Zimmermann et al. June 2 2004]. Interoperability is another major strength of SOA. Thus, it is an ideal architecture for creating large-scale SOS, such as global manufacturing and retail value systems, and of course C4ISR systems. Zimmermann discusses three levels of abstraction within SOA as follows [Zimmermann et al. June 2 2004]:

- *Operations:* They are operational leaf activities (or atomic activities) that represent single logical units of work. An execution of an activity will typically cause one or more persistent data records to be read, written, or modified. Consider activities for *Inventory Control* as a *Service*. The operations for this *Service* would be: *Lookup customer by telephone number, List customers by name and postal code*, and *Save data for new customer*. Please note that the term "leaf activity" is identical to the term used by MITRE and Lockheed-Martin ABM and Nyamekye's ABS, respectively.

- *Services:* They are logical groupings of operational leaf activities. For example, the *Inventory Control* as a *Service* that we discussed in the *Operations*.
- *Business Processes:* A long running set of *chained activities* performed to achieve some specific business goals. Business processes typically consists of multiple service invocations. Examples of business processes are: *Initiate New Employee*, *Sell Products or Services*, and *Fulfill Order*. Please note that the term, *chained activities,* is identical to OV-5 in DODAF, for achieving the goals of a DOD mission. We should emphasize that a business process consists of a series of operational leaf activities, which are executed in an ordered sequence according to a set of business rules. The sequencing, selection, and execution of operations, is termed a service, process *choreography* or *orchestration* [Arsanjani November 9 2004]. As mentioned before, choreographed or orchestrated services are invoked in respond to business events.

Figure 11 shows an abstract view of SOA, proposed by Arsanjani [Arsanjani November 9 2004]. The *Operational systems layer* consists of existing applications of the enterprise, for example, CRM and ERP packaged applications, or legacy systems. Through classification or logical groupings of leaf activities to form the composite services, the applications that support such composite services could also be grouped into appropriate enterprise components [Arsanjani November 9 2004]. The *Service components layer* is a set of enterprise assets, such as CRM solution, that fulfills the goals of a particular business unit, for example customer service unit [Arsanjani November 9 2004]. The *Services atomic or composite layer* comprises the services (atomic or leaf, or composite service) the business exposes to the service consumer or hides from the service consumer [Arsanjani November 9 2004]. The *Business process composition; choreography, business state machines layer* defines the compositions and choreographies of services exposed from the *Services layer* [Arsanjani November 9 2004]. A service consumer bundles the services into a workflow model or business process model through orchestration or choreography. The workflow model may consist of several integrated applications to support business processes. We call modeling of business processes Business Process Modeling or Business Process Management (BPM) [Havey 2005]. Generic Modeling Environment (GME) is an example of a modeling language [GME] for designing large-scale DOD SOS [Schmidt February 2006]. As noted before Milner and Wil van der Aalst et al. have done extensive work in process modeling for workflow management. The *Consumers layer* is the users interface for leveraging the services. Please note that SOA decouples the users interface from the services in the *Service components layer*. Besides aligning the information services with business goals, decoupling the user interface from the service provider is also another powerful reason for the SOA's adoption. Most importantly, AXIOM 1 is the scientific base for this decoupling concept. We will discuss decoupling shortly. Therefore, an SOS designer using an SOA template must provide an end-to-end solution from an access channel to a service or composition of services [Zimmermann et al. 2005]. The *Integration (enterprise service bus) layer* enables the integration of services through the introduction of a reliable set of capabilities, such as intelligent routing, protocol mediation, and other transformation mechanisms, often described as the Enterprise Service Bus (ESB) [Arsanjani November 9 2004]. The ESB also owes its concept from AXIOM 1. We will provide more discussion on ESB shortly. The *quality of service (QoS) [security, management and monitoring infrastructure services) layer* provides the capabilities required to monitor, manage, and maintain QoS such as security, performance, and availability [Arsanjani November 9 2004]. The *Data Architecture (meta-data) & business intelligence layer* defines the data modeling architecture, meta-data for the leaf activities or fine-grained services, and the business process management (BPM). The *Governance layer* is responsible for the governance of the SOA infrastructure for the SOS. That is, it is the system of governance mechanisms that ensure business and information systems projects achieve local level, enterprise level, and across the enterprise value system's objectives [Ross et al. 2006].

Proponents of SOA have argued that when an enterprise designs the value systems into logical groupings of uncoupled leaf activities, the enterprise then can create fine-grained services or simply services, expose such services to consumers for composition of services, regardless of the geographic location of the service consumer [Zimmermann et al. June 2 2004; Arsanjani November 9 2004]. The enterprise or service provider can use any operating system to host the services or create the services from any language independently of how the service consumer uses the services to achieve its business goals. That is, the service provider decouples the services from the service consumer. This concept, known as decoupling, is the most powerful argument behind the SOA's adoption. Intriguingly, it stems from Corollary 1 of AXIOM 1 as follows: *Corollary 1 (Decoupling of Coupled Designs): Decouple or separate parts or aspects of a solution if the FRs are coupled or become interdependent in the proposed design.* The implication of Corollary 1 is that in designing a part, software as a service, or a process, we must ensure the proposed design serves the goals independently. Thus, a service consumer, such as the server of an enterprise, can use a service created by another enterprise to achieve the business goals of the service consumer's enterprise independently of the language which the service provider chooses to create that service to serve the service provider's own business needs. Another caveat stems from Corollary 1. That is, when we design the value system of the SOS into uncoupled or stateless leaf activities or fine-grained services, we can use the same fine-grained services to compose new coarse-grained services (new OV-5s), without having to re-construct any part or the value system itself (no new OV-1s). That is, the OV-5 will not couple the OV-1, a major deficiency in DODAF. By stateless, we mean that each the leaf activity exists autonomously from each other. That is, we can execute each leaf activity independently of each other. Only when we choreograph or orchestrate them into services for a workflow model, will the execution of the downstream leaf activities be dependent on the execution of upstream leaf activities.

Furthermore, the proponents of SOA use the Enterprise Service Bus (ESB) to integrate the heterogeneous services in SOS ecosystem. Corollary 3 of AXIOM 1 and AXIOM 2 forms the theoretical concept for ESB [Suh 1990]*: Corollary 3 (Integration of Physical Parts): Integrate design features into a single physical process, device, or system when FRs can be independently satisfied in the proposed solution*. Corollary 3 states that the number of physical processes, devices, or systems should be reduced in order to decrease information content or eliminate coupling of FRs. Heterogeneous services exist in distributed SOS. To integrate such services, Corollary 3 says that we should not design one interface for each service. Rather we can create one system, ESB, to integrate these different services to fulfill a variety of business goals. Figure 12 is an example of an ESB configuration. Lublinsky has also recently noted the consequence of coupling in SOA [Lublinsky January 9 2007]. He emphasized that if we want to use services to build, maintain, and modify flexible enterprise solutions and to support a federated development approach throughout an enterprise, then loose coupling or decoupling of those services must be a prerequisite for designing the SOS.

Obviously, previous efforts show that we need a new approach with sound theoretical foundation to create a re-configurable C4ISR. Compounding such an approach is that the SOA, which has sound theoretical base for creating SOS, is still an evolving concept. Using axiomatic theory, the scientific concepts for creating Integrated Manufacturing Production Systems (IMPSs), and borrowing from Martin's work for creating information-based enterprises, we will discuss the generic design model for the C4ISR. The next section attempts to provide such a journey.

GENERIC DESIGN APPROACH FOR C4ISR

For years, manufacturing research scientists have found that the first step in designing agile Integrated Manufacturing Production Systems (IMPSs) [Black J T. 1991; Nyamekye July 28 2005], is first to transform the enterprise into logical functional groupings of processes that could form the subsystems of a manufacturing enterprise. In IMPSs, we call such a sub-system a manufacturing cell [Black J T. 1991]. The processes in each logical subsystem share some common property such as a group of common databases or entity types -- the processes use similar databases, or entity types, such as the part routing entity type. By logical functional groupings, we mean that natural business areas that are independent of any specific traditional department, such as accounting department, lathe or turning operations department, and so on. The researchers argued that by creating such natural logical groupings, redundant or stove-piped processes could be combined. Redundant processes require extra resources to execute the processes. Such resources may include expensive computer numerical control (CNC) machine tools, direct labor, and expensive CNC application programs to automate the machines. This could reduce the operating costs. Lower operating costs would generate business that is more profitable. Furthermore, statistical process control (from Design for Six Sigma (DFSS) concepts) could be applied to the processes to remove the sources of variability, such as setups of the processes, reworks, and so on [Nyamekye February 6-7 2006]. Eliminating the sources of variability within each process will enable the process to quickly respond to the needs of the customer, or in simple terms make the process become agile. When the natural subsystems are integrated with management information systems and mobile agents, we can then truly achieve agile manufacturing enterprises. In fact, such a concept is the basis for creating the Toyota Production System [Monden 1983]. We should emphasize that in Toyota Production System each worker, on the shop floor, is a mobile agent. He or she periodically checks the health of each machine in his or her own production line as well as the health status of other production lines through information transfer boards, positioned between successive production lines. When a machine breaks down, the worker publishes the problem to all workers on the shop floor by turning on a switch on an adjacent information transfer board that broadcasts the problem. Upon seeing a particular production line with the problem, other workers leave their production lines to assist the worker to fix the problem. The workers return to their respective production lines after helping the particular worker to fix his or her problem. Figure 13 is an example of distributed intelligent control systems simulation model for an unmanned manufacturing cell that integrates into Manufacturing Automation Protocol/Technical and Office Protocol (MAP/TOP) infrastructure [Chang and Nyamekye 1993]. We should emphasize that MAP/TOP [Jones 1988], originally envisioned by Boeing and General Motors in the early 1980s as the information technology architecture for integrating distributed manufacturing enterprise systems or Computer-Integrated Manufacturing (CIM), never materialized [Black J T. 1991]. The proponents of MAP/TOP quickly realized that redesigning the manufacturing enterprise into cellular manufacturing and integrating quality control to consistently achieve superior quality products must be addressed first before implementing any information technology [Black J T. 1991].

Martin [Martin Book I 1989; Martin Book II 1990; Martin Book III 1990;], the godfather of Information Engineering, also recognized the importance of grouping processes and entity types (data objects) into natural information systems as the first step in creating information-based enterprises. Porter [Porter 1985], the pioneer of the value chain model, also noted the significance of clustering activities into logical natural subsystems in a value chain. It is quite interesting to note that Corollary 3 of axiomatic theory supports integration of processes into subsystems. Using axiomatic theory, partitioning theory, borrowing from Martin's work, SOA principles, and the concepts from IMPSs, we will now provide the generic design approach for the C4ISR.

Axiomatic theory has extensive theoretical derivations. For this paper, we will omit any complex derivations and focus our attention on the basic models and the basic corollaries and theorems to discuss our design approach. Future publications will delve into detailed theoretical derivations. For more information on detailed mathematical derivations, for example proofs of corollaries and theorems please see Suh [Suh 1990; Suh 2001]. As noted in Figure 8, design is the mapping from the functional space to the physical space. Equation 1 gives the mathematical relationship for the mapping process from the functional space to the physical space. We call Equation 1 the design equation [Suh 1990]. Equation 1 is very important, because it establishes the basic equation for designing any process, product, system, or systems-of-systems, such as the C4ISR.

$$\{FR_i\} = [DM]\{DP_i\} \qquad \text{Equation 1}$$

The symbols in Equation 1 are defined as follows:

$\{FR_i\} =$ the vector that represents the functional requirements in the functional domain, Figure 8

$\{DP_i\} =$ the vector that represents the design parameters in the physical domain, Figure 8

$[DM] =$ the design matrix that relates $\{FR_i\}$ to $\{DP_i\}$

Equation 2 gives the generalized form for a design matrix.

$$\begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ A_{m1} & A_{m2} & \cdots & A_{mn} \end{bmatrix}$$ Equation 2

where, $A_{ij} =$ an element of a design matrix that relates a component of the $\{FR_i\}$ vector to a component of the $\{DP_i\}$ vector. The nature of $[DM]$ determines if the proposed design violates AXIOM 1 by compromising the independence of functional requirements. If $[DM]$ is a diagonal matrix with all the non-diagonal elements in Equation 2 being zero (that is only the diagonal elements A_{11}, A_{22} to A_{mn} are the non-zero elements) each of the $FR's$ can be independently changed by changing one and only one of the $DP's$. When $[DM]$ is a triangular matrix, that is, either the upper non-diagonal elements or the lower non-diagonal elements in Equation 2 are zeros, $FR's$ can be independently changed by changing $DP's$ in a specified sequence. In all other cases, $FR's$ cannot be independently changed. Thus, AXIOM 1 is violated; the old design must be discarded and a new design created.

The FR_i represents the top-level (global requirement(s)) or leaf requirement(s), for example the overall vision statement(s), or the goals of the ***edge*** or unit level of the enterprise, together with its partners (suppliers, customers, etc.) -- Command in C2. Please note that in addition to the functional requirements, we should also specify the constraints that establish the bounds within which the functional requirements could be achieved. The DP_i represents the top-level (global design parameter(s)) or the leaf design parameter(s), for example the control factor(s) of the overall, the ***edge*** or unit level condition(s) that the enterprise, together with its partners (suppliers, customers, etc.) must create in order for the enterprise and its partners to achieve the desired response (i.e., FR_i) -- Control in C2. We use Equation 1 to construct the design hierarchy of the SOS. The design hierarchy involves the creation of FR/DP decomposition through zigzagging between the FR domains and the DP domains until the leaf level FRs and DPs are reached [Suh 1990; Suh 2001]. To design a large-scale adaptive value system, Suh emphasizes that we should design such a system as an ideal large-scale flexible system that will permit the users of the system to adapt it to a variety of situations. Here, an ideal large-scale flexible system is an uncoupled design with infinite adaptability or flexibility. Infinite adaptability means that an acceptable set of DP_s can always be chosen to satisfy the given subset of FR_s. For example, suppose the subsets of FR_s change as a function of time as follows, Equation 3 [Suh 2001]:

@ t = 0 $\{FR\}_0 = \{FR_1, FR_5, FR_7, FR_m\}$ Equation 3

@ t = T1 $\{FR\}_1 = \{FR_3, FR_5, FR_8, FR_z\}$

@ t = T2 $\{FR\}_2 = \{FR_3, FR_9, FR_{10}, FR_m\}$

To satisfy $\{FR\}_0$, the chosen $\{DP\}_0 = \{DP_1, DP_5, DP_7, DP_m\}$ must ensure the independence of FR_1, FR_5, FR_7, FR_m as required by AXIOM 1. A functional requirement here could be an OV-5 (a chained activities, or mission thread) executed by a specific soldier on the battlefield to achieve an overall mission. The corresponding design parameter could be the operating conditions, such as the engagement distance for attacking the enemy. At t = T_1, a different subset of FR_s must be satisfied. This means the system must reconfigure itself to satisfy $\{FR_1, FR_5, FR_7, FR_m\}$ independently. Among the corollaries and theorems derived from AXIOM 1 and AXIOM 2, the following four corollaries and a theorem, are essential for designing C4ISR, namely [Suh 1990; Suh 2001]:

Corollary 1: Decoupling of Coupled Design: Decouple or separate parts or aspects of a solution if FRs are coupled or become interdependent in the proposed designs.

Corollary 2: Minimization of FRs: Minimize the number of functional requirements and constraints. Strive for maximum simplicity in overall design or the utmost simplicity in physical and functional characteristics.

Corollary 3: Integration of Physical Parts: Integrate design features into a single physical process, device, or system when FRs can be independently satisfied in the proposed solution.

Corollary 4: Use of Standardization: Use standardized or interchangeable parts if the use of these parts is consistent with the FRs and constraints.

THEOREM M2 (Large System with Several Subunits) When a large (e.g., organization) consists of several subunits, each unit must satisfy independent subsets of FRs so as to eliminate the possibility of creating a resource-intensive system or a coupled design for the entire system.

Corollaries 1 and 3 have already been discussed. Corollary 2 states that as the number of FRs and constraints increases, the system becomes more complex and thus raises the information content. Corollary 4 states a well-known design rule: use standard parts. Corollary 4 is the theoretical foundation for achieving interoperability or plug-and-play in designing any information system architecture, such as the SOA.

In Theorem M2, we represent a large organization as systems-of-systems, such as the C4ISR. Each subunit represents the individual subsystems, for example a logistical node, or a soldier on the battlefield. Because in adaptive SOS, each subunit operates autonomously to achieve its own local objectives, it must choose to satisfy some independent subsets of the global objectives at its local level. Consider the warfighter in the C4ISR ecosystem. The soldier sees a narrow view of the entire C4ISR. The warfighter cannot directly meet the global objectives of the C4ISR. Therefore, to achieve the overall goals of the C4ISR, the soldier must perform the battlefield activities whose local performance measures must still support the global performance measures (overall mission objectives).

The concept of electronic commerce in the DOD, including the relevant business areas and the war-fighter operations, Figure 14, is an ideal blueprint for the DOD value system [Defense Electronic Business 2000]. For example, in Porter's value system model, the *Industry (various ops)* could represent the DOD prime contractors as suppliers of the C4ISR ecosystem to the DOD. *Budget Planning* is an example of a *firm infrastructure's* activity in Porter's value chain model. However, insufficient data exist in the literature to use Figure 14 to illustrate the generic concept for designing the C4ISR. Thus, we will borrow from Martin's work, Figure 15, as a representation of the DOD business operations to create the diversified enterprise (enterprise value chain) of the business value system, Figure 7b, for the C4ISR. Martin has noted a similar approach for creating an information-based enterprise for the DOD [Martin Book III 1990]. Please note that Figure 15 supports Martin's four-stage lifecycle of products (Figure 16), as noted before. Thus, it is fitting to use Figure 15 for the generic design model for the C4ISR. The design model could be easily adapted to the full-scale DOD operations, including both the warfighter operations and business operations.

The MIT SOS modeling group has done much work on Design Structure Matrix (DSM) for building models for subsystems [Design Structure Matrix]. Though their work is very intriguing, they did not establish the generic theoretical framework for their method. In fact, their definition of partitioning contradicts the definition of partition envisioned by Gersting [Gersting 1998] in transforming sets into subsets. The design model chose the definition of partition by Gersting, because it is in agreement with the definition of partition published in many textbooks in discrete mathematics for the computer science field [Gersting 1998].

In the subsequent sections, we will use the basic design equation to construct the generic design architecture for the C4ISR value system. We will select a module from the design architecture and construct the cluster analysis for the module. According to Suh, a module, M_k, in axiomatic theory, is the row of design matrix (Equation 2), which produces a functional requirement (FR) when the row of design matrix receives an input or mathematically, the row of design matrix is multiplied by its corresponding design parameter (DP). Equation 4 gives the generalized definition of a module [Suh 2001].

$$M_k = \sum_{j=1}^{j=k} \frac{\partial FR_k}{\partial DP_j} \frac{DP_j}{DP_k} \qquad \text{Equation 4}$$

Please note that the module-junction structure diagram or simply the module diagram, constructed from Equation 4, constitutes another way of representing a system or system-of-systems [Suh 2001]. When integrated with SOA, the module-junction diagram will establish the blueprint or the generic design architecture for designing integrated C4ISR systems-of-systems or the C4ISR value system. Using Equation 5 we will then construct a set S defined by the Cartesian cross product of rows (processes or activities) and columns (subject databases or high-level groupings of entity types) in Figure 15.

$$S = \{(x, y) \mid x \in \text{ Database Set and } y \in \text{Process Set}\} \qquad \text{Equation 5}$$

Equation 5 says that the Cartesian cross product of two sets “Database Set” and “Process Set” is a set S of all ordered pairs (x, y) whose first component x comes from “Database Set” and whose second component y comes from “Process Set”. Gersting has established the theoretical model for partition of a set S. She defines it as follows [Gersting 1988]: “*Definition: Partition of a Set: A* ***partition*** *of a set S is a collection of nonempty disjoint subsets of S whose union equals S.”* From Equation 6, we partition the set S into subsets of logical functional groupings of processes and databases or equivalent classes – groups of related processes and databases [Gersting 1988].

$$[x] = \{y \mid y \in S \cap x \rho y\} \qquad \text{Equation 6}$$

The symbols in Equation 6 are defined as follows:
$[x]$ = set of all members of S to which x is related, called equivalence class of x
ρ = equivalent relation on a set S

Using AcclaroDFSS [AcclaroDFSS], a software package for axiomatic design, we first construct the generic model of the C4ISR value system using Porter’s value system model, Figure 7b. Figure 17 shows the module diagram of the design hierarchy, from AcclaroDFSS, for the DOD business value system. We should emphasize that the details of AcclaroDFSS follow Equations 1, 2, and 4 in constructing the module diagram, Figure 17. Using an algorithm proposed by Martin [Martin Book II 1990], and Figure 15, we select the enterprise-module, M2, from Figure 17 and partition the business processes and subject databases into mutually exclusive clusters of leaf activities and entity types. Please note that we have assumed Figure 15 to represent the business processes for a diversified enterprise, Figure 7b, of the C4ISR. The DFSS could then be used to eliminate the sources of variability of each leave activity before using the SOA modeling concepts proposed by Arsanjani to group the services into components – natural business areas. We will discuss Arsanjani concepts for SOA design shortly. For simplicity, we have not shown the DFSS model. Future publications will address integration of quality models into the initial design of C4ISR. Please see the previous work of Nyamekye et al. [Nyamekye et al. July 28 2005] for detailed discussion of integrated quality control into the initial design of cellular manufacturing. It is quite interesting to note that Ross et al. have extensively emphasized Six Sigma as a major requirement for creating any information-based enterprise architecture [Ross et al. 2006]. Figure 18 shows the construction of the logical functional groupings of processes and databases using Equations 5 and 6. Equations 5 and 6 are used again on Figure 19 to obtain the groupings of leaf activities and entity types in Figure 20.

Arsanjani has extensively discussed that service-oriented modeling and architecture consists of three general steps: identification, specification and realization of services, components and flows (typically, choreography of services) [Arsanjani November 9 2004].

Service identification: The Service identification consists of top-down, bottom-up, and middle-out approaches of domain decomposition, existing asset analysis, and goal-service modeling. The top-down approach involves creating the logical functional groupings into natural subsystems. Each leaf activity becomes the fine-grained service. The fine-grained services can then be composed into coarse-grained services, which can be chained into activity-flow model to create logical business area or natural subsystem, such a machining cell, or customer service unit – create the ***edge*** organizations. More importantly, the coarse-grained services can also be exposed for the users.

The *bottom-up* approach involves analyzing the existing application codes to see which codes could be grouped together to enable the leaf activities in the logical functional groupings or subsystems for the logical business areas, for example the Customer Relationship Management for the customer service unit. In cellular manufacturing, the CNC application codes are grouped together to machine the parts within a manufacturing cell, Figure 13.

The middle-out approach consists of *goal-service modeling* to validate and unearth other services not captured by either top-down or bottom-up service identification approaches [Arsanjani November 9 2004]. According to Arsanjani *goal-service modeling* ties services to goals and sub-goals, key performance indicators, and metrics.

Service classification or categorization: The Service classification is the same as the composition of fine-grained services into coarse-grained services, which can be chained into activity-flow model to create logical business area or natural subsystem, such a machining cell, or customer service unit – create the ***edge*** organizations- as noted in the previous section. Because service invocation involves the network, the exposed service interfaces, discussed next in the ***Subsystem analysis***, must be designed as coarse-grained to avoid network overload [Lublinsky January 9 2007]. Rather than exposing many service interfaces that provide limited functions, services should expose a small number of service interfaces that allow individual requests to perform complete business function, such as *checking out of customers* at the point-of-sale (POS), or the warfighter executing a mission thread on the battlefield [Lublinsky January 9 2007].

Subsystem analysis: The Subsystem analysis involves exposing the logical service groups at the appropriate subsystem interfaces. Furthermore, it involves creating executable models or process flow models to check for abnormal behaviors such as *death path elimination* [Girault et al 2003] for each subsystem. *Death path elimination* is a modeling technique used in languages, for example, Business Process Execution Language (BPEL) to bypass activities whose preconditions are not met in a workflow model [Girault et al 2003; Havey 2005]. According to Arsanjani, each subsystem becomes *Service or enterprise component*, depicted in the SOA model, Figure 11.

Component specification: *Component specification* entails specifying the details of each *Service or enterprise* component that implement the services. Such details includes the following [Arsanjani November 9 2004]:

- Data
- Rules
- Services
- Configurable profile
- Variations

Service allocation: *Service allocation* consists of assigning services to the subsystems. Please note that in functional logical groupings obtained from partition model, naturally groups of the leaf activities or fine-grained services are naturally formed. Coarse-grained services can then be created to obtain the natural subsystems of *Service components*. Through Service allocation, we can combine Service components to create other subsystems within the enterprise value chains and throughout the enterprise value system.

Service realization: According to Arsanjani this step recognizes that the software that realizes a given service must be selected or custom built, or outsourced [Arsanjani November 9 2004]. Which legacy system module to use for realizing a given service and which services should be built from the "ground-up", should also be decided during *Service realization* [Arsanjani November 9 2004]. Security, management and monitoring of services are other realization decisions for services besides business functionality of services [Arsanjani November 9 2004].

Figure 22 shows the leaf activities of *Purchasing Process*, from Figure 15, for creating the *Service component* for the *Purchasing Process*. Such transaction could represent purchasing food supplies for the soldiers on the battlefield, as an example of the business process in the C4ISR systems. Please note that in Figure 22, the number 1 represents the *Purchasing Process*. The symbols A, B, C, etc., designate the leaf activities (with asterisks) or fine-grained services. For example symbol, A, represents the leaf activity or fine-grained service "CREATE REQUISITION". (Appendix B lists the properties of coherent leaf activities [Winter et al. 1981].) We use the composition theory [Milner 2004; Van der Aalst et al. 2004; Girault et al. 2003] to create the *Service component* for the *Purchasing Process*.

Let *PURCHASING (1)* represent the *Service component* for the *Purchasing Process*.
Let 2 represent the fine-grained service for *CREATE REQUISITION*.
Let 3 represent the fine-grained service for *RECORD SUPPLIER PERFORMANCE DATA*.
Let 4 represent the fine-grained service for *ANALYZE SUPPLIER PERFORMANCE*.
Let 5 represent the fine-grained service for *SELECT SUPPLIER*.
Let $6 = 2 \cap 3 \cap 4 \cap 5$.

where, the symbol $\cap$ represents the logical symbol for union of services.

Let 7 represent the fine-grained service for *CREATE PURCHASE ORDER*.
Let 8 represent the fine-grained service for *CANCEL PURCHASE ORDER.*
Let 9 represent the fine-grained service for *FOLLOW UP DELIVERY.*
Let 10 represent the fine-grained service for *CREATE INFORMATION FOR ACCOUNTS PAYABLE.*
Let 11 represent the fine-grained service for *CREATE SPECIAL ORDER*.
Let 12 represent the fine-grained service for *TERMINATE ORDER*.
Let 13 represent the fine-grained service for *CREATE NEW ORDER*.
Let $14 = 12 \cap 13$

For normal purchase order processing, the *Service component* for *PURCHASING (1)* can be logically expressed as follows:

PURCHASING (1) $= 6 \cap 7 \cap 9 \cap 10$

If the purchase order is cancelled, only the fine-grained service, designated as 8, will be executed. As noted before, to eliminate *Death path elimination*, BPEL will ensure that all process flow paths are completed before final execution of the fine-grained service is completed [Girault et al 2003; Havey 2005].

For special order processing (without non-delivery of the purchase order), *Service component* for *PURCHASING (1)* can be logically expressed as follows:

PURCHASING (1) = $6 \cap 11 \cap 10$

For special order processing which involves creating a new order to replace a non-delivery order, *Service component* for *PURCHASING (1)* can be logically expressed as follows:

PURCHASING (1) = $6 \cap 14 \cap 10$

Figure 23 is the SOA model of the generic C4ISR. To create new business processes or mission threads at different times for the C4ISR, Equations 1 and 3 are used at the ***edge*** of the enterprise to create the new service flow models (OV-5). From the executable models, the full-scale integrated simulation models [Nyamekye February 6-7 2006] could be constructed for the C4ISR. Mobile agents could then be added to each *Service component* to serve the same purpose as that discussed in Toyota Production System. With such an approach, we can eliminate the issue raised by Kleinrock on mobile agents in system designs. Using AXIOM 2, and the appropriate constraints, we can predict the performance of the C4ISR. For this paper we have omitted simulation models and analysis from AXIOM 2. Future publications will address designing integrated Activity-Based Simulations for C4ISR that will include mobile agents, AXIOM 2 for performance evaluation, and more importantly the total cost of ownership (TCO) of the C4ISR, Figure 5.

Please note that we have assumed the generic SOA model for the C4ISR, based on the Porter's value system model in Figure 7b, for a diversified C4ISR. Suppliers' value chains can represent the operational activities performed by NGOs in providing the appropriate humanitarian needs for the civilians during combat and non-combat operations. The diversified enterprise value chains could represent the operational activities performed by a major DOD prime contractor coordinating all the DOD business operational activities during combat and non-combat operations. The channel value chains can represent the operational activities performed by distributors, such as the movement of food suppliers or weapons to the soldiers on the battlefield, or clothing to the civilians for humanitarian efforts. The buyer or the customer value chains can represent the operational activities of the warfighters or the civilians.

In Figure 23, the symbol I, associates with UDDI BUSINESS REGISTRY I, WEB SERVICE I and Data Source I for the enterprise and the value channels. Please note that UDDI stands for Universal Description, Discovery and Integration. The WEB SERVICE I contains *Service components* for the enterprise and the value channels. For simplicity, we have assumed in Figure 23 that the channel value chains are part of the enterprise operational activities. The UDDI BUSINESS REGISTRY I, serves the same function as the Service Broker discussed previously in Figure 10. Similarly, the symbol J associates with the UDDI BUSINESS REGISTRY J, WEB SERVICE J and Data Source J for the suppliers, and the symbol K associates with the UDDI BUSINESS REGISTRY K, WEB SERVICE K and Data Source K, for the buyers or customers (warfighters or civilians). Each Data Source can also independently receive data through a satellite system. For simplicity we have omitted such configuration in Figure 23. We should emphasize that the issue of coalition partners in adopting SOA concepts should not be an issue for creating the generic model for the C4ISR since SOA is now being internationally adopted [Schmelzer February 1 2007]. Please notice in Figure 23 that we have integrated ESBs into the SOA infrastructure.

SUMMARY AND FUTURE POTENTIAL CAPABILITIES

This paper has discussed a generic model as an alternative blue print for creating re-configurable C4ISR systems. The paper recognizes that despite detailed guidelines provided by DODAF, it may not be an appropriate architecture for designing an adaptive C4ISR that must respond to dynamic battlefield conditions, and more importantly for designing the DOD's visionary GIG.

Recent publications and the new requirements for the warfighters adaptability to urban warfare show that future C4ISR systems-of-systems must address such critical issues. Using axiomatic theory, the scientific concepts for creating Integrated Manufacturing Production Systems (IMPSs), the value system model, and borrowing from Martin's work for creating information-based enterprises, we have constructed the generic model that could serve as a template for designing an adaptive C4ISR. More importantly, the paper recognizes that future C4ISR SOS must follow the DOD's Net-Centric Checklist for creating re-configurable SOS to operate within the SOA environment. Thus, it has presented the generic architectural model to address such a need.

The proposed generic model provides potential capabilities for an experimental scientific research test bed in the on-going research activities on ***Power to the Edge***, such as Interoperability and Agility and more importantly as a test bed for creating integrated SOS for the Department of Homeland Security. It could as serve a test bed for creating integrated Activity-Based Simulation models -- hybrid activity-based models (service models) for discrete and continuous simulations that execute independently -- for distributed enterprises and also for integrating mobile agents without introducing complexities into the SOS. Using AXIOM 2 and Design for Six Sigma (DFSS) scientific concepts, the generic model could serve as a template for predicting in advance the performance of ***edge*** enterprises under the constraints defined by the enterprises.

REFERENCES

Alberts, S.D., and R. E. Hayes, THE FUTURE OF COMMAND AND CONTROL: UNDERSTANDING COMMAND AND CONTROL, CCRP Publication Series, 2006.

Alberts, S.D., and R. E. Hayes, Power to the Edge, CCRP Publication Series, 2003.

Alberts, S.D., J. J. Gartksa, and P., Stein, and P. F., NETWORK CENTRIC WARFARE: Developing and Leveraging Information Superiority, 2nd Edition (Revised), CCRP Publication Series, 1999.

AcclaroDFSS, Axiomatic Design Software, Inc., Boston, MA.

AMR Research, "Supply chain Excellence: Today's Best Driver of Bottom-Line Performance," Business Week, April 25, 2005.

Anthes, G., "Simpler IT Approach Needed, Former CIO Says," http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=274613, COMPUTERWORLD, p. 41, November 27, 2006.

Anthes, G., "Q&A: An Internet Pioneer Looks Ahead," http://computerworld.com/printthis/2005/0,4814,102862,00.html, COMPUTERWORLD, July 4, 2005.

Arsanjani, A., "Service-Oriented Modeling And Architecture: How To Identify, Specify, And Realize Services For Your SOA, Level: Intermediate," IBM DeveloperWorks, November 9, 2004.

Barsoum, Y., Dimarogonas, J., Kenneally, W., Kelly, K., Mahon, C., McBride, F., Meinert, B., Mullins, J., Osgoodby, K., Providakes, J. F., Wiener, S., Walsh, T. J., Farah-Stapleton, M. K., "C4ISR Architectural Design Tenets for Future Tactical Army Systems," IEEE MILCOM 2002 PROCEEDINGS, Volume: 2, pp. 1466 - 1471, October 7-10, 2002.

Black, J T., THE DESIGN OF THE FACTORY WITH A FUTURE, McGraw-Hill, Inc., New York, NY, 1991.

Bloomberg, J., and Schmelzer, R., Service Orient or Be Doomed: How Service Orientation Will Change Your Business, John Wiley & Sons, Hoboken, NJ, March 2006.

Boeing, Integrated Defense Systems, http://www.boeing.com/defense-space/ic/fcs/bia/index.html.

Bradley, P. S., and R. L., Sense & Respond: Capturing Value in the Network Era, Harvard Business School Press, Boston, MA, 1998.

Brooks, R., "Newer Math? A New High-School Mathematics Might Someday Model Complex Adaptive Systems," Technology Review: AN MIT ENTERPRISE, http://www.technologyreview.com//wtr_16010,1,p1.html, December 2005/January 2006.

Chang K., H., and Nyamekye, K., "Distributed Intelligent Control Systems for an Unmanned

Manufacturing Cell," Expert Systems in Engineering Systems, Chapter 19, pp. 353-364, Spyros Tzafestas, Editor, Spring-Verlag, New York, NY, 1993.

Chang, K., H., Nyamekye, K., and Black, J T., "A Distributed Expert System for Manufacturing Cell Control," Chapter in Recent Development in Production Research, Vol. 6, pp. 861-867.

CMMI, What is the CMMI? Process Academy's White Paper, July 2003.

Codd, E., 1971a, "Relational Completeness of Data Base Sub-languages," Courant Computer Science Symposium 6, Data Base Systems, Prentice Hall.

Colan, M., "Service-Oriented Architecture Expands The Vision Of Web Services, Part 1, Characteristics of Service-Oriented Architecture, Level: Introductory," http://www-128.ibm.com/developerworks/webservices/library/ws-soaintro.html, April 21, 2004.

Defense Modeling and Simulation (DMSO), "Activity-Based Simulation: A discrete simulation that represents the components of a system as they proceed from activity to activity; for example, a simulation in which a manufactured product moves from station to station in an assembly line. *Reference*: (**b**) "A Glossary of Modeling and Simulation Terms for Distributed Interactive Simulation (DIS),"" https://www.dmso.mil/public/resources/glossary/results?do=get&def=14, August, 1995.

Defense Electronic Business (DEB), *Joint Electronic Commerce Architecture*, 2000.

Dellith, E., "Another Look at Toyota," http://www.investor.reuters.com/Article.aspx?docid=9489&target=companyoftheday&src=cms, May 11, 2006.

Design Structure Matrix, http://www.dsmweb.org/index.php?option=com_content&task=view&id=33&Itemid=26.

DODAF (DOD Architecture Framework), Version 1: Deskbook, February 9, 2004.

DODAF (DOD Architecture Framework), Version 1, Volume I: Definitions and Guidelines, February 9, 2004.

DODAF (DOD Architecture Framework), Version 1, Volume II: Product Descriptions, February 9, 2004.

eWeek, "By the Numbers SOA what," June 26/July 3, 2006.

Fokkink, W., Introduction to Process Algebra, Springer-Verlag, New York, NY, 2000.

Gersting, J., L. Mathematical Structures for Computer Science, 4th Edition, W.H. Freeman and Company, New York, 1998.

Girault, C., and R. Valk, Petri Nets for Systems Engineering: A Guide to Modeling, Verification, and Applications, Springer, New York, 2003.

Glick, F., and D. Kohn, Darwin On Evolution: The Development and Theory of Natural Selection, Hackett Publishing Company, Inc., Cambridge MA, 1996.

GME (Generic Modeling Environment), "GME Technical Overview," http://www.isis.vanderbilt.edu/projects/gme/Tech.html.

Handfield, B. R., and E. L. Nichols, Jr., Supply Chain Redesign: Transforming Supply Chains into Integrated Value Systems, Prentice Hall, Upper Saddle River, NJ, 2002.

Hammer, M., and J., Champy, Reengineering the Corporation, HarperBusiness, Revised Edition, New York, NY 1022, 2001.

Havenstein, H., "Web Services Spread Slowly Into IT, But More Work Needs To Be Done To Address Architectural, Process And Security Issues," COMPUTERWORLD, pp. 6-7, January 24, 2005.

Havey, M., Essential Business Process Modeling, O'REILLY Media, Inc, Massachusetts, 2005.

IBM, "Running An On Demand Business To Succeed In The Connected World," WebSphere Software for E-business, May 2003.

IBM Web Seminar, "Introduction to the IBM Rational Unified Process," May 26, 2004.

IMS MISSION, http://www.ims.org/projects/project_info/mission.html.

Jones, V. C. MAO/TOP Networking: A Foundation for Computer-Integrated Manufacturing, McGraw-Hill, New York, NY, 1988.

Knowledge Wharton, Leadership and Change, "Tip of the Spear: Leadership Lessons from the U.S.-led Armed Forces in the Middle East," http://knowledge.wharton.upenn.edu/index.cfm?fa=viewfeature&id=1484, Knowledge Wharton, May 17, 2006.

Krone, R., "Managing a Complex System-of-Systems," http://govinfo.library.unt.edu/moontomars/docs/050404SlidesKrone.pdf, White Paper, The Boeing Company, May 4, 2004.
Ledeczi, A., Maroti, M., Bakay, A., Karsai, G., Garrett, J., Thomason, C., Nordstrom, G., Sprinkle, J., and Volgyesi, P., "The Generic Modeling Environment," White Paper, Vanderbilt University, Institute for Software Integrated Systems, Nashville, TN 37235, USA.
Lublinsky, O., "Defining SOA as an architectural style: Align your business model with technology, Level: Intermediate," IBM DeveloperWorks, January 9, 2007.
Martin, J., Information Engineering, Book I: Introduction, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1989.
Martin, J., Information Engineering, Book II: Planning and Analysis, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1990.
Martin, J., Information Engineering, Book III: Design and Construction, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1990.
Milner R., Communicating and Mobile Systems: the Pi-Calculus, Cambridge University Press, 2004.
Monden, Y., Toyota Production System: Practical Approach to Production Management, IIE Press, 1983.
Murch, R, Autonomic Computing: On Demand Series, Prentice Hall, Upper Saddle River, NJ 07458, 2004.
Nakazawa, H., and N. P., "Process Planning Based on Information Concept," Robotics and Computer Integrated Manufacturing, Vol. 1, No. 1, pp. 115-123, 1984.
National Institute of Standards and Technology Manufacturing, IMS MISSION, http://www.nsrp.org/projects/project10/sim_model.ppt, 1999.
National Research Council, Surviving Supply Chain Integration: Strategies for Small Manufacturers, National Academy Press, Washington, D.C., 2000.
National Research Council, Modeling and Simulation in Manufacturing and Defense Acquisition: Pathways to Success, National Academy Press, Washington, DC, 2002.
NATO SAS-050, "Exploring New Command and Control Concepts and Capabilities," Final Report, Prepared for NATO, January 2006.
Net-Centric Checklist, Office of the Assistant Secretary of Defense for Networks and Information Integration/Department of Defense Chief Information Officer, http://www.dod.mil/cio-nii/docs/NetCentric_Checklist_v2-1-3_.pdf, Version 2.1.3, May 12, 2004.
Nyamekye, K., J. A. Clendenin, and H. Burleson, "Information System Architecture for Wired Battlefield," 2004 Proceedings of Navy Marine Corps Intranet, June 2004 [CD Version].
Nyamekye, K., "The Modified Version of HP Adaptive Enterprise Architecture, Based on Service-Oriented-Grid (SO-G) Architecture Model," Working Paper, April 2006.
Nyamekye, K., "A Web-Based Distributed Enterprise Model For A Sustainable Growth And Low Cost Cocoa Supply Chain For A Ghanaian Cocoa Farmer," White Paper, July 2006.
Nyamekye, K., "Supply Chain Design for Six Sigma (DFSS): an Axiomatic Quality Approach, " American Society for Quality 2006 Six Sigma Conference Proceedings, Palm Springs, CA February 6-7, 2006.
Nyamekye Research and Consulting, National Institute of Standards and Technology Manufacturing Simulation and Visualization Program, http://www.mel.nist.gov/div826/msid/sima/item2000/06mclean.pdf, 2000.
Nyamekye, K., and J., A, Clendenin, "ICLOGISTICS JOINT BATTLESPACE INFOSPHERE (JBI)-GRID ARCHITECTURE, White Paper, July 2006.
Nyamekye, K., "Review Of Service-Oriented Architecture, Systems-of-Systems Architectures And Construction Of Generic Technical And Scientific Framework For Modeling And Simulation Of Service-Based Integrated Defense Systems", Boeing Integrated Defense Systems: NRC Technical Report, Boeing, November 6, 2006.
Nyamekye, K., and Y.-K. An, "Using Rapid Modeling Technology in a Permanent Mold Casting Production Facility," AFS Transactions, Volume 96, pp. 96-188, 1996.
Nyamekye, K., "New Tool for Business Process Reengineering: Activity-Based Simulation Offers Functionality that We've Never Experienced Before Until Now," IIE SOLUTIONS, March 2000, pp. 36-41, http://findarticles.com/p/articles/mi_hb3002/is_200003/ai_n7865133.
Nyamekye, K., Sutterfield, S., Askeland, D. R., and Bain, R., and Cunningham, M., Classification and

Coding: The First Step In Designing Manufacturing Cells: An article from: Modern Casting [HTML] (Digital), November 1994, Reprinted and Published, by Amazon Dot Com, http://www.amazon.com/gp/product/B00092YG9A/ref=sr_11_1/104-7516179-4397551?ie=UTF8, July 28, 2005.

Ohno, T., "Toyota Production System: Beyond Large-Scale Production," Productivity Press, New York, NY, 1988.

Performance-Based Logistics (PBL), http://akss.dau.mil/dag/Guidebook/IG_c5.3.asp.

Porter Michael E., Competitive Advantage: Creating and Sustaining Superior Performance: With a New Introduction, The Free Press, A Division of Simon & Schuster Inc., New York, NY, 1985.

Rob, P., and Coronel, C., Database Systems: Design, Implementation, and Management, 3rd Edition, Course Technology, A Division of International Thomson Publishing, Cambridge MA, 1997.

Roberts Julie, S., "The Buzz About Supply Chain: Does it mean to you what it means to your organization, suppliers, and customers?" Inside Supply Management, Vol. 14, No. 7, pp. 24-28, July 2003.

Ross, W. J., P. Weill, and D. C. Robertson, Enterprise Architecture as Strategy: Creating a Foundation For Business Execution, Harvard Business School Press, Boston, MA, 2006.

SAP, "Building the Lean Extended Enterprise through Adaptive Supply Chain Networks: CEO Breakthrough! Special Research Report!" KNOWLEDGE STORM, February 9, 2007

SCOR Version 7 Overview, Supply chain Operations Reference-Model, Supply chain Council.

Schmelzer, R., "The State of Worldwide SOA Adoption: Is the US Behind?" http://www.zapthink.com/report.html?id=ZAPFLASH-200721, ZapThink, February 1, 2007.

Schmidt, D., C., "Model-Driven Engineering," IEEE Computer Society, February 2006.

Shapiro J. F., Modeling the Supply Chain, Duxbury Thomson Learning, Pacific Groove, CA, 2001.

Shin and Ravindran, "Interactive Multiple Objective Optimization: Survey I-Continuous Case," Computers and Operations Research, Vol.18, 97-114, 1991.

Smith, D., "What Does IT Cost? Maximizing the IT Yield While Minimizing the IT Investment," from Micromation, http://www.micromationinc.com/, Presented at Better Management Web Seminar, http://www.bettermanagement.com/seminars/seminar.aspx?L=12913, September 15, 2005.

Softech, Inc 1981.*Integrated Computer-Aided Manufacturing (ICAM) Architecture Part II, Vol. IV Function Modeling Manual (IDEF0)*, Technical Report AFWAL-TR-81 4023, Materials Laboratory (AFWAL/MLTC), AF Wright Aeronautical Laboratories (AFSC), Wright-Patterson AFB, Dayton, Ohio.Federal Information Processing Standards Publication 183.

Suh, N. P., The Principles of Design, Oxford University Press, New York, 1990.

Suh, N. P., Axiomatic design: Advances and Applications, Oxford University Press, New York, 2001.

Tofts, C.M.N., "Describing Social Insect Behaviour Using Process Algebra," Transactions of the Society for Computer Simulation, pp. 227-283, 1992.

Van Der Aalst, W., and Hee, V., WORKFLOW MANAGEMENT, The MIT Press, Cambridge, MA, 2004.

Vernadat, F. B., Enterprise Modeling and Integration: Principles and Application, Berlin: Chapman and Hall, 1996.

Weiss, G. Multi-agent Systems: A Modern Approach to Distributed Artificial Intelligence, The MIT Press, Cambridge, MA, 1999.

Winter, K., and R. Belew, "From a Methodology Working Paper," Database Design, Inc., Ann Arbor, MI, 1981.

Wisnosky, D., J. Vogel, and S. Ring, "The Road to Executable Architectures," http://www.wizdom.com/roadtoexecutablearch.ppt, November 2004.

Wisconsin Manufacturing Extension Partnership (WMEP), "Turn Your Supply Chain into a Value Chain," http://www.wmep.org/valuechainmanagement.html.

Wooldridge, M., An Introduction to Multi-agent Systems, John Wiley & Sons, Hoboken, NJ, 2002.

Zachman J. A., "CONCEPTS OF THE FRAMEWORK FOR ENTERPRISE ARCHITECTURE," Zachman International, Inc., La Cañada, Ca., 1997.

Zimmermann, O., and M., Tomlinson, "Building Service-Oriented Architectures With Web Services: Extracts from Perspectives on Web Services – Applying SOAP, UDDI and WSDL to Real-World Projects," OOPSLA, 2005.

Zimmermann, O., P. Krogdahl, and C. Gee, "Elements Of Service-Oriented Analysis And Design: An Interdisciplinary Modeling Approach For SOA Projects, Intermediate Level," IBM

DeveloperWorks, June 2, 2004.
Zimmermann, O., M., Tomlinson, and S. Peuser, Perspectives on Web Services – Applying SOAP, UDDI and WSDL to Real-World Projects; Springer, 2nd Edition, New York, NY, 2005.

BIOGRAPHICAL SKETCH OF THE AUTHOR

Dr Kofi Nyamekye is the author of this report. He is the founder, President and Chief Executive Officer of Integrated Activity-Based Simulation Research, Incorporated, formerly known as Nyamekye Research & Consulting (NRC). Through his career, he has developed the educational, research and industrial experience to design and operate any manufacturing production system. As an educator (a former college professor) and a researcher, Dr. Nyamekye has published extensive articles on the design and operation of Cellular Manufacturing Production Systems and Simulation and Modeling of the Traditional Manufacturing Processes. Dr. Nyamekye has received many awards, among them the *1994 Educator of the Year Award from the St. Louis Society of Manufacturing Engineers Chapter 17 and the 1996 Best Paper Award selected by the Engineering Division of the American Foundrymen's Society*.

Under a large research grant from the U.S. Department of Energy, he pioneered a computer simulation model for the mold failure of the permanent mold casting process. The model will play an essential part as a federate in a distributed enterprise simulation model for automobile manufacturing supply chain federations.

Dr. Nyamekye is a member of the NIST's International Consortium of Research Scientists called the IMS MISSION. The goal of the MISSION is to integrate and utilize new, knowledge-aware, technologies of distributed persistent data management, as well as conventional methods and tools in various enterprise domains, to meet the needs of globally distributed modeling and simulation. Under a grant from NIST, Dr. Nyamekye has established the methodology for designing Activity-Based Simulation (ABS) System as a federate in the NIST distributed enterprise simulation model for the NIST IMS MISSION. The NIST federation uses the HLA RTI architecture. The ABS uses the entity relationship (E-R) modeling technique for designing an Activity-Based Costing model with input data from the simulation federates for supply chain optimization.

Recent emphasis on Activity-Based Methodology by MITRE and Lockheed-Martin, for service-based simulation of NCW [Wisnosky et al. November 2004], and previous emphasis on Activity-Based Simulation by the Defense and Modeling Simulation Office [DMSO, https://www.dmso.mil/public/resources/glossary/results?do=get&def=14], have made Activity-Based Simulation the essential concept for dynamic simulation and optimization of Service-Oriented Architecture (SOA). Dr. Nyamekye is a senior research scientist in modeling and simulation of *complex adaptive distributed enterprise systems* for a DOD prime contractor.

Dr. Nyamekye is also a member of the NIST Simulation Standards Consortium, and American Society for Engineering Education (ASEE). He holds a PhD in Industrial Engineering from Pennsylvania State University, an MS in Mechanical Engineering from Pennsylvania State University, and a BS in Mechanical Engineering from the University of Wisconsin. He can be reached at kofinsoyameye@earthlink.net.

APPENDIX A: LIST OF FIGURES

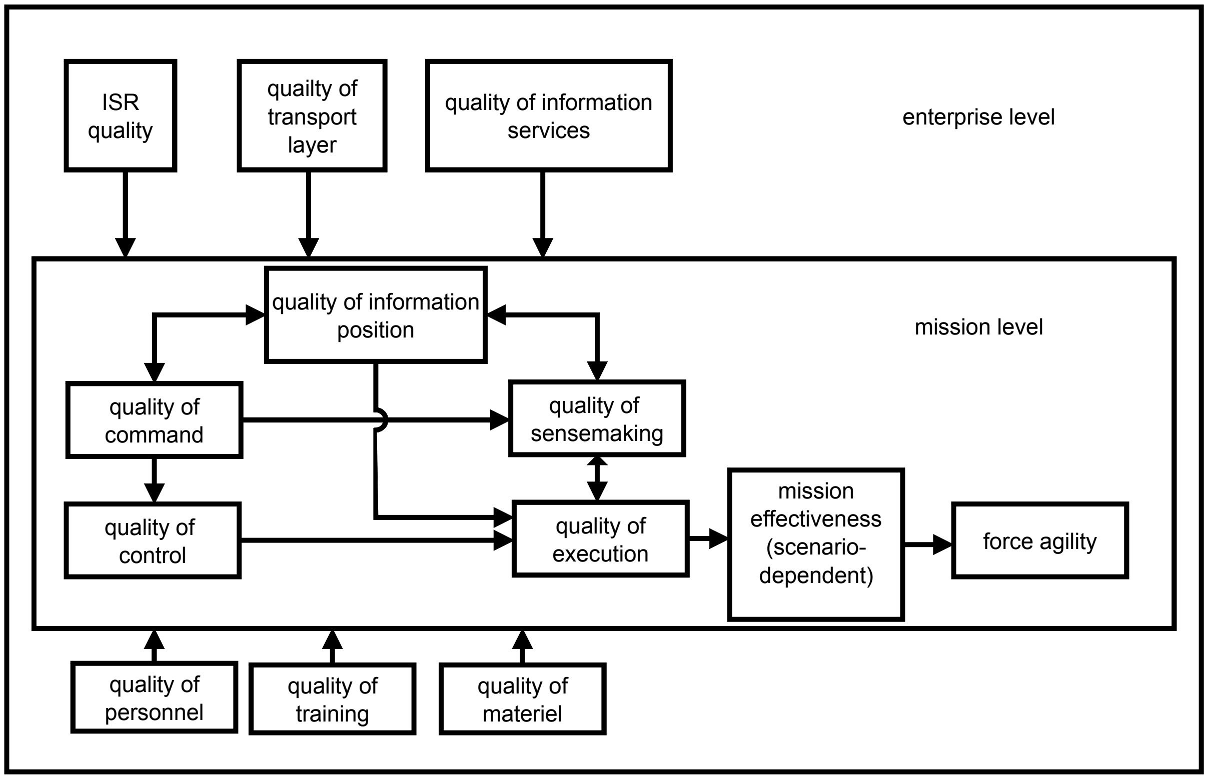
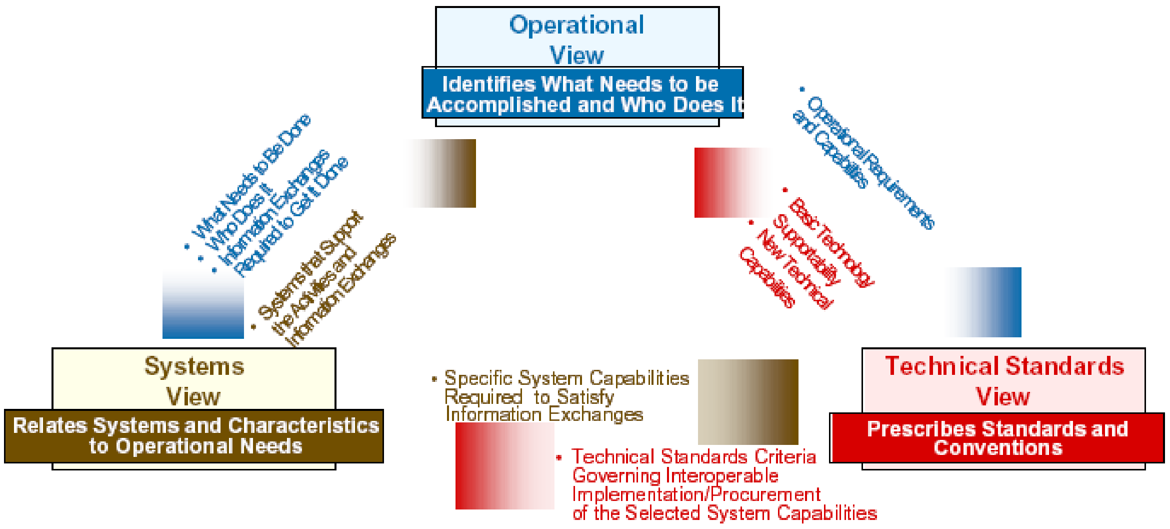
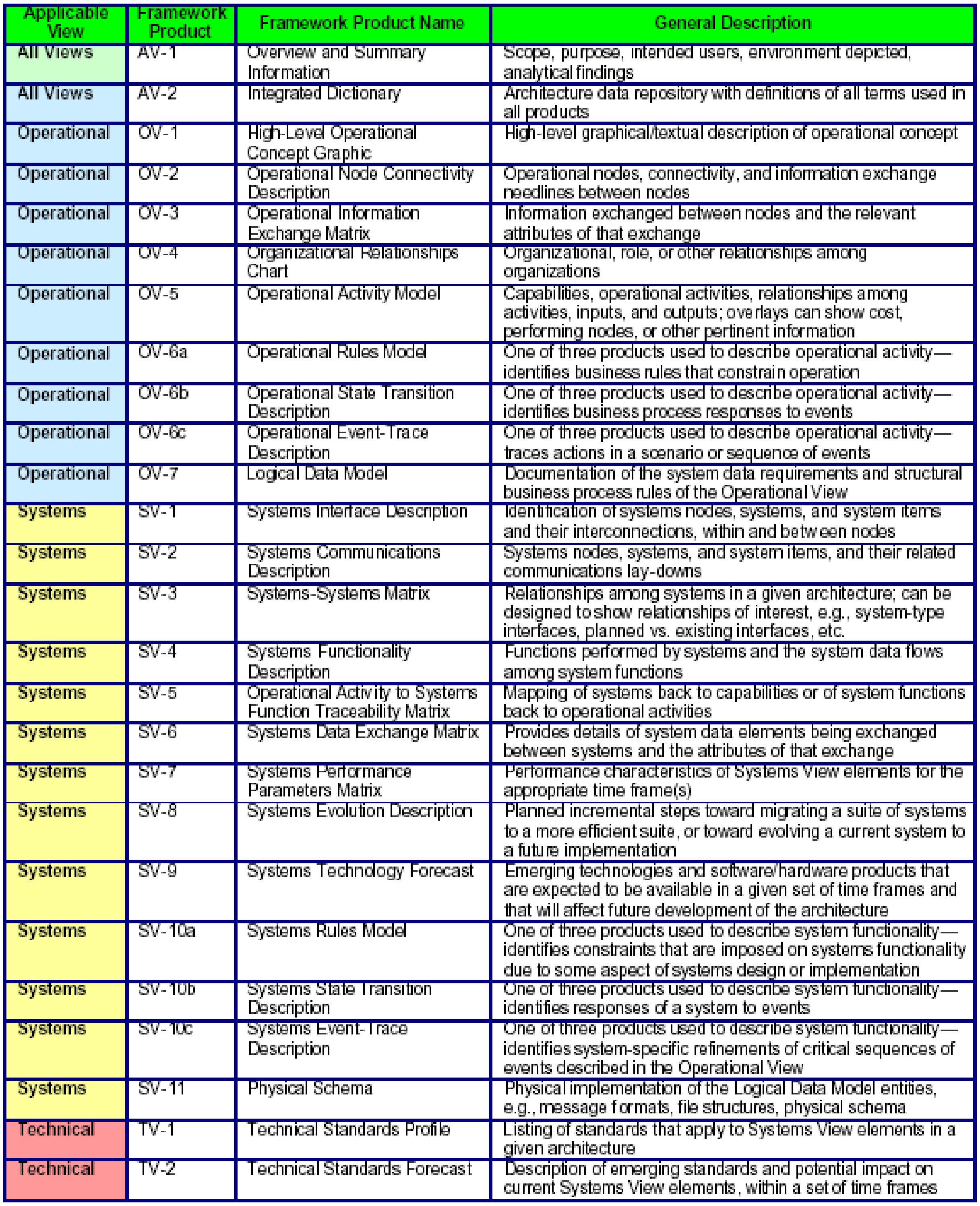


Figure 1. The Value Chain for C2 Showing the C4ISR Systems [Alberts et al. 2006.]



Operational View
Identifies What Needs to be Accomplished and Who Does It
• What Needs to Be Done
• Who Does It
• Information Exchanges Required to Get It Done
• Systems that Support the Activities and Information Exchanges
• Operational Requirements and Capabilities
• Basic Technology Supportability
• New Technical Capabilities
Systems View
Relates Systems and Characteristics to Operational Needs
• Specific System Capabilities Required to Satisfy Information Exchanges
• Technical Standards Criteria Governing Interoperable Implementation/Procurement of the Selected System Capabilities
Technical Standards View
Prescribes Standards and Conventions

Figure 2. The Three Views of DODAF [DODAF Version 1, Volume II February 2004.]



Applicable View	Framework Product	Framework Product Name	General Description
All Views	AV-1	Overview and Summary Information	Scope, purpose, intended users, environment depicted, analytical findings
All Views	AV-2	Integrated Dictionary	Architecture data repository with definitions of all terms used in all products
Operational	OV-1	High-Level Operational Concept Graphic	High-level graphical/textual description of operational concept
Operational	OV-2	Operational Node Connectivity Description	Operational nodes, connectivity, and information exchange needlines between nodes
Operational	OV-3	Operational Information Exchange Matrix	Information exchanged between nodes and the relevant attributes of that exchange
Operational	OV-4	Organizational Relationships Chart	Organizational, role, or other relationships among organizations
Operational	OV-5	Operational Activity Model	Capabilities, operational activities, relationships among activities, inputs, and outputs; overlays can show cost, performing nodes, or other pertinent information
Operational	OV-6a	Operational Rules Model	One of three products used to describe operational activity—identifies business rules that constrain operation
Operational	OV-6b	Operational State Transition Description	One of three products used to describe operational activity—identifies business process responses to events
Operational	OV-6c	Operational Event-Trace Description	One of three products used to describe operational activity—traces actions in a scenario or sequence of events
Operational	OV-7	Logical Data Model	Documentation of the system data requirements and structural business process rules of the Operational View
Systems	SV-1	Systems Interface Description	Identification of systems nodes, systems, and system items and their interconnections, within and between nodes
Systems	SV-2	Systems Communications Description	Systems nodes, systems, and system items, and their related communications lay-downs
Systems	SV-3	Systems-Systems Matrix	Relationships among systems in a given architecture; can be designed to show relationships of interest, e.g., system-type interfaces, planned vs. existing interfaces, etc.
Systems	SV-4	Systems Functionality Description	Functions performed by systems and the system data flows among system functions
Systems	SV-5	Operational Activity to Systems Function Traceability Matrix	Mapping of systems back to capabilities or of system functions back to operational activities
Systems	SV-6	Systems Data Exchange Matrix	Provides details of system data elements being exchanged between systems and the attributes of that exchange
Systems	SV-7	Systems Performance Parameters Matrix	Performance characteristics of Systems View elements for the appropriate time frame(s)
Systems	SV-8	Systems Evolution Description	Planned incremental steps toward migrating a suite of systems to a more efficient suite, or toward evolving a current system to a future implementation
Systems	SV-9	Systems Technology Forecast	Emerging technologies and software/hardware products that are expected to be available in a given set of time frames and that will affect future development of the architecture
Systems	SV-10a	Systems Rules Model	One of three products used to describe system functionality—identifies constraints that are imposed on systems functionality due to some aspect of systems design or implementation
Systems	SV-10b	Systems State Transition Description	One of three products used to describe system functionality—identifies responses of a system to events
Systems	SV-10c	Systems Event-Trace Description	One of three products used to describe system functionality—identifies system-specific refinements of critical sequences of events described in the Operational View
Systems	SV-11	Physical Schema	Physical implementation of the Logical Data Model entities, e.g., message formats, file structures, physical schema
Technical	TV-1	Technical Standards Profile	Listing of standards that apply to Systems View elements in a given architecture
Technical	TV-2	Technical Standards Forecast	Description of emerging standards and potential impact on current Systems View elements, within a set of time frames

Figure 3. The 26 Products of DODAF [DODAF Version 1, Volume II February 2004.]

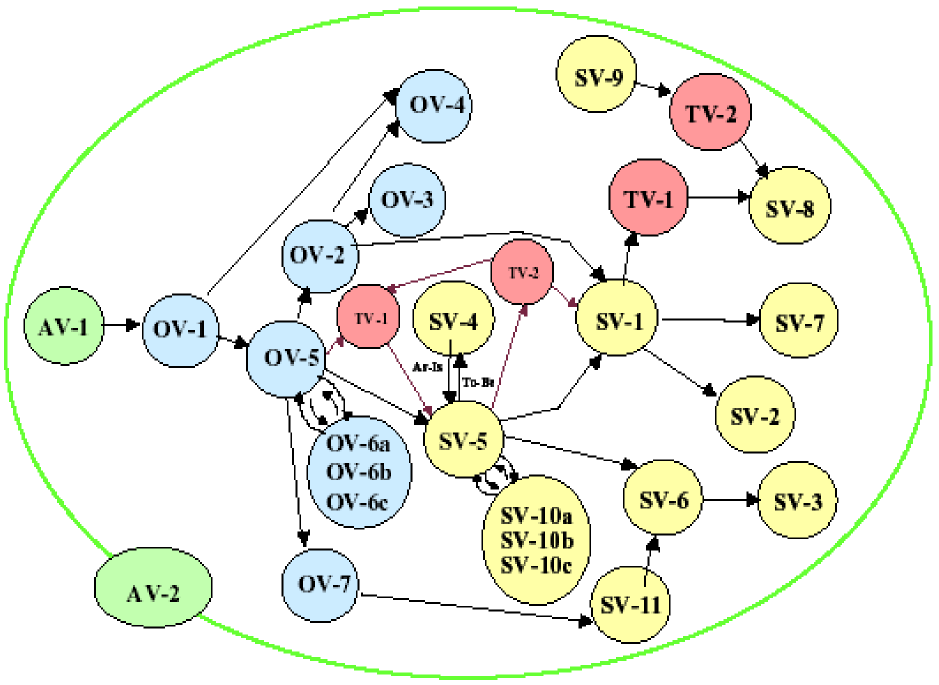


Figure 4. Relationships Among the Data Elements of DODAF [DODAF Version 1, Volume II February 2004.]

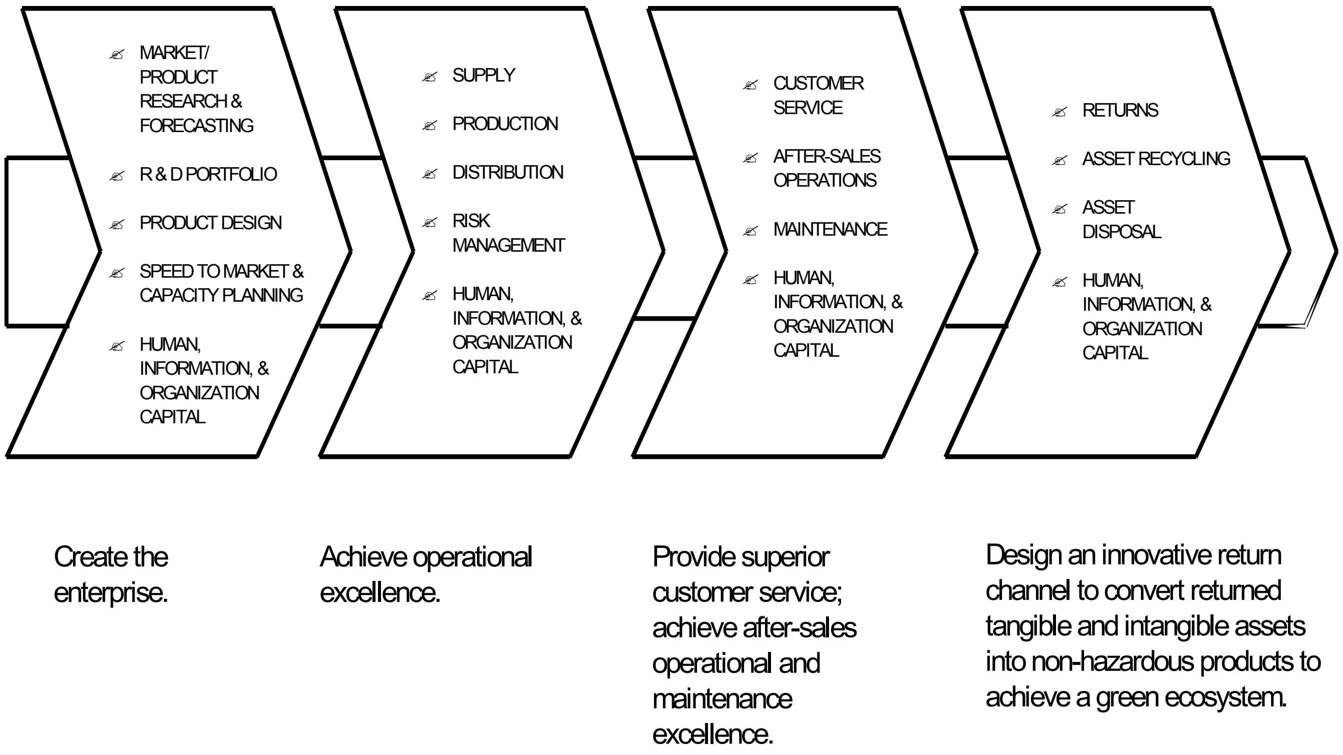

MARKET/ PRODUCT RESEARCH & FORECASTING
R & D PORTFOLIO
PRODUCT DESIGN
SPEED TO MARKET & CAPACITY PLANNING
HUMAN, INFORMATION, & ORGANIZATION CAPITAL
SUPPLY
PRODUCTION
DISTRIBUTION
RISK MANAGEMENT
HUMAN, INFORMATION, & ORGANIZATION CAPITAL
CUSTOMER SERVICE
AFTER-SALES OPERATIONS
MAINTENANCE
HUMAN, INFORMATION, & ORGANIZATION CAPITAL
RETURNS
ASSET RECYCLING
ASSET DISPOSAL
HUMAN, INFORMATION, & ORGANIZATION CAPITAL
Create the enterprise.
Achieve operational excellence.
Provide superior customer service; achieve after-sales operational and maintenance excellence.
Design an innovative return channel to convert returned tangible and intangible assets into non-hazardous products to achieve a green ecosystem.

Figure 5. Proposed Value Chain Model, Based On The Four-Stage Lifecycle of a Product, Service or Systems-Of-Systems [Nyamekye April 2006.]

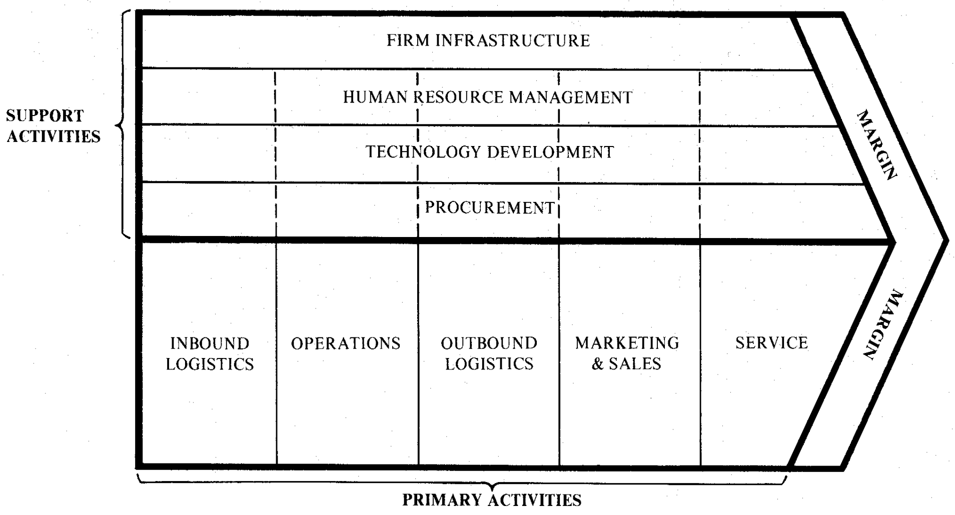


Figure 6. Value Chain [Porter 1985.]

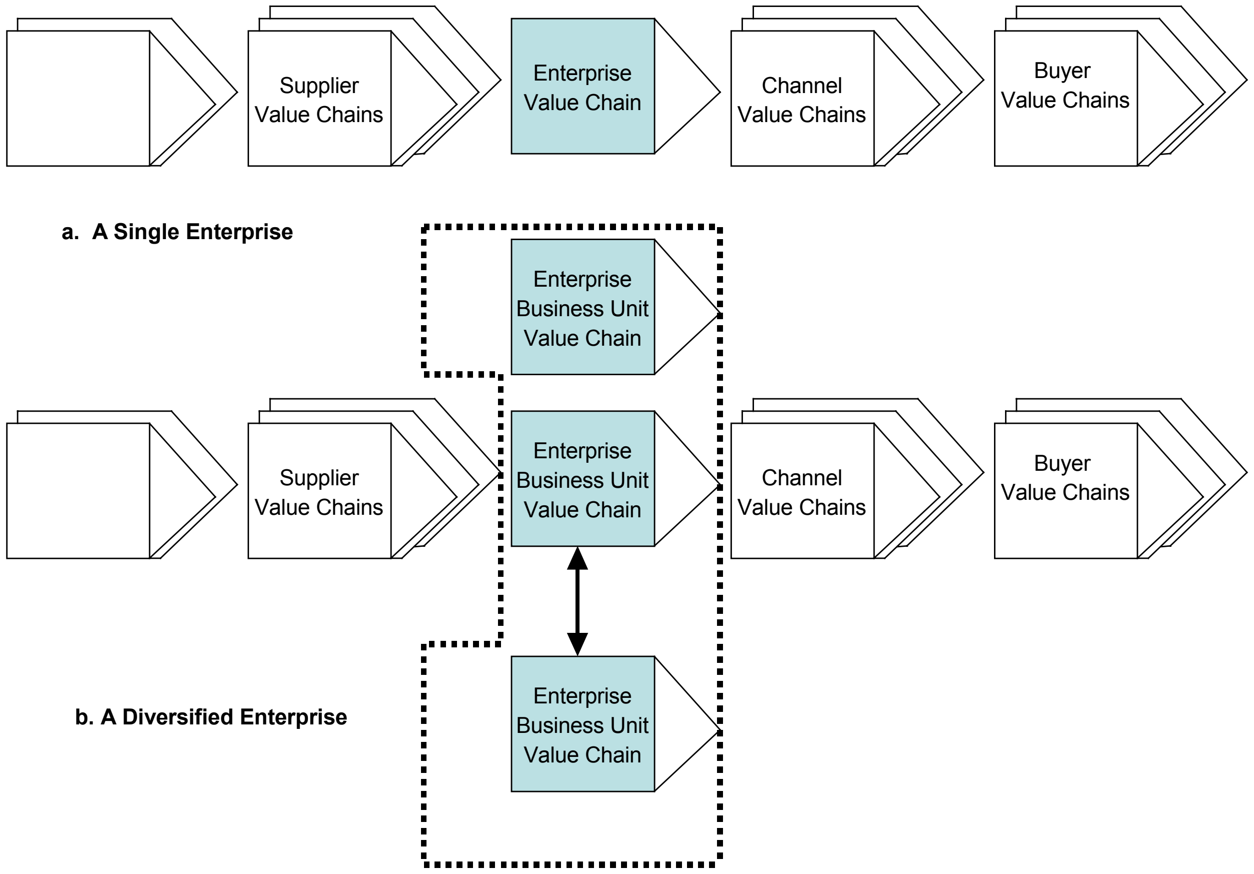


Figure 7. The Value System [Porter 1985.]; a. For a Single Enterprise; b. For a Diversified Enterprise.

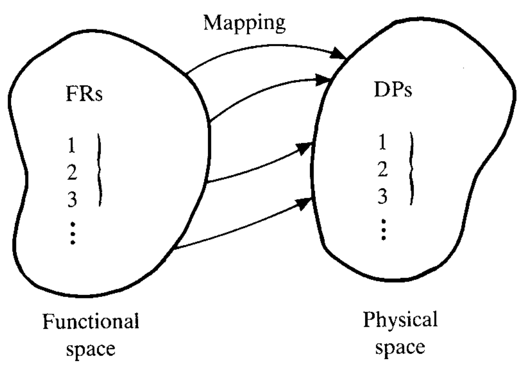


Figure 8. The Mapping From The Functional Space (Or Domain) To The Physical Space (Or Domain), [Suh 1990.] Please note that the DPs in the physical space are chosen to satisfy the FRs in the functional domain.

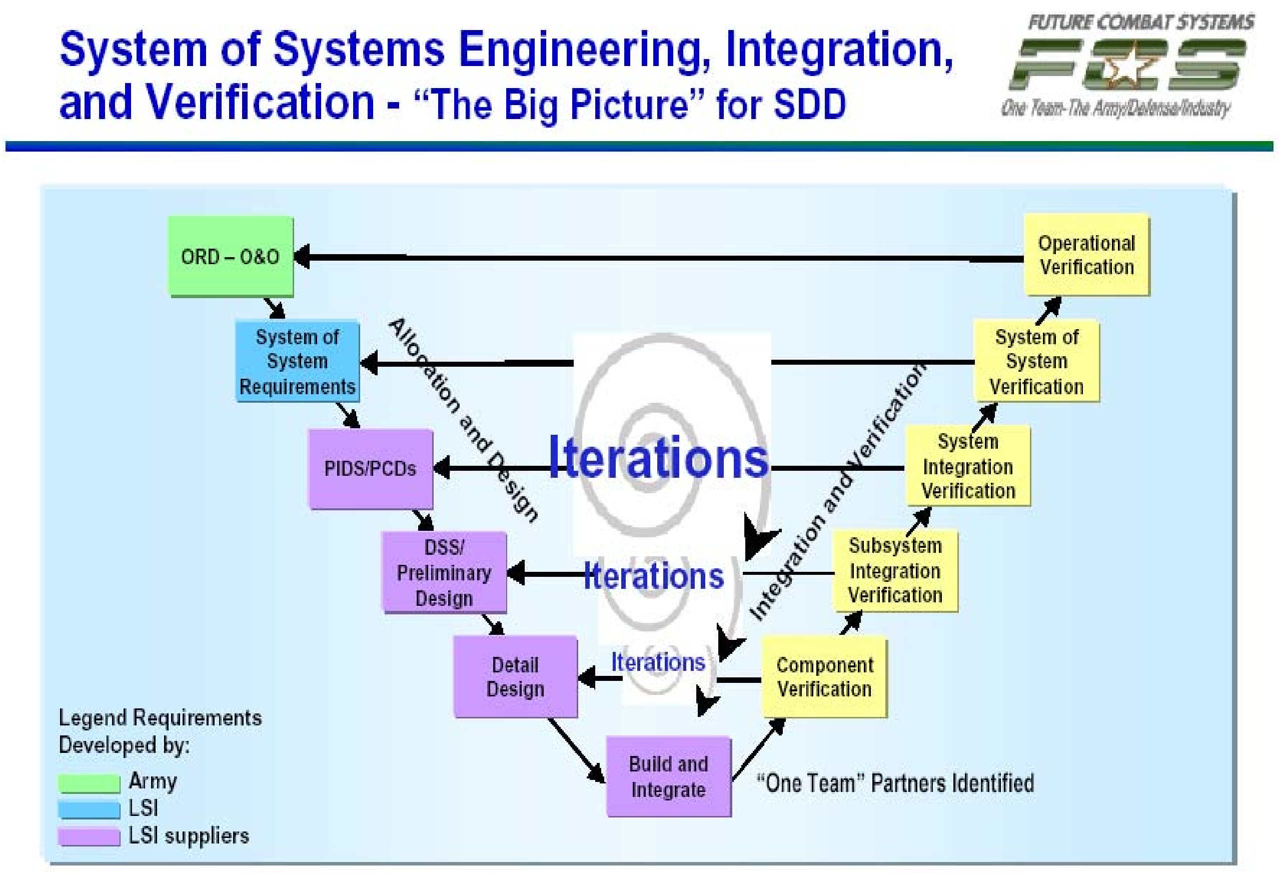

System of Systems Engineering, Integration, and Verification - "The Big Picture" for SDD
FUTURE COMBAT SYSTEMS
FCS
One Team-The Army/Defense/Industry
ORD – O&O
System of System Requirements
PIDS/PCDs
DSS/ Preliminary Design
Detail Design
Build and Integrate
Component Verification
Subsystem Integration Verification
System Integration Verification
System of System Verification
Operational Verification
Allocation and Design
Integration and Verification
Iterations
Iterations
Iterations
"One Team" Partners Identified
Legend Requirements Developed by:
🟩 Army
🟦 LSI
🟪 LSI suppliers

Figure 9. The Future Combat System "V" Model for the Re-configurable Service-Based Simulation Architectures for Boeing FCS [Krone 2004.]

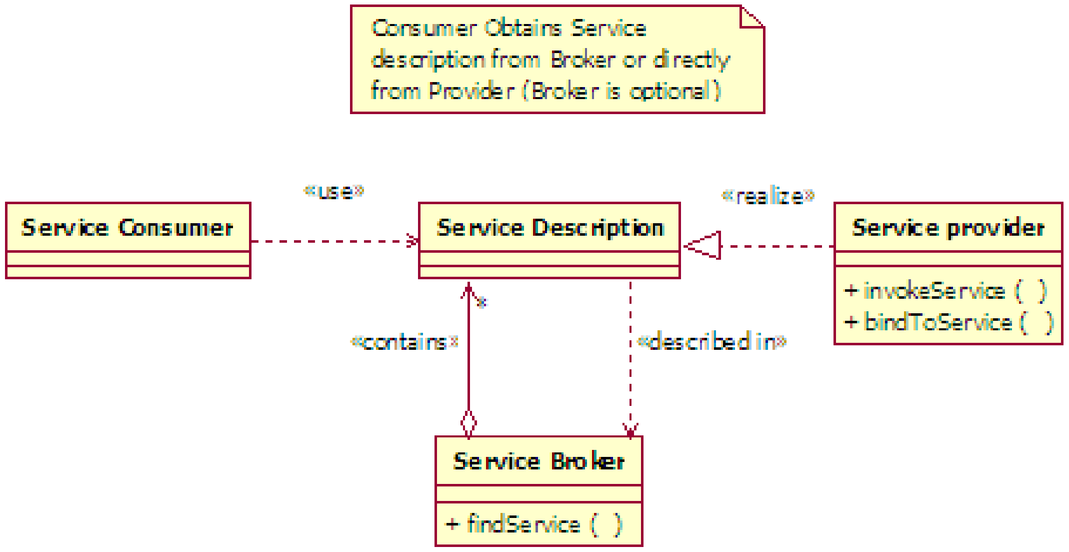


Figure 10. Conceptual Model Of Service-Oriented Architecture (SOA) [Arsanjani November 9 2004.]

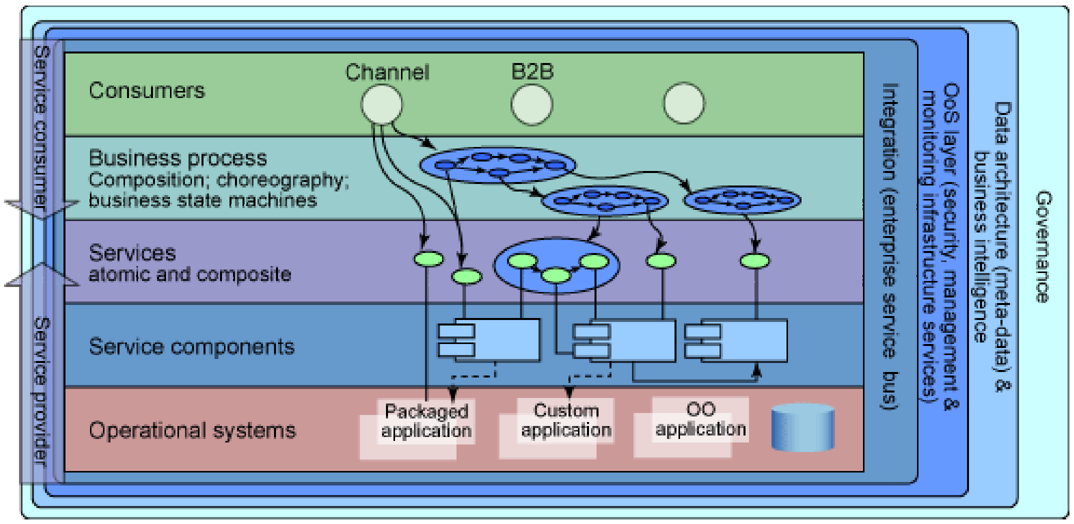

Service consumer
Service provider
Consumers
Channel
B2B
Business process
Composition; choreography; business state machines
Services
atomic and composite
Service components
Operational systems
Packaged application
Custom application
OO application
Integration (enterprise service bus)
QoS layer (security, management & monitoring infrastructure services)
Data architecture (meta-data) & business intelligence
Governance

Figure 11. The Different Layers of Service-Oriented Architecture (SOA) [Arsanjani November 9 2004.]

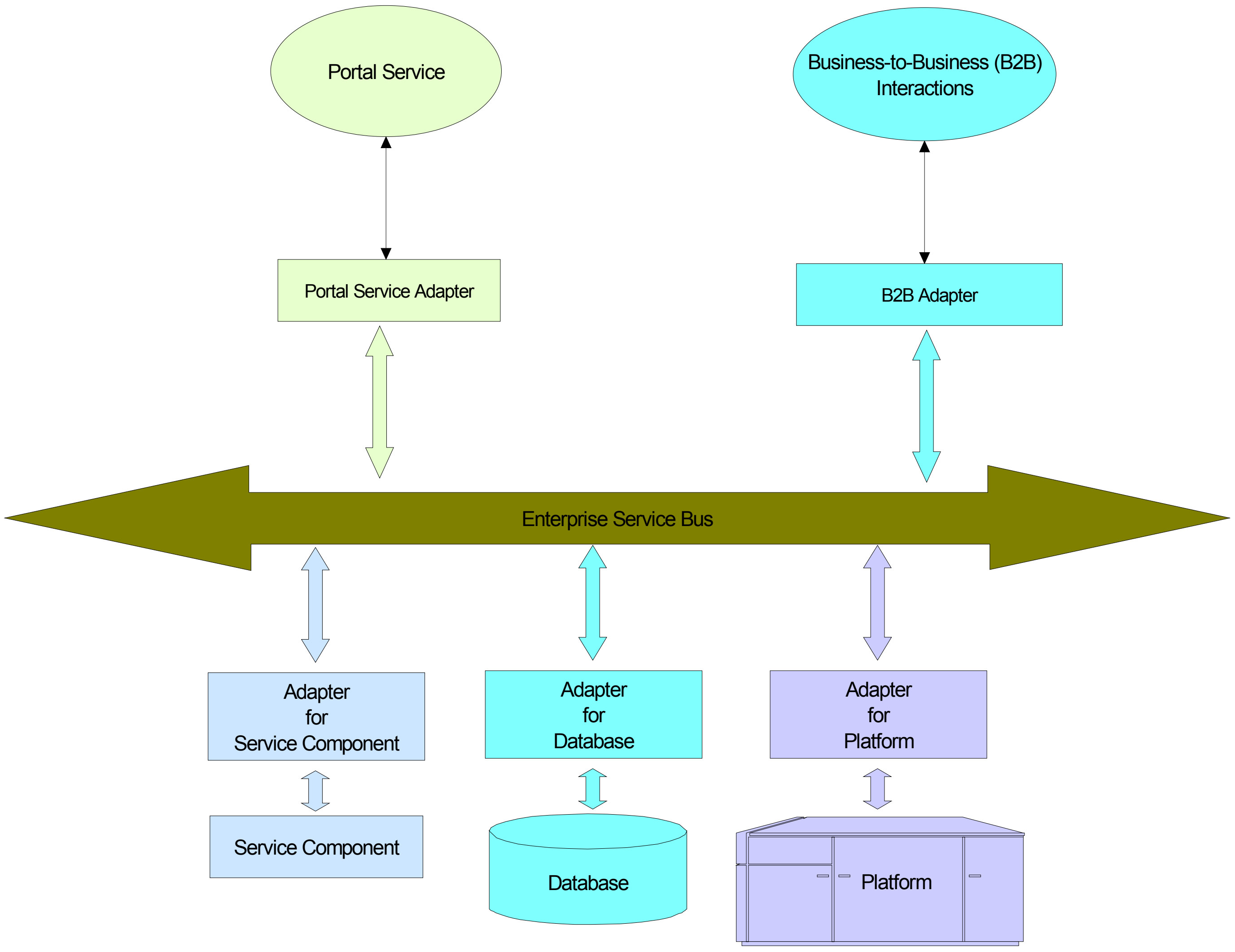


Figure 12. Enterprise Service Bus.

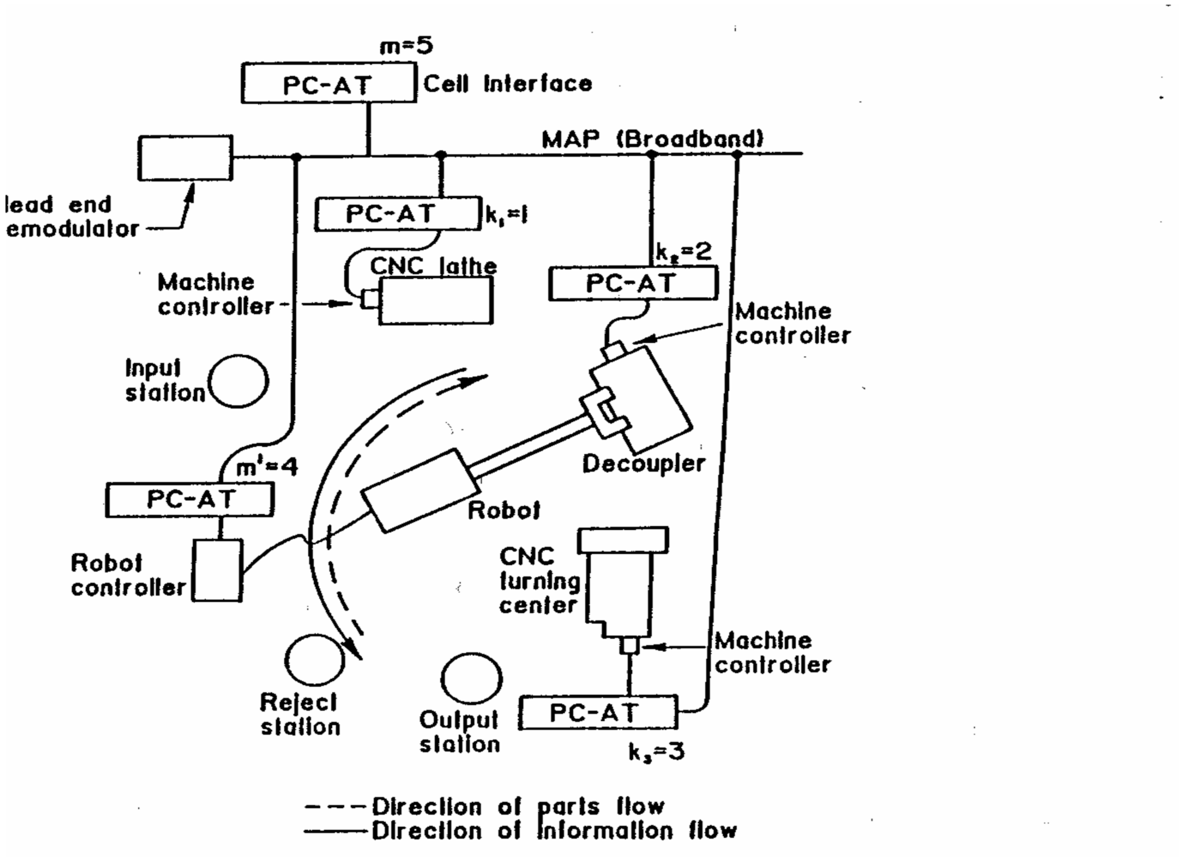


Figure 13. Distributed Intelligent Control Systems Simulation Model For Unmanned Robot Cell In A Manufacturing Automation Protocol (MAP) Environment [Chang and Nyamekye 1993.]

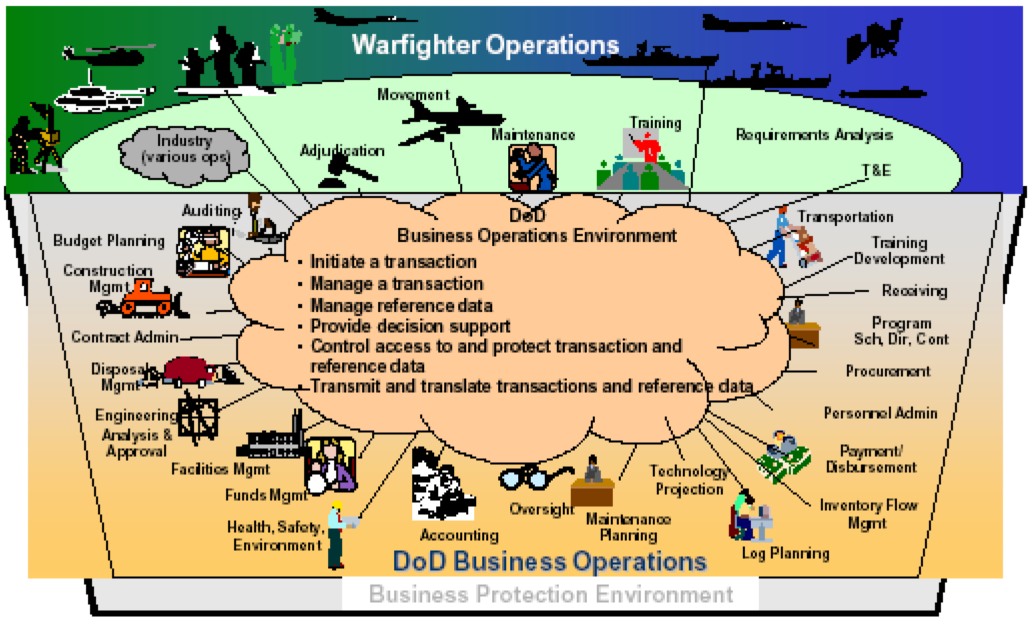
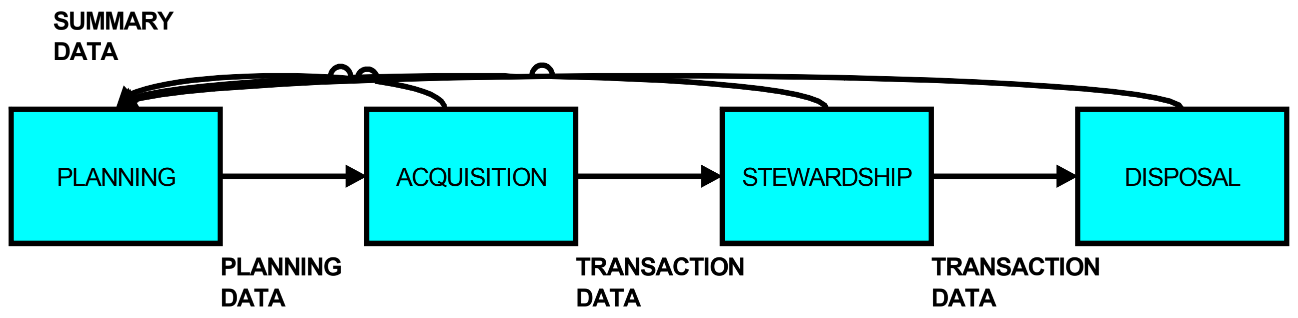


Figure 14. The Concept Of Electronic Commerce In The DOD, Including The Relevant Business Areas And Support To The Warfighter [Defense Electronic Business 2000.]

	SUBJECT DATABASES																							
PROCESSES	Customer	Budget	Financial	Vendor	Procurements	Materials inventory	Fin. goods inventory	Orders	Cost	Sales	Sales territory	Payments	Planning	Employee	Salaries	Facilities	Work in progress	Machine load	Open requirement	Shop floor routings	Product	Product design	Parts master	Bill of materials
Market analysis	R									R	R		R								R			
Product range review									R				R											
Sales forecasting	R	C								R	R		C								R			
Financial planning		R							R				C											
Capital acquisition		R	C										R											
Funds management		R	R																					
Product design													R								C	C	C	
Product pricing		R							R												C	R		
Product spec. maint.																					R		C	C
Materials requirements				R															C		R		R	R
Purchasing				C	C																R		R	
Receiving				R	R	R															R		R	
Inventory control						C											R							
Quality control					C																			
Capacity planning				R												R		C	R	R				
Plant scheduling																R	C	R		R	R			
Workflow layout																C				C	R			
Materials control				R													R				R			R
Sizing and cutting																		R		C	R			R
Machine operations																		R		C				
Territory management	C							R			R													
Selling										C	C													
Sales administration								R			R													
Customer relations	R							R		R														
Finished stock control							C										R				R			
Order servicing	R							C													R			
Packing								R													R			
Shipping																					R			
Creditors and debtors	R			R								R												
Cash flow	R			R	R			R	R			R			R		R							
Payroll												C		R	R									
Cost accounting					R				C			C												R
Budget planning		C	R						R	R			R				R							
Profitability analysis									R	R														
Personnel planning			R											C	C									
Recruiting														R	R									
Compensation policy			R											R	R									

Figure 15. Cartesian Product (Cross Product) of Databases And Processes [Martin Book II 1990.]; C= Create, R=Read, U= Update, D = Delete



Requirements	Procurement	Warehousing	Selling
Design	Recruiting	Inventory Control	Delivery
Measurement	Implementation	Maintenance	Order Selling
Control	Creation	Support	Shipping
Accounting	Fabrication	Tracking	Fleet Management
Market Research	Development	Modification	Accounts Payable
Forecasting	Engineering	Quality Control	Retirement
Capacity Planning	Production Scheduling	Packing	Equipment Dispositio
Evaluation	Testing	Repair	Scrap

Figure 16. The Four-Stage Lifecycle Of Products, Services, And Resources Showing The Sequence Of Processes Through The Stages In The Lifecycle Of Each Product, Service, Or Resource: ***Planning, Acquisition, Stewardship, And Disposal*** [Martin Book II 1990.]

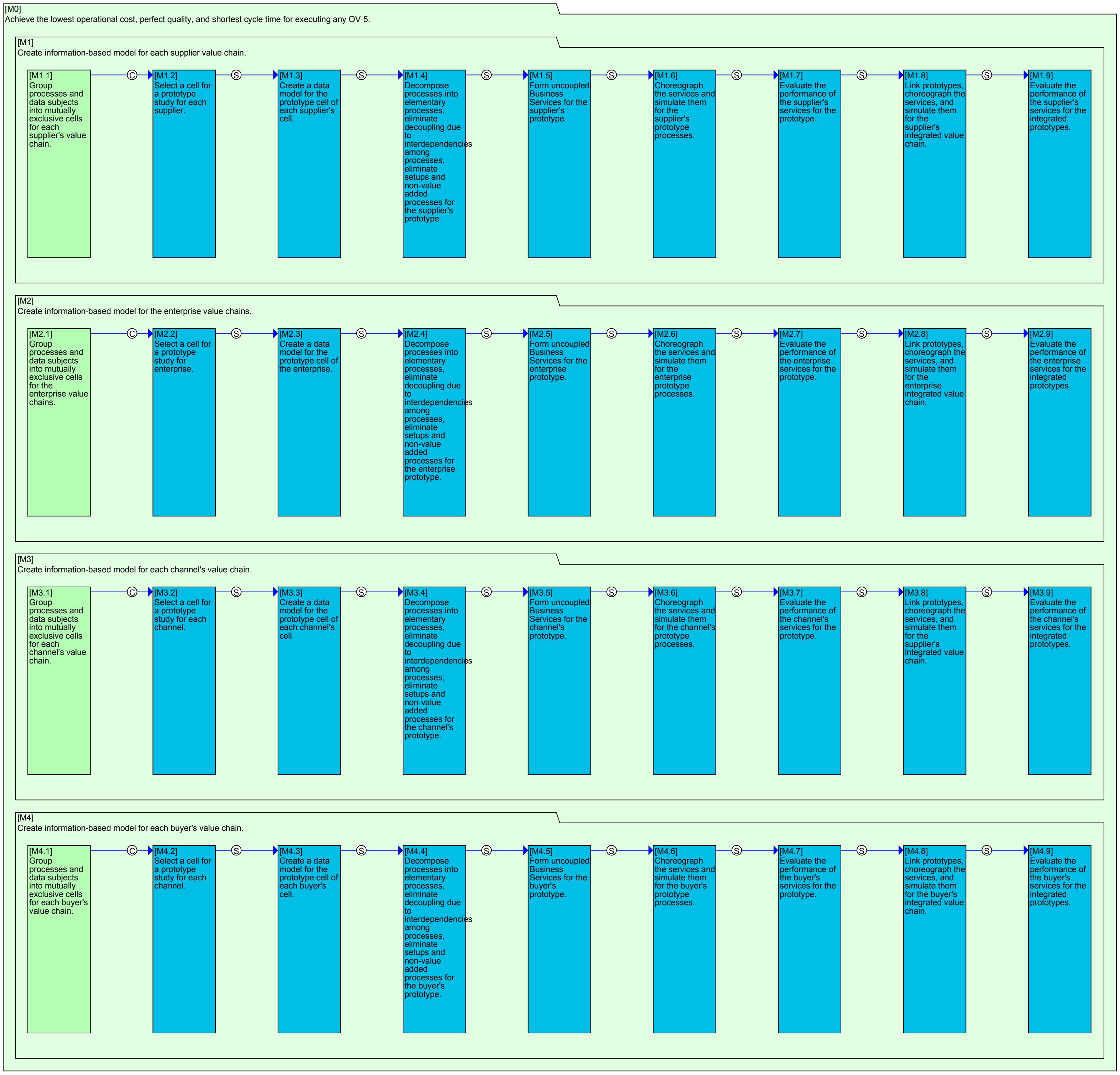

[M0]
Achieve the lowest operational cost, perfect quality, and shortest cycle time for executing any OV-5.
[M1]
Create information-based model for each supplier value chain.
[M1.1] Group processes and data subjects into mutually exclusive cells for each supplier's value chain.
[M1.2] Select a cell for a prototype study for each supplier.
[M1.3] Create a data model for the prototype cell of each supplier's cell.
[M1.4] Decompose processes into elementary processes, eliminate decoupling due to interdependencies among processes, eliminate setups and non-value added processes for the supplier's prototype.
[M1.5] Form uncoupled Business Services for the supplier's prototype.
[M1.6] Choreograph the services and simulate them for the supplier's prototype processes.
[M1.7] Evaluate the performance of the supplier's services for the prototype.
[M1.8] Link prototypes, choreograph the services, and simulate them for the supplier's integrated value chain.
[M1.9] Evaluate the performance of the supplier's services for the integrated prototypes.
[M2]
Create information-based model for the enterprise value chains.
[M2.1] Group processes and data subjects into mutually exclusive cells for the enterprise value chains.
[M2.2] Select a cell for a prototype study for enterprise.
[M2.3] Create a data model for the prototype cell of the enterprise.
[M2.4] Decompose processes into elementary processes, eliminate decoupling due to interdependencies among processes, eliminate setups and non-value added processes for the enterprise prototype.
[M2.5] Form uncoupled Business Services for the enterprise prototype.
[M2.6] Choreograph the services and simulate them for the enterprise prototype processes.
[M2.7] Evaluate the performance of the enterprise services for the prototype.
[M2.8] Link prototypes, choreograph the services, and simulate them for the enterprise integrated value chain.
[M2.9] Evaluate the performance of the enterprise services for the integrated prototypes.
[M3]
Create information-based model for each channel's value chain.
[M3.1] Group processes and data subjects into mutually exclusive cells for each channel's value chain.
[M3.2] Select a cell for a prototype study for each channel.
[M3.3] Create a data model for the prototype cell of each channel's cell.
[M3.4] Decompose processes into elementary processes, eliminate decoupling due to interdependencies among processes, eliminate setups and non-value added processes for the channel's prototype.
[M3.5] Form uncoupled Business Services for the channel's prototype.
[M3.6] Choreograph the services and simulate them for the channel's prototype processes.
[M3.7] Evaluate the performance of the channel's services for the prototype.
[M3.8] Link prototypes, choreograph the services, and simulate them for the supplier's integrated value chain.
[M3.9] Evaluate the performance of the channel's services for the integrated prototypes.
[M4]
Create information-based model for each buyer's value chain.
[M4.1] Group processes and data subjects into mutually exclusive cells for each buyer's value chain.
[M4.2] Select a cell for a prototype study for each channel.
[M4.3] Create a data model for the prototype cell of each buyer's cell.
[M4.4] Decompose processes into elementary processes, eliminate decoupling due to interdependencies among processes, eliminate setups and non-value added processes for the buyer's prototype.
[M4.5] Form uncoupled Business Services for the buyer's prototype.
[M4.6] Choreograph the services and simulate them for the buyer's prototype processes.
[M4.7] Evaluate the performance of the buyer's services for the prototype.
[M4.8] Link prototypes, choreograph the services, and simulate them for the buyer's integrated value chain.
[M4.9] Evaluate the performance of the buyer's services for the integrated prototypes.
C
S

Figure 17. Module Diagram, from AcclaroDFSS [AcclaroDFSS], for a Diversified Enterprise Value System (Figure 7b) for the Business Activities for C4ISR. Please zoom page at 200% or more to see the text on Figure 17. Please note that module M2 is selected as an example for business value system of a Diversified Enterprise for the C4ISR.

	SUBJECT DATABASES																							
PROCESSES	Planning	Budget	Financial	Product	Product design	Parts master	Bill of materials	Open requirements	Vendor	Procurements	Materials inventory	Machine load	Work in progress	Facilities	Shop floor routings	Customer	Sales	Sales territory	Fin. goods inventory	Orders	Payments	Cost	Employee	Salaries
Market analysis	R			R												R	R	R						
Product range review	R																					R		
Sales forecasting	C	C		R												R	R	R						
Financial planning	C	R																				R		
Capital acquisition	R	R	C																					
Funds management		R	R																					
Product design	R			C	C	C																		
Product pricing		R		C	R																	R		
Product spec. maint.				R		C	C																	
Materials requirements				R		R	R	C	R															
Purchasing				R		R			C	C														
Receiving				R		R			R	R	R													
Inventory control											C		R											
Quality control										C														
Capacity planning								R	R			C		R	R									
Plant scheduling				R								R	C	R	R									
Workflow layout				R										C	C									
Materials control				R			R		R				R											
Sizing and cutting				R			R					R			C									
Machine operations												R			C									
Territory management																C		R		R				
Selling																	C	C						
Sales administration																		R		R				
Customer relations																R	R			R				
Finished stock control				R									R						C					
Order servicing				R												R				C				
Packing				R																R				
Shipping				R																				
Creditors and debtors									R							R					R			
Cash flow									R	R			R			R				R	R	R		R
Payroll																					C		R	R
Cost accounting							R			R											C	C		
Budget planning	R	C	R										R				R					R		
Profitability analysis																	R					R		
Personnel planning			R																				C	C
Recruiting																							R	R
Compensation policy			R																				R	R

Figure 18. Partition of Databases and Processes Set Into Logical Functional Groupings Or Natural Business Areas, from Figure 15 [Martin Book II 1990; Gersting 1998.]; C= Create, R=Read, U= Update, D = Delete

	ENTITY TYPES															
LEAF ACTIVITIES	Employee	Contact Employee	Applicant	HR Compensation Regs, Plans, etc.	HR Benefits Regs & Plans	HR Staffing Requirements & Plans	Job Requisition	Stockholder	Boardmember	Misc. Contacts/VIPs	Financial Plans	Accounting Regs, Practices	Ledger Accounts	Customer Purchase Order/Invoice	Customer Payments	Other Invoice
Evaluate Financial Proposals																
Estimate Near - Term Earnings														R		
Prepare Budget	R	R		R	R						C	R	U			
Receive Funds												R		R	C	U
Pay Funds	R											R				
Report Finances	R											R	U	R	R	R
Administer Taxes												R	R		R	R
Maintain Financial Reg, Policies											R	U				
Audit Finances												R	R		R	R
Manage Financial Investments								C				R				
Plan Human Resources	R	R				U	U		R		R					
Acquire Personnel	C	C	C			R	R		U							
Position People in Jobs			R			R	U		R							
Terminate/Retire People	U	U							U							
Plan Career Paths	U			R	R	R										
Develop Skills/Motivation	U	U			R	R										
Manage Individual Emp Relations	U	U			R											
Manage Benefits Programs					C											
Comply with Govt HR Regulations	R			R												
Maintain HR Regs, Policies				C		C										

Figure 19. Cartesian Product (Cross Product) of Entity Types And Leaf Activities [Martin Book II 1990.] Note: The Last Two Natural Business Areas, Subsets 7 & 8 (From Figure 18) Are Used For Creating Figure 19. C= Create, R=Read, U= Update, D = Delete

LEAF ACTIVITIES	ENTITY TYPES															
	Financial Plans	Ledger Accounts	Customer Payments	Other Invoice	Accounting Regs, Practices	Stockholder	HR Staffing Requirements & Plans	Job Requisition	Contact Employee	Employee	Boardmember	Applicant	HR Benefits Regs & Plans	Misc. Contacts/VIPs	Customer Purchase Order/Invoice	HR Compensation Regs, Plans, etc.
Evaluate Financial Proposals																
Estimate Near - Term Earnings															R	
Prepare Budget	C	U			R				R	R			R			R
Receive Funds			C	U	R										R	
Pay Funds					R					R						
Report Finances		U	R	R	R					R					R	
Administer Taxes		R	R	R	R											
Maintain Financial Reg, Policies	R				U											
Audit Finances		R	R	R	R											
Manage Financial Investments					R	C										
Plan Human Resources	R						U	U	R	R	R					
Acquire Personnel							R	R	C	C	U	C				
Position People in Jobs							R	U			R	R				
Terminate/Retire People									U	U	U					
Plan Career Paths							R			U			R			R
Develop Skills/Motivation							R		U	U			R			
Manage Individual Emp Relations									U	U			R			
Manage Benefits Programs													C			
Comply with Govt HR Regulations										R						R
Maintain HR Regs, Policies							C									C

Figure 20. Partition Of Subsets, In Figure 19, Into Mutually Exclusive Subsets Of Leaf Activity-Entity Types Using The Cartesian Cross Product of Row And Column [Martin Book II 1990; Gersting 1998.]; C= Create, R=Read, U= Update, D = Delete

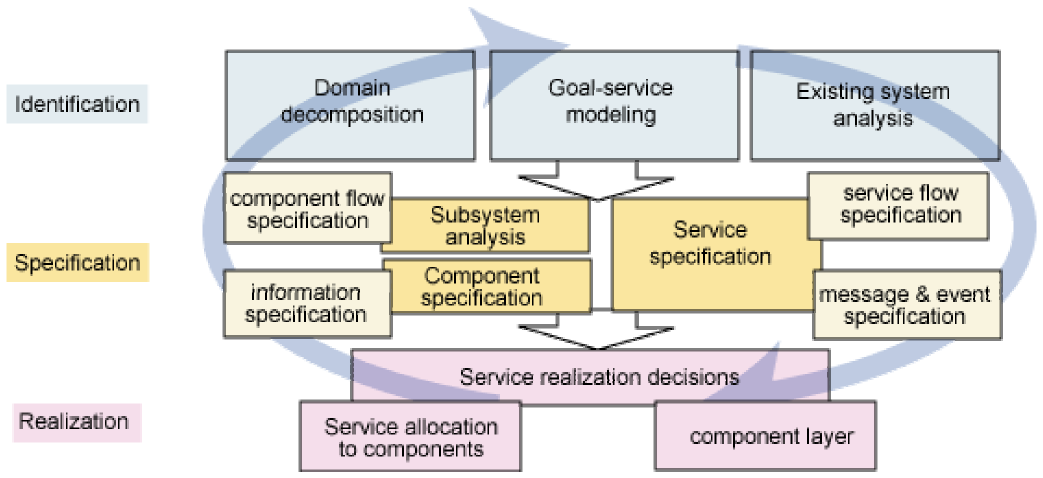


Figure 21. The Service-Oriented Modeling And Architecture Methodology [Arsanjani November 9 2004.]

```
MATERIALS
.
.   PURCHASING (1)
.   .
.   .       CREATE  REQUISITION (A)                                   *
.   .
.   .       SUPPLIER INFORMATION
.   .   .
.   .   .       RECORD SUPPLIER PERFORMANCE DATA (B)                  *
.   .   .
.   .   .       ANALYZE SUPPLIER PERFORMANCE (C)                      *
.   .   .
.   .   .       SELECT SUPPLIER (D)                                   *
.   .   .
.   .   CREATE PURCHASE ORDER (E)                                     *
.   .
.   .   CANCEL PURCHASE ORDER (F)                                     *
.   .
.   .   FOLLOW UP DELIVERY (G)                                        *
.   .
.   .   EXCEPTION PROCESSING
.   .   .
.   .   .   CREATE SPECIAL ORDER  (H)                                 *
.   .   .
.   .   .   SUPPLIER NONDELIVERY
.   .   .   .
.   .   .   .   TERMINATE ORDER (I)                                   *
.   .   .   .
.   .   .   .   CREATE NEW ORDER (J)                                  *
.   .   .
.   .   CREATE INFORMATION FOR ACCOUNTS PAYABLE (K)                   *
.   .
.   RECEIVING
```

Figure 22. Leaf Activities for Service Classification Or Categorization and Service Allocation for The ***Purchasing*** Process (Data from Figure 15 or 18) [Martin Book II 1990.]

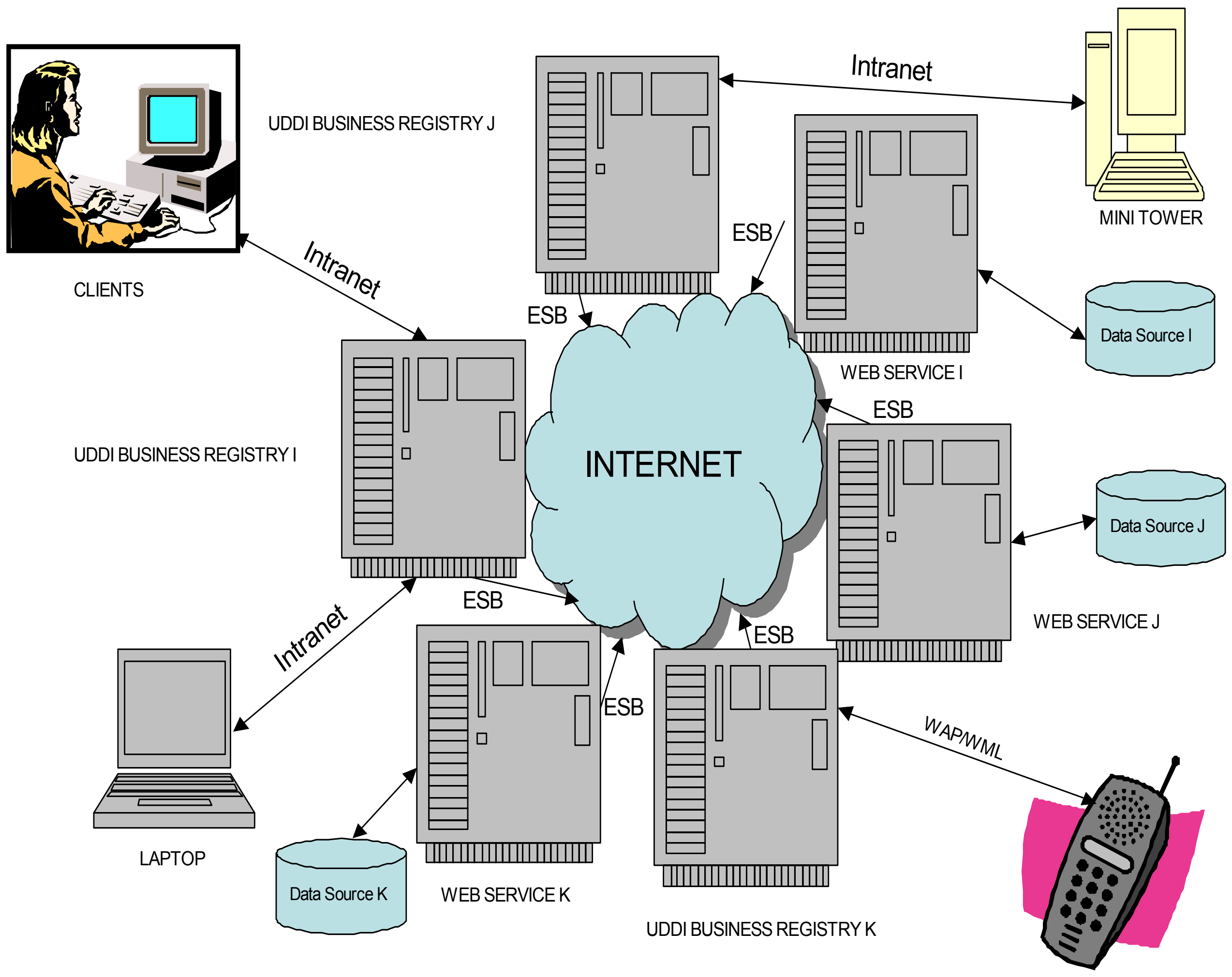


Figure 23. Service-Oriented Architecture (SOA) Model Of The Generic C4ISR; UDDI = Universal Description, Discovery and Integration. The Universal Description, Discovery and Integration), UDDI provides a standard mechanism for registering and discovering Web services. According to Bloomberg et al. [Bloomberg et al. 2006], a Web service is *a standard-based, contracted interface to software functionality*. The Web Services Description Language (WSDL) provides a standard mechanism for describing the programmatic interfaces of the Web services [Zimmermann et al. 2005].

APPENDIX B: PROPERTIES OF COHERENT LEAF ACTIVITIES

We have borrowed from the work of Winter et al. [Winter et al. 1981] to list the qualities to look for in well-formed leaf activities, coherent leaf activities, or simply coherent activities [Winter et al. 1981.]

1. *A coherent activity produces some clearly identifiable result. Its purpose is to produce this result. The result may be a marketable product, a part of a product, an idea, a decision, a sale, a set of alternatives, a paycheck, a prospect, etc. It should be possible to identify the purpose/result of the activity in a single simple sentence. By contrast, a poorly formed activity yields no identifiable result at all, or it produces a number of unrelated results.*
2. *A coherent activity has clear boundaries. At a given time, one can say unambiguously who is working on it and who is not. Over time, one can identify the moments when work on the activity starts and stops. Transitions between coherent activities are well-marked. Incoherent activities overlap and blend into one another; one cannot localize when and where they are going on.*
3. *A coherent activity is carried out as a unit. It is done by a single person or well-defined group of people who work as a team to produce the result. Management responsibility for the activity is similarly well-defined and vested in a single person or group. An ill-defined activity may be carried out by an ill-defined group of people, i.e., it may be unclear who does it. Or it may be done by a well-defined group of people whose jobs have something in common, but who do not work as a team: they do not interact, communicate, cooperate to produce the activity's result, perhaps because they are dispersed throughout the enterprise in a way that never brings them into contact with one another.*
4. *Once initiated, a coherent activity is self-contained – it proceeds largely independently of other activities. If an alleged activity requires intense interaction along the way with another alleged activity, consider recasting them as a single activity. Another way of putting this is that the interactions within the team that carries out a coherent activity will be rich compared to the interactions between teams working on different activities.*

Though not absolute requirements when forming an activity model, Winter et al. recommend using such points as heuristics in creating an activity model. Winter et al. points are in agreement with the design of a leaf activity or fine-grained service in Service-Oriented Architecture (SOA), the design of leaf activity in Integrated Manufacturing Production Systems (IMPSs) and the design of leaf activity in Activity-Based Simulation (ABS).

As noted before a leaf activity or fine-grained service is the fundamental basis for the Service-Oriented Architecture (SOA).

Colan also puts it this way for creating services in SOA [Colan April 21 2004]: *Services should be independent, self-contained requests, which do not require information or state from one request to another when implemented. Services should not be dependent on the context or state of other services. When dependencies are required, they are best defined in terms of common business processes, functions, and data models, not implementation artifacts (like a session key). Of course, requester applications require persistent state between service invocations, but this should be separate from the service provider.*

Here is an example of the ***wrong way*** *to define a conversation:*

Requester: “What is Bruce’s checking account balance?"
Provider: “$x"
Requester: “And what is his credit limit?"
Provider: “$y"

The provider is required to remember Bruce's account between requests, which introduces complexity into the service implementation. Stateless service design would redefine the conversation as follows:

Requester: "What is Bruce's checking account balance?"
Provider: "$x"
Requester: "What is Bruce's credit limit?"
Provider: "$y"

Please note the term stateless services mean the same as uncoupled services in axiomatic theory.