NPS-TE-12-003

ACQUISITION RESEARCH SPONSORED REPORT SERIES

Virtualization of System of Systems Test and Evaluation

4 June 2012

by

Major Seth F. Gibson, USMC

Advisors: Dr. John S. Osmundson, Associate Professor, and
Brad Naegle, Senior Lecturer
Graduate School of Operational & Information Sciences

Naval Postgraduate School

Prepared for: Naval Postgraduate School, Monterey, California 93943

ACQUISITION RESEARCH PROGRAM
GRADUATE SCHOOL OF BUSINESS & PUBLIC POLICY
NAVAL POSTGRADUATE SCHOOL

The research presented in this report was supported by the Acquisition Chair of the Graduate School of Business & Public Policy at the Naval Postgraduate School.

To request Defense Acquisition Research or to become a research sponsor, please contact:

NPS Acquisition Research Program
Attn: James B. Greene, RADM, USN, (Ret.)
Acquisition Chair
Graduate School of Business and Public Policy
Naval Postgraduate School
555 Dyer Road, Room 332
Monterey, CA 93943-5103
Tel: (831) 656-2092
Fax: (831) 656-2253
E-mail: jbgreene@nps.edu

Copies of the Acquisition Sponsored Research Reports may be printed from our website: www.acquisitionresearch.net

ACQUISITION RESEARCH PROGRAM
GRADUATE SCHOOL OF BUSINESS & PUBLIC POLICY
NAVAL POSTGRADUATE SCHOOL

ABSTRACT

Virtualization is the use of a software application to emulate the physical performance of a computer, including the central processing unit (CPU), storage, network device, random access memory (RAM), and operating system (OS) through executable data files. The virtualization software application allows for multiple virtual machines to exist on a single set of physical hardware. This technology can increase the flexibility of the hardware while reducing hardware configuration time. Virtualization technology will improve the Department of Defense (DoD) system of systems (SoS) test and evaluation (T&E) process. The implementation of virtualized systems within SoS will create three primary benefits. First, test personnel can improve configuration management for all component systems. Second, test personnel can reduce test environment setup time. Third, test personnel can improve the scalability of SoS architectures. The success of a DoD information system depends on its ability to meet the established criteria of cost, schedule, and performance. By appropriately integrating virtualization technology into the SoS T&E process, system program managers can improve the likelihood of meeting these criteria.

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank my dear wife, Natalie, for her support and encouragement throughout my time at the Naval Postgraduate School. You are my best friend, and I am blessed to have you as my wife. Thank you to my children, Shey, Megan, and Cole, for their inspiration to enjoy each moment and their reminder that I was a kid once, too. Special thanks to my parents, who have provided unwavering support and continue to inspire me to reach new personal, professional, and academic heights. I also want to share my appreciation with my advisors, Dr. John Osmundson, Dr. Deborah Goshorn, and LTCOL Brad Neagle (Ret.). Professor Osmundson, your insight, and experience have allowed me to develop a thesis worthy of the Naval Postgraduate School. Professor Goshorn, without the aid of your research, I would have been unable to complete this thesis. LTCOL Neagle, your instruction in the field of acquisition has transformed my professional and educational experience. Finally, I would like to express my gratitude for the tremendous guidance and support of Ms. Karey Shaffer and Ms. Tera Yoder and the team of staff and editors of the Acquisition Research Program. Thank you all for your effort in support of my research.

THIS PAGE INTENTIONALLY LEFT BLANK

NPS-TE-12-003

ACQUISITION RESEARCH SPONSORED REPORT SERIES

Virtualization of System of Systems Test and Evaluation

4 June 2012

by

Major Seth F. Gibson, USMC

Advisors: Dr. John S. Osmundson, Associate Professor, and
Brad Naegle, Senior Lecturer
Graduate School of Operational & Information Sciences

Naval Postgraduate School

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

ADCCP	Army Data Center Consolidation Plan
AFATDS	Advanced Field Artillery Tactical Data System
C2	Command and Control
C4I	Command, Control, Communication, Computers, and Intelligence
CAC2S	Common Aviation Command and Control System
CoC	Combat Operation Center
COI	Community of Interest
COTS	Commercial Off-the-Shelf
CPOF	Command Post of the Future
CPU	Central Processing Unit
DAS	Defense Acquisition System
DCGS	Digital Common Ground System
DDR	Double Data Rate
DGIAS	Distributed Global Information Grid (GIG) Intelligence Automation System
DIPR	Detect–Identify–Predict–React
DoD	Department of Defense
DVD	Digital Video Disc
E2E	End to End
ECO	Engineering Change Order
EMD	Engineering and Manufacturing and Development
FCS	Fixed Camera System
FoS	Family of Systems

GB	Gigabyte
GHub	Geospatial Hub
GHz	Gigahertz
GPS	Global Positioning System
GUI	Graphical User Interface
HDX	High Definition User Experience
IaaS	Infrastructure as a Service
IBM	International Business Machine
IC	Internal Clock
IP	Internet Protocol
ISR	Intelligence Surveillance and Reconnaissance
IT	Information Technology
JCIDS	Joint Capabilities Integration Development System
LUN	Logical Unit Number
Mbit/sec	Megabit Per Second
MAGTF	Marine Air Ground Task Force
MC3T	MAGTF C4I Capability and Certification Test
MCIC	MAGTF C4I Integration and Certification
MCTSSA	Marine Corps Tactical Systems Support Activity
MHz	Megahertz
ms	Millisecond
NGO	Non-Governmental Organization
NIC	Network Interface Card
NIST	National Institute of Standards and Technology
ODBC	Open Database Connectivity

OS	Operating System
PaaS	Platform as a Service
PC	Personal Computer
PCoIP	Personal Computer over Internet Protocol
PM	Program Manager
PoE	Power over Ethernet
PPBE	Planning, Programming, Budgeting, and Execution
QFD	Quality Function Deployment
R&D	Research and Development
RAM	Random Access Memory
RDP	Remote Desktop Protocol
RTC	Real Time Clock
SaaS	Software as a Service
SAN	Storage Area Network
SDK	Software Development Kit
SoS	System of Systems
SQL	Standard Query Language
T&E	Test and Evaluation
TB	Terabyte
TD	Test Director
TEMP	Test and Evaluation Master Plan
TES	Test and Evaluation Strategy
UAV	Unmanned Aerial Vehicle
USB	Universal Serial Bus
VDI	Virtual Desktop Infrastructure

VM	Virtual Machine
VMM	Virtual Machine Monitor
VV&T	Validation, Verification, and Testing
WIPT	Working-Level Integrated Product Team
Y2K	Year 2000

I. INTRODUCTION

A. BACKGROUND AND HYPOTHESES

An information technology (IT) or command and control (C2) system's performance in the test and evaluation (T&E) phase of a program's life cycle will impact its success or failure. Program managers (PM) must choose wisely where to distribute their budget in order to control development costs and program schedules. To maximize limited budgets, it is a manager's duty to find improved productivity in business processes and ensure the effective use of IT infrastructure. One technology designed to achieve both efficient business processes and the efficient use of infrastructure is virtualization. Virtualization software decomposes the physical elements of a computer into a set of executable software files. This transformation allows for the emulation of a physical computer through software, which provides administrators with improved process efficiencies throughout the IT infrastructure. The implementation of virtualization technology in T&E can reduce hardware and manpower costs while decreasing lab configuration schedules. This increase in productivity reduces schedule time and cost, thus managers can apply resources to other critical areas of the program.

To realize the efficiencies gained by virtualization in T&E, the test environment should ideally mimic a large system of systems (SoS) setting. System of systems architectures incorporate multiple IT systems working together either in sequence or in parallel to produce some output. For example, the Department of Defense (DoD) intelligence community relies on multiple intelligence surveillance and reconnaissance (ISR) platforms such as unmanned aerial vehicles (UAVs) or manned fixed-wing aircraft to collect data on a given target. The collection data must be processed by unique systems and then transmitted to analysts for further study. These disparate inputs will eventually come together within a single system to provide a cohesive understanding of a given target. To ensure each of these systems work together, they must be

tested together in an environment closely matching the operational setting. By creating SoS architectures in virtualization, administrators can create functional models of systems, which allow for new software updates or patches to be easily integrated or new products to be tested without adjusting the existing system connections.

To demonstrate the benefits of virtualization in the test environment, a study was completed of the Distributed Global Information Grid (GIG) Intelligence Automation System (DGIAS). The DGIAS laboratory combined numerous systems working together to connect several collection platforms and database systems. To validate the hypothesis, a hardware consolidation plan and a process model of the system were developed to verify a reduction of hardware requirements and configuration time.

B. BENEFITS OF STUDY

With this thesis, I seek to identify the efficiencies gained from the use of virtualization in system of systems test and evaluation. Given the large scale of most SoS environments, a solution must be developed which combines IT flexibility and scalability without increasing manpower. In this thesis, I outline for system testers the benefits and limitations of implementing virtualization technology in an SoS T&E setting.

C. RESEARCH QUESTIONS

1. What are the ideal system traits for implementing virtualized system of systems test and evaluation?
2. What type of virtualization environment should be created to benefit the system of systems test and evaluation process?
3. What are the efficiencies achieved through the use of virtualization in system of systems test and evaluation?

4. What are the limitations of using virtualization in a system of systems test and evaluation environment?

D. THESIS ORGANIZATION

Following the current chapter's introduction to virtualization and SoS T&E, in Chapter II, I introduce some important concepts in the field of virtualization. Then I discuss the concept of cloud computing, including the different service and deployment models. Finally, I review the limitations of the virtualization technology.

In Chapter III, I discuss the fundamentals of system of systems T&E in the DoD acquisition process as well as T&E methodologies. Although numerous T&E methodologies exist, I only discuss the most accepted and practiced techniques. Next, I introduce the DoD SoS initiatives as a framework for DoD acquisition T&E.

Following the presentation of the background information, in Chapter IV I examine in detail the Distributed Global Information Grid (GIG) Intelligence Automation System (DGIAS). This chapter explores the physical hardware, software, and processes of the DGIAS. Then I propose a virtualized architecture for the DGIAS to determine the differences between a fully physical implementation and a hybrid (physical and virtualized) architecture. Finally, I present a process model of the system to identify efficiencies achieved by a virtualization implementation in a T&E environment.

In Chapter V, I summarize the findings of this thesis and suggest several opportunities for future research at the intersection of virtualization or cloud computing and SoS T&E.

THIS PAGE INTENTIONALLY LEFT BLANK

II. VIRTUALIZATION AND CLOUD COMPUTING

A. BACKGROUND

1. Early Virtualization

Throughout their history, virtual machines (VMs) have sought to divide the computing components of hard disk storage, random access memory (RAM), and central processing unit (CPU) of a single large computer into several smaller computers for use by multiple users. The separation of components is achieved through virtualization software. Virtualization software has matured since its first introduction in the late 1960s at IBM® (International Business Machine). The production of the CP-40 (Control Program-40), developed in concert with the IBM System/360 Model 40 (Adair, Bayles, Comeau, & Creasy, 1966), was the first system to host multiple operating systems (OSs) on a shared platform and provided the foundation for virtualization (Varian, 1991). Current computer capabilities and network throughput have completed the original vision of virtualization. Today, organizations can operate the equivalent of a historic mainframe on a single rack of servers. This is made possible by continued miniaturization and commoditization of computer components. Computers in the form of blade servers now contain multi-terabyte internal storage, hundreds of gigabytes of RAM, and multi-core processors. This computing power is equivalent to seven to ten desktop computers and is the primary enabler for virtualization.

2. Virtualization System Elements

Parmalee, Peterson, Tillman, & Hatfield (1972) outlined the capabilities of virtualization in the early days of VM with some guiding principles. The following four principles define the VM tenets and have influenced current virtualization software:

- Concurrent running of dissimilar operating systems by different users. While one virtual machine is used to develop

and test code for the current release level of an operating system, another virtual machine can be using a back-level release of the same system.

- Both system and application programs may be developed and debugged for machine configurations that are different from that of the host machine. Thus, a host machine with a modest amount of main storage can provide the environment for development and testing of a system to run on a machine with a large amount of main storage.

- One virtual machine is totally insulated from the effects of software failures occurring in other virtual machines.

- The host machine can aid in the measurement of hardware and software usage by the various virtual machines. Specific virtual machines built for monitoring can communicate directly with the host without impacting the machines being monitored. (Parmalee et al.,1972, p. 109)

It should be intuitive that a single powerful set of hardware or platform could perform the work of several smaller sets of hardware. This is what virtualization seeks to achieve. As the age of mainframes in the 1960s and 1970s gave way to the personal computer (PC) in the 1980s, the need to develop large powerful systems diminished. Only major corporations, universities, and governmental agencies continued to maintain and operate large mainframes. The business world's focus on the PC reduced the need for virtualization as a means to service multiple VMs and multiple users. Today, consumers can purchase small, high-performance computers as commodities, thus removing the size and cost barriers of the past. The industry's current emphasis on cloud-based architectures will further push the IT market to a greater reliance on a centralized server-based model.

3. Virtualization Architecture

The application of virtualization to SoS architectures will provide a platform for multiple system designers to share a common infrastructure. "This leads to ease of use and optimal product design and testing, which decreases costs and

lead times" (Swaminathan & Murthy, 2006, p. 67). Swaminathan and Murthy used the concept of virtualization to develop the representation shown in Figure 1. The architecture as depicted allows VMs to communicate with each other, much like hosting an entire network from a single computer. The structure illustrated represents an ideal environment for testing complex network topologies given the multiple possibilities for VM connectivity. The authors also acknowledge a possible work around to traditional virtualization architectures by adding non-virtual machine entities into the environment to help simulate a piece of hardware that is not easily virtualized, seen in Figure 2 as a *stub.*

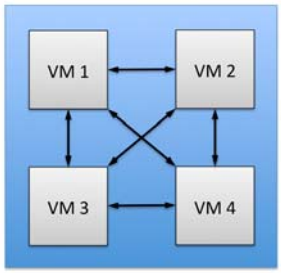

Figure 1. Four VMs on a Single Set of Hardware
(Swaminathan & Murthy, 2006)

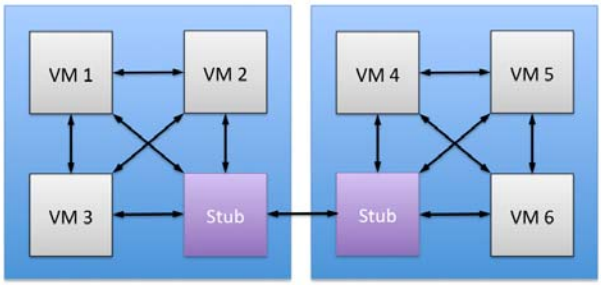

Figure 2. Six VMs on Two Sets of Hardware
(Swaminathan & Murthy, 2006)

The stub represents a physical component such as a switch or a network device that allows communication between two virtualization environments that normally would not have the capability to organically communicate. For example, a stub would provide the necessary interface for a system hosted in a virtual environment that requires a satellite communication link. The stub would be

physically connected to the server to translate the message into the appropriate format for transmission. Because many components of a system cannot be virtualized, it is important to understand how alternatives can be developed to simulate or replicate interactions from end-to-end of the system. This is specifically important for testing of systems given the need for operationally accurate and repeatable test conditions.

B. COMPONENTS OF VIRTUALIZATION

1. Hardware

a. Server

The server provides the processing power necessary to begin a virtual environment. Servers have replaced the mainframes of previous generations. For the purposes of this thesis, a server is defined as a set of hardware components (CPU and RAM) that perform the tasks of a given set of software. The term *server* can also be associated with a type of software such as an email server, which performs the function of organizing and distributing emails to a group of users. Throughout this thesis, the term *server* refers to the hardware, unless explicitly stated. Servers are traditionally housed in racks with multiple servers per rack. This configuration allows for the centralized access to electricity, air conditioning, and high bandwidth networking necessary for maximum performance. Today's servers can contain multiple CPUs and hundreds of gigabytes of RAM.

The continued improvement of the hardware components has enhanced the types of functions performed by servers. With the improved performance many servers are capable of hosting a virtual environment with dozens of VMs. By hosting multiple VMs on a single set of hardware, administrators gain efficiency in power consumption, physical footprint of computing devices, and ease of management of the hosted VMs.

b. Client

In a traditional client–server architecture, the workstation or client is a desktop or laptop computer. A client may also be known as a node in a network architecture. A network in the physical world is comprised of multiple nodes or clients. In a virtual environment, the client can take many forms such as a thin-client, thick-client, zero-client, or web-client. Each type provides characteristics specific for a given environment. The administrator of the network must determine which types of clients provide users with the required functionality. All clients offer a user the requisite keyboard, video display, and input options such as a Universal Serial Bus (USB) or Digital Video Disc (DVD) drive.

A thin-client contains a specially designed client software with minimal functionality required to perform hardware interface and to communicate with the server. The use of a thin-client requires a specially designed hardware device with onboard processing, memory, and networking. These devices contain the minimal components necessary to provide an operative user experience. A thin-client is normally housed in a device approximately 6” x 6” x 2”. The small size and onboard processing is ideal for organizations looking to reduce the physical footprint and power consumption, without sacrificing computing performance.

A thick-client is a traditional desktop with an additional virtualization software application installed to enable communication with the server. The desktop does not perform any application instructions, but it does provide video display and network messaging, much like that of a thin-client. The thick-client initiative is an attempt to repurpose or reuse existing desktops within an organization without having to dispose of desktops or purchase new hardware. Although physical footprint and power consumption efficiencies are not achieved with a thick-client, it does give access to multi-core processing, increased memory, and storage from the existing desktop.

The zero-client is a form of a virtual client that makes use of the PCoIP (personal computer over Internet protocol), discussed later in this chapter, to stream images of the VM state from the server to a device that does not contain any internal processing, memory, or storage capacity. The device does not record any portion of a virtual session; it displays images or screenshots of the VM hosted on the server. This type of device is the smallest in scale of all clients and the most secure of the options available to administrators.

A virtualized web-client allows a VM to be accessed through an Internet web browser. Any computer with access to the network can operate a VM through the host-based OS remote protocol made available through a browser plug-in. Remote protocols are discussed later in this chapter. The web-client provides the greatest degree of flexibility for a user. However, because the hardware and software were not designed specifically for the purpose of web-based virtualization, some performance is degraded due to latency.

c. Storage Area Network (SAN)

All servers are designed with some amount of storage available to them, usually on the scale of multiple terabytes per server. However, in a virtual environment, it may become necessary to make additional storage available, such as for the purposes of maintaining VMs or creating snapshots of VMs in a test environment. Given reduced costs and greater accessibility to storage, administrators have employed racks of storage and allocated them to the virtual environment. These dedicated storage devices, called storage area networks (SAN), contain multiple terabytes of data and can be shared by several servers, thus increasing the flexibility of the resources available through virtualization. Servers can be assigned a specific LUN (logical unit number) or memory address on a SAN, or the storage can be dynamically assigned according to the need of a given VM. Not all VMs perform the same functions or execute the same applications; therefore, the architecture should offer flexible storage options based on user needs.

2. Software Architecture

Three specific types of software enable a virtual environment, as depicted in Figure 3. These software allocate the physical hardware among the VMs, provide OS platforms, and manage the entire VM architecture. The first type, known as the virtual machine monitor (VMM), or hypervisor, is installed upon a set of hardware, much like an OS is installed upon a traditional desktop computer. The VMM does not perform all of the traditional OS functions; instead, it controls access to the CPU, RAM, network interface card (NIC), and storage between the VMs. Next is the VM OS, such as Microsoft® Windows, Ubuntu Linux, or Red Hat Linux. The VM OS sends requests to the VMM for central processing, memory usage, storage, and network access. Finally, to manage the configuration of the VMs, management software is installed on a specific VM within the environment. This management software, such as VMWare's VCenter Server, provides several functions including software application access and update support to the entire group of VMs.

a. x86 Platforms

Today, virtualization, in its most developed form, has remained within the x86 platform. The x86 platform includes the traditional Microsoft® Windows–based OS family in its many versions (XP, Windows 7, Windows 8) and Ubuntu with its Linux OS. The platform, created by the chipset technology of Intel and AMD Corporations, allows software that resides three layers above the hardware to have direct access to the hardware. In 2006, Intel and AMD modified the CPU instructions to allow virtualization to occur more easily, thus reducing the need for software workarounds to achieve the resource sharing (Neiger, Santoni, Leung, Rodgers, & Uhlig, 2006; AMD, n.d.). The VT-x technology by Intel (Neiger et al., 2006) and AMD-V by AMD (AMD, n.d.) provide the virtualization chipset instructions necessary to enable resource sharing. Figure 3 illustrates the concept of a bare metal or full virtualization implementation, in which the VMM is installed directly on top of the x86 platform or server.

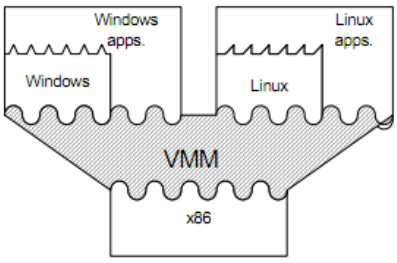

Figure 3. Depiction of x86 Platform Hosting Windows- and Linux-Based OS
(Smith & Nair, 2005)

b. Virtual Machine Monitor or Hypervisor

The VMM performs a central role in a virtualization environment. It is a small encapsulated piece software that may appear to perform as an OS; however, its functionality is more limited. The VMM is able to control the hardware, that is, make calls to the CPU, RAM, and storage, like an OS, but it does this in support of the VMs installed upon it. A bare metal VMM configuration, as seen in Figure 3, is known to be the most efficient means of allowing VMs access to the physical hardware. This configuration reduces the software calls, or messages, transmitted between a VMM and the platform for the purposes of providing resources to the VMs. In a typical server rack, multiple servers would be mounted, each with a VMM installed. The mix of two or more servers is known as a cluster. These clusters provide one of the unique benefits of virtualization, which is the ability to share resources based on CPU and RAM demand. As the workload of a group of VMs increases, the VMM can allocate more resources to the VMs in need to provide the most efficient instruction execution.

c. Virtual Machine Operating System

The OS installed on a VM is the software component most familiar to users. It is the family of Microsoft® or Ubuntu OSs most often used in a traditional desktop or laptop environment. The OS in a virtual environment executes the same functions performed in a traditional user environment, such

as running applications, controlling access to the network, managing system attributes, and performing basic calls to the CPU, RAM, and storage. Within a virtual environment, the performance of an OS and its associated applications remains the same as it would if it existed in a physical machine.

d. Virtual Machine Configuration Management

To manage the VMs hosted upon a VMM, server configuration management software has been developed to manage the VMs hosted in an environment. This software allows for the creation of new VMs from a template or by copying an existing VM. The software also provides the ability to take snapshots or back-ups of a system state of a VM in a certain state or configuration. These snapshots allow a VM to be restored in the case of file corruption or a system conflict with the integration of new software. One efficiency provided through VM management software is the ability to roll back or revert to a previous system state. In the case of a disaster, the management software can recognize if a VM or a cluster of VMs has shut down unexpectedly and quickly boot up a new cluster of VMs to compensate. This type of administrator control cannot be easily duplicated with physical machines.

3. Network

a. Components

In a virtual environment, the network is the heart of the architecture. It provides the connectivity necessary for both the physical components (e.g., fiber, cables, wireless, backplane, switch, router) and the virtual components (e.g., virtual LAN, virtual switch). Although it is not the purpose of this thesis to discuss all of the detailed components of the network and their functionality, it is necessary to mention two basic ideas that impact a virtual environment, specifically in a virtual desktop infrastructure (VDI) implementation: latency and protocol. With the following discussions, I explore the impact these ideas have on virtual environment.

b. Latency

The metric of latency, often used within the field of IT, and specifically virtualization, “can be measured one-way, from source to destination, or two-way round-trip, from source to destination and back to source (usually excluding the processing time at the destination to generate the response)” (Fehse, 2011, p. 12). For virtual systems, which interact with a server and a possible VDI client, the latency can determine the success of a virtual environment. Given that processing occurs at the server, a user must rely on a high-speed transmission of an input, the processing of that input, and the retransmission of the output. If this process exceeds 20 milliseconds (ms), the user is delayed in performing any other action until the last request has been completed. For multi-step processes, this interaction can inhibit user productivity if the delay becomes significant. Therefore, networks must minimize latency to improve user experience.

c. Protocols

To address latency and the communication between the server and the client, three types of protocols are widely accepted as standards. These standards facilitate the server–client communication necessary to perform any set of instructions. The Remote Desktop Protocol (RDP), developed by the Microsoft® Corporation, is designed to create remote displays and application support for users operating a Microsoft® OS. The protocol contains a “bandwidth reduction feature comprised of data compression, caching of graphical elements, and network load balancing” (Fehse, 2011, p. 16). These features reduce latency while improving the user experience and enhancing screen refresh rates.

The PCoIP technology, developed by the Teradaci™ Corporation for use by VMWare, focuses on bandwidth reduction through pixel transmission to reduce latency. This type of protocol streams the video of the user’s screen to the client. No data transmission occurs between the server and the client, which removes the need to have any client-side processing or storage capability. The

technology uses a series of video codecs to encode and decode the video stream at real-time speeds (Teradaci, n.d.).

The HDX™ (High Definition User Experience), developed by Citrix, relies on server, network, and client processing to effectively transmit data according to network congestion and available bandwidth (Citrix, n.d.). HDX's dynamic adjustments to network latency by the VMM, VMs, and network devices suggest that users will experience improved VM performance.

4. Server Virtualization

Server virtualization has been the focus for most businesses seeking to adopt virtualization technology. Server virtualization makes use of the position that any server that consistently operates below 50% of capacity is wasting capacity. To remove the waste, additional services must be hosted on the hardware. Virtualization provides the means for multiple server-based applications such as a web or email server to be hosted on separate VMs within a single set of hardware, as seen in Figure 4.

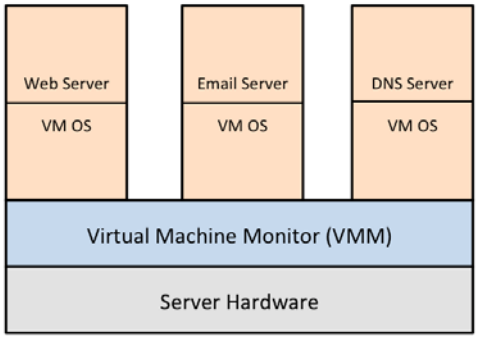

Figure 4. Depiction of Server Application Virtualization

5. Virtual Desktop Infrastructure

The VDI initiative has recently grown into a significant portion of the virtualization movement. In its purest form, VDI is a return to the server-terminal architecture of the 1960s and 1970s. Within VDI, the application hosting, processing, and networking all occurs at the server with a user interface in the

form of a thin-client or thick-client. The clients are connected to the server via a switch/router, Ethernet cable, or wirelessly. This type of implementation is ideal for established environments with high bandwidth.

C. VIRTUALIZATION: THE BUILDING BLOCK OF THE CLOUD

From virtualization has emerged the concept of *cloud computing*. The National Institute of Standards and Technology (NIST; 2011) offers the following definition for cloud computing: "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" (NIST, 2011, p. 2). This definition shares those early ideas of virtualization by allocating the computer resources across multiple VMs. By understanding the different service models available, the DoD can implement the appropriate model to assist in the T&E process. The services maintain a great deal of server and workstation hardware that can, in effect, be repurposed to implement private clouds in the T&E environment.

1. Cloud Service Models

The cloud service models provide a road map for future virtualization implementations. Figure 5 depicts the three enterprise service models: software as a service (SaaS), platform as a service (Paas), and infrastructure as a service (IaaS). Figure 5 distinguishes between a cloud service provider's control (highlighted in gray) and consumer's control (highlighted in white) for each type of service. Each model provides an enterprise different levels of control for the key elements of an IT infrastructure. SaaS limits a user to only minor application configuration settings without providing full access to the OS. This differs from PaaS, where consumers are authorized full application permissions, that is, they can install, uninstall, and manage applications through the OS. The IaaS model provides a consumer the option to create unique VMs or platforms, install specific OSs, and manage all applications.

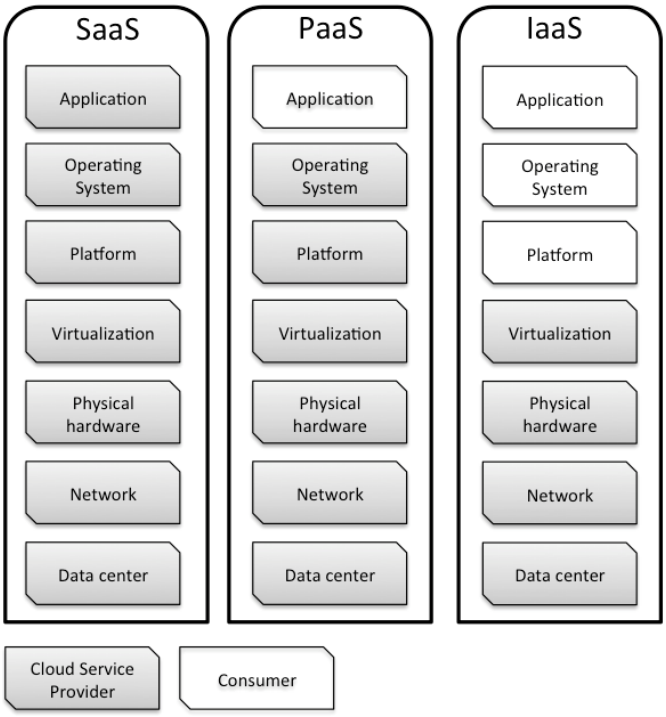

Figure 5. Cloud Service Models

An organization's ability to manage its own infrastructure determines the appropriate service model. Virtualization enables SaaS, PaaS, and IaaS to fulfill the desired user functionality. Virtualization provides the organization the ability to create a variety of VMs tailored to their needs. For example, if a developer requires a Windows-based x86 platform on which to test a specific application's performance, in an IaaS agreement, the developer can specify that requirement and build a VM to those criteria. The developer can then execute the testing in the Windows environment. Once complete, if he or she desires to test in a Linux environment, a new environment can be established all from the same client.

2. Cloud Deployment Models

a. Private Cloud

The private cloud model, depicted in Figure 6, limits access of the computing resources to consumers of a specific organization (NIST, 2011). This is the most secure form of the cloud deployment models and is ideal for test

activities or system development, because it provides the best possible computing resources with little risk of compromise. Test environments could be quickly established, employed, and saved for future use in this type of model. The private cloud gives an administrator the maximum amount of control, while providing consumers with the available computer resources on request.

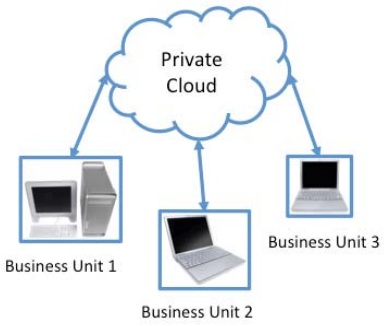

Figure 6. Private Cloud Model

b. Community Cloud

In a community cloud, the computing resources are shared among a community of consumers with mutual interests. A community of interest (COI) could leverage a community cloud to share limited applications or to give access to a common set of tools for consumers with a specific skill set. This type of cloud also allows for common concerns such as security issues, policy compliance, or mission accomplishment (NIST, 2011). For example, TopCoder Inc. has established a community of software developers and provided them with the necessary software development kit (SDK) through their community cloud (TopCoder, n.d.). As stated by NIST, a community cloud "may be owned, managed, and operated by one or more of the organizations in the community, a third party, or some combination of them, and it may exist on or off premises" (NIST, 2011, p. 3). Figure 7 depicts how three disparate organizations share the same resources to accomplish common goals.

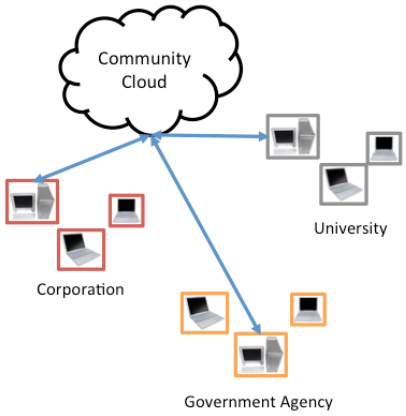

Figure 7. Community Cloud Model

c. Public Cloud

The public cloud model, depicted in Figure 8, permits open use by the public (NIST, 2011). The cloud provider enforces the goals or purposes of the architecture. The figure illustrates how non-governmental organizations (NGOs), small businesses, individuals, academic institutions, governmental agencies, and major corporations can all work together to achieve a common goal. An example of this coordination is evident in organizations such as InRelief.org, which provides improved response time for "Humanitarian Assistance and Disaster Relief (HADR) events by connecting military/civilian organizations" (InRelief, n.d.).

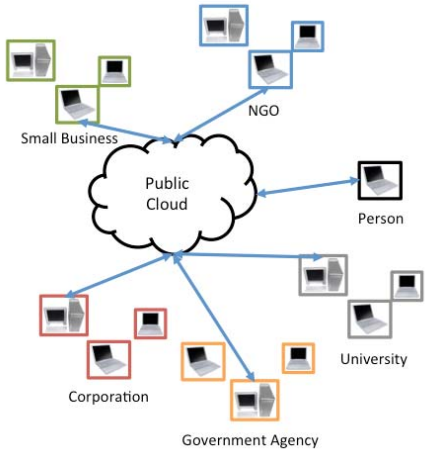

Figure 8. Public Cloud Model

d. Hybrid Cloud

The hybrid cloud permits the combination of two or more established clouds. The independent cloud architectures remain private objects, but allow for sharing of resources to accommodate load-balancing, fail-over protection, and application sharing (NIST, 2011). The hybrid cloud represented in Figure 9 portrays the combination of three unique clouds connected via common standards for the purposes of shared interests. This type of model represents the greatest degree of collaboration, because the communication standards established within an individual cloud must be duplicated across multiple cloud providers.

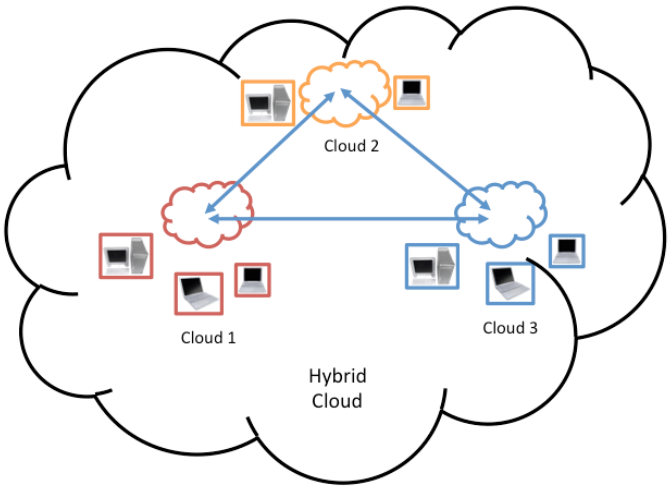

Figure 9. Hybrid Cloud Model

D. DEPARTMENT OF DEFENSE VIRTUALIZATION INITIATIVES

The DoD, in an attempt to adopt virtualization, has released service-based IT strategies that reflect the capabilities gained by implementing the technology. Each department has unique virtualization goals that focus on primary missions and existing IT platforms. For the focus of this thesis, I discuss the United States Marine Corps', the United States Navy's, and the United States Army's virtualization and cloud strategies.

1. United States Marine Corps

The Marine Corps operational forces were early adopters of virtualization technology. During the 2005 tsunami in South East Asia, the Marine Expeditionary Unit used server virtualization to consolidate the host nation's critical systems by partitioning the blade server hardware for multiple applications (Brodhun, 2008). In 2008, the information architect for Product Group–10 at Marine Corps Systems Command outlined a goal of 98% server virtualization. The objective was to achieve this goal in a three-phase process, as outlined in Figure 10. At the completion of the Marine Corps' implementation, the VM to physical server consolidation should be 2:1.

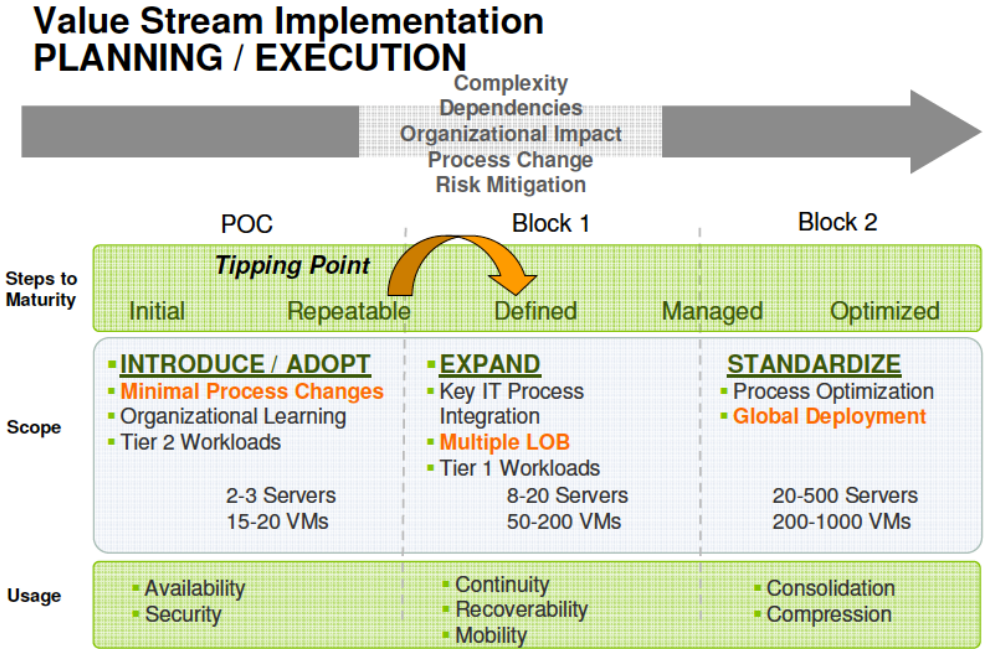

Figure 10. Marine Corps Virtualization Strategy
(Brodhun, 2008)

2. United States Navy

In NAVADMIN 008/11 (U.S. Navy, 2011), VADM Dorsett, Deputy Chief of Naval Operations for Information Dominance, outlined several future Navy initiatives that focus on virtualization. The first step outlined the requirement to reduce the number of data centers operated by the Navy. To achieve a goal of a

25% reduction, the Navy would need to leverage server virtualization: “Maximum effort should be applied to reduce the IT footprint and infrastructure in an effort to save Navy resources in hardware, software, manpower and to promote Navy green IT efforts” (U.S. Navy, 2011). To find this reduction, the Navy must consolidate server-based applications to “increase server utilization by 40 percent or more (not to exceed 80 percent utilization) and increase server virtualization by 50 percent” (U.S. Navy, 2011).

In addition to server virtualization, the Navy will also begin a thin-client pilot program: “DDCIO [Department of Navy Deputy Chief Information Officer], in coordination with the Navy Technical Authority, will lead a thin-client initiative, replacing traditional computing desktops with less expensive, mobile hardware that is engineered to support migration to a mobile workforce environment” (U.S. Navy, 2011). The Navy’s stated virtualization goals exhibit the impact of the technology in increasing IT efficiency and flexibility. By entering the early adoption phase of VDI, the Navy can eliminate the excess hardware associated with traditional desktops and shift to the PaaS, IaaS, and SaaS service models (U.S. Navy, 2011).

3. United States Army

As part of the Army Data Center Consolidation Plan (ADCCP), the Army will replace data centers with a unified cloud-computing architecture (U.S. Army, 2011). The U.S. Army plans will “reduce expenses associated with data center hardware, software and operations, and will be able to shift IT investments to more efficient computing technologies” (U.S. Army, 2011). The Army, like the aforementioned Marine Corps and Navy, must adhere to the DoD mandate to accommodate a reduction of data centers and a move to more energy-efficient IT solutions. This requirement paves the way for technologies like virtualization to play a greater role in IT strategies.

E. LIMITATIONS

1. Hardware

The principles of virtualization allow hardware to be allocated easily to the required virtual environment. However, it does not adequately allow for the hardware to simulate a system of greater capability than the existing hardware. For example, if the hardware contains a single dual-core processor and 100 megabytes of RAM, the hardware cannot host VMs emulating computers with quad-core processors and 200 megabytes of RAM. Virtualization can only divide the existing hardware into smaller elements. If a user were to allocate the hardware in excess of the system specifications, the system would stop functioning and crash. Therefore, when designing virtual environments, it is critical to match the existing hardware to the expected VM architecture.

2. Software

Virtualization software requires several components to achieve full functionality. The functions of the VMM, environment management software, and client-side software must all work together. This requirement limits the options that administrators can take to implement a virtualization environment. Once a vendor of the virtualization software is selected, the remainder of the architecture must, in most situations, remain with that vendor. The different elements of the virtualization hierarchy do not work well with different developer models. This realization can create problems for administrators who are trying to find a hybrid solution for each element of the environment. For example, if Citrix is selected for the client-side software, the client software must work with the management software, which must work with the VMM. This requirement often limits architectures to a single vendor based on interface requirements.

3. Network

To sufficiently host a virtual environment, the organizational network must be robust. The data rate required to host server-side VMs without a client does

not require a network at all, given that all communication between the VMs occurs at the backplane of the server rack. However, if the VMs require a VDI implementation, then the throughput requirement increases substantially. Each VDI client requires a connection of 25 megabits per second (Mbit/sec). This type of connection ensures that the latency of the server to client communication is minimized to 20 ms. When the network is unable to sustain the 25 Mbit/sec level, the client-side interface can slow to an unusable level, limiting the user's ability to perform any functions. Therefore, the network must account for the number of VMs hosted by the server and prepare for high throughput requirements both in wireless and cabled environments.

4. Real-Time Systems

Real-time systems do not perform well in virtualization environments due to the problem of time drift or clock drift. Computers account for time traditionally with a physical clock in the hardware known as the real-time clock or RTC (the term internal clock or IC has also been used). However, in virtualization, no physical RTC exists inside the VM and the RTC must be replaced through methods of time correction such as a Network Time Protocol server. Although the software shows promise in the scalability and management of virtual environments, it does not satisfactorily handle requirements for precise timing. The concept of time keeping or time synchronization is important to computers because it provides a system with an understanding of how it relates to other systems. If a system or application requires precision timing, such as those found in track-based systems which use a GPS (Global Positioning System) to indicate the time, the system could be at a disadvantage given the time drift induced between the hardware and the VMs.

F. CONCLUSION

Virtualization has evolved significantly in the decades since the CP-40. However, from the early days at IBM to the cutting edge developments today with companies such as Citrix and VMWare, the tenets have remained the same.

Developers and users have sought to find efficiency in maximizing computer resources across multiple users. Virtualization does not represent a single solution for all the technological challenges of today's IT environment. Yet, it does provide a specific set of ideal capabilities for establishing large computer environments quickly and completely.

THIS PAGE INTENTIONALLY LEFT BLANK

III. TEST AND EVALUATION IN SYSTEM OF SYSTEMS ARCHITECTURES

A. TEST AND EVALUATION

1. Overview

The test and evaluation aspect of a system's development is a small piece of three larger DoD support systems: the Defense Acquisition System (DAS), the Joint Capabilities Integrated Development System (JCIDS), and the Planning Programming, Budgeting, and Execution (PPBE) Process. Figure 11 illustrates how the three separate processes mutually support and overlap to form a system of checks and balances. The DAS provides the specific management of the T&E processes, with JCIDS providing program oversight, and PPBE providing the program funding.

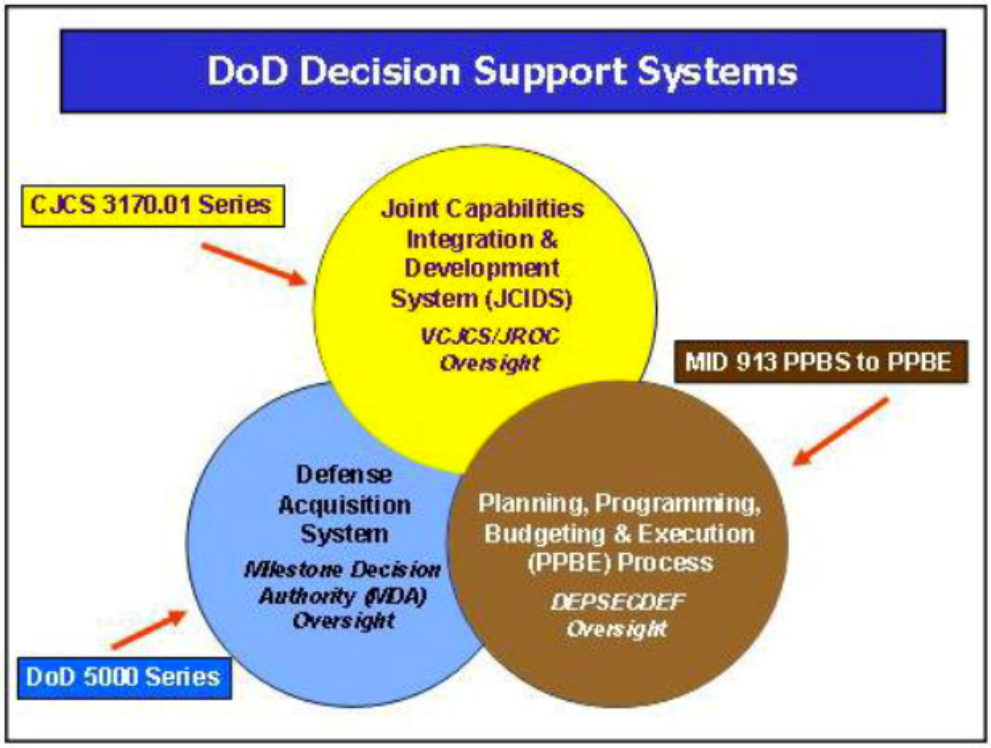

Figure 11. DoD Decision Support Systems
(DoD, 2012, p. 6)

2. Purpose

The purpose of T&E is to provide information to help mitigate the risks involved in developing systems and capabilities (Under Secretary of Defense for Acquisition, Technology, & Logistics [USD(AT&L)], 2008). A system's test and evaluation methodology is developed based on the system's requirements. To understand the methodology, it is necessary to discuss the elements of test and evaluation. The test is an action to verify operability, supportability, or performance of an item by subjecting it to real or simulated conditions with special test equipment or tools to obtain measurements or data for analysis (Blanchard, 2011). The test is designed to measure a specific system objective or requirement. An evaluation is a continuous iterative process to examine and assess a system or an element of a system with regard to relative worth, quality of performance, degrees of effectiveness, and anticipated cost (Blanchard, 2011). T&E standards are initially defined during the conceptual design period of a system by translating user needs into formal statements. Subsequently, specific test methods are established to determine the system's performance against the requirements.

Test and evaluation gauges the progress of a system and its capabilities throughout development. It provides awareness of system capabilities and limitations to the DAS for use in improving performance. To be effective, T&E must be integrated at the beginning of the system development to identify system strengths and weaknesses. The objective is to recognize system defects so components or processes can be retooled prior to system release (USD[AT&L], 2008). Figure 12 depicts the multiple test activities that are required throughout a system life cycle. Two key documents that inform the test processes are the Test and Evaluation Strategy (TES, or Eval Strategy) and the Test and Evaluation Master Plan (TEMP).

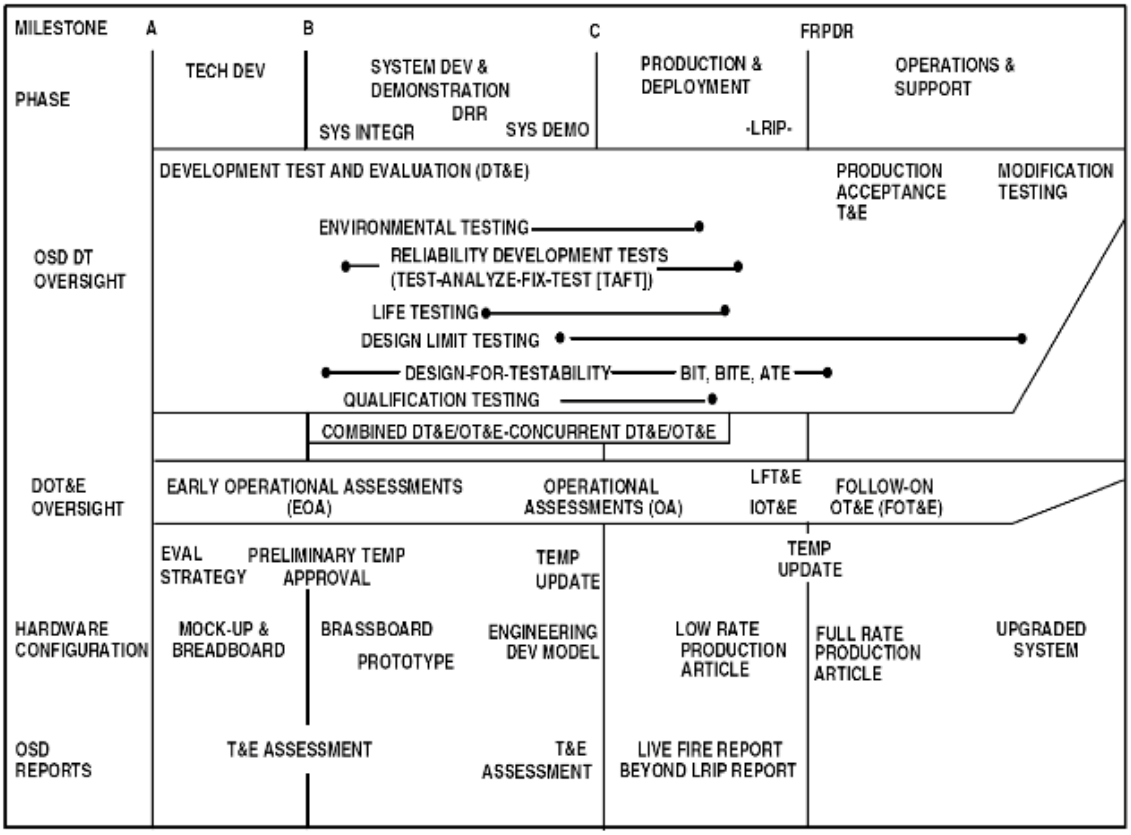

Figure 12. Test and Evaluation Framework
(Naegle, 2011)

3. Test and Evaluation Strategy (TES)

The TES describes the concept for tests and evaluations throughout the program life cycle, starting with technology development and continuing through engineering and manufacturing development (EMD) into production and deployment. The TES requires approval prior to Milestone A. The TES informs the TEMP at Milestone B, which becomes the primary source of guidance for all test activities. Development of a TES involves testers, evaluators, and program managers to ensure buy-in and suitability of the test procedures and timeline. These personnel specify the technical, functional, and operational test details to ensure the TES meets the established criteria (DoD, 2012).

4. Test and Evaluation Master Plan (TEMP)

The TEMP describes the total T&E planning from component development through operational T&E into production and acceptance. The T&E Working-level Integrated Product Team (WIPT) provides input for the TEMP to the PM regarding each test event. The TEMP identifies the T&E activities and the personnel and infrastructure requirements. The TEMP is reevaluated throughout the production phase to adapt to changes to system requirements (DoD, 2012).

B. SYSTEMS BACKGROUND

1. Systems Science

In the physical world, systems can exist as organic or human-made systems. In both types of systems, the elements of components, attributes, and relationships define the system and its purpose. In the sphere of IT, systems predominantly take the form of some combination of computer hardware or software. Therefore in IT, the components traditionally define the parts of a system, whether defined in software or hardware. The attributes are the characteristics which describe the components, such as the speed of the CPU or the type of user interface. The relationships or connecting medium of IT systems would be the physical cable lines or the protocol used to transmit data between components. These components work together to achieve a common purpose or goal and the system components depend on each other to achieve the purpose (Blanchard & Fabrycky, 2011).

2. Systems Engineering

The discipline of systems engineering has become a core piece of the DoD acquisition process. The DoD Instruction 5000.02 defines it as

> An approach to translate operational needs and requirements into operationally suitable blocks of systems. The approach shall consist of a top-down, iterative process of requirements analysis, functional analysis, and allocation, design synthesis and verification, and system analysis and control. Systems engineering

> shall permeate design, manufacturing, test and evaluation, and support of the product. Systems engineering principles shall influence the balance between performance, risk, cost and schedule. (USD[AT&L], 2008)

The system purpose drives the design, development, and T&E of a systems engineering approach. In turn, requirements determine the system components, attributes, and relationships. Systems engineers use a top-down approach to verify the interfaces of the system components by observing the interactions. Then as part of the systems engineering process, system components and relationships are analyzed from a life-cycle perspective from the system's first operational use to its retirement. By completing this analysis, system upgrades and future changes can be anticipated and built-in to the system design. To achieve these varied tasks, system engineers use interdisciplinary teams to meet technical demands and management to ensure that each design discipline is represented and that their methods, techniques, and tools are integrated in the development of the system (Blanchard & Fabrycky, 2011).

3. Systems Framework

The emergence of SoS engineering, integration, and testing has given rise to several theories or frameworks to understand the complexities of working within SoS or FoS architectures. For this thesis, the work of Goshorn (2010) provides a fundamental structure to categorize both system and SoS engineering. Figure 13 lists the Systems Engineering Core model phases from (X)—The Need, to J—Disposal. The eleven phases of the Core model denote the processes of the systems engineering life cycle. The Core model and other systems engineering activities begin with an operational need. This need determines the form, function, and technical specifications of the system to be developed.

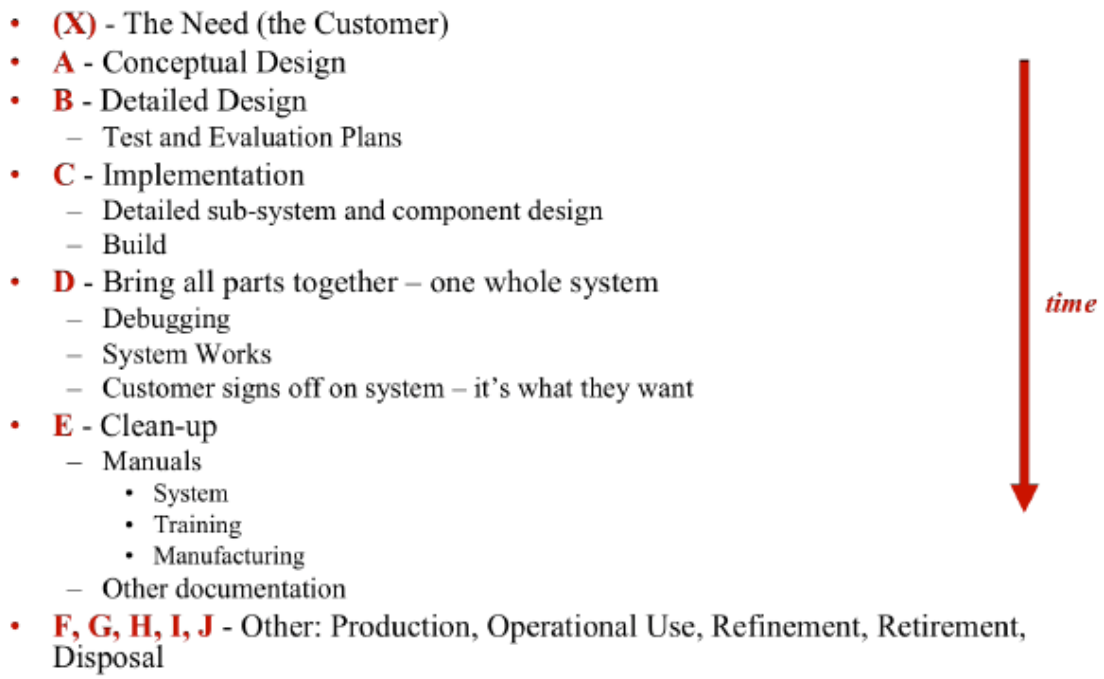

Figure 13. Overview of Phases in the Systems Engineering Core Model
(Goshorn, 2010)

After the customer's operational need has been defined, the Core model phase of Conceptual Design (A) begins. This phase mirrors the work of Boehm's nine-level waterfall life cycle (1981) and allows designers to determine the technical feasibility of a system for a given need. In Phase A, a top-down approach to the system design is started and a refinement of the customer needs is completed. The refined customer needs ensure a thorough understanding of the problem by the engineers and the customer, much like that of a quality function deployment (QFD; Yang, 2008). Phase B, Detailed Design, follows the conceptual design as the big picture architecture is decomposed into functional diagrams (Goshorn, 2010). The design plans should include the system description, components, and technical specifications. The Implementation process of Phase C enacts the detailed designs of Phase B. System components are built to specifications. System domain activities work largely independently as they prepare their component or subsystem for integration testing. Phase D, termed *Bring all parts together*, integrates the components for test and evaluation

and debugging. This phase is a targeted phase of this thesis, given the focus on T&E. Phase D requires a fast-paced tempo relative to the other aspects of the life cycle (Goshorn, 2007). The dimension of pace becomes more important given the impending end of the Core model. The speed of the decision-making and autonomy of the system integrators create a more compressed schedule to deliver the system to the customer on time (Goshorn, 2007). Following the T&E and the approval of the system by the customer is the Clean-up phase (E). This phase is marked by the creation of operating and training manuals, as well as the necessary instructions to manufacture or change the system. The subsequent phases of F through J do not apply specifically to this thesis. Although they do play an important role in the life cycle of a system, they are not relevant to the discussion of system development and system testing.

Figure 14 depicts the phases on an x-y axis to demonstrate the linear progression of the phases through time and their relative cost per unit. The proportions indicate approximations of where designers and engineers spend their time and resources for a given phase of the system development. The Core model, in Figure 14, is carried forward as a basic framework for how all systems and subsystems progress through the development cycle.

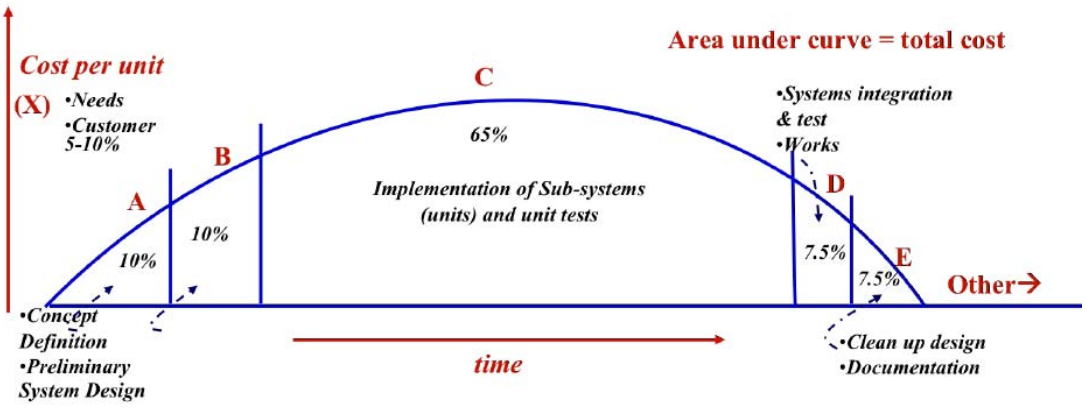

Figure 14. Systems Engineering Core Model
(Goshorn, 2010)

As a system is developed, multiple components and subsystems are created in support of the larger system. A system or SoS upgrade is completed by an engineering change order (ECO). As ECOs are implemented and changes are made to the system, new versions of the system are created as indicated by the Version 2.0 or V2 classification. Throughout the upgrade or ECO process, it is essential for system developers to deliver a functional system or SoS to complete the testing process. To meet the system availability requirement, developers should ensure that ECOs meet the necessary design traits prior to integration. Figure 15 helps show the relationship between subsystems and a higher level system.

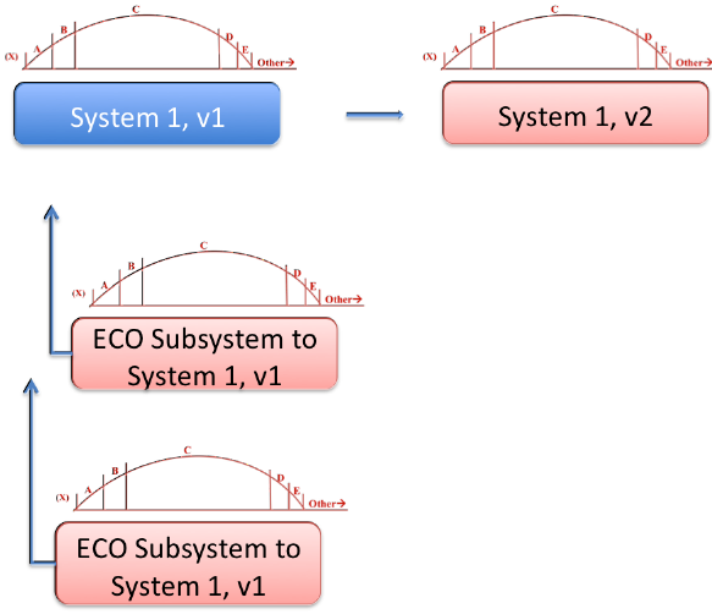

Figure 15. Systems Engineering of a System
(Goshorn, 2010)

A system or SoS hierarchy is depicted in Figure 16, which reveals the relationship of changes for systems and SoS. This overall picture of the SoS helps a test director (TD) or program manager see the progression of an SoS and the component systems that constitute the SoS.

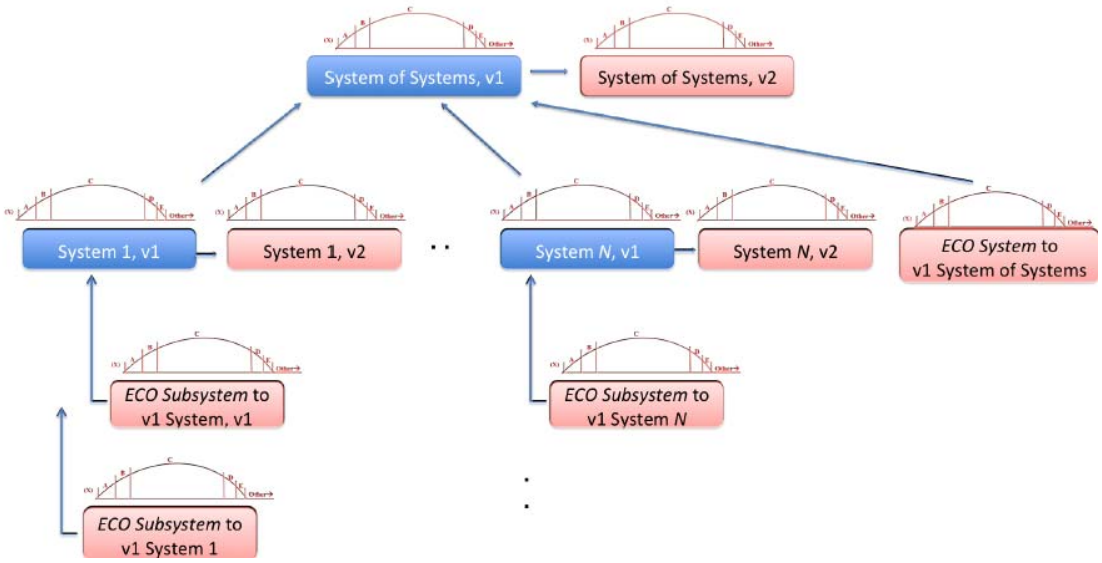

Figure 16. Applied Methodology for Systems Engineering of Systems of Systems
(Goshorn, 2010)

4. System of Systems

The DoD established a systems engineering methodology for program development, which required a modular open-systems approach for systems development. This vision for component-based systems gave way to the system of systems and family of systems (FoS) framework. The SoS or FoS approach is applied throughout the DAS. The DoD defines SoS *design* as the following:

> A set or arrangement of interdependent systems that are related or connected to provide a given capability. The loss of any part of the system could significantly degrade the performance or capabilities of the whole. The development of an SoS solution will involve trade space between the systems as well as within an individual system performance. (Chairman of the Joint Chiefs of Staff [CJCS], 2007)

The purpose of system of systems (SoS) testing is to integrate multiple component systems into a single TEMP. Each system tested within an SoS architecture may prove operational in a stand-alone environment, but may fail when combined with other component systems. Given that most command, control, communication, computers, and intelligence (C4I) systems were developed and fielded independently, SoS testers attempt to link the systems

together into a single network to measure the combined performance, or the overall SoS functionality.

5. System of Systems in the Department of Defense

There have been several initiatives within the DoD which have sought to develop SoS engineering methodologies and integrate them into the field of T&E. This SoS method sprung from the increased complexity and individuality of our systems. As stated by Miller (2008), "early C4I systems were designed, acquired, and fielded independently. Each addressed a single warfighting function, such as logistics, fire support, or intelligence" (Miller, 2008, p. 1). Programs such as the Marine Air Ground Task Force (MAGTF), Command Control Communication Computer (C4I) Capability and Certification Test (MC3T) sought to integrate several Marine Corps programs of records, such as the Combat Operation Center (CoC) Digital Common Ground System (DCGS; Marine Corps Tactical System Support Activity [MCTSSA], 2010). MC3T fulfilled the requirement for a metric to compare SoS performance to the needs of a warfighter (Miller, 2008). The follow-on program to MC3T, known as MCIC (MAGTF C4I integration and certification), seeks to continue the goal of linking multiple systems such as the Advanced Field Artillery Tactical Data System (AFATDS), Common Aviation Command and Control System (CAC2S), and the Command Post of the Future (CPOF) to verify that the system of systems integration performs as expected. The MC3T and MCIC SoS events provide end-to-end thread-based or task-based mission simulations which link forward-deployed systems to rear-echelon systems via a direct link or intermediary systems. By recreating these operational architectures in test environments, MCTSSA strives to improve the integration of these systems when they are fielded.

C. SYSTEM TEST METHODOLOGIES

The testing methodologies discussed in this section all have origins in software developmental testing. The methods contain testing elements unique to the field of software design. In the work of Abu-Taieh & El Sheikh (2007), the

authors consider several test methods, from cursory to detailed, such as audit, inspections, face validity, structured walkthrough, syntax analysis, Turing tests, bottom-up, top-down, black-box (functional), white-box (structural), regression, and thread-based testing. The work of Abu-Taieh and El Sheikh, based on the work of Balci (1994, 1995), and Balci et al. (1996), seeks to organize types of tests and to align requirements to the appropriate type of test.

Test methodologies also make use of what is known in the software domain as validation, verification, and testing (VV&T). These processes are akin to the T&E processes that focus on software stability, functionality, and security. The field of VV&T as organized by Balci (1995) includes the following types of techniques: informal, static, dynamic, symbolic, and formal (depicted in Figure 17). Each category of technique has a stylistic approach suited for testing the different states of a system. In the following section of this chapter, I review the techniques most suited for SoS testing.

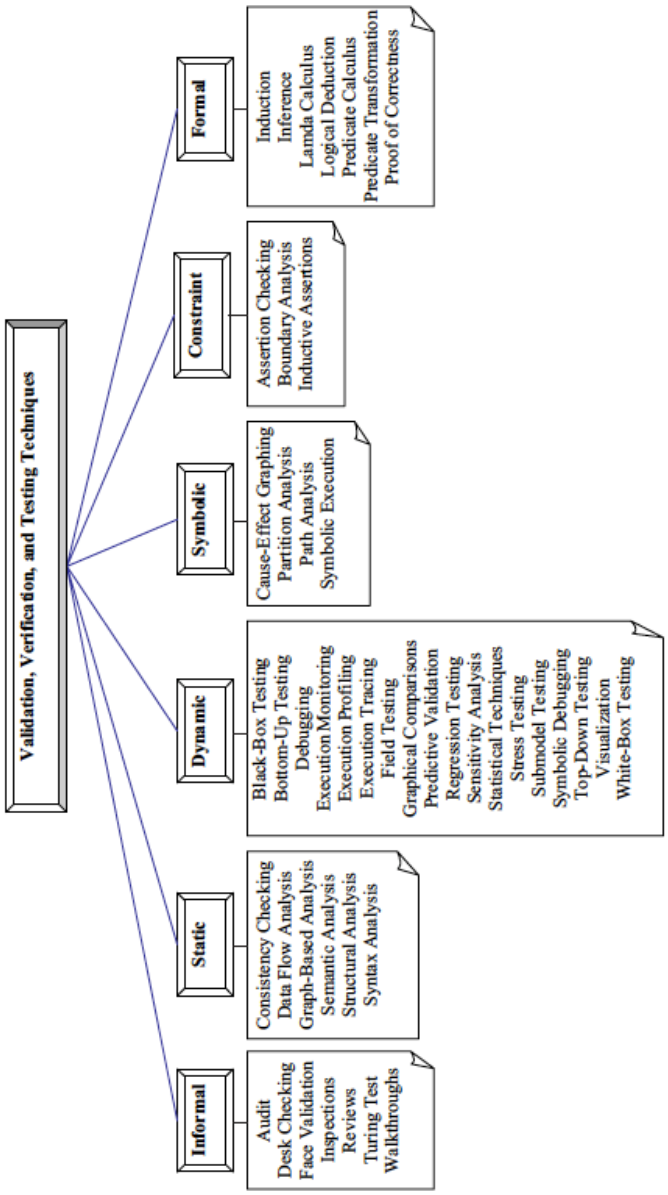

Figure 17. VV&T Techniques
(Balci, 1995, p. 152)

1. Bottom-Up Testing

In bottom-up testing, the lowest level of a system's components are tested first, with subsequent testing building on the successful tests of these components. The lowest level components contribute to subsystem testing, which leads to overall system testing. This type of methodology can be applied at

any level of T&E, both in the system development phase and the operational test phase. Bottom-up testing produces benefits such as the following:

- assessment of the lowest level components first,
- future testing built upon verified components, and
- reduced complexity at initial stages of testing.

2. Top-Down Testing

In top-down testing, high-level components are tested followed by lower level components. This type of testing relies on substitute components, also known as stubs, to perform in place of lower level components to mimic functionality to be developed later. This type of testing requires an understanding of the high-level architecture to account for all the primary systems.

3. Black-Box Testing (Functional)

The focus of black-box testing is the output of the test. The test originates with some input, and the resultant output is measured against some existing criteria. The test does not explicitly examine if the system is performing the tasks properly; instead, it determines if the results of the process produce the expected values. This type of testing is a more pragmatic approach to system development and is most likely performed by users or higher level operators who do not have the knowledge base to understand the interworkings of a given system (Abu-Taieh & El Sheikh, 2007).

4. White-Box Testing (Structural)

White-box testing examines the internal systems and subsystems of a given application to determine if the precise tasks are being executed in the manner in which they were designed. This form of testing requires a detailed understanding of each module and how each module handles a given piece of information. This type of testing can become complex and, therefore, should be performed with low-level components with few processes (Abu-Taieh & El Sheikh, 2007).

5. Regression Testing

In order to perform regression testing, an administrator must understand the previous states of the system. The system tester regresses, or returns to, a previous system state to understand how a modification, in the form of an update or engineering change order (ECO), may have led to a system failure or undesired state (Abu-Taieh & El Sheikh, 2007). This type of testing is preferred within a virtual environment given the ease of system state snapshots and rollbacks.

6. Mission Thread Based Testing

Mission thread based testing is the evolution of thin thread based testing first used by the DoD for end-to-end (E2E) Year 2000 (Y2K) testing. Thin-thread testing executed a software macro that would link multiple systems to determine the integration of the systems (Pham, 2006). The value in thread testing is the ability to use small amounts of software to link several systems. The code was easily understood by the systems integrators of the multiple systems and the threads did not rely on a single programming language to achieve their goals. The threads did have weaknesses; for example, several threads were required to determine the functionality of the system, and they often required manual development and verification (Pham, 2006). These weaknesses were addressed with scenario based or mission thread based testing. Mission thread based testing identifies the critical processes that must work and exercises them across multiple systems. By using mission thread testing, the SoS performs as a single system, thus verifying the interconnections between component systems and component system performance.

D. CONCLUSION

The test and evaluation of a system of systems architecture is a complicated endeavor requiring a detailed understanding of the system capabilities, technologies involved, program costs, and program timelines. The

process is driven by system requirements and stakeholder input into the Test and Evaluation Master Plan to achieve the desired system performance at the time of delivery to the customer. To assist in the process, systems engineering provides a repeatable framework to address many of the difficulties encountered during T&E. The Core model (Goshorn, 2010) offers one approach for how to view system and SoS development. With a structure in place, specific testing of the system performance can be completed using many of the methods outlined by Abu-Taieh & El Sheikh (2007). To achieve thorough and efficient T&E for an SoS, engineers and test officers must understand the processes to complete the tests and they must have the environment capable of performing the tests.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. CASE STUDY OF THE DISTRIBUTED GLOBAL INFORMATION GRID (GIG) INTELLIGENCE AUTOMATION SYSTEM

A. INTRODUCTION

The purpose for this chapter is to use as a case study the application of virtualization to an SoS T&E environment. In this chapter, I seek to answer proposed research questions as they apply to the Distributed Global Information Grid (GIG) Intelligence Automation System (DGIAS). The work of Goshorn (2010) provides the necessary information about the component systems of the DGIAS including the architecture of the system, the physical hardware, and the software components. Then, I provide an analysis of the suitability of the current system for virtualization. Next, I discuss a proposed virtualized architecture of the DGIAS. Following that, I introduce a process model incorporating the Core model to determine the efficiencies gained through virtualization in the test and evaluation of a new engineering change order. Finally, I discuss the limitations of virtualization as they apply to the DGIAS.

B. DGIAS SUITABILITY ANALYSIS

As one of the guiding principle of this thesis, the research question, what are the ideal system traits for implementing virtualized system of systems test and evaluation?, helped to shape a series of questions for assessing a system's suitability for virtualization. Given that not every system is right for virtualization, these questions help determine the qualities of the system that need to be assessed to determine whether the system fits into a virtualization environment. The questions were designed around the operating systems of the clients, the processor requirements of the operating systems and applications, and the storage requirements of the data. These constraints are the minimum to consider and do not fully account for every possibility but they provide a guideline for system designers. As a note, the system processor and storage specifications of the clients and servers were compared against the capabilities of the Dell

PowerEdge R610 Server (Dell, 2010) with two six-core processors and the Hewlett-Packard P4300 G2 SAS Starter SAN Solution (Hewlett-Packard [HP], 2012) with 20 terabytes (TB) of available capacity. The following are the questions I developed to assess a system's suitability for virtualization:

- Are the system nodes or clients comprised of Windows or Linux x86 operating systems? Answer: Yes, the clients use the Windows XP operating system and the servers use Windows Server 2004.
- Can the number of CPU cores currently used by the system clients be hosted by the available server? Answer: Yes, the minimum number of CPU cores employed throughout the system is 16. The hardware available can support up to 24 CPU cores.
- Can the storage requirement of the system clients be stored on the available Storage Area Network? Answer: Yes, the maximum amount of storage required by all the clients and servers is 16,240 GBs or less than 17 TBs of storage. The storage available can support up to 20 TBs.

Affirmative answers to all three of these questions indicate that the system would likely be suited for virtualization. A negative response to any one of the three questions indicates that a system would not support using the technology.

C. DGIAS SYSTEM COMPOSITION

1. Description

The DGIAS demonstrates the capability of an SoS that integrates multiple ISR assets for the purpose of fusing video collection with real-time facial analysis. The DGIAS uses commercial off-the-shelf (COTS) products in the design and development of the system. The DGIAS is a proof-of-concept SoS which combines top-down, bottom-up systems, and middleware to Detect–Identify–Predict–React (DIPR) to a set of inputs and provides cueing to higher level intelligence systems. Figure 18 depicts the original DGIAS high-level architecture with the component systems and their relationships.

The bottom-up systems are the Fixed Camera System, a Kiosk System, Unmanned Ground Vehicle System, Unmanned Aerial Vehicle System, and Cyber System. The middleware system is made of service-oriented architecture

which supports smart “push/pull” of sensor data and intelligence products between GIG-nodes. The top-down systems are the Command and Control System for supporting viewers for commanding officers, intelligence analysts, and tactical operators of sensors (Goshorn, 2012).

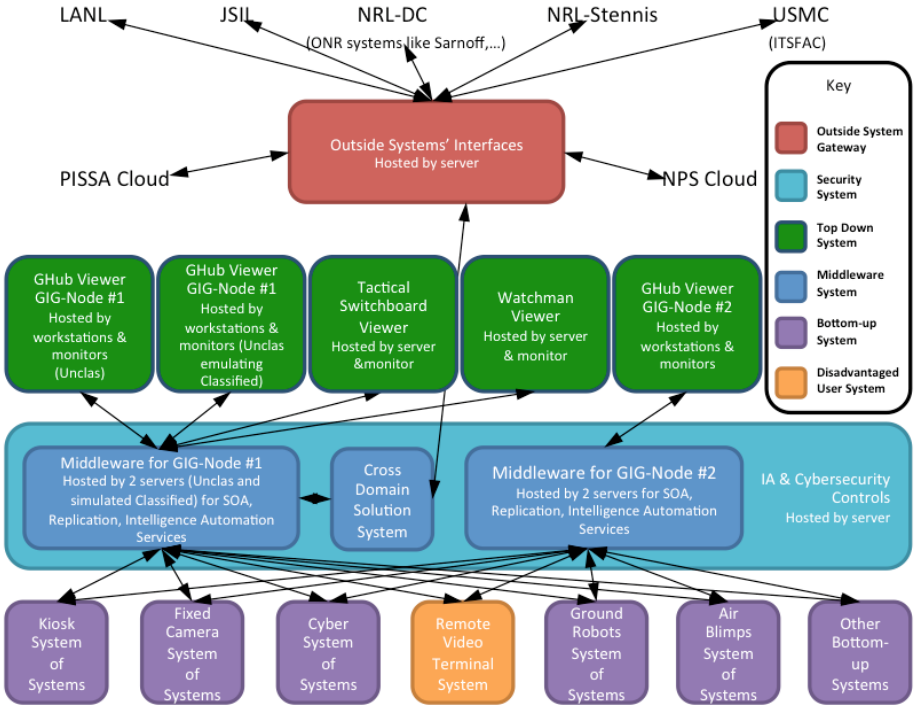

Figure 18. System View Diagram for DGIAS
(Goshorn, 2010)

As part of this research, I discuss four of the 12 component systems: the Kiosk System, Fixed Camera System, Middleware, and Watchman Viewer System. These systems are representative of the entire top-down and bottom-up system functionality. Figure 19 highlights the systems of the DGIAS to be analyzed.

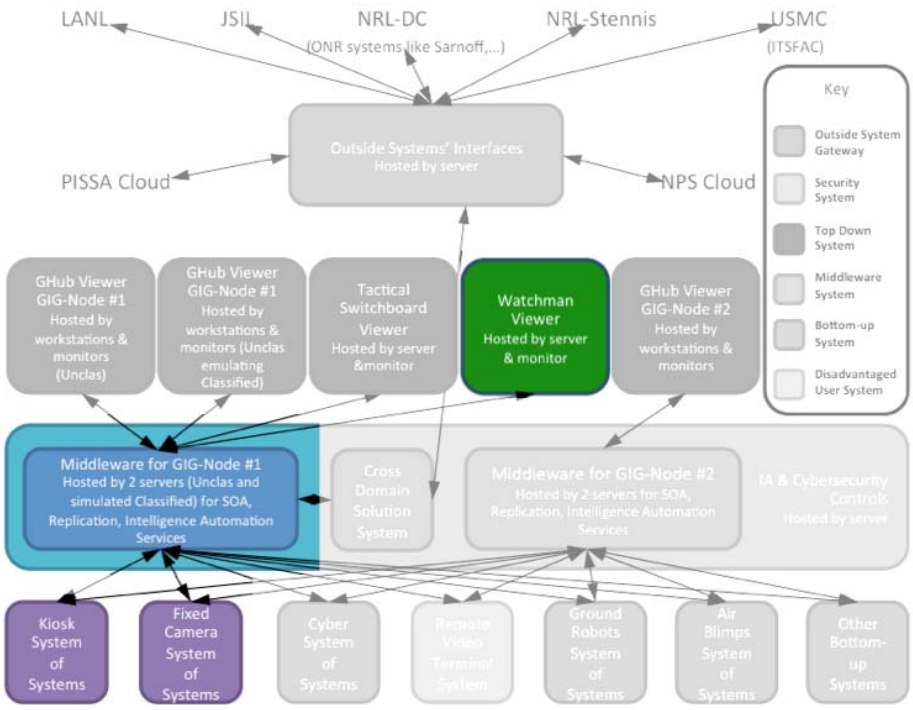

Figure 19. Selected DGIAS Systems
(Goshorn, 2010)

Figure 20 illustrates the DGIAS physical architecture of the selected Kiosk System, Fixed Camera System, Middleware System (Geospatial Hub [GHub]), and Watchman Server System.

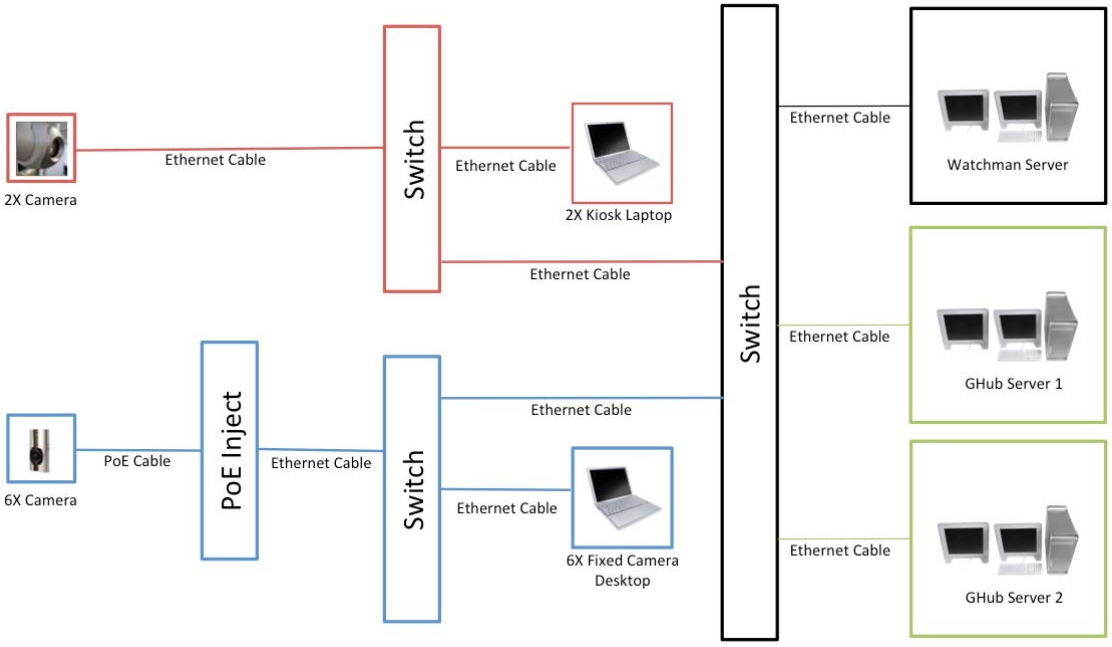

Figure 20. Physical Architecture of Selected DGIAS Systems

2. Component Systems of DGIAS

a. Kiosk System

The Kiosk System is an assemblage of multiple hardware components including two Dell D820 Latitude™ laptop computers with the Windows XP OS, two Sony pan tilt zoom (PTZ) cameras, a network switch, one wireless microphone system, seven microphones, two speakers, an audio mixer, and cabling. The system's purpose is to act as a component system in the larger DGIAS architecture by providing "interactive facial recognition, audio recording, and analysis" (Goshorn, 2010, p. 303). Figure 21 represents the Kiosk System architecture.

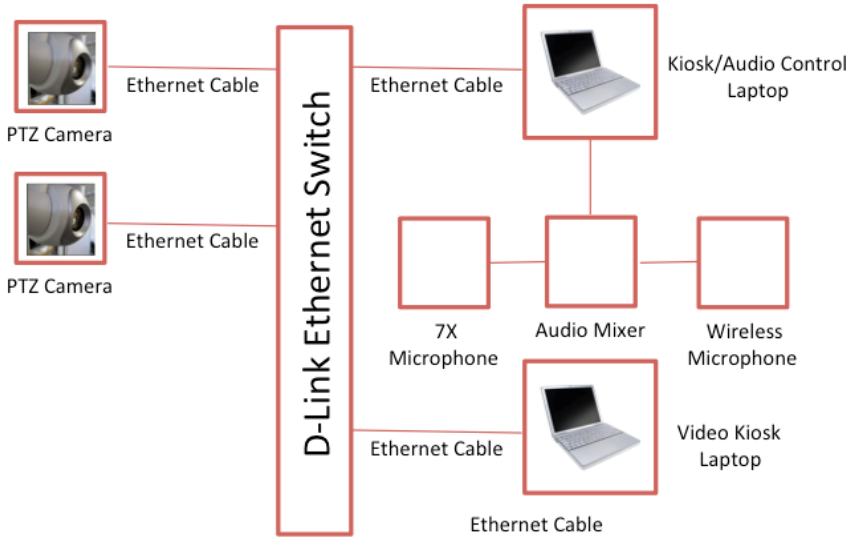

Figure 21. Kiosk System Physical Architecture

The visual and auditory data collected by the two Kiosk laptops is sent to the Watchman Server for further analysis and integration with other component system data. The Video Kiosk laptop uses MATLAB software to detect known faces through a detection algorithm. The MATLAB writes facial recognition data to the Watchman Server in SQL (Standard Query Language) format to the Watchman database. This database messaging is completed using Open Database Connectivity (ODBC) standards across each node and server. The Audio Kiosk laptop uses Audacity software to provide spectral analysis of collected audio data. Both laptops have the following software installed: Mozilla Firefox, Filezilla FTP Server, and Wireshark. As part of the network plan, each Kiosk laptop was assigned a unique IP address for deconfliction (Goshorn, 2010). Figure 22 lists the hardware specifications of the Dell Latitude™ D820 laptops, and Figure 23 is a photo of the laptops in the Kiosk System.

FEATURES	Dell Latitude D820
SYSTEM	
Processor Type	Intel® Core™ Solo and Duo processors 667MHz Front Side Bus & 2M Smart L2 Cache
Processor Features	Intel® Core™ Solo processor T1400 (1.86 GHz) Intel® Core™ Duo processor T2300E (1.66GHz) T2400 (1.86GHz) T2500 (2.00GHz) and T2600 (2.16GHz) Intel® Core™ 2 Duo processor T7200 (2.00GHz), T7400 (2.13GHz), T7600 (2.66GHz)
Operating systems	Microsoft® Windows® XP Professional SP2, Microsoft Windows XP Home SP2
Chipset	945GM (667MHz front side bus) with Intel onboard graphics or 945PM with NVidia graphics
Memory	Min: 512MB DDR2 shared[1] 533 or 667MHz Max: 4GB[2] DDR2 shared[1] 533 or 667MHz
Displays	15.4" WXGA (1280 X 800 resolution); 15.4" WSXGA+ (1680 X 1050 resolution); 15.4" WUXGA (1920 X 1200 resolution)
External Display	Supports up to a maximum resolution of WUXGA (1920 x 1200)
Graphics	Choice of Intel® Graphics Media Accelerator 950 (Up to 224MB shared); 256MB NVIDIA® Quadro NVS 110M TurboCache™ [3]; or 512MB NVIDIA® Quadro NVS 120M TurboCache™ [3]
Hard drives	40, 60, 80, 100 and 120GB[4] primary; 80GB[4] secondary
Keyboard	87-Key US; 88-Key Europe; 91-Key Japan; key travel 2.5mm; key spacing 19.05mm
Pointing Device	Touch Pad - PS/2 compatible, Track Stick - PS/2 compatible
Audio	High Definition Audio codec. Dual speakers, 8 ohm. Integrated omni-directional microphone.
Dimensions	H: 35.3mm/1.39" x W: 361mm/14.2" x D: 262.6mm/10.34"
Weight	Starting at 6.0Lbs/2.73Kgs*
POWER	
Power Supply	90 Watt AC adapter with cord wrapping
Batteries	Primary 6-cell/53 WHr "Smart" Li-Ion battery featuring ExpressCharge™ Primary 9-cell/85 WHr "Smart" Li-Ion battery featuring ExpressCharge™ Secondary 6-cell/48 WHr "Smart" Li-Polymer battery featuring ExpressCharge™
CONNECTIVITY	
Wired	56K[3] v.92 Internal Modem; 10/100/1000 Gigabit[6] Ethernet network interface adaptor
Mobile Broadband:	Cellular Broadband: Dell Wireless 5500 Mobile Broadband 3G HSDPA (Cingular US) Dell Wireless 5700 Mobile Broadband CDMA EVDO (Verizon US)
Wi-Fi:	Intel® PRO/Wireless 3945A/G (802.11a/g), Dell Wireless 1490 (802.11a/g), Dell Wireless 1390 (802.11g)
Bluetooth:	Dell Wireless 350 Bluetooth internal wireless card
EXPANDABILITY	
Express Card	54mm Express Card Slot; supports both 54mm and 34mm Express Cards
PC Card	One Type I or Type II
I/O Ports	Serial, 1394, docking connector, 4 USB, powered USB (D-Bay), VGA, headphone/speaker out, infrared port, RJ-11, RJ-45, AC power
Docking	D/Port , D/Dock, D/View Notebook Stand, D/Monitor Stand
Modular Options	24X CD-ROM, 8X DVD-ROM[7], 8X DVD+/-RW[8], 24X CDRW/DVD, Floppy Disk Drive, Secondary 6-cell/48 WHr "Smart" Li-Polymer battery, 2nd 80GB[4] hard drive, or TravelLite module
USB Memory Keys	128MB, 256MB and 512MB USB Memory Keys[5] (optional)
SECURITY	
Physical Security	Cable Lock Slots, Media Module and Hard Drive locks
User & System Security	Integrated Smart Card Reader, Trusted Platform Module 1.2 and optional UPEK® finger print reader. Dell Embassy® Trust Suite by Wave Systems security software.
Network Security	802.11 WiFi Protected Access (WPA), 802.11i (WPA2), Virtual Private Networks (VPN) and 802.1x with EAP modes, CCX V4.0

Figure 22. Kiosk System's Dell Latitude™ D820 Hardware Specifications
(Dell, 2005a)

Figure 23. Kiosk System's Dell D820 Latitudes

b. Fixed Camera System

The Fixed Camera System (FCS), like the Kiosk System, is a mix of multiple hardware components, primarily a series of cameras and laptop computers. The system hosts 36 WiLife cameras which are controlled by a correlating laptop computer. The laptops are connected via cabling to a switch that also links the Watchman Server where data is stored and facial recognition analysis applications are hosted. The system's purpose is to provide persistent observation of the second floor of Bullard Hall at the Naval Postgraduate School. To ensure maximum coverage, the 36 cameras were distributed between the major corridors of the building and select rooms.

The WiLife Logitech cameras used for the system provide an onboard 400 Megahertz (MHz) processor with 24 bits per pixel and 8-bit color data. The camera resolution of 320 x 240 or 640 x 480 pixels may be selected, as well as frame rates of 5, 10, or 15 frames per second. The cameras are connected by Power over Ethernet (PoE) cabling into a PoE injector to provide continuous 48-VDC power to the camera as well as connectivity for data transmission (Goshorn, 2010). The PoE injectors then connect to a switch which links the camera data to the controller laptop. Given a software constraint of the WiLife Command Center application, only six cameras can be paired with a single laptop. This requirement dictated the need to operate and maintain six laptops as part of the system function. Figure 24 depicts the physical architecture of the Fixed Camera System.

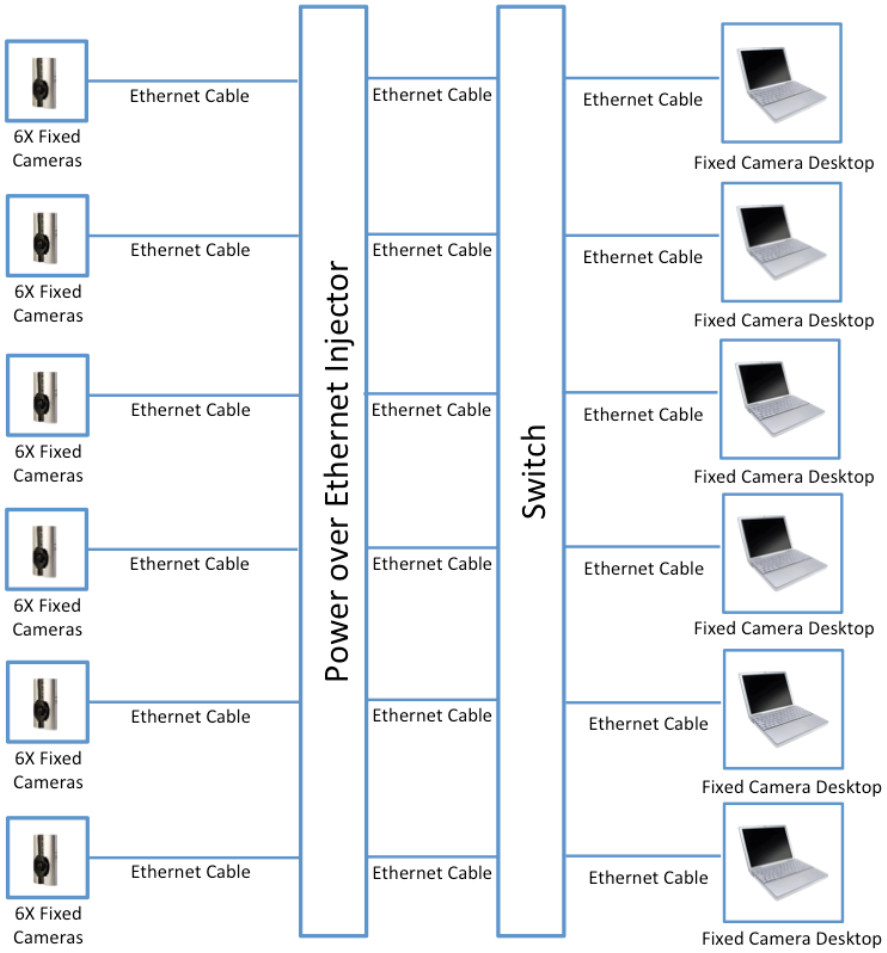

Figure 24. Fixed Camera System Physical Architecture

The six desktops used for the Fixed Camera System are Dell Precision™ 490 Desktops. The desktops use the Windows XP OS and 17-inch monitors to perform the functions of the system. Figure 25 lists the hardware specifications for the desktops and Figure 26 is an image of the computer chassis.

Dell Precision™ 490 Workstation

FEATURES	Dell Precision™ 490 Workstation
SYSTEM	
Processors	Dual-core Intel® Xeon® 5100 series processors with up to 1333MHz front side bus and 4MB shared cache; Quad-core Intel Xeon 5300 series processors with up to 1333MHz front side bus and 2 x 4MB shared cache; All processors are 64-bit, XD and VT capable
Operating Systems	Genuine Windows® XP Pro SP2 Genuine Windows® XP Professional x64 Edition Windows Vista™ capable[6] Red Hat Enterprise Linux WS v.4 (Intel EM64T)
Chipset	Intel® 5000X chipset
Memory	Up to 32GB[2] quad-channel[3] architecture DDR2 Fully Buffered DIMM (FBD) 533 and 667MHz ECC memory; Eight DIMM slots
Flash BIOS	8MB flash memory for system BIOS; SMBIOS 2.3.4 support
Graphics	Support for PCI Express x16 graphics cards up to 150 watts and with up to 512MB graphics memory including: NVIDIA Quadro® FX 4500; NVIDIA Quadro® FX 3500; ATI FireGL 7200; NVIDIA Quadro® FX 3450; NVIDIA Quadro® FX550; ATI™ FireGL™ V3400; NVIDIA Quadro® NVS 285; All graphics cards support dual monitor configurations
Hard Drives	SATA 3.0Gb/s with NCQ 7200 RPM with 16MB DataBurst Cache™ up to 750 GB[7]; SATA 3.0Gb/s with NCQ 7200RPM with 8MB DataBurst Cache™ up to 250GB[7]; SATA 10K RPM with 16MB DataBurst Cache™ up to 160GB[7]; SAS 10K RPM up to 300GB[7]; SAS 15K RPM up to 146GB[7]; Chassis supports up to three internal drives (2.0TB[7] maximum storage capacity)
Hard Drive Controller	Integrated SATA 3.0Gb/s controller with support for RAID 0, 1, and 5; SAS RAID 0, 1 with optional SAS 5i/r PCI-E card; SAS RAID 0, 1, 5, (in mini-tower orientation only) with optional PERC 5/i PCI-e card Integrated Broadcom® 5751 Gigabit4 Ethernet controller
Audio Controller	Integrated High Definition Audio (Rev 1.0 Specification) with Sigmatel STAC9200 High Definition Audio CODEC and Intel ESB2's AC97/ High Definition digital controller
Standard I/O Ports	Eight USB 2.0: two on front panel, five on back panel, one internal on motherboard; Two serial; One parallel; Two PS/2; One RJ-45; Stereo line-in and headphone line-out on back panel; Microphone and headphone connector on front panel
Optional I/O	IEEE 1394a connector available on front panel with add-in card
CHASSIS	
Dual-orientation Desktop	Desktop orientation with feet: (WxHxD) 17.64" x 6.73" x 18.54" (maximum including badge); 44.8 cm x 17.1 cm x 47.1 cm (maximum including badge)Mini-tower orientation with feet: (WxHxD) 6.73" x 17.64" x 18.54"; 17.1 cm x 44.8 cm x 47.1 cm (maximum including badge)Two internal 3.5" HDD bays; Two external 5.25" optical bays, one of which can accommodate a third HDD in mini-tower orientation.One external 3.5" flex bay for floppy or media card reader, one of which can accommodate a third HDD in desktop orientation; One PCI-e x16 graphics slot, Two PCI-e x8 slots wired as x4, Two PCI-X 64bit/100MHz slots with support for 3.3v or universal cards, One PCI 32bit/33Mhz slot; 750 watts Power Factor Correcting (PFC) power supply
PERIPHERALS	
Monitors	Performance flat panel displays, Dell UltraSharp™ widescreen and standard flat panel displays from 17" viewable to 30" viewable; Analog flat panel displays and CRT monitors also available
Keyboard	Dell Enhanced QuietKey USB; Enhanced Multimedia USB; Smart Card keyboard USB
Mouse	Dell USB two-button mouse and Dell USB optical two-button scroll mouse
Optional Speakers	Internal chassis speaker; Dell two and three piece stereo system; Dell sound bar available for all flat panel displays
STORAGE DEVICES	
Optional Removable Storage	CD-ROM, CD-RW, CD-RW/DVD Combo, DVD-ROM[5], DVD+/-RW[10], USB Floppy Drive, USB media card reader
Optional Modem	Dell 56K[12] v.92 Data/Fax PCI modem
SECURITY	
Software	Trusted Platform Module 1.2 (TPM 1.2); Chassis intrusion switch; Setup/BIOS Password; I/O Interface Security
ENVIRONMENTAL & REGULATORY	
Standards	TCO99, Blue Angel, Green PC, Energy Star, BSMI, C-TICK, CE, FCC, IRAM, NEMKO, NFPA 99, SABBS, SASO, TCO, TUV, UL, VCCI, USB 2.0; WEEE
Lead Free	Environmentally conscious design is RoHS Compliant[13]/Lead Free[14]
SERVICE & SUPPORT	
Base	3-Year Limited Warranty[15] with 3 years standard Next Business Day (NBD) onsite[16] parts replacement and 3 years NBD onsite[16] labor (US Only)
Recommended	3-Year Same Business Day 4 hour On-site Service[16] – 5 days a week, M-F 10 hours a day (8-6PM) 3-Year Same Business Day 4 hour On-site Service[16] – 7 days a week, 24 hours a day 3 & 4-year Gold Technical Support, expert support via phone, e-mail and online chat - 7 days a week, 24 hours a day

Figure 25. Fixed Camera System's Dell Precision™ 490 Desktop Hardware Specifications
(Dell, 2005b)

Figure 26. Fixed Camera System's Dell Precision™ 490 Desktop
(ImageShack, n.d.)

The six desktops use the WiLife Command Center software to control the cameras assigned to the individual computers. Video and images from the cameras are transmitted to the desktops and managed by the vendor software. The Command Center software controls the functions of the cameras and provides motion detection criteria. Additionally, the software auto-generates and organizes video files in the Windows Media Video format or .wmv. The raw .wmv files are then analyzed by MATLAB Simulink software as part of the system Detect function. The Detect function searches the raw video for human forms based on established criteria written into the MATLAB software. Videos containing positively identified human forms are sent to the Watchman Server for further analysis. Finally, MATLAB writes data regarding an observed instance to the Watchman Server in SQL format using ODBC to the Watchman database.

c. Middleware System

The Middleware System enabling interoperability standards are the Geospatial Hub (GHub) system and the database system created for the DGIAS. The GHub is a geospatially conscious content management system that classifies and distributes information developed by users, analysts, or sensor platforms (Sample & Ioup, 2010). The middleware is made up of two instances of GHub (one to emulate an Unclassified instance and one to emulate a Classified instance). The outside systems' interfaces are a subsystem within GHub that allows for the middleware to connect to other services outside of the Naval Postgraduate School. Figure 27 represents the physical architecture of the Middleware System.

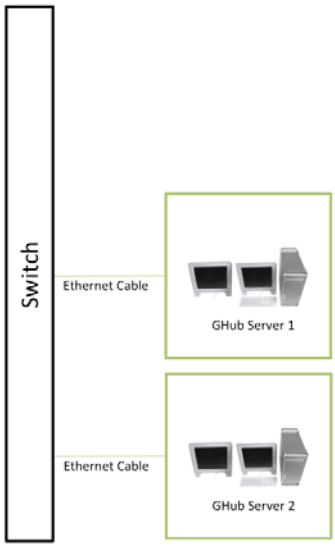

Figure 27. Middleware System (Geospatial Hub) Server Physical Architecture

d. Watchman Viewer System

The Watchman Viewer acts as the Command and Control center for the DGIAS system. It is the central component system of the DGIAS where high-level analysis is performed. The Watchman System is hosted on an Apple Mac Pro computer with the Mac OS X hosting a VMWare VM of Windows Server 2004. The system requires a VM to permit the applications of SQL Server, Microsoft Access, and WiLife Command Center to run in their native Windows environment. To run the Windows Server 2004, a minimum number of two CPU cores must exist. The Mac Pro more than meets this need with its two 3.2 GHz, Quad Core Intel Xeon processors (or eight cores), and 32 GBs of double data rate (DDR) RAM. Through two monitors the system offers a user the choice between viewing the Mac OS X display or the Windows Server 2004 display. As in the Kiosk and Fixed Camera systems, the MATLAB software writes facial recognition data to the Watchman Server in SQL format to the Watchman database. Once data is recorded in the Watchman database, it can be retrieved through the Microsoft Access GUI (graphical user interface) available through the

Windows Server display. Figures 28 and 29 depict the physical and virtualization architectures of the Watchman Viewer System respectively.

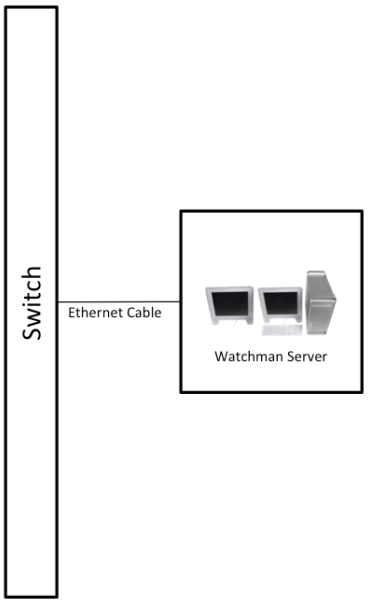

Figure 28. Watchman Viewer System Physical Architecture

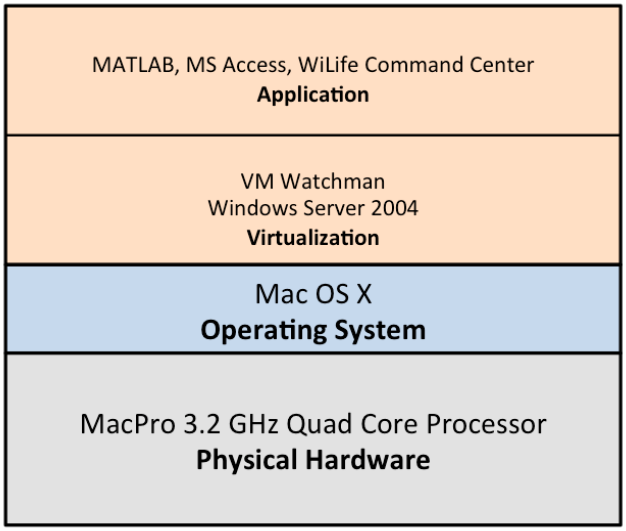

Figure 29. Watchman Viewer System Virtualization Architecture

Figures 30 and 31 identify the hardware specifications of the Mac Pro and its appearance.

Product Details

Processor	3.2GHz Quad-Core Intel Xeon "Nehalem" processor
Cache	8MB fully shared L3 cache per processor
Memory	3GB (three 1GB DIMMs) of DDR3 ECC SDRAM at 1066MHz; supports up to 32GB
PCI Express 2.0 graphics	ATI Radeon HD 5770 with 1GB of GDDR5 memory
Hard disk drive	1TB[1]Serial ATA (3Gb/s); 7200 rpm
Optical drive	18x SuperDrive with double-layer support(DVD±R DL/DVD±RW/CD-RW)
Wireless	Built-in AirPort Extreme 802.11n Wi-Fi[3] and Bluetooth 2.1 + EDR (Enhanced Data Rate) wireless technology
PCI Express	Three open full-length PCI Express 2.0 expansion slots with mechanical support for 16-lane cards; 300W combined maximum for all PCI Express slots (PCI Express slots are not compatible with PCI or PCI-X expansion cards)
Connections and audio	Four FireWire 800 ports (two on front panel, two on back panel) Five USB 2.0 ports (two on front panel, three on back panel) Two USB 2.0 ports on included keyboard Front-panel headphone minijack and internal speaker Optical digital audio input and output TOSLINK ports Analog stereo line-level input and output minijacks
Size and weight	Height: 20.1 inches (51.1 cm) Width: 8.1 inches (20.6 cm) Depth: 18.7 inches (47.5 cm) Weight (standard configuration): 39.9 pounds (18.1 kg)[2]

Figure 30. Watchman System's Mac Pro Hardware Specifications
(Apple, 2012)

Figure 31. Watchman System's Mac Pro Server
(Apple, 2012)

D. PROPOSED DGIAS VIRTUALIZATION

1. Description

The purpose of the proposed DGIAS is to model how the system would be architected if it were migrated to a virtualization environment. Not all systems can

function with a virtualization layer of software. For example, the camera system's PoE injectors do not have the possibility of being virtualized given their distinctive function. However, a significant number of the components, specifically the laptops and desktops, can all be consolidated to two blade servers. Figure 32 depicts the new physical architecture. In this diagram, the two Kiosk System laptops are replaced by two thin-clients, and the six FCS desktops are eliminated from the system. The server systems of Watchman and GHub now share the same hardware as the Kiosk System (vAlpha), and the FCS is self-contained on a single server (vBeta).

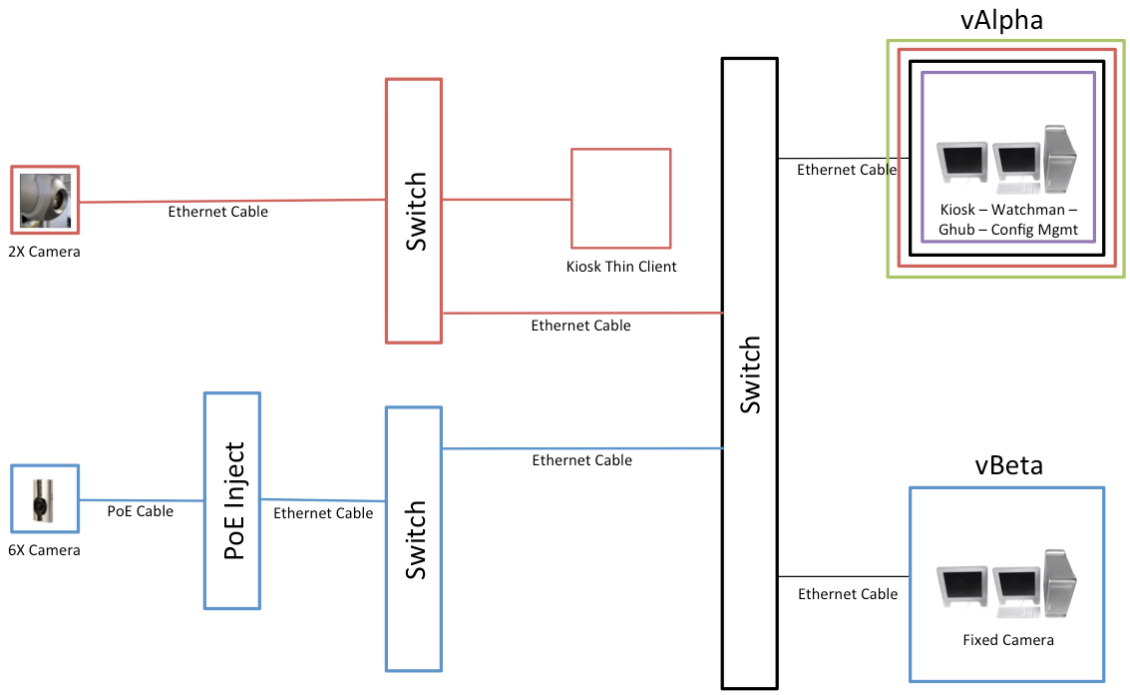

Figure 32. Proposed Physical Architecture of DGIAS

The architecture also accounts for the addition of the Configuration Management software required to manage the environment. The Configuration Management software, as discussed in Chapter II, provides an administrator the management tools necessary to create, replicate, and control all VMs in an environment. Figure 33 represents the virtualization architecture of the proposed DGIAS system consolidated to vAlpha and vBeta.

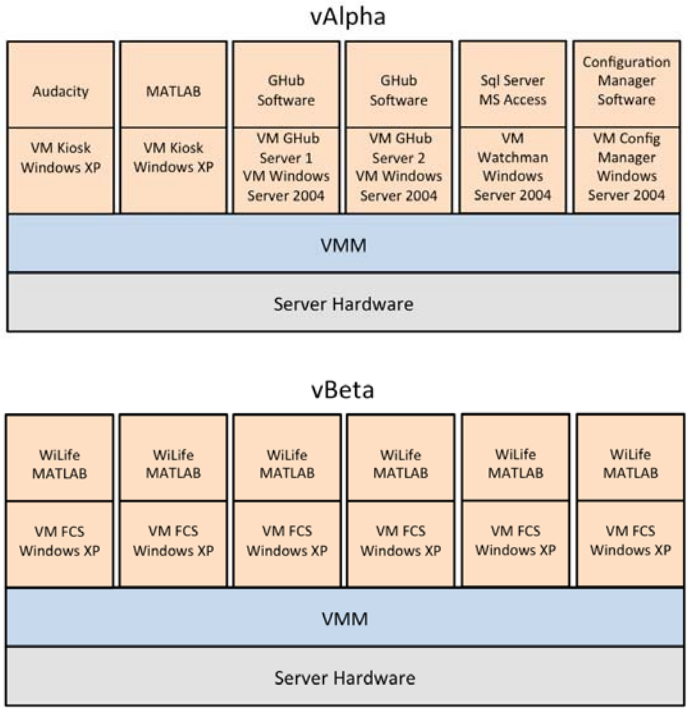

Figure 33. Virtualization Architecture of vAlpha & vBeta

A consolidation rate of six VMs to one server is conceivable given that most of the systems require only a single core processor to run the Windows XP OS, as is the case with all of the FCS VMs. For the systems which require multicore processors to run the Windows Server 2004, their workloads are more infrequent given that only higher level analysis is performed and therefore will not overload the system hardware. However, if the systems hosted on vAlpha do require more resources, then vBeta, as part of a system preference, could automatically accept the Kiosk systems' VMs to balance the processor and memory requirements across both sets of the hardware. The movement of VMs across hardware is a process available for most virtualization vendor platforms. If all the DGIAS systems were to be virtualized, two more blade servers, at a minimum, would be required to accommodate the 12 other systems. This would provide the minimum resources necessary while maintaining a 20% reserve capacity.

E. DGIAS TEST AND EVALUATION PROCESS MODEL

To identify some of the efficiencies gained by using virtualization, a process model was created to simulate a generic test and evaluation process. To model the T&E process for an SoS, I again reference the work of Goshorn (2010) to provide the necessary framework. Recall from Chapter III the Core model, as seen in Figure 34; it provides the empirical data of a system's development cycle, but also the development cycle of a new ECO as it is integrated across the component systems of an SoS or FoS. To provide the necessary context, I assumed that a probable ECO would be a modification to the face detection algorithm of the DGIAS. An ECO of this type would require the change to be enacted across multiple computers. For simplicity, the period of a 40-hour week was selected as a realistic and manageable time frame across which to distribute the Core model for the purposes of modeling.

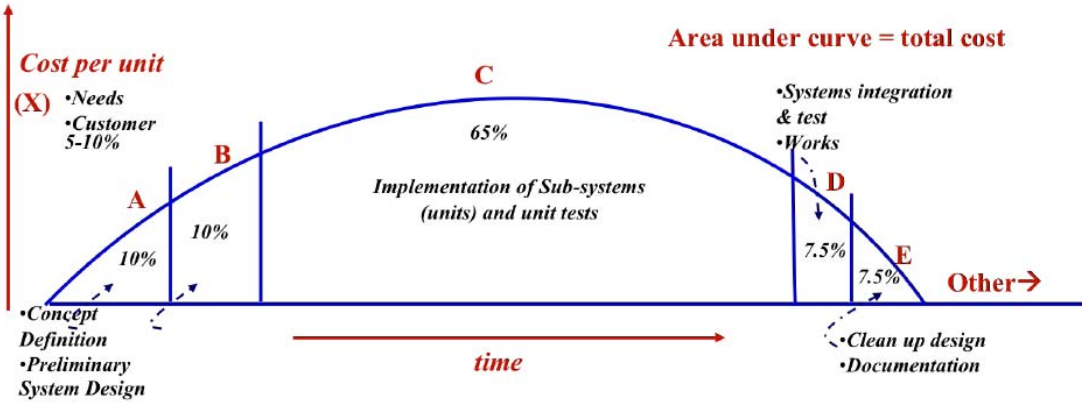

Figure 34. Core Model
(Goshorn, 2010)

Table 1 lists the distribution of the 40 hours across the phases of the Core model. Additionally, the calculations in the model account for the different types of testing to be completed throughout the ECO integration. The different types of testing that were selected are titled Functional, Low-level thread, Medium-level thread, and High-level thread. The thread based testing categories and related time requirements are all assumptions necessary to account for the varying types

of tests that occur, while meeting the precise percentages suggested by the Core model. An example of a Low-level thread might be requiring the system to observe, detect, and react to a single individual. An example of a Medium-level thread would be requiring the system to respond to a group of three to five people. A High-level thread would require the system to respond to 10 to 20 people.

Table 1. DGIAS ECO Integration With Core Model Hours Breakdown for 40-Hour Period

Core Model	Process	Time %	Functional	Low	Med	High	Report	Cumulative
A	Concept	10%	1.75	0	0	0	0	1.75
B	Detail	10%	1.75	1.5	1.5	1.5	0	6.25
C	Build	65%	13	6.5	3.25	3.25	0	26
D	Test	7.50%	1	1	0.5	0.5	0	3
E	Document	7.50%					3	3
							Total	**40**

Note. All units are in hours.

The DGIAS ECO T&E process model begins with the issuing of an ECO requirement by the lead systems engineer. The ECO is simultaneously passed to the three component system teams of Kiosk, Fixed Camera, and the combined GHub and Watchman System Team. Once the teams have completed the conceptual and detailed design for the Functional test, the designs are passed to the T&E Environment Team to allocate the appropriate hardware and software and then configure the environment correctly. Following a complete system build, the SoS Integration Test Team conducts a functional test. The results are recorded and sent to the component system teams again for a subsequent period of detailed design. The designs are again sent to the T&E Environment Team, which modifies the hardware and software and configures the SoS. Once complete, the SoS Integration Test Team conducts a Low-level thread test. The processes completed for the Low-level thread test are repeated for the Medium- and High-level thread tests. After the completion of the High-level thread test, the SoS Integration Team prepares the documentation and submits the report to end

the process model. Table 2 outlines the participants of the model and their full-time equivalent wage per hour. Figure 35 depicts the implementation of the model; each swim lane represents a participant. Appendix A contains the data output from the model with 100% of the Phase C activities occurring with physical machines. As a parameter for the model, 10 ECOs were simulated arriving every 40 hours for a total of 400 hours' worth of work completed. The costs accrued totaled $17,773.60 for the work completed.

Table 2. DGIAS ECO Integration Participants

Participant	Process	Wage per hour
Lead Systems Engineer	Issue ECO	NA
Kiosk System Team	Conceptual Design and Detailed Design	$39.48
Fixed Camera System Team	Conceptual Design and Detailed Design	$39.48
GHub and Watchman System Team	Conceptual Design and Detailed Design	$39.48
T&E Environment Team	Implement designs through a mix of hardware, software, and virtualization.	$24.14
SoS Integration Test Team	Conduct Functional and Low, Med, and High-level thread based tests. Produce documentation for final report.	$33.70

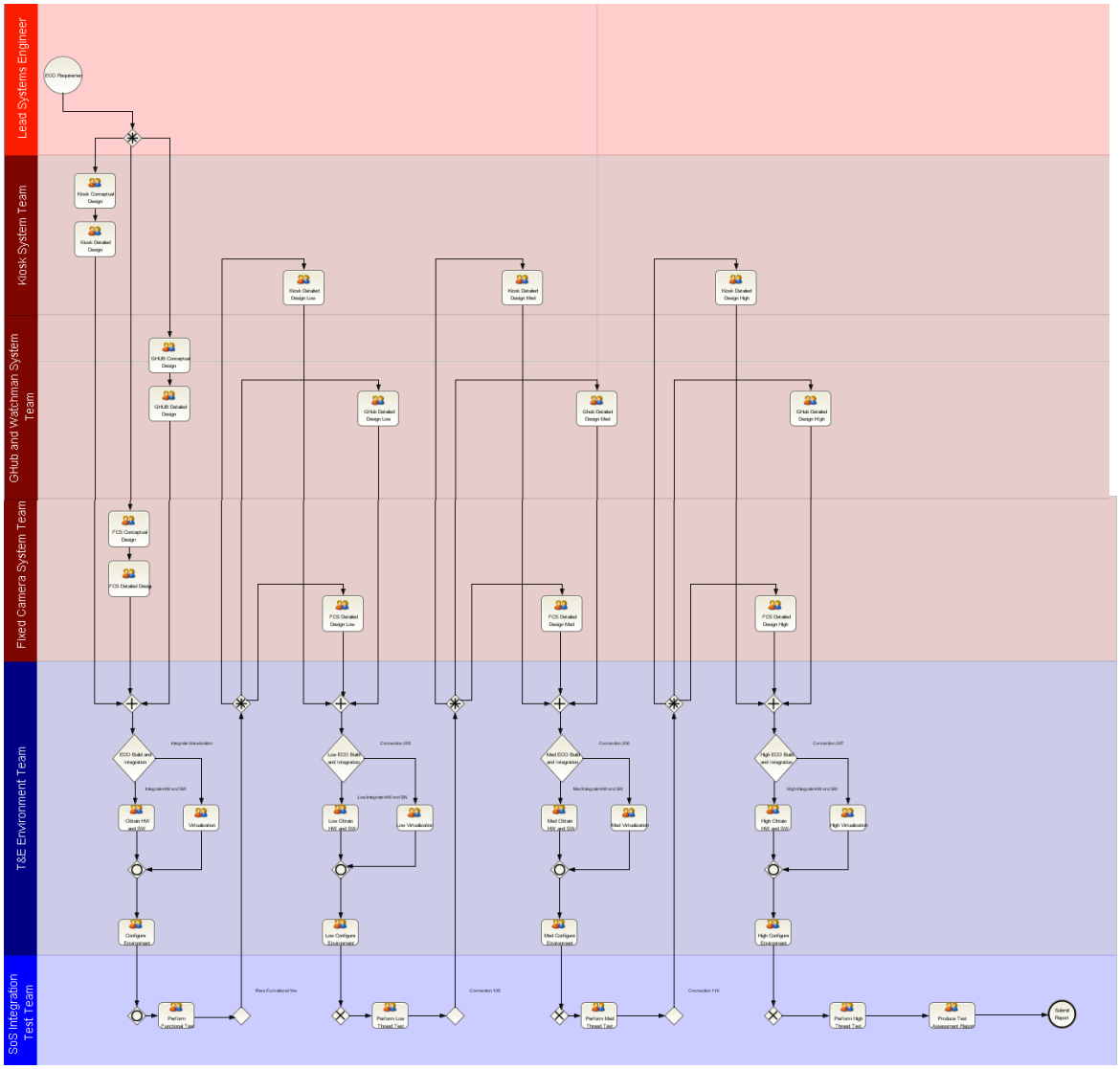

Figure 35. DGIAS ECO Implementation Model

To achieve virtualization, efficiencies in the process Phase C, the Build and Implementation processes, must be addressed. The first detail that I had to determine in my research was the ratio of building physical machines compared to virtual machines. Table 3 establishes the baseline times for the activities normally performed in a T&E computer environment. In the first column of the table are the activities connected with building a physical machine. Next, in the center are the actions related to building a VM. Finally, on the right are the activities required for copying a VM from a known good copy. While all the times

in this table are not universal, they demonstrate the time efficiency that is gained by using virtualization.

Table 3. Activity Times

Physical Machine Build Activity	Time	Virtual Machine Build Activity	Time	Virtual Machine Copy Activity	Time
Build Physical Computer	29	Build Virtual Machine	1	Copy Virtual Machine	1
Install OS	20	Install OS	20	Install Applications	9
Install Applications	9	Install Applications	9	Configure for network	2
Configure for network	2	Configure for network	2		
Total	**60**		**32**		**12**

Note. All units are in minutes.

It should be clear from Table 3 that there are significant time savings when building a VM compared to a physical machine. There are still fixed periods in the process, such as installing an OS, installing applications, and configuring the client for the network. The difference between building a complete physical machine compared to building a complete VM is approximately 28 minutes, or a savings of 47%. Once a VM is built, it can be copied and pasted in the environment, thereby eliminating the need to install an OS. The difference between copying a VM and building a physical machine is approximately 48 minutes, or a savings of 80%. To help an engineer determine some of the efficiencies gained by using virtualization, the following calculation was developed: 32 minutes multiplied by x, where x equals the number of unique systems in the SoS, plus 12 minutes times y, where y equals the number of clients or instantiations of the different types of systems (see Equation 1).

$$32x + 12y = VirtualizationEnvBuildTime \quad (1)$$

For example, the DGIAS has four different types of systems: Kiosk, FCS, Watchman, and GHub and seven copies. Following the calculation (32*4) + (12*7), it takes 212 minutes to create a suitable virtualization environment. The ratio of the time it would take to create the environment with VMs compared to physical machines is approximately 1:3. This efficiency is created simply by

building the SoS with VMs. By consolidating the system to two blade servers and a SAN storage array, the system can rapidly be updated or altered should a new ECO be required. For example, if a change needed to be completed to the MATLAB Simulink Detect algorithm in the FCS, six separate actions would be required by the T&E Environment Team responsible for the modification. Likewise, if the algorithm change created an error or changed the system stability, it would have to be removed six separate times. In the proposed system, a change to a single FCS VM could be replicated across the system in a single process, thereby reducing the amount of work significantly. Additionally, the Environment Team could record the entire system state to the SAN array prior to the integration of the ECO. The captures or snapshots of the system state before the integration would allow a quick rollback to the previous system configuration. Figure 36 shows the delineation between build times of physical machines and virtual machines. The Virt (Worst) line, which is still considerably faster than building a physical machine, is calculated by only using unique system types with no additional copies. The Virt (Worst) calculation would be (32*11) + (12*0) = 352 minutes. The Virt (Best) was calculated using (32*1) + (12*10) = 152 minutes. This would be a system in which all the clients are copies of the original system. The DGIAS is plotted between the best and the worst given its mix of systems and copies.

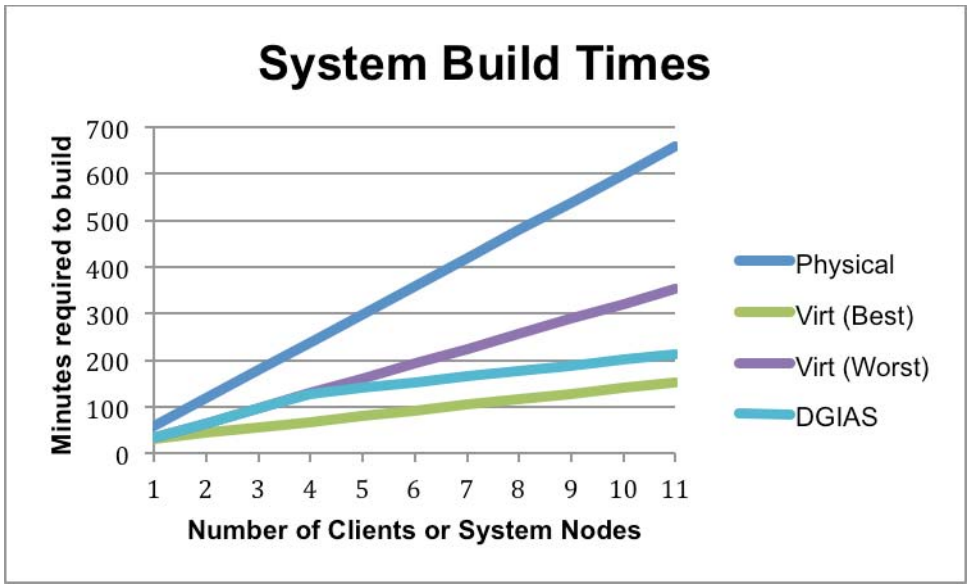

Figure 36. System Build Times

The more common the systems are and the more VMs that are created, the greater the time savings. The configuration time between two VMs and two physical machines is not as significant as the configuration time between 11 VMs and 11 physical machines.

Finally, Table 4 lists the different tasks of Phase C in the Core model. The following information was used in part to generate Appendix B data, which is a full virtualization environment. Although it may be unlikely to convert 100% of the system to virtualization, it is important to understand the limitations of the technology as it applies to the process. An 11% reduction could be achieved from the overall Core model process as it applies to the DGIAS. Not all systems will be as suited as the DGIAS; therefore, the savings will be some number less than 11%. Table 4 highlights the shift from 65% to 54% when comparing C_{1+3} to C_{2+3}. Appendix B also shows the reduction in overall cost from $17,773.60 in the original model to $16,705.00.

Table 4. DGIAS ECO Integration Phase C—Process Tasks

Core Model	Task	Time %	Functional	Low	Med	High	Report	Cumulative
C_1	Obtain HW SW	16%	3.25	1.63	.81	.81	0	6.5
C_2	Virtualization	5%	1.04	.52	.26	.26	0	2.08
C_3	Configure Env	49%	9.75	4.88	2.44	2.44	0	19.50
C_{1+3}	Obtain HW SW Configure Env	65%	13	6.5	3.25	3.25	0	26
C_{2+3}	Virtualization Configure	54%	10.79	5.40	2.7	2.7	0	21.58

Note. All units are in hours.

F. CONCLUSION

The DGIAS is an SoS that suits the virtualization platform given its Windows-based OSs and standard desktop, laptop, and server hardware. The migration to virtualization will improve the engineering team's abilities to integrate and test new ECOs. The system combined with the Core model is an ideal case study for the strengths of virtualization. By centralizing the computing to two capable servers, the configuration management of the system will be significantly improved.

V. CONCLUSION

A. SUMMARY

Virtualization, when matched with a compatible technology, offers immense benefits to the test and evaluation phase of an IT project. Virtualization can improve efficiencies in time including reduced labor hours, elimination of redundant tasks, easy rollback to previous system states, and reduced labor expenses. Within the case study of the DGIAS system of systems testing, virtualization resulted in an 11% reduction in time spent for each integrated engineering change order. Although virtualization is not ideal in all circumstances, it has shown great promise as a way to improve the T&E process within SoS.

For virtualization to be a viable option, several conditions must be met. First, the system nodes must be comprised of Windows or Linux x86 operating systems. Second, the existing servers or prospective servers must be capable of managing the system's workloads or user demands. Lastly, the storage requirement of the system clients must be less than the available SAN storage array of the test environment. If a system meets all of these requirements, then migration to virtualization is a possible option for the IT environment.

DGIAS is a candidate for virtualization because several of its component systems meet the three key requirements described in this summary. The DGIAS uses the Windows XP and Windows Server OSs, can be consolidated to two servers, and requires less than 17 TBs of storage. Also, because it is such a complex system of systems, there are likely to be many software-based ECOs within the test and evaluation process. By implementing virtualization, time savings can be gained with each ECO.

By involving virtualization in the ECO process, system developers can save an average of 11% time savings over the life cycle of the testing process. As shown in Chapter IV, the average time saved is four hours per ECO.

Assuming the number of ECOs per system is 10 with virtualization of the DGIAS, the system could save over $1,000 and several hours. These data were substantiated with a simple formula (see Equation 2) developed to account for the time savings achieved when incorporating virtualization into the system architecture.

$$32x + 12y = VirtulizationEnvironmentTime \tag{2}$$

This formula helps to account for the variety of systems and the number of clients or copies the system hosts. It is not meant to replace a detailed modeling and simulation process, but rather to be used for high-level analysis when deciding between virtualization and physical machines. Virtualization is not for every system and it requires specific types of system traits to provide efficiencies. But when it is paired with the right type of system architecture, it quickly can provide dividends to the system engineers and designers who leverage it.

B. FURTHER RESEARCH AND RECOMMENDATIONS

1. Limits of Virtualization

While virtualization is a useful tool, it does have its limitations. Virtualization is susceptible to a time drift problem, which is more likely during times of high workloads. This causes VMs to lose time which may impact the performance of a given application or system. Engineers should consider the importance of time to the overall system performance before implementing a virtualization environment. If time is critical to the system performance, then virtualization should be avoided. Methods do exist to minimize the impact of time drift on VMs, but they must be built into the system design. Further research and adaptation is needed to solve this problem of time drift. A solution would allow projects susceptible to time to be able to utilize virtualization when currently they cannot.

2. Improved Capabilities

Currently virtualization can only operate with Windows and Linux operating systems. If it could be expanded to include Apple OS, it would open even more projects and systems to virtualization. Another improvement would be to allow for mobile devices to be easily integrated into the virtual environment .

3. Further Case Studies

Extensive research into the field of virtualization has shown very little empirical data qualifying or quantifying the validity of the process. It is difficult to determine through statistical means the actual efficiencies gained from virtualizing the test and evaluation process because there are no specific reference case studies to turn to. As more companies adopt virtualization as a valid tool, there needs to be more literature on the process outcomes to guide future decision-makers. There needs to be more cooperation within the growing IT business market to share the virtues of virtualization. If there is a wide spectrum of outcomes after virtualization, then upgrades can be made to the software to try and ensure that it leads to future efficiency gains. However, until the benefits and limitations of virtualization are studied on a grander scale, then a system developer must make utilization decisions based solely on assumptions and trial and error.

4. Specific Measuring Tool

Another recommendation for advancing and improving virtualization is to create a specific, universal formula for determining time-efficiencies using virtualization. This mathematically based formula would work for all projects universally and would aid process managers in deciding whether virtualizing all or some of their test and evaluation will result in improved efficiencies and thus cost savings. This would require further study of the specific components of virtualization as well as the study of other cases where it was implemented to determine if a standard of measure can be created.

Virtualization, when compatible within a given technology, offers immense benefits to the test and evaluation phase of an IT project. It can improve efficiencies in time including labor hours, reduce redundancy in effort, eliminate potential loss of test results, and save money on hardware expenses. Within the case study of DGIAS system of systems testing, virtualization has shown a 20% reduction in time spend for each ECO ordered. Although virtualization is not ideal in all circumstances, it has shown great promise as a way to improve the T&E process within system of systems.

APPENDIX A

The data in this appendix is the result of modeling 10 ECOs introduced at an interval of one every 40 hours through the Core model within the DGIAS. This report represents the current system T&E environment.

Simulation Results					
Duration	400:00:00	Time			
Process Time And Cost					
Process	**Scenario**	**Instances**	**Total Cost**	**Waiting Time (Time)**	**Total Time (Time)**
DGIAS_Virtualization	(default)	10	17773.6	0:00:00	400:00:00
DGIAS_Virtualization					
Instances	10				
Activity	**Performer**	**Occurs**	**Waiting Time (Time)**	**Time To Complete (Time)**	**Total Time (Time)**
Configure Environment	Any member of T&E Environment Team	10	0:00:00	97:30:00	97:30:00
FCS Conceptual Design	Any member of Fixed Camera System Team	10	0:00:00	17:30:00	17:30:00
FCS Detailed Design	Any member of Fixed Camera System Team	10	0:00:00	17:30:00	17:30:00
FCS Detailed Design High	Any member of Fixed Camera System Team	10	0:00:00	15:00:00	15:00:00
FCS Detailed Design Low	Any member of Fixed Camera System Team	10	0:00:00	15:00:00	15:00:00
FCS Detailed Design Med	Any member of Fixed Camera System Team	10	0:00:00	15:00:00	15:00:00
GHub Conceptual Design	Any member of GHub System Team	10	0:00:00	17:30:00	17:30:00
GHub Detailed Design	Any member of GHub System Team	10	0:00:00	17:30:00	17:30:00
GHub Detailed Design High	Any member of GHub System Team	10	0:00:00	15:00:00	15:00:00
GHub Detailed Design Low	Any member of GHub System Team	10	0:00:00	15:00:00	15:00:00
GHub Detailed Design Med	Any member of GHub System Team	10	0:00:00	15:00:00	15:00:00
High Configure Environment	Any member of T&E Environment Team	10	0:00:00	24:22:30	24:22:30
High Obtain HW and SW	Any member of T&E Environment Team	10	0:00:00	8:07:30	8:07:30
Kiosk Conceptual Design	Any member of Kiosk System Team	10	0:00:00	17:30:00	17:30:00
Kiosk Detailed Design	Any member of Kiosk System Team	10	0:00:00	17:30:00	17:30:00
Kiosk Detailed Design High	Any member of Kiosk System Team	10	0:00:00	15:00:00	15:00:00
Kiosk Detailed Design Low	Any member of Kiosk System Team	10	0:00:00	15:00:00	15:00:00
Kiosk Detailed Design Med	Any member of Kiosk System Team	10	0:00:00	15:00:00	15:00:00
Low Configure Environment	Any member of T&E Environment	10	0:00:00	48:45:00	48:45:0

	Team				0
Low Obtain HW and SW	Any member of T&E Environment Team	10	0:00:00	16:15:00	16:15:00
Med Configure Environment	Any member of T&E Environment Team	10	0:00:00	24:22:30	24:22:30
Med Obtain HW and SW	Any member of T&E Environment Team	10	0:00:00	8:07:30	8:07:30
Obtain HW and SW	Any member of T&E Environment Team	10	0:00:00	32:30:00	32:30:00
Perform Functional Test	Any member of SoS Integration Test Team	10	0:00:00	10:00:00	10:00:00
Perform High Thread Test	Any member of SoS Integration Test Team	10	0:00:00	5:00:00	5:00:00
Perform Low Thread Test	Any member of SoS Integration Test Team	10	0:00:00	10:00:00	10:00:00
Perform Med Thread Test	Any member of SoS Integration Test Team	10	0:00:00	5:00:00	5:00:00
Produce Test Assessment Report	Any member of SoS Integration Test Team	10	0:00:00	30:00:00	30:00:00

Resource	Unit	Cost/Unit	Threshold	Usage	Cost
Any member of T&E Environment Team	Hour	24.14	0	260	$6,276.40
Any member of SoS Integration Test Team	Hour	33.7	0	60	$2,022.00
Any member of Kiosk System Team	Hour	39.48	0	80	$3,158.40
Any member of GHub System Team	Hour	39.48	0	80	$3,158.40
Any member of Fixed Camera System Team	Hour	39.48	0	80	$3,158.40

Performers Queue Length and Utilization

Name	Average	Min	Max	Utilized(%)	Idle(%)
Any member of T&E Environment Team	0	0	0	65	35
Any member of SoS Integration Test Team	0	0	0	15	85
Lead Systems Engineer	0	0	0	0	100
Any member of Kiosk System Team	0	0	0	20	80
Any member of GHub System Team	0	0	0	20	80
Value of 'Creator'	0	0	0	0	100
Generic	0	0	0	0	100
Any member of Fixed Camera System Team	0	0	0	20	80

APPENDIX B

The data in this appendix is the result of modeling 10 ECOs introduced at an interval of one every 40 hours through the Core model within the DGIAS. This report represents the proposed virtualization system T&E environment.

Simulation Results					
Duration	395:34:24	Time			
Process Time And Cost					
Process	**Scenario**	**Instances**	**Total Cost**	**Waiting Time (Time)**	**Total Time (Time)**
DGIAS_Virtualization	(default)	10	16705	0:00:00	355:44:00
DGIAS_Virtualization					
Instances	10				
Activity	**Performer**	**Occurs**	**Waiting Time (Time)**	**Time To Complete (Time)**	**Total Time (Time)**
Configure Environment	Any member of T&E Environment Team	10	0:00:00	97:30:00	97:30:00
FCS Conceptual Design	Any member of Fixed Camera System Team	10	0:00:00	17:30:00	17:30:00
FCS Detailed Design	Any member of Fixed Camera System Team	10	0:00:00	17:30:00	17:30:00
FCS Detailed Design High	Any member of Fixed Camera System Team	10	0:00:00	15:00:00	15:00:00
FCS Detailed Design Low	Any member of Fixed Camera System Team	10	0:00:00	15:00:00	15:00:00
FCS Detailed Design Med	Any member of Fixed Camera System Team	10	0:00:00	15:00:00	15:00:00
GHub Conceptual Design	Any member of GHub System Team	10	0:00:00	17:30:00	17:30:00
GHub Detailed Design	Any member of GHub System Team	10	0:00:00	17:30:00	17:30:00
GHub Detailed Design High	Any member of GHub System Team	10	0:00:00	15:00:00	15:00:00
GHub Detailed Design Low	Any member of GHub System Team	10	0:00:00	15:00:00	15:00:00
GHub Detailed Design Med	Any member of GHub System Team	10	0:00:00	15:00:00	15:00:00
High Configure Environment	Any member of T&E Environment Team	10	0:00:00	24:22:30	24:22:30
High Virtualization	Any member of T&E Environment Team	10	0:00:00	2:36:00	2:36:00
Kiosk Conceptual Design	Any member of Kiosk System Team	10	0:00:00	17:30:00	17:30:00
Kiosk Detailed Design	Any member of Kiosk System Team	10	0:00:00	17:30:00	17:30:00
Kiosk Detailed Design High	Any member of Kiosk System Team	10	0:00:00	15:00:00	15:00:00
Kiosk Detailed Design Low	Any member of Kiosk System Team	10	0:00:00	15:00:00	15:00:00
Kiosk Detailed Design Med	Any member of Kiosk System Team	10	0:00:00	15:00:00	15:00:00

Low Configure Environment	Any member of T&E Environment Team	10	0:00:00	48:45:00	48:45:00
Low Virtualization	Any member of T&E Environment Team	10	0:00:00	5:12:00	5:12:00
Med Configure Environment	Any member of T&E Environment Team	10	0:00:00	24:22:30	24:22:30
Med Virtualization	Any member of T&E Environment Team	10	0:00:00	2:36:00	2:36:00
Perform Functional Test	Any member of SoS Integration Test Team	10	0:00:00	10:00:00	10:00:00
Perform High Thread Test	Any member of SoS Integration Test Team	10	0:00:00	5:00:00	5:00:00
Perform Low Thread Test	Any member of SoS Integration Test Team	10	0:00:00	10:00:00	10:00:00
Perform Med Thread Test	Any member of SoS Integration Test Team	10	0:00:00	5:00:00	5:00:00
Produce Test Assessment Report	Any member of SoS Integration Test Team	10	0:00:00	30:00:00	30:00:00
Virtualization	Any member of T&E Environment Team	10	0:00:00	10:20:00	10:20:00

Resource	Unit	Cost/Unit	Threshold	Usage	Cost
Any member of GHub System Team	Hour	39.48	0	80	$3,158.40
Any member of Fixed Camera System Team	Hour	39.48	0	80	$3,158.40
Any member of T&E Environment Team	Hour	24.14	0	215	$5,190.10
Any member of SoS Integration Test Team	Hour	33.7	0	60	$2,022.00
Any member of Kiosk System Team	Hour	39.48	0	80	$3,158.40

Performers Queue Length and Utilization

Name	Average	Min	Max	Utilized(%)	Idle(%)
Any member of GHub System Team	0	0	0	20.22	79.78
Any member of Fixed Camera System Team	0	0	0	20.22	79.78
Any member of T&E Environment Team	0	0	0	54.54	45.46
Any member of SoS Integration Test Team	0	0	0	15.17	84.83
Value of 'Creator'	0	0	0	0	100
Generic	0	0	0	0	100
Lead Systems Engineer	0	0	0	0	100
Any member of Kiosk System Team	0	0	0	20.22	79.78

LIST OF REFERENCES

Abu-Taieh, M. O., & El Sheikh, A. A. R. (2007). Discrete event simulation process validation, verification, and testing. In A. Dasso & A. Funes (Eds.), *Verification, validation, and testing in software engineering* (pp. 155–184). Hershey, PA: Idea Group.

Adair, R. J., Bayles, R. U., Comeau, L. W., & Creasy, R. J. (1966). *A virtual machine for the 360/40* (Report No. 320-2007). Cambridge, MA: IBM Cambridge Scientific Center.

Advanced Micro Design (AMD). (n.d.). AMD virtualization. Retrieved from http://sites.amd.com/us/business/it-solutions/virtualization/Pages/virtualization.aspx#2

Apple. (2012). Refurbished Mac Pro 3.2 GHz Quad-Core Intel Xeon. Retrieved from http://store.apple.com/us/product/G0LF0LL/A

Balci, O. (1994). Validation, verification, and testing techniques throughout the life cycle of a simulation study. *Annals of Operations Research*, *53*, 215–220.

Balci, O. (1995). Principles and techniques of simulation validation, verification, and testing. In C. Alexopoulos, K. Kang, W. R. Lilegdon, & D. Goldsman (Eds.), *Proceedings of the 1995 Winter Simulation Conference* (pp. 147–154). New York, NY: ACM Press.

Balci, O., Glasow, P. A., Muessig, P., Page, E. H., Sikora, J., Solick, S., & Youngblood, S. (1996). *Department of Defense verification, validation and accreditation (VV&A) recommended practices guide*. Retrieved from http://vva.msco.mil/Mini_Elabs/VVtech-dynamic.htm#dyn1

Blanchard, B. S., Fabrycky, W. J. (2011). *Systems engineering and analysis*. Upper Saddle River, NJ: Prentice Hall.

Boehm, B. W. (1981). *Software engineering economics*. Englewood Cliffs, NJ: Prentice Hall.

Brodhun, C. P., III. (2008). *Virtualization: Case study of the USMC*. Breakout session PO2769 at the meeting of VMWorld 2008, Las Vegas, NV.

Chairman of the Joint Chiefs of Staff (CJCS). (2007). *Joint capabilities integration and development system* (CJCS Instruction 3170.01F). Washington, DC: Pentagon.

Citrix. (n.d.). Citrix HDX technology brings high-definition user experience to virtual desktops and application. Retrieved from http://www.citrix.com/English/ne/news/news.asp?newsID=1686302

Dell. (2005a). Dell Latitude D820. Retrieved from http://www.dell.com/downloads/global/products/latit/en/spec_latit_d820_en.pdf

Dell. (2005b). Dell Precision 490. Retrieved from http://www.dell.com/downloads/global/products/precn/en/spec_precn_490_en.pdf

Dell. (2010). Dell PowerEdge R610. Retrieved from http://www.dell.com/downloads/global/products/pedge/en/server-poweredge-r610-specs_en.pdf

Department of Defense (DoD). (2012). *Defense acquisition guidebook (DAG)*. Retrieved from http://akss.dau.mil/dag

Fehse, C. (2011). *Infrastructure suitability assessment modeling for cloud computing solutions* (Master's thesis). Monterey, CA: Naval Postgraduate School.

Goshorn, D. (2010). *The systems engineering of a network-centric distributed intelligent system of systems for robust human behavior classifications* (Unpublished doctoral dissertation). University of California, San Diego.

Goshorn, L. (2007). *Project management/engineering concepts and definitions* (Technical report). Framingham, MA: JLG Technologies.

Goshorn, R. (2012). *Distributed-GIG intelligence automation systems lab for military and homeland security*. Monterey, CA: Naval Postgraduate School.

Hewlett-Packard (HP). (2012). HP P4300 G2 7.2TB SAS Starter SAN Solution (BK716A)—Specifications and warranty. Retrieved from http://h10010.www1.hp.com/wwpc/us/en/sm/WF06b/12169-304616-3930449-3930449-3930449-4118659-4118705-4118707.html?dnr=1

ImageShack. (n.d.). Dell Precision 490 [Image]. Retrieved from http://img1.imageshack.us/img1/1266/sp490chassis1wn.jpg

InRelief. (n.d.). About InRelief.org. Retrieved from http://www.inrelief.org

Marine Corps Tactical System Support Activity (MCTSSA). (2010). MAGTF C4I capability certification test MC3T 09-01 event report version 1.0 (MCTSSA CM MC3T-0029). Camp Pendleton, CA: MCTSSA Systems Engineering & Integration Support Division.

Miller, G. (2008). *Alternative designs for a joint command, control, communications, computers and intelligence (C4I) capability certification management*. Paper presented at the meeting of the 13th International Command and Control Research and Technology Symposia (ICCRTS), Seattle, WA.

Naegle, B. (2011). Test planning and temp [Coursework, Class MN4602, Lesson 6]. Graduate School of Business and Public Policy, Naval Postgraduate School, Monterey, CA.

National Institute of Standards and Technology (NIST). (2011). *The NIST definition of cloud computing* (NIST Special Publication 800-145). Retrieved from http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

Neiger, G., Santoni, A., Leung, F., Rodgers, D., & Uhlig, R. (2006). Intel virtualization technology: Hardware support for efficient processor virtualization. *Intel Technology Journal, 10*(3), 167–177. doi:101535/itj.1003

Parmalee, R. P., Peterson, T. I., Tillman, C. C., & Hatfield, D. J. (1972). Virtual storage and virtual machine concepts. *IBM Systems Journal*, *11*(2), 99–130.

Pham, H. (2006). *Springer handbook of engineering statistics*. doi:10.1007/978-1-84628-288-1_24

Sample, J. T., & Ioup, E. Z. (2010). Forging geospatial tools. *Geospatial Intelligence Forum*. Retrieved from http://www.kmimediagroup.com/mgt-home/248-gif-2010-volume-8-issue-4-may/2889-forging-geospatial-tools.html

Smith, J. E., & Nair, R. (2005). *Virtual machines: Versatile platforms for systems and processes* (1st ed.). San Francisco, CA: Morgan Kaufmann.

Swaminathan, S., & Murthy, K. (2006). Test optimization using software virtualization. *IEEE Software*, *23*(5), 66–69.

Teradici. (n.d.). PCoIP technology. Retrieved from http://www.teradici.com/pcoip/pcoip-technology.php#PCoIP_is_a_host_rendering_protocol

TopCoder. (n.d.). About us. Retrieved from http://www.topcoder.com/aboutus/

Under Secretary of Defense for Acquisition, Technology, and Logistics (USD[AT&L]). (2008). Operation of the defense acquisition system (DoD Instruction 5000.02). Washington, DC: Author.

U.S. Army. (2011). *Army Data Center Consolidation Plan (ADCCP)*. Retrieved from http://ciog6.army.mil/LinkClick.aspx?fileticket=_knFGXDuaOI%3d&tabid=122

U.S. Navy. (2011). *Navy information management information technology efficiencies* (NAVADMIN 008/11). Retrieved from http://www.public.navy.mil/bupers-npc/reference/messages/Documents/NAVADMINS/NAV2011/NAV11008.txt

Varian, M. (1991). *VM and the VM community: Past, present, and future.* Retrieved from http://web.me.com/melinda.varian/Site/Melinda_Varians_Home_Page_files/neuvm.pdf

Yang, K. (2008). Voice of the customer—Capture and analysis. Retrieved from http://www.knovel.com/web/portal/browse/display?_EXT_KNOVEL_DISPLAY_bookid=2618&VerticalID=0

2003 - 2012 SPONSORED RESEARCH TOPICS

Acquisition Management

- Acquiring Combat Capability via Public-Private Partnerships (PPPs)
- BCA: Contractor vs. Organic Growth
- Defense Industry Consolidation
- EU-US Defense Industrial Relationships
- Knowledge Value Added (KVA) + Real Options (RO) Applied to Shipyard Planning Processes
- Managing the Services Supply Chain
- MOSA Contracting Implications
- Portfolio Optimization via KVA + RO
- Private Military Sector
- Software Requirements for OA
- Spiral Development
- Strategy for Defense Acquisition Research
- The Software, Hardware Asset Reuse Enterprise (SHARE) repository

Contract Management

- Commodity Sourcing Strategies
- Contracting Government Procurement Functions
- Contractors in 21st-century Combat Zone
- Joint Contingency Contracting
- Model for Optimizing Contingency Contracting, Planning and Execution
- Navy Contract Writing Guide
- Past Performance in Source Selection
- Strategic Contingency Contracting
- Transforming DoD Contract Closeout
- USAF Energy Savings Performance Contracts
- USAF IT Commodity Council
- USMC Contingency Contracting

Financial Management

- Acquisitions via Leasing: MPS case

- Budget Scoring
- Budgeting for Capabilities-based Planning
- Capital Budgeting for the DoD
- Energy Saving Contracts/DoD Mobile Assets
- Financing DoD Budget via PPPs
- Lessons from Private Sector Capital Budgeting for DoD Acquisition Budgeting Reform
- PPPs and Government Financing
- ROI of Information Warfare Systems
- Special Termination Liability in MDAPs
- Strategic Sourcing
- Transaction Cost Economics (TCE) to Improve Cost Estimates

Human Resources

- Indefinite Reenlistment
- Individual Augmentation
- Learning Management Systems
- Moral Conduct Waivers and First-term Attrition
- Retention
- The Navy's Selective Reenlistment Bonus (SRB) Management System
- Tuition Assistance

Logistics Management

- Analysis of LAV Depot Maintenance
- Army LOG MOD
- ASDS Product Support Analysis
- Cold-chain Logistics
- Contractors Supporting Military Operations
- Diffusion/Variability on Vendor Performance Evaluation
- Evolutionary Acquisition
- Lean Six Sigma to Reduce Costs and Improve Readiness
- Naval Aviation Maintenance and Process Improvement (2)
- Optimizing CIWS Lifecycle Support (LCS)

- Outsourcing the Pearl Harbor MK-48 Intermediate Maintenance Activity
- Pallet Management System
- PBL (4)
- Privatization-NOSL/NAWCI
- RFID (6)
- Risk Analysis for Performance-based Logistics
- R-TOC AEGIS Microwave Power Tubes
- Sense-and-Respond Logistics Network
- Strategic Sourcing

Program Management

- Building Collaborative Capacity
- Business Process Reengineering (BPR) for LCS Mission Module Acquisition
- Collaborative IT Tools Leveraging Competence
- Contractor vs. Organic Support
- Knowledge, Responsibilities and Decision Rights in MDAPs
- KVA Applied to AEGIS and SSDS
- Managing the Service Supply Chain
- Measuring Uncertainty in Earned Value
- Organizational Modeling and Simulation
- Public-Private Partnership
- Terminating Your Own Program
- Utilizing Collaborative and Three-dimensional Imaging Technology

A complete listing and electronic copies of published research are available on our website: www.acquisitionresearch.net

THIS PAGE INTENTIONALLY LEFT BLANK

ACQUISITION RESEARCH PROGRAM
GRADUATE SCHOOL OF BUSINESS & PUBLIC POLICY
NAVAL POSTGRADUATE SCHOOL
555 DYER ROAD, INGERSOLL HALL
MONTEREY, CALIFORNIA 93943

www.acquisitionresearch.net