



**THE CYBERSPACE DEVELOPMENT DOGFIGHT:
TIGHTENING THE ACQUISITIONS TURN CIRCLE**

GRADUATE RESEARCH PAPER

Matthew P. Larkowski, Major, USAF
AFIT/ICW/ENG/09-03

**DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY**

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION IS UNLIMITED

The views expressed in this graduate research project are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government.

AFIT/ICW/ENG/09-03

The Cyberspace Development Dogfight: Tightening the Acquisitions Turn Circle

GRADUATE RESEARCH PAPER

Presented to the Faculty

Department of Electrical & Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Cyber Warfare`

Matthew P. Larkowski, BS

Major, USAF

June 2009

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION IS UNLIMITED

**THE CYBERSPACE DEVELOPMENT DOGFIGHT:
TIGHTENING THE ACQUISITIONS TURN CIRCLE**

Matthew P. Larkowski, BS
Major, USAF

Approved:



Robert F. Mills, PhD (Chairman)

11 Jun 09

Date



John M. Colombi, PhD (Member)

11 Jun 09

Date

Abstract

The purpose of this research was to assess the ability for DoD software development to keep up with the increasing rate of technological change, then propose avenues for improvement. Specifically, this research attempts to answer fundamental questions based on the concerns for the 2010 Quadrennial Defense Review. In general, how do we adapt software acquisitions strategy to cope with the increasing rate of technological change?

The following conclusions were reached: (a) software projects must be scoped and scheduled for development cycles on the order of months, not years, and use open architecture, Agile Development methods, and scalable designs with modular code; (b) budgets must be stabilized for long-term integrity, with a software development working capital fund reserved for JUONS-like urgent IT needs; (c) increased use of MAJCOM- or AOC-level business centers must be encouraged and funded to produce tailored software modules that interface with larger agile programs built to accept these modules; (d) we must take advantage of ATCD and ATD efforts from research laboratories, giving MAJCOM and AOC business centers budget authority to “pull” a limited amount of ATDs, ACTDs, and JACTDs from the labs, through the appropriate System Program Office, to produce and field operational software (by default, not by exception); and (e) periodic software development working groups and conferences should be continued, but with emphasis on standardization and sharing of lessons learned between services, MAJCOMs, and AOCs.

Acknowledgements

I would like to thank my parents for teaching me how to find the answers to my own questions; for their encouragement and support while shuffling three children through the hectic tapestry of numerous school, sports, music, and computer events; and for not knowing how to program the VCR, thus requiring me to learn how to teach, which is perhaps my most valuable skill.

I would like to thank my fellow Cyber Warfare classmates of AFIT Class ICW-09J: for their unique and valuable skills and contributions throughout the last several months; for tolerating my philosophical rants, for providing a sounding board for my off-the-wall ideas, and for their great friendship and family; and for motivating and challenging me with obscure computer questions, recipe requests, and grammar semantics issues (not including this page).

I would like to thank my numerous mentors and commanders throughout my career, my professors here at AFIT, and the United States Air Force, for giving me this opportunity to take valuable time away from the operational Air Force so I could learn without distraction, for providing an outstanding educational atmosphere with unique military characteristics I could not get elsewhere, and for encouraging me to contribute my skills and military experience toward a subject area about which I am very passionate.

Matthew P. Larkowski

Table of Contents

	Page
Abstract	iv
Acknowledgements	v
Table of Contents	vi
List of Figures	viii
List of Tables	ix
I. Introduction	1
Background.....	1
Motivation	2
Purpose	3
Scope	4
Results	4
Thesis Organization.....	5
II. Paralysis by Analysis: Getting to know thyself	6
A Typical Technology Purchase for an Average Consumer	6
History of Acquisition Reform and Software Development	7
The Good.....	8
The Bad	8
JCIDS to the rescue	10
Paralysis in the current DoD JCIDS Process:.....	11
Is our acquisition process fast enough for IT acquisition and software development?.	15
III. So what do we do about it?	19
How do we change our acquisition strategy to keep pace with our peers?	19
Decrease scope and increase frequency of software development projects.....	19
Consider event-based strategy, technical maturity, and flexible budgets	20
Develop a separate cyber acquisition category	21
What are technology-centric organizations doing to remain relevant? How do they remain agile and responsive, while minimizing risk?	25
Best Vantage, Inc., and the OODA loop	25
Google, Inc., on ‘Beta’ software and Agile Practices	28

How Google estimates complexity instead of duration or budget	36
Does Google’s agile programming method scale well to large projects?	37
DISA on improving software development using agile methods.....	38
Software Development.....	38
Software Testing	39
Improving on JITC certification.....	40
Determine how to integrate with existing JCIDS process	41
Balancing between JCIDS and Agile: Learning from the JPADS ATCD success	42
A better way to integrate advanced technology: The WIDE/GRS ATD Struggle....	44
Increase the use of MAJCOM and AOC ‘business centers’	49
IV. Conclusions and Future Research.....	54
Findings	54
Future Research	55
Develop Separate DoD 5000 Instructions for Information Technology	55
Reassess required Key-Performance Parameters (KPP) for IT.....	56
Determine acquisition categories based on risk and complexity, not cost estimates.	57
Appendix A: Fundamental Concepts	60
Moore’s Law	60
Rock’s Law.....	62
Joint Capabilities Integration & Development System (JCIDS).....	63
Colonel John R. Boyd.....	66
Law of Competing Motives.....	68
Bibliography	70
Vita.....	72

List of Figures

	Page
Figure 1 - The Program Management Pendulum Swing [11]	16
Figure 2 - Acquisition Categories and Milestone Decision Authority [14]	22
Figure 3 - DODI 5000.2 ACAT I-III Programs Detail	24
Figure 4 - Best Vantage Inc., Strategic Food Marketing and OODA Loops	25
Figure 5 - Google software development 'Agile 101' [19]	31
Figure 6 - 'Stories' are requirements the customer and developer understand	33
Figure 7 - JPADS enables standoff precision airdrop using guided parachutes[26]	43
Figure 8 - Legacy Software Tool showing hard-to-read text displays [27]	45
Figure 9 - WIDE/GRS ATD showing the Multi-Mission Timeline View	46
Figure 10 – WIDE/GRS ATD showing interactive timeline [28]	47
Figure 11 - WIDE ATD reduced time and error by more than half [28]	48
Figure 12 - Crate Warehouse from the movie "Raiders of the Lost Ark"	49
Figure 13 - Software Development Turn Circles	51
Figure 14 - The Software Development Dogfight	55
Figure 15 - Moore's Law Original Graph	60
Figure 16 - Moore's Law Linear Relationship [32]	61
Figure 17 - JCIDS & Defense Acquisition System Integrated Framework Chart [5]	64
Figure 18 - DoD Architecture, Requirements & Acquisition Process [34]	65
Figure 19 – JCIDS and the Defense Acquisition System [34]	65
Figure 20 - The One and Only OODA Loop [34]	67
Figure 21 - Law of Competing Motives, Example Color Wheel	69

List of Tables

	Page
Table 1 - Acquisition Reform Initiatives [2].....	7
Table 2 - Redesigned Acquisition Categories for Information Technology.....	23
Table 3 - Important Questions Gleaned From Best Vantage Inc.....	28
Table 4 - The Google point system for estimating complexity [19].....	36

THE CYBERSPACE DEVELOPMENT DOGFIGHT: TIGHTENING THE ACQUISITIONS TURN CIRCLE

I. Introduction

“DoD has been able to develop and acquire the best weapons and support systems in the world. DoD and contractor personnel accomplished this feat not because of the system, but in spite of it. And they did so at a price... the nation can no longer afford to pay.”

— Former Secretary of Defense William J. Perry

Background

Moore’s Law describes the miniaturization trend for doubling the density of microchip transistor components approximately every 24 months. This basic tenet of Moore’s Law is often generalized to describe the increasing pace of technology change. Technology has, is, and will continue to double on the order of months, not years. In contrast, Rock’s Law is often conversely associated with Moore’s Law in that the cost of producing the future’s increased density and technology also increases exponentially. Thus the ages-old balancing act between resources and capability is set in motion. See Appendix A: Fundamental Concepts, for further discussion on these areas.

Given limited resources and a finite economy, equilibrium is reached at some point and Rock’s Law dominates the otherwise exponential growth of Moore’s Law. I believe, however, that we are more limited by how we are *using* our resources, not just the *amount* of our resources. We are also limited by a dwindling understanding of how our people fit into the process; after all, a process doesn’t run itself! Regardless, the rate of technology change is only getting faster and resources aren’t getting more generous... yet our mechanism for spending money, allocating resources, and adapting to changing technology requirements is either unchanged or taking even longer to complete for each

development cycle. The development cycle can be described using the Observe, Orient, Decide, Act (OODA) model. Colonel John R. Boyd, considered to be the father of the ‘OODA loop’ for process analysis, would be the first to say that we should not simplistically limit our focus only to the ‘OODA’ parts of the process. “People, ideas, hardware — in that order”, he would often preach, and “machines don’t fight wars, people do, and they use their minds”, were commonly heard from Colonel Boyd [1].

As the relative length of time required to complete any project increases, we must attempt to predict future capability requirements earlier and earlier. Thus, the precision and accuracy of these requirements fall off rapidly the more we try to peer into the crystal ball and see the future. I sense that the theoretical value and efficiency attained by our current acquisitions process is often a false one when *all* the costs are all tallied. These costs are due to the lack of timely technological adaptations, increasingly unstable process transitions, and outright regression in providing timely, unique high-priority capabilities required by operators while instead preserving the lower priority integrated and over-centralized capabilities desired by managers. It is important to note that this is not necessarily the fault of the acquisition process itself. The process fulfills valid and important ambitions; nonetheless its current manifestation is inadequate and incomplete for addressing particularly urgent or unique capability requirements.

Motivation

This research supports the Quadrennial Defense Review (QDR) request for an acquisition strategy in an accelerating world. Based on the QDR 2010 topics, I posed the following questions: How do we change our acquisition strategy to keep pace with our

peers? How do we bridge the gap between specific mission-focused systems, whose requirements are well-defined and scoped, and large-scale enterprise systems, which tend to be one-size fits all, yet often don't scale well? What are technology-centric organizations in the business sector doing to remain relevant? How do they remain agile and responsive, while minimizing risk?

Purpose

The purpose of this research project is to evaluate our current acquisition capabilities for software development then present methods for mitigating challenges and improving our strategy. To do this, this study will look at the history of acquisition reform in the broad sense, before focusing on software development process improvements. It is my feeling that software development is the metaphorical 'canary in the coal mine' in terms of highlighting acquisitions process breakdowns; software development can often be the first to suffer from any weaknesses or mismanagement. This study will look at changes we can make to improve software development, researching pros and cons of a smaller, faster acquisitions cycle by using medium-sized business centers, as opposed to only using large enterprise architecture development efforts. Better methods for transition and integration of Advanced Concept Technology Demonstrations (ACTD), Joint ACTDs (JACTD), and Advanced Technology Demonstrations (ATD) into the Joint Capabilities Integration & Development System (JCIDS) process will also be explored.

Scope

This study is limited to a review of existing software development strategies, including an overview of relevant theories as needed, and an exploration of new approaches for improving software development speed and quality. I am writing this with both operations and acquisitions readers in mind, so I've made the attempt to minimize the use of acronyms or jargon, while introducing a modicum of key terms everyone should know. An in-depth examination of the various strategies, engineering studies, and cost-benefit analyses are also beyond the scope of this study.

Results

The following conclusions were reached: (a) software projects must be scoped and scheduled for development cycles on the order of months, not years, and use open architecture, Agile Development methods, and scalable designs with modular code; (b) budgets must be stabilized for long-term integrity, with a software development working capital fund reserved for JUONS-like urgent IT needs; (c) increased use of MAJCOM- or AOC-level business centers must be encouraged and funded to produce tailored software modules that interface with larger agile programs built to accept these modules; (d) we must take advantage of ATCD and ATD efforts from research laboratories, giving MAJCOM and AOC business centers budget authority to “pull” a limited amount of ATDs, ACTDs, and JACTDs from the labs, through the appropriate System Program Office, to produce and field operational software (by default, not by exception); and (e) periodic software development working groups and conferences

should be continued, but with emphasis on standardization and sharing of lessons learned between services, MAJCOMs, and AOCs.

Thesis Organization

This chapter presents the motivation, purpose, scope and results for this research, and concludes with the document's organization. Chapter II explores the history of acquisition reform, including lessons learned from selected acquisition projects and existing studies. Chapter III aggregates and discusses methods for improved software development — including pros and cons of each concept. Chapter IV presents the conclusions and ideas for future research.

II. Paralysis by Analysis: Getting to know thyself

"If you know the enemy and know yourself, you need not fear the result of a hundred battles. If you know yourself but not the enemy, for every victory gained you will also suffer a defeat. If you know neither the enemy nor yourself, you will succumb in every battle."

—Sun Tzu

A Typical Technology Purchase for an Average Consumer

Nearly everyone reading this is probably familiar with the basic challenges of software acquisition. Even the most tepid computer user has likely experienced the dreaded software upgrade, purchased a software program that did not do what we wanted, or used software that was unreliable and untrustworthy. I submit that software development and acquisition within the DoD is not altogether different from the typical experience of the average person, only the scope and scale of the software is much larger.

We have probably purchased a technology item, such as software, a computer, television, or stereo. In doing this, we've gone through the classic struggle of trying to decide which features were *needed* compared to which features were merely *wanted*. We considered whether or not to buy the economy model with fewer features and perhaps lesser quality, or maybe decided to go for the top of the line, hoping the technology features and quality will mean it will last several years. Once our choice was made, we've perhaps been disappointed when we discovered that a newer model is already under development, and the model purchased will no longer be the leading edge of the technology wave. If we were already aware of this and made a conscious decision as to which 'wave' to try for — the sharp, 'bleeding' edge, expensive first wave, the economy of last year's wave, or the 'bargain basement' wave of technology at risk of becoming

obsolete. Ultimately, we must make a decision sometime, or the waves of technology pass will us by. This experience is paralysis by analysis.

History of Acquisition Reform and Software Development

“What would you do if you were stuck in one place and every day was exactly the same, and nothing that you did mattered?”
—character Phil Connors from the film “Groundhog Day”

The DoD also experiences a similar paralysis and the United States has made many efforts to improve defense acquisitions such that we prevent the waste of valuable resources, while providing timely and superior equipment to our fighting forces. Table 1 shows a brief history of recent US acquisition reform efforts, prior to the current JCIDS process. It is important to understand that there have been many reform efforts, by many intelligent people, and these reforms have made positive changes, but there’s still room for improvements, in order to speed up software acquisitions:

Table 1 - Acquisition Reform Initiatives [2]

1961	McNamara Initiative
1970	Fitzhugh Commission
1972	Commission on Government Procurement
1976	OMB Circular 4-109
1978	Defense Science Board Acquisition Cycle Study
1979	Defense Resources Management Study
1981	Defense Acquisition Improvement Program
1983	Grace Commission
1986	Packard Commission
1986	Goldwater Nichols
1989	Defense Management Review
1990	Quadrennial Defense Reviews

The Good

In one of many Rand Corporation studies of DoD Acquisition Reform (AR) efforts, Dr. Ken Oscar, Acting Assistant Secretary of the Army for Acquisition, Logistics, and Technology, described the positive aspect of AR that laid the foundation for JCIDS in an article titled “*Reexamining Military Acquisition Reform: Are We There Yet?*”:

Overall, Oscar characterized the AR movement in the 1990s as having been energized by Secretary Perry’s “Mandate for Change” speech in 1994, and as having achieved three very important legislative accomplishments over the period: the Defense Acquisition Workforce Improvement Act (DAWIA) of 1990, the Federal Acquisition Streamlining Act of 1994, and the Federal Acquisition Reform Act (FARA) of 1996. In his view, those legislative actions (along with the AR efforts to internally reform the acquisition process — e.g., the rewrite of the 5000 Series2) have helped to improve the education and skills of the acquisition workforce, remove unnecessary laws, and reduce regulations — thereby contributing to an environment that allows for more creative approaches to acquisition than were previously possible.[3]

The reforms of the 1990’s reduced the “stovepipe” effect where organizations tended to limit communication vertically up and down the chain of command. Lateral collaboration did not happen naturally. Dr. Oscar also highlighted the birth of “evolutionary acquisition”, which divided large systems into smaller chunks — increasing delivery flexibility and decreasing scheduling risk [3].

The Bad

While these reforms reduced confusion and “red tape”, paving the way for Program Managers (PM) to have more creative control over the acquisition process, there was still more to be done. “AR gives PMs authority to take risks but not the resources...” and “We reformed the acquisition process but not the financial process that supports it...” were among the complaints fielded from PMs. More so, I’ve often felt that many of the

changes were more “lip service” than anything else. The RAND study’s findings echoed these sentiments:

Several of the participants provided frank assessments of the changes — or lack thereof — brought about under AR. A senior deputy PEO commented that “AR has been good at cranking out policies, but hasn’t made anything faster, better, or cheaper,” a remark with which many others participating in the group interview concurred. One participant noted, “There is no such thing as acquisition reform. We’ve changed the way PMs deal with contractors, but nothing else has changed.” [3]

In another key finding, there was a general observation that not all organizations were playing along with AR efforts. It was widely felt that external organization resistance can still dominate AR:

AR will remain suboptimized until they reform the financial, logistics, test, engineering, contracting, and legal communities. These communities can unilaterally kill any AR program, since they have full veto authority in most cases, while not being held accountable for their decisions. [3]

In criticism of the testing portions of AR, several PMs felt this same refusal to change:

The testing community is still in the old ways of doing business . . . The test community is still living 30 years in the past. . . . The test community is still focused on their reporting requirements rather than testing to fix. [3]

These observations all seem to involve the concepts of authority, responsibility, and accountability. In my opinion, these three things must be distributed with equal measure within an organization. Whenever one organization is responsible for performing tasks, but is not authorized to control its resources, there is a breakdown in the overall system of accountability. Quite often this lack of authority is felt in terms of funding control: “Many PMs felt constrained due to ‘color-of-money’ restrictions on how they could spend the moneys within their budgets.” [3] It seems that it is not just the lack of money or resources, but often the control over the same that is inadequate.

JCIDS to the rescue

Acquisition reforms have led to the current JCIDS process, and we've evolved JCIDS as a capabilities-based method for deciding what to buy and what features we need and want — feeding the overall acquisitions process. Quite literally, the needs are called threshold requirements and the wants are called objective requirements.

Previous to JCIDS, acquisition was often service focused and platform-centric [4]. For example, the Army might be inclined to build a newer tank, improving upon the current design. This might have been done without considering whether or not a new tank was really needed. Indeed, the capabilities of another service may have already better addressed the threat for which the new tank would be designed, or a simple change doctrine could mitigate the issues at hand without requiring a new tank. Will the new tank fit onto our cargo planes or ships? Either way, precious resources would be wasted on a new tank that is not needed, won't work with our other equipment, or duplicates a capability that already exists.

In general, JCIDS seeks to preserve resources with a process that begins with the identification of desired capabilities and an organization's current ability to address that capability (known as a gap analysis). Once a gap in capability is identified, JCIDS seeks to determine whether or not we need to make something to fix it — a materiel solution. Otherwise, a non-materiel solution, such as new training or processes, may fill the gap. These new requirements are mixed with existing requirements, and then refined, prioritized, and reviewed by the Joint Requirements Oversight Council (JROC).

JCIDS continues with concept refinement. Next, it proceeds to a Functional Solutions Analysis (FSA), which determines whether or not a material solution is

required vs. a non-material or training solution. Ultimately an Analysis of Alternatives (AOA) is done for choosing a specific material solution to procure. The process of building a new technology proceeds through three milestones (A, B, and C), where key documents are produced and reviewed as to the technology is designed, prototypes are built and tested, and finally the technology is fielded under full-rate production. The entire life-cycle of the technology is considered, from initial operational capability to the eventual retirement of the technology.

Other processes operate along with JCIDS, such as the Planning, Programming, Budgeting, and Execution (PPBE) process for continually reassessing the DoD's resource allocation. Any new JCIDS program must be prioritized within the PPBE for consideration. Software must also seek Joint Interoperability Test Center (JITC) certification in order to verify that it meets the Net-Ready Key Performance Parameters (NR-KPP), that qualify the software for full-rate production. This is the last of several certifications before the J-6 issues the Interoperability System Validation letter that allows the software to be used operationally [5]. For more information on the JCIDS process, see Appendix A: Fundamental Concepts.

Paralysis in the current DoD JCIDS Process:

“You will never understand bureaucracies until you understand that for bureaucrats procedure is everything and outcomes are nothing.”

—Thomas Sowell, The Hoover Institution

The paralysis within DoD acquisitions manifests itself in a slightly different way than it does for the typical consumer purchase, described earlier, but the analogy holds true. To understand this, we must look at what happens in the JCIDS efforts that begin

the acquisition process. In JCIDS, once a capability gap is identified, and possible solutions developed, an analysis is done to determine how “joint” the new proposal is and how much oversight it deserves. JCIDS is designed to save resources, avoid duplication, and ensure analyses consider all services jointly; in JCIDS everything is “born joint.” A new proposal may have impacts that affect all the services, and is categorized as “Joint Integration”, or the proposal may only affect a single service, and is categorized as “Independent”. Furthermore, the JROC has oversight for and reviews all large acquisition category (ACAT) programs, including any other programs for which they have an interest [4].

This categorization and oversight is all well and good, except that it may have drastic impacts on the size and scope of the subsequent acquisition effort. For large programs, it ensures risk is minimized, resources are efficiently spent, and ‘no stone is left unturned’. For urgent war-fighter needs, however, JCIDS would add excess burden and discourage experimentation. Thus the Joint Urgent Operational Needs (JUONs) process was developed in July 2005, to provide a mechanism for rapidly validating and resourcing these needs [6].

JUONs, according to CJCSI 3170.01G, 1 March 2009, defines urgent operational needs as those that:

(1) fall outside of the established Service processes; and (2) most importantly, if not addressed immediately, will seriously endanger personnel or pose a major threat to ongoing operations. They should not involve the development of a new technology or capability; however, the acceleration of a Joint Capability Technology Demonstration or minor modification of an existing system to adapt to a new or similar mission....

These needs, according to the older 15 July 2005 version of CJCSI 3470.01, "... can be considered as life- or combat-mission-threatening needs, based on unforeseen military requirements that must be resolved in days, weeks or months." [7]. Although not always 'life or death', JUONs is a useful non-JCIDS method which could benefit IT acquisitions. The inclusion of ACTDs within JUONs also provides an avenue for fast IT development.

The "born joint" aspect of JCIDS, JUONs, and JACTDS shifts the focus toward resource efficiency rather than time-to-field efficiency or mission-effectiveness efficiency, although the JUONs process attempts to mitigate this. This is doing exactly what 'Joint-ness' was designed to do, but perhaps we need to reconsider the ramifications of this new focus. A simple, urgently needed IT software tool may initially be paralyzed by JCIDS joint analysis and, unless it qualifies as a JUON, the joint analysis time can be excessive. Furthermore, assuming a joint service need for development of a new tool, all services must coordinate together as they march through the acquisition process steps.

Joint interoperability is significantly improved through joint integration efforts, but at a cost. In this synchronized fashion, integration ensures that no one is left out of the loop and that all requirements are considered, but the drawback of integration is that we now must run at the speed of the slowest runner. We cannot move through successive OODA-loop cycles until all interdependent pieces are ready. One service may have a unique requirement for a tool capability that another service does not have, yet all services must wait until the analysis, design, and testing for that requirement are integrated into the whole process.

In 2005, Vice Chairman of the Joint Chiefs of Staff, Admiral Edmund Giambastiani gave written testimony on JCIDS to the House Armed Services committee.

Vice Admiral Evan Chanik, chief of the Joint Staff J-8, Force Structure Directorate, commented on Admiral Giambastiani's testimony, suggesting future improvements to JCIDS, according to this excerpt from this February, 2006, *Inside the Air Force* report, "Joint Staff officials will 'tweak' JCIDS to better address urgent needs" [8]:

...Chanik's comments come several months after [Giambastiani], in written response to advance questions from the Senate Armed Services Committee during his confirmation process last summer, first raised the issue of tweaking JCIDS to address urgent requirements. In his responses, Giambastiani noted that the JCIDS process "is designed to impact mid- to far-term capabilities and funding (three years and beyond)" but has "less flexibility to quickly respond to emerging requirements...in the near-term budget years (one to two years)."

Ultimately, limited acquisition authority and other ad hoc measures Congress has enacted to address the problem should give way to more permanent solutions, Giambastiani wrote. "In the long term, the JCIDS process needs to change to fall more in line with the demands and pace of today's operations," Giambastiani noted.

Further criticisms of JCIDS, by current and former military officials, were published in *Inside the Navy*, and reproduced in the same *Inside the Air Force* article. Retired Marine Corps Lt. Gen. Paul Van Riper, in an email to Chairman of the Joint of Staff General Peter Pace, and other senior leadership, "... slammed JCIDS for being "overly bureaucratic and procedurally focused." Van Riper continued:

"My greatest concern is that as these concepts migrate into the curricula of professional military schools they will undermine a coherent body of doctrine creating confusion within the officer corps," Van Riper continued. "In fact, I have begun to see signs of just that!" In a response sent several days following Van Riper's e-mail, [USMC LtGen James] Mattis -- who is now in the post Van Riper held when he retired in 1977 -- agreed wholeheartedly. [8]

Many of these criticisms have been taken to heart and on 1 May, 2007, CJCSI 3170.01F was released in order to continue to "...refine the JCIDS process and the information they

require to ensure they are making effective, appropriate decisions in a timely manner. This update to the policies and processes continues that evolution of JCIDS to ensure our ability to continue to meet the needs of the joint warfighter.” As of this writing, CJCSI3170.01G, 1 March 2009, is the current JCIDS policy release.

Is our acquisition process fast enough for IT acquisition and software development?

“The global situation is so unpredictable that the Acquisition system must be flexible and agile.”

— Colleen A. Preston, Deputy Under Secretary of Defense for Acquisition Reform [9]

Since the acquisition criticisms of 2005 through 2007 were voiced by high-level military leaders, it still needs more ‘tweaks’. In a *DefenseNews* online article titled “DoD IT Procurement Too Slow: Cartwright”, published 4 Mar 2009, Joint Chiefs Vice Chairman General James Cartwright’s comments at the eighth annual Naval IT Day conference were recorded. The article describes the state of IT procurement as follows:

The current method of procurement for information technology is so slow that by the time software systems and the like are purchased, they're out of date, Joint Chiefs Vice Chairman Gen. James Cartwright said March 4 at a conference for the IT industry.

"It takes longer to declare a new [program] start than the lifecycle of the software package," Cartwright told an audience of IT industry and Armed Services representatives...[10]

With the advent of JCIDS, the AR pendulum has swung such that we are now excessively risk averse, with the tendency to not accept failure as a necessary part of development. General Cartwright highlighted this concept, as the article continues:

Aiming for a "perfect" IT solution is often the problem, Cartwright said. "We have this mindset that somehow whatever we field has to be perfect, so we'll spend a life of an application's utility testing it to make sure it's invulnerable and makes no mistakes," Cartwright said. "Looking for the

perfect solution is almost always a recipe for irrelevance, and we've proved that over and over and over again." [10]

The challenge facing the Defense Department with IT procurement isn't that technology is too advanced, it's that the culture for procurement isn't working and needs to change, Cartwright said. [10]

This pendulum swing effect is shown in Figure 1, in a graph excerpt from a slide presentation by Colonel Pete Rustan, of the National Reconnaissance Office. Note that as of 2005, DoD is in a state of risk aversion which drives higher costs. Whereas in 1957 the state of total risk acceptance, although cheaper, was also undesirable in that there was a higher 'potential energy' for failure. Furthermore, I submit that the fact that our process is swinging and unstable at all is a major factor. We must seek stable equilibrium, balancing between cost and risk. This equilibrium is likely different for large acquisitions, such as an aircraft carrier, than for smaller IT software development projects — our system must accommodate the range between these extremes.

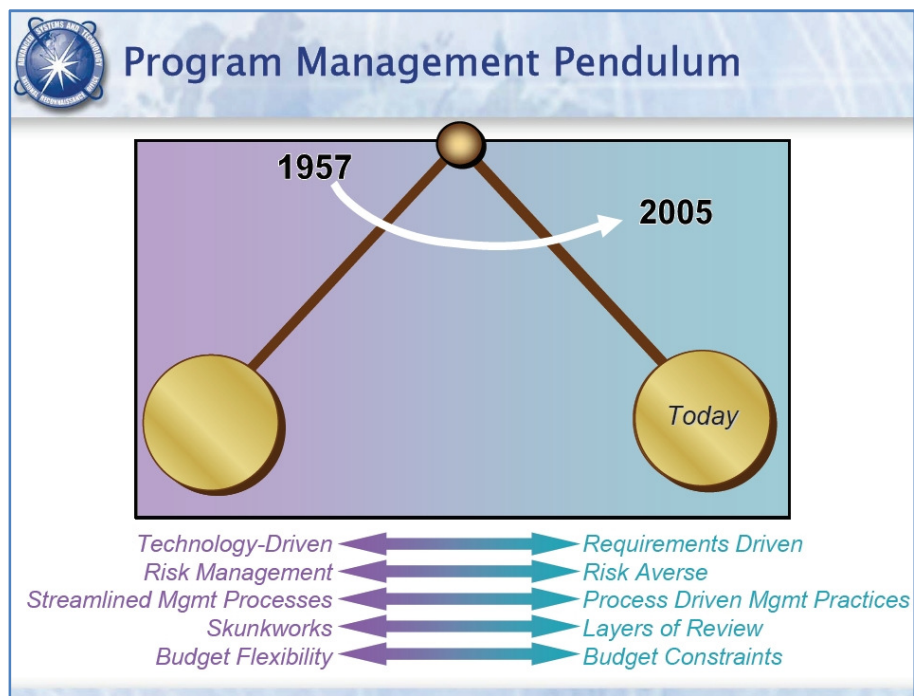


Figure 1 - The Program Management Pendulum Swing [11]

General Cartwright then described that the JCIDS procurement method may be acceptable for building an aircraft carrier, but for IT procurement, the software we develop is "...irrelevant before they even get to milestone A" [10]. Cartwright suggests that although we should proceed with IT procurement improvement, the "DoD may not buy the most advanced solutions – on a large scale – that industry has to sell" [10]. In other words, like the typical consumer purchase example, we do not need to be on the leading edge for IT procurement. I personally disagree with this in part, since IT security risks are continually discovered and exploited by hackers. If we purposely decide not to stay ahead of this exploitation wave, we leave our software systems exposed to infiltration and manipulation. I concede that many 'wants' are truly not 'needs', but security is usually in the 'needs' category. Mr. Terry Halvorsen, deputy commander for Naval Network Warfare Command, confirms this in the same DefenseNews article:

... we're not always going to use all of the most leading edge technology in large areas. We may use it to build a specific mission or to meet a specific requirement in small ways and then ... we'll adapt it. Those of you on that leading edge need to sometimes step back a little and think about the tactical, operational problems that we have, particularly with that balance between security and getting missions [done][10].

With these changes in mind, I view the lack of AR as analogous to monetary inflation. Holding onto \$1000 cash for years and years, the cash will lose value relative to the economy, due to inflation; by doing nothing we lose money. Similarly, the pace of change has the same effect if our procurement process is static. If we do not continually invest in and evolve our procurement process through AR, in time we will be unable to keep up. In deference to the earlier criticisms that the financial communities have not mirrored the AR movement, since 1986 we've had a two-year Program Objective

Memorandum (POM) process for budgeting DoD requests from the federal budget.

Although changes can be made annually, this financial process is often the ‘long pole’ in the software development tent, introducing a large amount of lag into the system. Due to future ‘technology inflation’, this 2-year POM cycle will become relatively shorter and shorter, in effect. Again, by doing nothing, we lose money.

Robert Carey, the Navy's chief information officer, commented on the acquisition system challenges for IT, and the inflexibility of our budgeting system:

Things are moving really fast. The acquisition system, and more importantly the budgeting system, move at a different pace. Today, if most of you come in and say, 'I've got this great idea. I want to give it to you,' all of our money has been displaced. There is little agility in that system. When you go spend it on something else, an opportunity, you generally have to break something else. That being said, we have the opportunity within the (Federal Acquisition Regulation) [and other areas to adopt new technology]. There's a lot of agility that affords us. We need to look at ... how do we move faster while maintaining the right levels of control on the system.[10]

Finally, and most recently, Defense Secretary Robert Gates identified AR as an emphasis area for the 2009 QDR. The DoD Buzz online Defense and Acquisition journal describes Secretary Gates’ desires for better acquisition business processes as follows:

Gates has repeatedly called for acquisition reform to get control over costs and the need to make “tough choices” on weapons and spending. “We must ensure that requirements are reasonable and technology is adequately mature” before programs go forward, he said. His moves to add acquisition personnel and reduce the role of contractors in procurement is likely just a first step. He wants to reform the acquisition system so it can speed weapons to the battlefield and lessen the need for ad-hoc procurement arrangements, such as JIEDDO, that were created to try and avoid the often Byzantine weapons buying process. On Gates’ comments on the need for acquisition reform, CSIS’ Anthony Cordesman had this to say: “The secretary advanced some key issues and priorities for reform. Unfortunately, exactly the same comments could have been made during the Eisenhower administration – and were.”[12]

III. So what do we do about it?

How do we change our acquisition strategy to keep pace with our peers?

“Be quick but don’t hurry.”
—John Wooden

Decrease scope and increase frequency of software development projects

In short, we need a tighter OODA loop for cyberspace IT software development. We must accept the risk of what appears to be resource inefficiency and increased cost for the sake of development speed and increased experimentation. I believe that the short-term excess cost will be outweighed by the long-term savings obtained by completing software development projects that better meet user requirements, in a more timely fashion.

Increasing the frequency and decreasing the scope for each spiral or increment will give software developers a smaller and more recent set of requirements to work with. What is often termed ‘requirements creep’ is actually, in many cases, the nature of technology change. Unfortunately, we are usually not able to obtain the adversary’s schedule in advance; therefore, we must concede that much of our strategy should allow for agile, flexible reaction to hostile action. If our adversary’s software technology changes weekly, but our software development can only respond years later, we will likely lose the software battle — this reality should not be considered ‘requirements creep’. Taking smaller bites, together with flexible budget *authority* for the PM (emphasis on authority), will help ensure that we have a better response to all the bumps in the software road. Perhaps we can even evolve to a continuous integration and

development strategy for software builds, using the patching concept between major releases.

Consider event-based strategy, technical maturity, and flexible budgets

The current acquisitions process still depends highly upon accurate estimates for costs and depends upon highly refined requirements. These estimates are refined too slowly in the JCIDS process to work well for a fast-paced IT project. Failing to meet these imprecise, inaccurate, and optimistic estimates may doom a program to failure. Thomas Christie, former DoD Director of Operation Test and Evaluation (OT&E), in his *Military.com* forum article titled “Questioning Acquisition Reform”, states that “undertaking major developments without understanding key technical issues is the root cause of major cost and schedule problems.”

One solution for this estimation dilemma has been to only allow the use of mature technology, as assessed using the Technology Readiness Level measurement system. Often, though, we cannot wait for this maturity to happen as we compete with an unconstrained adversary. Indeed, without a program budget to start things off, some concepts and technologies may never get the incentive or ability to be matured. Christie further highlighted this need for technical maturity:

Of critical importance is demonstrating the technical maturity of the technologies embedded in a new system development prior to proceeding into accelerated development. Sufficient up-front funding and time for effective system and sub-system prototype demonstration and testing should be programmed to ensure an informed decision concerning the technical risk entailed in proceeding.[13]

Christie concludes that rigid time-based scheduling strategies create pressures to shortcut the proper steps and, in one case, “drove decisions to severely reduce

development testing to save dollars and stay on schedule.” Dollars, time, and risk often compete in a sort of ‘Law of Competing Motives’ fashion (see Appendix A: Fundamental Concepts). To alleviate this shortcutting pressure, Christie proposes using an event-based strategy instead of a schedule-based strategy:

The decision authority should impose an event-based (as opposed to a schedule-based) strategy on the program to include meaningful and realistic "exit criteria" for each stage of development and production. Only if these criteria are demonstrated and satisfied should the program proceed to its next stage. [13]

Mr. Christie also underscored his sentiments that the initial vector of a program is most important. Root causes such as this, not excessive oversight, lead to problems:

Most knowledgeable observers of and participants in this process have already identified most problems and proposed solutions for them. Pointing fingers at oversight agencies in the executive and legislative branches for the lengthy times from program starts to deliveries to the troops in the field does not address the root causes for those schedule slips. Neither does the cyclical invention of acquisition strategies with catchy buzzword titles come to grips with those root causes...

...more informed management attention and discipline at the front end of the process should go a long way toward solving many of the problems plaguing defense acquisition. Nothing is new here. Time and again, major defense management reviews have reached the same conclusions. It is high time that decision-makers take seriously these findings, most of which are embedded in existing directives and instructions that govern the acquisition process, and make them an integral part of their program-review and decision process. [13]

Develop a separate cyber acquisition category

“Nothing is so unequal as the equal treatment of unequals”.
—P. Larkowski, father of the author, adapted from a popular quote from George Orwell’s book “Animal Farm”

After observing the pendulum swing effect described by Pete Rustan, I came to the conclusion that what is an ‘optimal’ cost-risk balance for a large acquisition program

may be different than what is ‘optimal’ for smaller, more agile, IT software development projects. I thought of the following slide, Figure 2, from the March 2007 Air Force Acquisition Action Officer 101 briefing:



Acquisition Categories and Milestone Decision Authority (MDA)

		Non-Space Programs	Space Programs
Classification	Sub-Designation	MDA	MDA
MDAP	ACAT ID	USD(AT&L)	
	ACAT IC	SAF/AQ as delegated by USD(AT&L) and SecAF	USecAF as delegated by USD(AT&L) and SecAF*
MAIS	ACAT IAM	ASD(NII) / DoD CIO	
	ACAT IAC	SAF/AQ as delegated by DoD CIO and SecAF	
Major System	ACAT II	SAF/AQ (or appropriate AFPEO if delegated)	AFPEO/SP
Non-major System	ACAT III	Appropriate AFPEO or Deputy AFPEO	AFPEO/SP or Deputy
Technology and other projects	ATDs, ACTDs, joint warfighting experiments, concept refinement	Varies with category	Varies with category

**Authority presently retained by USD(AT&L)*

Figure 2 - Acquisition Categories and Milestone Decision Authority [14]

What I noticed in the slide is that the Space Programs, due to the unique aspects of the space environment, are treated differently from all other acquisition efforts. Currently, non-space IT software development falls under ACAT I or ACAT III. There is no ACAT II for IT system software and Major Defense Acquisition Programs (MDAP) do not solely include IT. Thus only Major Automated Information Systems (MAIS),

Non-major ACAT III systems, and “Technology and other projects” categories include IT software development programs.

I propose that we simplify this by adding a new column for ‘pure IT’ programs, (see Table 2). Note 3, of the DoDI 5000.2, from December 8, 2008, helps discern what is or is not ‘pure IT’, (see Figure 3). In general, Automated Information Systems are primarily IT, where the software *itself* is a service or capability, rather than an indirect or supporting entity. Supporting software should develop according to its parent program.

Table 2 - Redesigned Acquisition Categories for Information Technology

		Information Technology Programs	Non-Space Programs	Space Programs
Classification	Sub-Designation	MDA	MDA	MDA
MDAP	ACAT I(D)	(IAM) ASD/DoD CIO	USD(AT&L)	USecAF and SecAF
	ACAT I(C)	(IAC) SAF/AQ and SecAF	SAF/AQ and SecAF	
Major System	ACAT II	TBD	SAF/AQ or AFPEO	AFPEO/SP
Non-Major System	ACAT III	AFPEO or DAFPEO	AFPEO or DAFPEO	AFPEO/SP or DAFPEO/SP
Technology and other projects	ATD/ACTD, Joint War-fighting Experiments, Concept Refinements	Varies with category	Varies with category	Varies with Category
Relative Cycle		Short-term	Medium-Term	Long-term

Note: MAIS is removed and incorporated within MDAP. Also, it is possible that we should rename “A”, “B”, “C”, or “D” suffixes for ACAT I programs.

In short, we must not treat software development like an aircraft carrier — which is what MAIS/MDAP programs tend to do. Separating IT in this way makes it easier to assess the cost-risk pendulum balance differently for the short-term natured ‘pure IT’ projects. Furthermore, software development that is integral to non-IT programs should not fall under this new IT column. Also note that acquisition categories are largely cost-based and perhaps we should change this focus, (see the *Future Research* section).

Table 1. Description and Decision Authority for ACAT I – III Programs.

Acquisition Category	Reason for ACAT Designation	Decision Authority
ACAT I	- MDAP (section 2430 of Reference (k)) - Dollar value: estimated by the USD(AT&L) to require an eventual total expenditure for research, development, test and evaluation (RDT&E) of more than \$365 million in fiscal year (FY) 2000 constant dollars or, for procurement, of more than \$2.190 billion in FY 2000 constant dollars - MDA designation - MDA designation as special interest	ACAT ID: USD(AT&L) ACAT IC: Head of the DoD Component or, if delegated, the CAE (not further delegable)
ACAT IA ^{1,2}	- MAIS (Chapter 144A of Reference (k)): A DoD acquisition program for an Automated Information System³ (either as a product or a service) that is either: - Designated by the MDA as a MAIS; or - Estimated to exceed: \$32 million in FY 2000 constant dollars for all expenditures, for all increments, regardless of the appropriation or fund source, directly related to the AIS definition, design, development, and deployment, and incurred in any single fiscal year; or \$126 million in FY 2000 constant dollars for all expenditures, for all increments, regardless of the appropriation or fund source, directly related to the AIS definition, design, development, and deployment, and incurred from the beginning of the Materiel Solution Analysis Phase through deployment at all sites; or \$378 million in FY 2000 constant dollars for all expenditures, for all increments, regardless of the appropriation or fund source, directly related to the AIS definition, design, development, deployment, operations and maintenance, and incurred from the beginning of the Materiel Solution Analysis Phase through sustainment for the estimated useful life of the system. - MDA designation as special interest	ACAT IAM: USD(AT&L) or designee ACAT IAC: Head of the DoD Component or, if delegated, the CAE (not further delegable)
ACAT II	- Does not meet criteria for ACAT I - Major system - Dollar value: estimated by the DoD Component Head to require an eventual total expenditure for RDT&E of more than \$140 million in FY 2000 constant dollars, or for procurement of more than \$660 million in FY 2000 constant dollars (section 2302d of Reference (k)) - MDA designation⁴ (paragraph (5) of section 2302 of Reference (k))	CAE or the individual designated by the CAE ⁴
ACAT III	- Does not meet criteria for ACAT II or above - AIS that is not a MAIS	Designated by the CAE ⁴
1. In some cases, an ACAT IA program, as defined above, also meets the definition of an MDAP. The USD(AT&L) shall be the MDA for such programs unless delegated to a DoD Component. The statutory requirements that apply to MDAPs and MAIS shall apply to such programs. 2. The MDA (either the USD(AT&L) or, if delegated, the ASD(NII)/DoD CIO or another designee) shall designate MAIS programs as ACAT IAM or ACAT IAC. MAIS programs shall not be designated as ACAT II. 3. Automated Information System: A system of computer hardware, computer software, data or telecommunications that performs functions such as collecting, processing, storing, transmitting, and displaying information. Excluded are computer resources, both hardware and software, that are: - an integral part of a weapon or weapon system; - used for highly sensitive classified programs (as determined by the Secretary of Defense); - used for other highly sensitive information technology programs (as determined by the ASD(NII)/DoD CIO); or - determined by the USD(AT&L) or designee to be better overseen as a non-AIS program (e.g., a program with a low ratio of RDT&E funding to total program acquisition costs or that requires significant hardware development). 4. As delegated by the Secretary of Defense or Secretary of the Military Department.		

Figure 3 - DODI 5000.2 ACAT I-III Programs Detail

What are technology-centric organizations doing to remain relevant? How do they remain agile and responsive, while minimizing risk?

"A good hockey player plays where the puck is, a great hockey player plays where the puck is going to be."
—Wayne Gretzky

Best Vantage, Inc., and the OODA loop

During my research of technology-centric businesses, believe it or not, a food marketing company caught my eye. Best Vantage, Inc. followed the military's example for their use of the OODA loop. Best Vantage has developed some useful tenets that, ironically, the military will find useful as an example to follow for software development. If you look at the following figure, (Figure 4), you may see why this strategic food marketing company got my attention:



Figure 4 - Best Vantage Inc., Strategic Food Marketing and OODA Loops

So what is Best Vantage doing that is so useful for software development?

Within the *Best Vantage Inc. Newsletter*, from January 2009, author Daniel Best wrote an excellent article titled “Marketing inside the loop”. In this article, he attributes thanks for “...the privilege of flying combat aircraft competitively under the instruction of U.S. Air Force pilots, to whom he remains eternally grateful for the life lessons learned.” Mr. Best uses full military parlance, citing Col. John R. Boyd and Wayne Gretzky to describe how the food marketing business should think of marketing as a strategy. Although I found the entire article amusingly interesting, there are two main nuggets of wisdom I wish to discuss here.

I particularly like the aspects of “rapid decision-making capability”, “short product develop cycles”, “strong customer relationships”, and “ready access to capital” or “cash on hand”. Also, the Energy-Maneuverability concept translates well into the acquisition-equivalent concept of ‘Resource-Maneuverability’. These first nuggets of wisdom are in the following excerpt from Best’s article [15]:

Question: How does your company’s management structure empower you to get inside your competitors’ O.O.D.A. loops?

Energy and Maneuverability (E&M):

Speed of execution hinges upon overcoming the energy and maneuverability of your competitor. For fighter pilots, “energy” translates into altitude (potential energy), thrust, gravity and speed (kinetic energy) and flexibility or maneuverability (pilot experience, aircraft design).

What are the business equivalents of potential (stored) and kinetic (unleashed) energy?

A company's *potential* energy can be construed as follows:

- An effective and rapid decision-making capability to confront competitive threats.
- High-quality and motivated personnel.
- New, not-yet-introduced technologies and short product develop cycles.
- Strong customer relationships (stored goodwill or karma).
- Strong brand identity, a quality image and high visibility.
- A low debt /asset ratio and ready access to capital.
- Cash on hand.

A company's *kinetic* energy might include:

- Hard-hitting, highly visible and effective marketing campaigns.
- New technology introductions of demonstrated value that capture the imagination of the marketplace.
- An enthusiastic, aggressive sales force, armed with effective marketing tools, compelling sales propositions, and T&E budgets.

Companies should strive to overcome their competitors' energy levels. A company with poor marketing energy risks being plucked out of the air by higher-energy competitors.

In my view, all of these aspects need to be under the control of a single organization that is low-level enough to understand the customer, while high-level enough to understand the interdependencies and interaction of the project as a whole.

The second nugget of wisdom includes the set of all questions gleaned from within Best's article. I've extracted them into Table 3 and translated the questions into their software development equivalents:

Table 3 - Important Questions Gleaned From Best Vantage Inc.

Best Vantage Inc. Questions	Equivalent Software Development Lessons
Are you satisfied that you enjoy a clear, unobstructed view of your market, your customers, and yourself?	Do the Program Manager and the Software Developers know the full job process of the targeted software users?
Does your company have the internal capabilities to forge rapid decisions when necessary?	What is the shortest time for approving the simplest change? Do decisions require resolution at weekly meetings? You are only as speedy as your least speedy change.
Is your company pro-active or reactive to changing circumstances?	Are we behind schedule?!? Are we seeking feedback from the users?
Does your organization's command structure permit you to sift and prioritize available options to arrive at the best possible decision...quickly?	Who is required to approve changes? Does change require a committee, small group of relevant experts, the program manager, ... the President?
How does your company's management structure empower you to get inside your competitors' O.O.D.A. loops?	Is the decision authority delegated to the same group that is also responsible and accountable for success or failure?
What would a self-audit of your company's energy states reveal?	How good is our energy <u>position</u> ? How tight can we turn and adapt? How much authority or budget do we command? How good are we at implementing changes?
In an environment squeezed by reduced profit margins and overworked personnel, is your company becoming more, or less maneuverable?	What is the energy <u>trend</u> ? Is it "Good getting better", "Good getting worse", "Bad getting better", or "Bad getting worse"?

Google, Inc., on 'Beta' software and Agile Practices

"To say that companies or CIOs are reluctant to embrace agile is like saying they wouldn't take aspirin for a headache ... and they're not only not taking the aspirin, they're banging their heads against the wall and wondering why it hurts."

— Jim Johnson, Standish Group Chairman

Google, Inc., is a widely recognized company that is popular for its extremely fast and accurate internet search engine. To a slightly lesser extent, Google is also well-known for their lead in development of trend-setting software technologies for

simplifying, filtering, and presenting large amounts of data in a highly intuitive way that even computer-illiterate people can easily learn to use. Many businesses use Google technology for their own internal network search engines. Government organizations, such as NASA, have partnered with Google to display NASA's Hubble Space Telescope images, and other space imagery, within their Google Earth and Google Sky applications.

One may also be aware that most of Google's software always seems to be in "Beta" mode — an indicator typically meaning that software, although publicly released, it is still in draft. Perhaps it is Google's way of legally avoiding responsibility or liability when your data is lost from a Google Mail account? Mr. Paul McNamara, of Network World, asked Google about the beta issue and documented his correspondence in his *PCWorld* article "Google Has Gone and Redefined 'Beta'". I think the following quote from Google speaks a lot about Google's attitude toward software development in a fast-paced world [16]:

We believe beta has a different meaning when applied to applications on the Web, where people expect continual improvements in a product. On the Web, you don't have to wait for the next version to be on the shelf or an update to become available. Improvements are rolled out as they're developed.

Wow! I'm not sure that the military is ready to just call everything 'beta' and release software that is not tested and certified, but I do think we can learn from this philosophy of "continual improvements" that are "rolled out as they're developed". Acquisition is such a difficult process to get through, given JCIDS, the PPBE system, JTIC security certification, etc., that we tend to group as many capability requirements as we can within each release or software update. In essence, we are catering to the time it takes to cycle

through the process rather than making the process cater to how fast we wish to cycle through new changes.

It does not surprise me that Best Vantage, Google, and many other business industries seem to be grappling with the same fundamental challenges of keeping up with rapid change. In my research of these businesses, the idea of ‘Agile Development’, ‘Agile Practices’, ‘Agile Manifesto’, and other similar ‘Agile’ concepts occurred often as a common technique for adapting to the fast-paced world of IT software development. Like any buzzword methodology, ‘Agile’ has been overused and ‘stolen’ by many different organizations claiming to be ‘Agile’. The Air Force even adopted the term in November, 2001, according to the *Program Manager* magazine article “Acquisition Center of Excellence Will Drive New Capabilities to the Warfighter” [17]:

The new office, led by a Senior Executive Service member, also will be the driving force for implementing “Agile Acquisition,” a sweeping series of initiatives designed to streamline the Air Force's acquisition systems. The plan was endorsed at a meeting of the Air Force's four-star generals and senior civilians in November 2001.

Some organizations have adapted honest and productive efforts for ‘Agile’ while many others have only attempted to rebrand their poor processes with this new jargon. One programmer, Steve Yegge, likened ‘Agile’ to fad diets, in that there are many diet plans out there and people want them to work, but nobody seems to lose any weight. Thus Yegge describes the existence of both ‘Good Agile’ and ‘Bad Agile’ [18]:

Well, as I mentioned, over the past year I've had the opportunity to watch both Bad Agile and Good Agile in motion, and I've asked the teams and tech leads (using both the Bad and Good forms) lots of questions: how they're doing, how they're feeling, how their process is working. I was really curious, in part because I'd consented to try Agile last Christmas ("hey, it can't hurt"), and wound up arguing with a teammate over exactly what metadata is allowed on index cards before giving up in disgust. Also

in part because I had some friends on a team who were getting kind of exhausted from what appeared to be a Death March, and that kind of thing doesn't seem to happen very often at Google.

Yegge's final sentence says a lot — Google has 'Good Agile', while his friends did not.

In the same vein of thinking, I like to say “Before evaluating what is said, first evaluate who is saying it and why”.

So what does Google do for its agile software development? In 2006, Google published a slideshow document about “Agile Practices on Real-World Projects.” In this document, Google lists the following 'Agile' principles:



Figure 5 - Google software development 'Agile 101' [19]

To the uninitiated, some of this may or may not make sense; so I'll make my best effort to translate each of Google's 'Agile 101' bullet statements in the following paragraphs.

Clear, customer visible stories

First, a story is a human-recognizable statement or sentence that defines the customer's requirements for the software. An example: "As an authenticated user, non-administrator, I must be able to drag-and-drop graphic images from the scheduling module to the reports module so that I can present my schedule solutions to my commander." The key difference between 'stories' and JCIDS 'requirements' development is that 'Agile' advocates an "ongoing, unfolding conversation about the details" and focuses on the "conditions of satisfaction" [20]. "Customer visible" simply means that it is the customer that must understand the story statement. After all, they are the reason for the software programming. Several companies indicated that they teach the customer how to write their own stories along the way. Regardless, the customer-developer interaction is early and continuous through the use of stories. Stories, like requirements, are rank-ordered according to customer preference.

Test-Driven Development

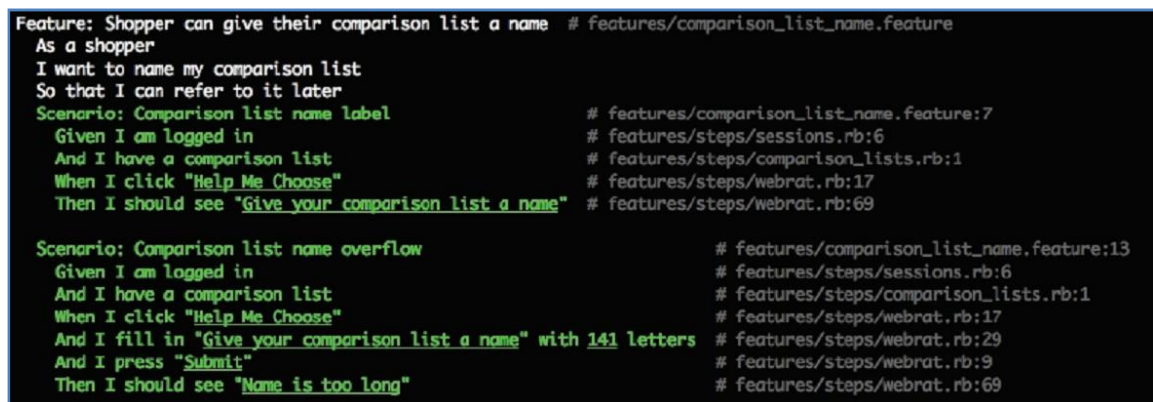
The 'test-driven' concept advocates the idea that the testing procedures for ensuring the software works should be written before the software is coded. No test pilot takes off on a test sortie without a detailed, coordinated, and approved test plan... neither should software developers start coding before knowing how to test. This test plan also aids documentation — something often skipped by 'bad agile' developers. Stories naturally lead into test statements, and this brings testing into the development cycle early in a concept called Story-Driven Test Development (STDD).

The HashRocket company, (named after the image likeness of the text called a hashrocket “=>”), is a pioneering web application consultancy group that specializes in the Ruby programming language and embraces the test-driven concept wholly.

According to HashRocket, risk is minimized by building and testing often:

Building a successful web application is rocket science ...but it doesn't have to take forever. Hashrocket is an expert consultancy group that uses best-of-breed technologies like Ruby on Rails to deliver the highest quality software in the least amount of time.

Figure 6 shows a screenshot of actual stories, from the HashRocket company, along with programming remarks on the right-hand side and scenario statements for how to test for ‘satisfaction’ of the requirement[21].

A screenshot of a text-based interface showing two sets of user stories and their corresponding test scenarios. The first set is for a feature where a shopper can name their comparison list. The second set is for a scenario where a comparison list name overflows. Each story is followed by a scenario with specific steps and expected outcomes, with file paths and line numbers provided for each step.

```
Feature: Shopper can give their comparison list a name # features/comparison_list_name.feature
As a shopper
  I want to name my comparison list
  So that I can refer to it later
Scenario: Comparison list name label # features/comparison_list_name.feature:7
  Given I am logged in # features/steps/sessions.rb:6
  And I have a comparison list # features/steps/comparison_lists.rb:1
  When I click "Help Me Choose" # features/steps/webrat.rb:17
  Then I should see "Give your comparison list a name" # features/steps/webrat.rb:69

Scenario: Comparison list name overflow # features/comparison_list_name.feature:13
  Given I am logged in # features/steps/sessions.rb:6
  And I have a comparison list # features/steps/comparison_lists.rb:1
  When I click "Help Me Choose" # features/steps/webrat.rb:17
  And I fill in "Give your comparison list a name" with 141 letters # features/steps/webrat.rb:29
  And I press "Submit" # features/steps/webrat.rb:9
  Then I should see "Name is too long" # features/steps/webrat.rb:69
```

Figure 6 - ‘Stories’ are requirements the customer *and* developer understand

Continuous Integration

With continuous integration, the various code modules are tested before mixing them together with the rest of the software application. Furthermore, according to Google, “the entire application is kept in a deployable state from the first week.” [19] In this way negative synergistic effects and other errors are detected early in the process rather than just in time before a design review.

Short Iterations

Google uses one week iterations “for easier course corrections, and [to] shorten the feedback cycle.” [19] This is crucially important in that the short cycle, combined with a continuously deployable application, allows the customer to validate the design requirements through “play testing”. Our current processes, by contrast do not provide anywhere near this amount of testing and customer validation. This means that any errant programming continues on the wrong vector much longer, until it is much costlier to fix. In the *IEEE Software* article, “Is Internet-Speed Software Development Different?”, the authors advocate agile development over traditional methods because it provides such a flexible environment for highly dynamic requirements: [22]

Developing software at Internet speed requires a flexible development environment that can cope with fast-changing requirements and an increasingly demanding market. Agile principles are better suited than traditional software development principles to provide such an environment. –IEEE

Pair Programming — crew concept/ formation flying for computer programmers

In pair programming, two programmers work together on a single machine — each programmer with a keyboard and mouse. I liken this to formation flying, or the crew concept, for Air Force aviators. It takes training and familiarity at first, but the benefits and synergy of two heads are better than one. Pair programming is one of the more contentious concepts of agile programming and ‘Extreme Programming’ (a form of agile). People tend to either love it or hate it. Even Google acknowledges that pair programming is “...probably the most controversial part of [Extreme Programming].” [19] Many feel that two programmers can get more done separately,

working in parallel on different code modules. In practice Google claims that pair programming “accelerates knowledge transfer” because “Your 80/20 rule overlaps favorably with your partner’s 80/20 rule.” [19] Visiting programmer Mr. Paul Barry describes learning about ‘ping pong pair programming’ at HashRocket, and how it aids test-driven development [23]:

Pair programming, at least to the degree that it is done at Hashrocket, was a new experience for me. Every developer works in a pair programming setup, with a laptop hooked up to a 30-inch display and two keyboards and two mice. Having a large monitor, two keyboards and two mice may seem like a luxury, but it really helps you get into the flow of pair programming.

One pair programming technique that makes TDD easier is ping pong pair programming. When doing ping pong pair programming, when you sit down to build a feature, the first person in the pair writes the test. Next, the second person in the pair writes the code to make the test pass. Then the second person writes a test for the next feature, and the first person in the pair implements it. By repeating this process throughout day, you have several benefits. First of all, it avoids the scenario where one developer does most of the work and the other developer just zones out and gets distracted by something else, because you are constantly switching back and forth. Also, like two people who go on a diet together to help each other stick to it, one developer doesn't let the other developer get lazy and skimp out on the tests for a specific feature. That doesn't apply just to the test. As long as you have a pair of two experienced programmers, one of them [isn't] going to let the other get away with writing a nasty piece of code. It's like a real-time code review.

Extensive Customer Involvement

Google believes that the “customer owns the priorities, [and] the developer owns the cost estimates.” This means that a feature isn’t marked as “done” until it is installed on the test demonstration server and approved by the customer. The story-driven and test-driven aspects ensure continuous customer-developer involvement, which in-turn ensures a solid software application. [19]

How Google estimates complexity instead of duration or budget

Google advocates estimating complexity instead of duration, since people are “better at estimating complexity than duration.” [19] They use a point system which estimates the complexity of each agile programming story in terms of 1, 2, or 3 points, based on the following table of criteria:

Table 4 - The Google point system for estimating complexity [19]

One point	“I know exactly how to do this, and can do it in half a day.”
Two points	“I know exactly how to do this, but it will be some work.”
Three points	“Somehow we will implement this feature.”

They go on to describe that three-point stories often become even more complex, driving the need to break down the story into smaller stories. “[Larger than three-point] tasks have fractal complexity: Small tasks are more predictable than large ones”, so Google always tries to break the stories down to three-points or less.

The customer (or PM) takes the stories the developer estimated for that week and prioritizes them, while keeping the complexity point estimates in mind. Google programmer development teams (pairs) begin coding the story modules. Using the point estimates, Google tracks how many points are coded by the development team each week. In their experience, “a focused team gets about as much done each week as it did the week before.” This velocity tracking method builds trust between customer and developer. Cost features, based on these estimates, are openly exposed to the customer, giving them greater control and creating “alignment between the developer and the customer.” [19]

Does Google's agile programming method scale well to large projects?

Google provided three examples to counter the following agile programming myths about scalability:

- “Agile is great for small projects, you can’t make it work in a big company.”
- “Agile is great for exploratory projects, where you don’t know the requirements up-front, but not for projects with big up-front requirements.”
- “Agile works for Greenfield projects, but not for big legacy applications.”

In the most telling example, a large web application was developed using traditional parallel development methods, but the persistence layer portion of the code wasn’t ready by the time the user interface coding needed to start — parallel anything breaks down when dependencies are involved. Google used agile programming methods and reduced the time-to-market for the application “by months on a 6-month release cycle.”

Google described how non-agile coding quality was far inferior to the test-driven development quality of agile programming. An application had been developed where one of the subsystems was coded with non-agile practices. This subsystem was less than 10 percent of the size of the whole application, but contained more than 75 percent of the coding errors. This one subsystem caused the whole application to delay release by 4 weeks — one third of the planned 12 week release cycle! [19]

DISA on improving software development using agile methods

Software Development

Dr. Steven J. Hutchison, Defense Information Systems Agency's (DISA) test and evaluation executive, recently wrote an extensive article titled "Reinventing IT Test and Evaluation", in April of 2009. On the subject of Software Development, Hutchison writes the following [24]:

Once upon a time, DoD acquired IT following a different process from weapons systems. In 1996, the department merged the acquisition of automated information systems into the DoD5000 series. When you think about that, the IT revolution was in full swing in 1996, with major online businesses, such as eBay, Amazon and Google, emerging as IT leaders. Given what emerged in the commercial sector in the decade-plus since, hindsight might suggest that DoD's "one size fits all" model has just about eliminated innovation in approaches to acquisition and results in much of the department's cost overruns, poor performance and lengthy schedule delays. In fact, a recent analysis of major automated information systems shows that the average time from program start to initial operational capability, following the DoD5000 model, is 91 months.

Hutchison then describes how the commercial sector has developed many approaches to improve software development and cites agile programming as "... one of the fastest growing and most productive approaches to software development." He further advocates that these agile principles "must be part of the requirements process, development, testing and oversight." He continues to describe how we should not front-load our expectations, nor expect everything in the Capability Development Document (CDD) to be delivered "... in the first increment". Hutchison feels we should be required to "...prioritize the requirements within the CDD, and focus the iterations on the

immediate needs of the war-fighter — the one or two capabilities that will have immediate beneficial impact if fielded” [24].

The agile PM has to transition from the waterfall approach to a highly collaborative environment where all stakeholders work together to develop, test, fix and verify solutions. In DoD’s evolutionary acquisition waterfall process, each increment is guided by the JROC, and managed to the same set of milestones. This is not an agile process [24].

Software Testing

On the subject of software testing, Hutchison declares the following [24]:

For T&E to be an enabler in such an environment, testers must be equally agile. Long lead time test plan approvals and lengthy staffing of test reports cannot be the standard, or T&E risks becoming an obstacle rather than an enabler. The approach I recommend is designed to achieve this objective. It is an integrating model where activity by one is accepted by all.

I refer to the model as capability test and evaluation (CT&E), and I describe it this way: “one team, one time, one set of conditions.” The intent of CT&E is to focus the four T&E activities on the goals of that iteration, and merge them into one short duration test period.

Hutchison considers testing as yet another “overly prescriptive process,” with four separate test activities occurring within each development cycle in the DoD IT T&E community. These four activities include developmental, operational, JITC, and IA certification and accreditation testing. These different organizations compete with each other for “time to conduct their events, under different test conditions, for different decision-makers”.

Despite the commonalities of these test activities, Hutchison laments our lack of coordination and teamwork, which leads to differing vectors for the different test activities [24]:

Unfortunately, we don't always conduct these test activities in an integrated manner; nor do we combine results in evaluation reports, and this can mean that different information is being presented to the decision-makers — the milestone decision authority, interoperability certifier and the IA certifier, or “designated approving authority.” Acquisition decision-making would be greatly improved if the various T&E activities were synchronized to produce a single evaluation that satisfies the needs of all three decision-makers.

Additionally, the test process itself “typically exceeds six months” and it is not conducive to agile programming processes that need to test all of the time [24].

Improving on JITC certification

In 2008, DISA made concerted efforts to improve upon lengthy and cumbersome portions of the test process. JITC certification is an important but huge obstacle in the grand scheme of things. Despite DISA efforts, they have yet to gain full support for these changes. According to Hutchison:

Last year, a combined industry/government council on OT&E recommended sweeping changes to the format and content of the T&E Master Plan (TEMP). During the committee discussions on the proposed TEMP, it was clear that everyone thought the ideas were good and should be pursued. Unfortunately, no decisions were made to experiment with the format, and it looked as if the good work of the group was going to fall by the wayside.

Significant time savings can be accomplished with simple reforms to the testing process, as Hutchison points out:

DISA volunteered to use the new TEMP for the Net-Centric Enterprise Services program, since a new TEMP was required for the pending Milestone C. For comparison purposes, the Milestone B TEMP exceeded 360 pages, required 18 signatures and took over six months to obtain approval once it began the staffing process. The new document prepared for Milestone C, which is generally a far more detailed plan, took only 70 pages, 14 signatures and 90 days from the beginning of document preparation to final approval. That's a successful pilot by any measure.

Given the DISA experience, the rest of the DoD should follow their lead in adopting agile methods to improve JCIDS. Hutchison recommends immediate update of the DoD 5000 regulations, but advises that the acquisition community should not wait for it and should migrate toward agile development and testing. Hutchison concludes his comments on agile and testing with recommendations for supervision [24]:

... we need agile oversight. We can improve the way we acquire and test information technologies by focusing on the immediate needs of the warfighter and development in short duration sprints. In T&E, a one-team, one-time, one-set-of-conditions approach is essential to delivering improved capabilities in significantly less time.

Determine how to integrate with existing JCIDS process

“... agility is an important attribute for a successful project since it is the most effective way of dealing with uncertainty and avoiding or if necessary responding effectively to crises.”

— Steve Adolph, What Lessons Can the Agile Community Learn from A Maverick Fighter Pilot?[25]

The last major area of study involves how we can better integrate efforts we already have into the acquisition process. Not only should we adopt the best practices of the agile development community, but we need to take advantage of the experimentation

efforts in action today. First, we need to establish a balance between JCIDS, traditional development, and agile development.

Balancing between JCIDS and Agile: Learning from the JPADS ATCD success

[JPADS] revolutionized the way we resupply the War-fighter...Saving American blood is a good thing, and this will definitely do it.
—Maj Gen Scott Gray, Commander, Air Mobility Warfare Center

Choosing between a traditional method and the agile method is not an “either/or” decision. We can use the most conservative way of doing acquisition, in terms of resource efficiency, or we can adopt agile techniques to shift our priority toward ‘time-to-market’ efficiency. I argue that we can adopt most of the agile concepts and improve development speed and efficiency without reducing quality or increasing risk. Yet, even if we make significant improvements, there will always be a need for rapid custom software development for unforeseen, unplanned, and unbudgeted urgent war-fighter needs.

We should simply use risk analysis to determine whether or not to use JCIDS and traditional acquisitions vs. JUONS-type of agile software development and procurement method. There is a recent non-IT example we can learn from — the Joint Precision Airdrop System (JPADS). This system uses GPS-guided steerable parachutes to accurately deliver needed supplies to the war-fighter — even in the high-altitude, mountainous terrain of Afghanistan. JPADS allows the cargo aircraft to remain at high altitude, out of the range of most enemy anti-aircraft weapons, and still get the cargo drop where it is needed.

In 2004, JPADS started out as a Joint ACTD with a few prototypes for various sizes of parachutes and weights of cargo. JPADS became a formally funded program in 2008, largely due to the ongoing operations in Afghanistan for Operation ENDURING FREEDOM. I often wonder if JPADS would have survived had the urgency of war not driven the high-level visibility needed to encourage funding for this revolutionary system. Even so, it took a “partnership of U.S. Joint Forces Command, U.S. Transportation Command, U.S. Special Operations Command, U.S. Army, U.S Air Force, U.S. Marine Corps, and others...” to help make it work, according to the Under Secretary of Defense, in the article, “Success Story: ACTD Helps War-fighters Get High Altitude, Accurate Parachute Resupplies”.



Figure 7 - JPADS enables standoff precision airdrop using guided parachutes[26]

There was difficulty of retroactively performing JCIDS processes and analyses on the JPADS ACTD. I saw this firsthand while working in the Mission Planning Systems branch of the Plans and Programs division of Headquarters, Air Mobility Command (AMC). Risk Analysis had determined that the JPADS ACTD was an urgent war-fighter need and it was funded accordingly for rapid development. Future increments of JPADS, however needed to proceed through normal acquisitions methods — including a

Functional Solutions Analysis, Analysis of Alternatives, the production of an Initial Capabilities Document (ICD), etc. When many of these initial JCIDS steps are made to determine the risk of proceeding toward low-rate initial production and eventually full-rate production, it seemed that JPADS had already passed these hurdles—since avionics integration requirements were already accomplished. Indeed the Capabilities Development Document (CDD) was approved for use in lieu of the Capability Production Document (CPD), which enabled Milestone C approval for full-rate production. Regardless, just as changing JCIDS ACAT levels generates cumbersome “what if” questions and retroactive process requirements, the JPADS ACTD also had to overcome these ‘square peg, round hole’ issues when it was refactored into JCIDS.

A better way to integrate advanced technology: The WIDE/GRS ATD Struggle

JPADS was a success story in that it was a valuable system that quickly achieved operational fielding to meet urgent war-fighter needs. JPADS was also successfully integrated into JCIDS for future increments. Another ATD that I had experience with has, to date, not been so privileged.

The Tanker Airlift Control Center (TACC) commands and controls all transient AMC aircraft throughout the world. A TACC-managed aircraft is landing or taking off somewhere in the world every 90 seconds, so the TACC can be a busy place at times. Several custom programming projects had been created by people with computer skills, usually evolving spreadsheets that had been made to help TACC controllers with repetitive tasks — improving some of the customized software used by the floor operators to keep track of the nuances of each AMC mission they controlled.

Much of the command and control (C2) software was developed like a large database, where the mission details, aircraft tail numbers and details, aircrew names and details, cargo details, flight plan and diplomatic clearance details, weather and other details, were all centralized for improved management. Although state-of-the-art at the time, many of these legacy systems consisted mostly of text displays with massive amounts of look-alike data, see Figure 8.

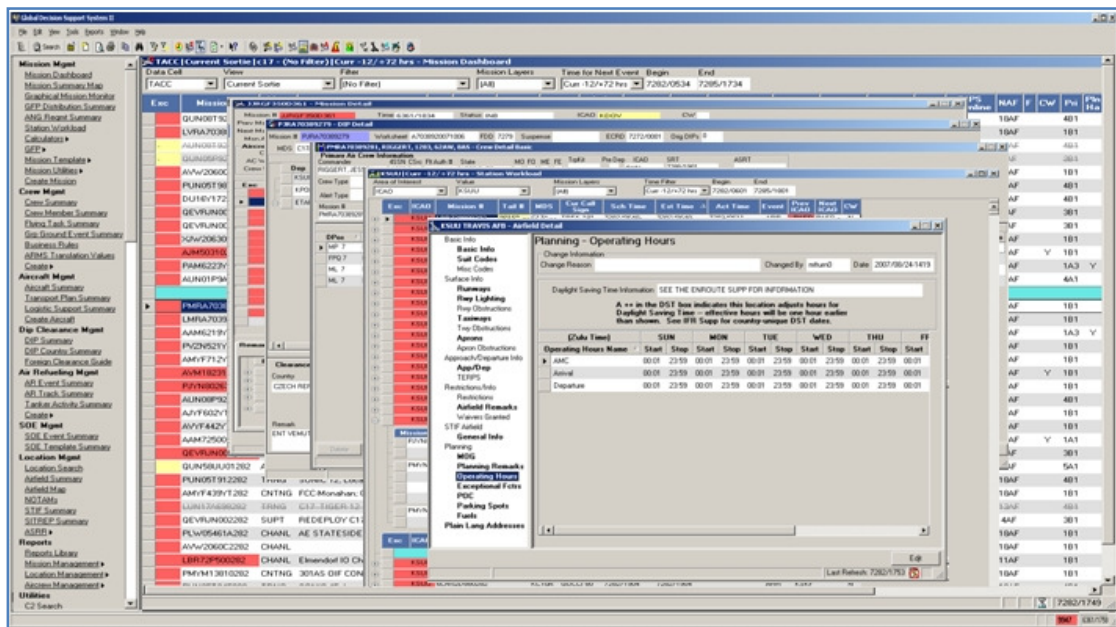


Figure 8 - Legacy Software Tool showing hard-to-read text displays [27]

In 2006, a team from the Air Force Research Laboratory (AFRL) was working on an ATD for visualizing C2 data. The Work-Centered Interface Distributed Environment (WIDE) and Global Response & Synchronization (GRS) ATD's were both managed by the Human Effectiveness and Information Directorates at AFRL. This team took a human factors approach toward simplifying and filtering massive amounts of mission-related data and presenting it in an intuitive graphics display for the users to better

intuitively, while also allowing the user to ‘drill down’ into the textual data as needed, (see Figure 10).

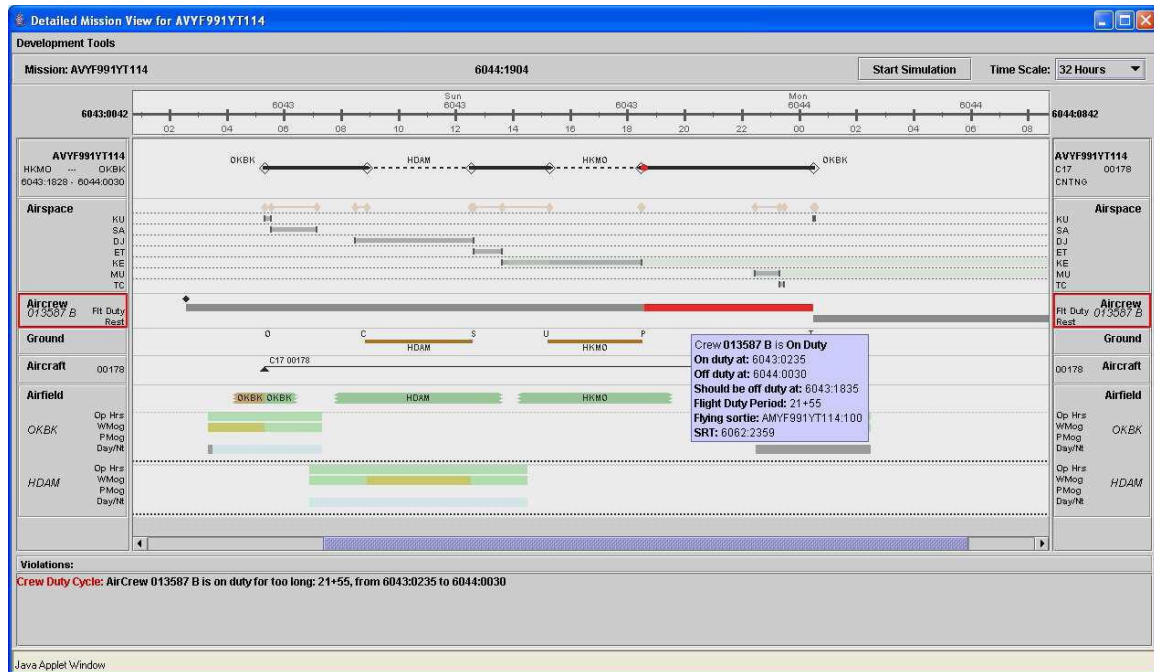


Figure 10 – WIDE/GRS ATD showing interactive timeline [28]

Experienced TACC floor controllers are able to fuse the massive amount of dynamic data in their head and come up with solutions, but this data fusion is difficult to learn and it takes a while to immerse and become familiar with the data once ‘on shift’. WIDE/GRS automated much of the fusion and elimination of incompatible solutions. This reduced the amount of data and simplified the decision process of weighing options in order to arrive at a solution.

The improvement in timeliness and increase in quality of solutions and decisions was born out in field testing with operational TACC controllers. Tests were run using both the legacy software systems (information-equivalents similar to what the controllers

were already familiar with in the TACC) and then using the WIDE/GRS software. After a 30 minute familiarization training session, the controllers were presented scenarios and evaluated on the timeliness and quality of their solution to the scenario. An example scenario might involve finding the best airplane to perform an emergency medical evacuation of a patient, while minimizing the impact on other missions. WIDE/GRS testing showed a drastic improvement over legacy methods, especially for the less experienced controllers, and enabled them to get the correct solution in significantly less time. A sample of the test results are shown below, in Figure 11.

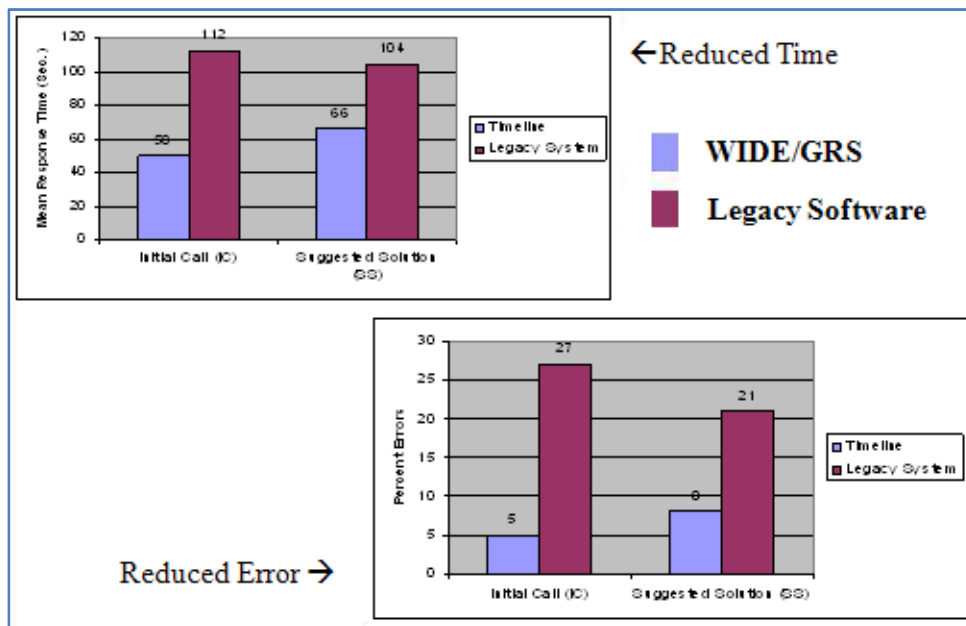


Figure 11 - WIDE ATD reduced time and error by more than half [28]

A concerted effort was made to integrate WIDE/GRS technology into existing TACC C2 software. However, due to funding difficulties and issues with building a dynamic interface to the centralized mission data, the efforts to field WIDE/GRS technology to the operational TACC controllers are currently on hold. There is an

ongoing effort to integrate this ATD into software used by USTRANSCOM, (the result of which has yet to be determined). Perhaps it will be saved from the metaphorical warehouse of crates ...once we can get “top men working on it ... top men” [29].



Figure 12 - Crate Warehouse from the movie "Raiders of the Lost Ark" [29].

Increase the use of MAJCOM and AOC ‘business centers’

For much of my career I’ve developed and improved upon custom software tools, saving users from endless stacks of sticky notes and automating many of the tedious and error-prone tasks often encountered when managing people, airplanes, and cargo. The problem with these ‘stove piped’ efforts is that most military officers move every couple of years and there is no maintenance plan for the stovepipe software, other than tracking down the original programmer for troubleshooting.

Despite the stove pipe issue, the alternative was also undesirable. With the increasing workload and decreasing manning, we've become more and more dependent on technology to fill the gap, yet that needed technology capability takes years to develop once the need for it is realized. Hence, there have been many occasions to create stovepipe software that was meant only to fill the gap until a formal software acquisitions effort came in to replace it.

Often, nothing has been developed to replace stove pipe software. In fact, a software tool I developed, in 2004, for deployed operations command and control was still being used the last time I checked in 2007. I released an update to the tool in 2006, (based on unsolved challenges I had since figured out), making sure it properly calculated for leap years and other 'future-proofing' efforts. In some respects, this software enabled the bad aspects of our system in that, had the software not been developed, the urgent operational need for a formal acquisitions effort would have demanded new technology.

The Happy Medium

The TACC has a software 'business center', as it was called in 2007, which has since become a 'rapid application and requirements development branch'. This branch has several civilian personnel working with military leadership. They developed, among other things, custom tools requested by the TACC commander, et al. They programmed web applications and designed custom interfaces between some of the stove pipe software and larger enterprise applications. They were not funded or manned adequately enough to take over all of the stove pipe software that had been developed, but they did

take over some of the software maintenance. In a TACC briefing, I likened the IT software development environment as pictured in Figure 13, below.

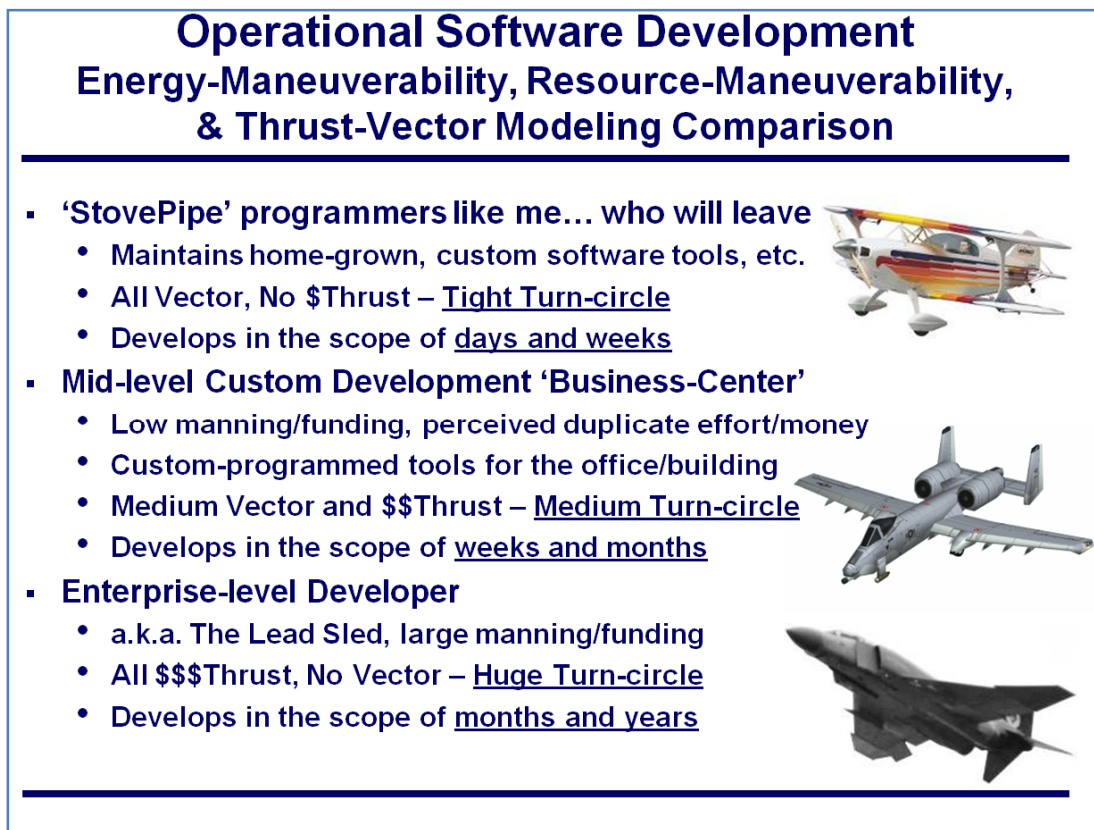


Figure 13 - Software Development Turn Circles

The TACC business center was a happy medium between stove piped programming and waiting too long for desperately needed software tools deemed not important enough for the budget. Again, I fault the *way* we budget, not necessarily *how much* we budget. Follow this scenario, for example, when your hard disk fills up with files, do you try to find one or two large unneeded files and delete them, or do you delete ten percent of every file? Of course, you don't delete ten percent of every file, since this would corrupt all of the files! Why then, when we need to trim our budgets, do we seem to think it is 'fairness' that matters, cutting all programs equally? The end result is that

every program is now corrupted and must adjust their acquisition strategy — often by dropping features or by extending the development timeline and costs into the next fiscal year's budget.

Linking MAJCOM and AOC business centers with ATDs, ACTDs and JACTDs

My overall argument with respect to business centers is that we lose time and money while we wait for large software programs that take too long to field. Stove pipe programmers are filling the gap, but are not a viable long-term answer either — there is too much risk in depending on unmaintained, often incompletely developed software tools. The answer is to increase the use of business centers. Furthermore, these business centers can and should link with ATDs, ACTDs, and JACTDs. I believe, given the agile principles explained earlier, that MAJCOM or AOC level business centers have the right people to play the 'customer' role with developers of all types, including research labs such as AFRL and DARPA. The business center can also create a viable, funded, sponsor for 'pulling' advanced technology efforts into the JCIDS process, feeding into the 'experimentation' and 'concept refinement' areas. We must make integration of at least some advanced technology efforts the 'rule', not the 'exception to the rule'.

Pros of MAJCOM and AOC business centers:

- Software is tailored and custom developed to the unique requirements of users
- There is more face-to-face interaction between customer and developer
 - This improves software quality by better refining requirements
- There are fewer competing priorities and budgets within a MAJCOM or AOC
- Less 'one-size-fits-all' software development
- Business centers can foster innovation through working groups and conferences
 - Business centers exchange lessons learned, standards, and frameworks
- Reduces single-point-of-failure dependencies on any particular software
- Users may get the software tools they need instead of waiting for enterprise developers to build low-priority features into already bloated applications

Cons of MAJCOM and AOC business centers:

- Duplication of effort occurs when two different centers develop similar tools
 - Budgeting can more efficient with increased economy of scale
- Tools for one business center may be incompatible with another's
- Different language/procedures/cultures develop because different tools are used
- Must force cross-flow and interplay between centers
 - Working groups and conferences require significant effort
- Resistance to adopting another center's products, organizational inertia
- Decentralization complicates centralized security management

IV. Conclusions and Future Research

“Program turbulence is #1 obstacle to commercial investment in defense projects”

- Norm Augustine

Findings

The following conclusions were reached: (a) software projects must be scoped and scheduled for development cycles on the order of months, not years, and use open architecture, Agile Development methods, and scalable designs with modular code; (b) budgets must be stabilized for long-term integrity, with a software development working capital fund reserved for JUONS-like urgent IT needs; (c) increased use of MAJCOM- or AOC-level business centers must be encouraged and funded to produce tailored software modules that interface with larger agile programs built to accept these modules; (d) we must take advantage of ATCD and ATD efforts from research laboratories, giving MAJCOM and AOC business centers budget authority to “pull” a limited amount of ATDs, ACTDs, and JACTDs from the labs, through the appropriate System Program Office, to produce and field operational software (by default, not by exception); and (e) periodic software development working groups and conferences should be continued, but with emphasis on standardization and sharing of lessons learned between services, MAJCOMs, and AOCs.

The aviator part of me thinks of the two-year POM process, JCIDS, and other process limitations as a large turning radius, limiting the maneuverability of our metaphorical acquisitions aircraft. The IT pace of change is like the tiny Mig-15 fighters of the Korean conflict... small, able to tightly turn, and able to out maneuver our less agile F-86 Sabre jets. Yet, with our superior tactics and training, our pilots were enabled,

empowered, and indeed expected to out-fly the Korean enemy — which they eventually did with better than an eight to one kill ratio. Will we be able to do as well when the ‘cyberspace dogfight’ comes upon us?

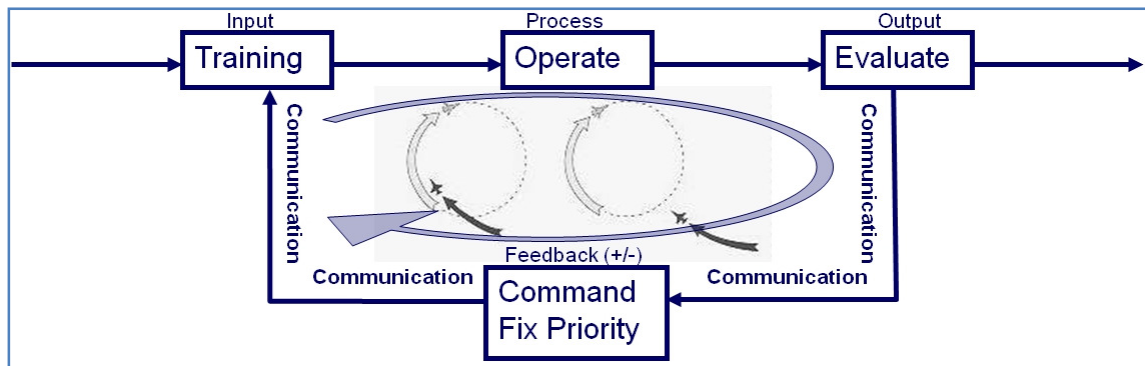


Figure 14 - The Software Development Dogfight

Future Research

Develop Separate DoD 5000 Instructions for Information Technology

Future research should continue with the development of IT acquisition as a separate entity from both Space and non-Space acquisition (refer to Figure 2 and Table 2). The history of the separation of space technology from non-space technology should be accomplished. We’ve developed space systems long before JCIDS and traditional acquisitions were developed. Perhaps the same reasoning that was used to treat Space acquisition differently from non-Space acquisition can be replicated to bolster the argument for the additional separation of IT acquisition from non-IT acquisition. The review should include applicable portions from the National Security Space (NSS) Acquisition Policy 03.01, the DOD Instruction 5000.1 and 5000.2, AFRD 63-1 Capabilities-Based Acquisition System, and any currently effective memoranda.

Reassess required Key-Performance Parameters (KPP) for IT

Every acquisition program must include the net-ready KPP, which ensures compliance with the net-centric operations and warfare model, compliance with Global Information Grid (GIG) profiles, Information Assurance (IA) security requirements, and Integrated Architecture (IA) concepts [5]. Future research should reassess and revise these requirements. In general the net-ready KPP has been positive, but we need to modernize it a bit. Net-centric *capability* is good, while net-centric *over-dependence* is not.

My experience is that IA security has always been an afterthought. Upfront and continuous security focus prior to JITC certification helps. There is still a need to improve the timeliness of JITC certification. If programs ‘think security’ from the beginning they will be much better off. Investigate ways to improve security focus, balanced with accessibility. Security requirements should include the ability to log or analyze appropriate information such that data manipulation (malicious or accidental) can be detected. We may not always prevent a security breach, but we must have enough data for forensics analysis to determine what happened.

Accessibility always competes with security. In the past we’ve often built security with the mindset of a fence between inner and outer domains. We must become more layered than this, and consider a sort of ‘Plan B’ KPP. In my mind, a ‘Plan B’ KPP provides a failsafe capability for when the power goes out, the network goes down, or parts of the system have been hacked or corrupted. All systems should have the ability to print to hard copy, capture screen shots, or similarly store data with some type of robust

portable means. The intent is to not only foster the failsafe capability, but to enable a ‘bug-out’ situation to a new location for continued operations under duress.

Lastly, Human Factors have long been highly considered in the development of fighter cockpits, with concepts such as the Heads-Up-Display (HUD) and HOTAS (Hands On Throttle And Stick). There has been a tendency to disregard human factors when it comes to IT software development — or it is too a low priority. Instead, we must treat computers like a fighter cockpit, monitors and displays like a HUD, and keyboards and mice like HOTAS. We should adopt a standardized description for software interface controls, (i.e. we should not have to explicitly request that the mouse-wheel functions for scrolling a text display!). Look to Apple Computer, Microsoft, Google, et al, for inspiration in this regard — they provide both good and bad examples of human factors integration. Perhaps we can hire them for training and project management?!?!

Determine acquisition categories based on risk and complexity, not cost estimates

Acquisition categories are primarily based on the costs estimates, rather than risk or complexity of the program. Of course, the JROC may always designate a program as “Joint Special Interest”, based on any criteria they may choose. However, an area for further research should investigate the use of a non-budget-centric method for categorizing our programs. This is especially so, given Google’s experiences with agile — using complexity estimates rather than cost or time estimates.

Similar to the desire for event-based criteria instead of time-based criteria for development strategy, we should have complexity-based criteria instead of cost-based criteria for determining acquisition category. A risk analysis process can determine this

categorization (which should still consider cost). The aim is to prevent shortcuts and increased risk in an attempt to avoid a Nunn-McCurdy breach (when cost overruns exceed 15%). Instead, the aim should be to reduce risk and avoid an escalation in risk-category. A fast-paced program that becomes high-risk can be slowed down, if necessary, without undue concern or focus on the budget impact. More people can be dedicated to help with the workload of the high-complexity areas.

The fact that our budget process discourages changes across fiscal years must be addressed in order to provide the flexibility for this system to work. Pilots learning to fly in poor weather learn to focus on control instruments, such as the artificial horizon indicator, rather than performance instruments like the altimeter or heading indicator. Budget measurements like performance instruments lag too much to be use for proactive control over the system. We must forge a different *way* to do our budgets and categories.

This page intentionally left blank.

Appendix A: Fundamental Concepts

Moore's Law

In 1965, Gordon E. Moore, then director of Fairchild Camera and Instrument Corporation, Semiconductor Division's Research and Development Laboratories, wrote an article in *Electronics* magazine titled "Cramming more components onto integrated circuits." In this article, he declared that the integrated circuit is "the future of electronic itself" and successfully predicted that integrated circuits would lead to such controversial wonders as "home computers — or at least terminals connected to a central computer — automatic controls for automobiles, and personal portable communication equipment."

Among other discussions of reliability and increased use of integrated electronics within industry, Moore presented a graph not unlike the one pictured below, in Figure 15, taken from Moore's original notes. This image is reprinted and captioned according to the Intel Corporation requirements. [30]

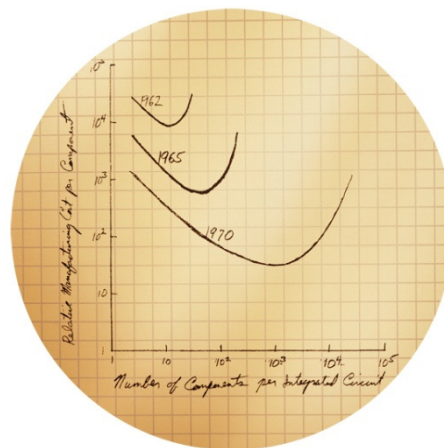


Figure 15 - Moore's Law Original Graph

In 1965, Gordon Moore sketched out his prediction of the pace of silicon technology. Decades later, Moore's Law remains true, driven largely by Intel's unparalleled silicon expertise.

Copyright © 2005 Intel Corporation. Used with Permission. [31]

http://download.intel.com/museum/Moores_Law/Images_Assets/Image_Usage_Guide_Readme.pdf

CPU Transistor Counts 1971-2008 & Moore's Law

Transistor count

Curve shows 'Moore's Law': transistor count doubling every two years

Date of introduction

Processor	Approx. Date of Introduction	Approximate Transistor Count
4004	1971	2,300
8008	1972	6,000
8080	1976	29,000
286	1982	120,000
386	1985	275,000
486	1989	1,200,000
Pentium	1993	3,100,000
K5	1996	4,000,000
K6	1997	5,500,000
PIII	1999	9,500,000
K7	2000	29,000,000
K8	2002	290,000,000
Barton	2003	100,000,000
Atom	2005	29,000,000
Core 2 Duo	2006	290,000,000
Core Cell	2006	290,000,000
K10	2007	730,000,000
RV770	2007	730,000,000
GT200	2007	730,000,000
Quad-Core Itanium Tukowila	2007	730,000,000
Dual-Core Itanium 2	2007	730,000,000
POWER6	2007	730,000,000
G80	2007	730,000,000
Itanium 2 with 9MB cache	2007	730,000,000
Core 2 Quad	2007	730,000,000
Itanium 2	2007	730,000,000
P4	2000	29,000,000

61

Rock's Law

According to Philip E. Ross, in his article “*5 Commandments: The rules engineers live by weren't always set in stone*”, Rock's Law was first attributed to venture capitalist Arthur Rock by Gordon Moore. Philip writes that the law is “...sometimes called Moore's Second Law, because Moore first spoke of it publicly in the mid-1990s, we are calling it Rock's Law because Moore himself attributes it to Arthur Rock, an early investor in Intel, who noted that the cost of semiconductor tools doubles every four years.” [33]

Rock's Law is considered by many to be the economic opposite to Moore's Law, in that Moore's law is limited only by the expansion of technology while Rock's Law is the counter force limited by economic reality. Eventually, the inability to cheaply fabricate circuitry becomes the dominant limiting factor. Although many contest that Rock's Law is not holding true, and our efforts should instead focus on the fabrication cost per transistor (not the fabrication tools), the Law is still useful in the broad sense. Rock's Law may need revision before it is to be used for accurate and precise predictions of the future, but it will probably always be correct in that economics will tend to counter and regulate any exponential technology growth.

Joint Capabilities Integration & Development System (JCIDS)

The JCIDS process is a highly refined and thorough framework for the DoD acquisitions process. Despite any shortcomings with respect to IT software development, JCIDS and traditional acquisitions are a positive culmination of years of previous acquisition reform efforts. I believe the best way to learn the nuances and details of JCIDS is to simply dive right in and review the Defense Acquisition University's (DAU) website on JCIDS: <https://acc.dau.mil/CommunityBrowser.aspx?id=28947>. Also search for the Defense Acquisition Guidebook for more information [5], and CJCSI 3170.01G (or the most current reference), which outlines JCIDS policy.

The DAU site has several training courses, documents, and briefings. There is even an interactive Integrated Framework Chart: <https://acc.dau.mil/ifc/>. You may click on links within the chart to zoom in on detailed descriptions, with links to even more information. Figure 17 shows the overview of the entire chart, where you can notice the “V”-shaped patterns characteristic of systems engineering processes. Figure 18 and Figure 19 both show a bit more detail on how JCIDS fits into the overall Defense Acquisition System.

The challenge, as you study the acquisitions process, is to envision it within the larger OODA loop of the USA, DoD, and military services. We must not allow the control-performance lag introduced by JCIDS and the Defense Acquisitions System to dominate our operations. Yet, we cannot just skip steps without consequence, so we must be prudent in our continued reform of JCIDS and the Defense Acquisitions System.

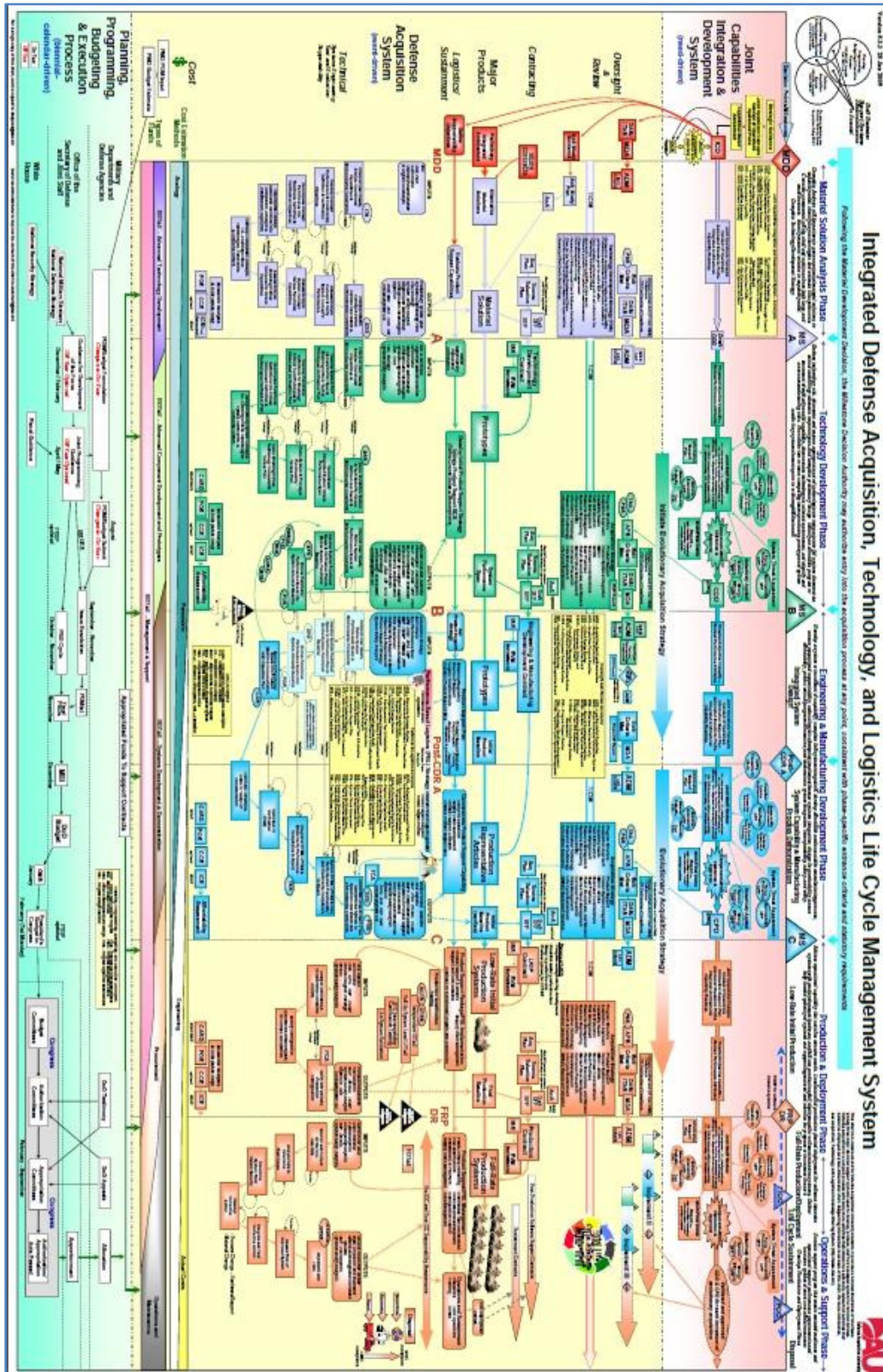


Figure 17 - JCIDS & Defense Acquisition System Integrated Framework Chart [5]

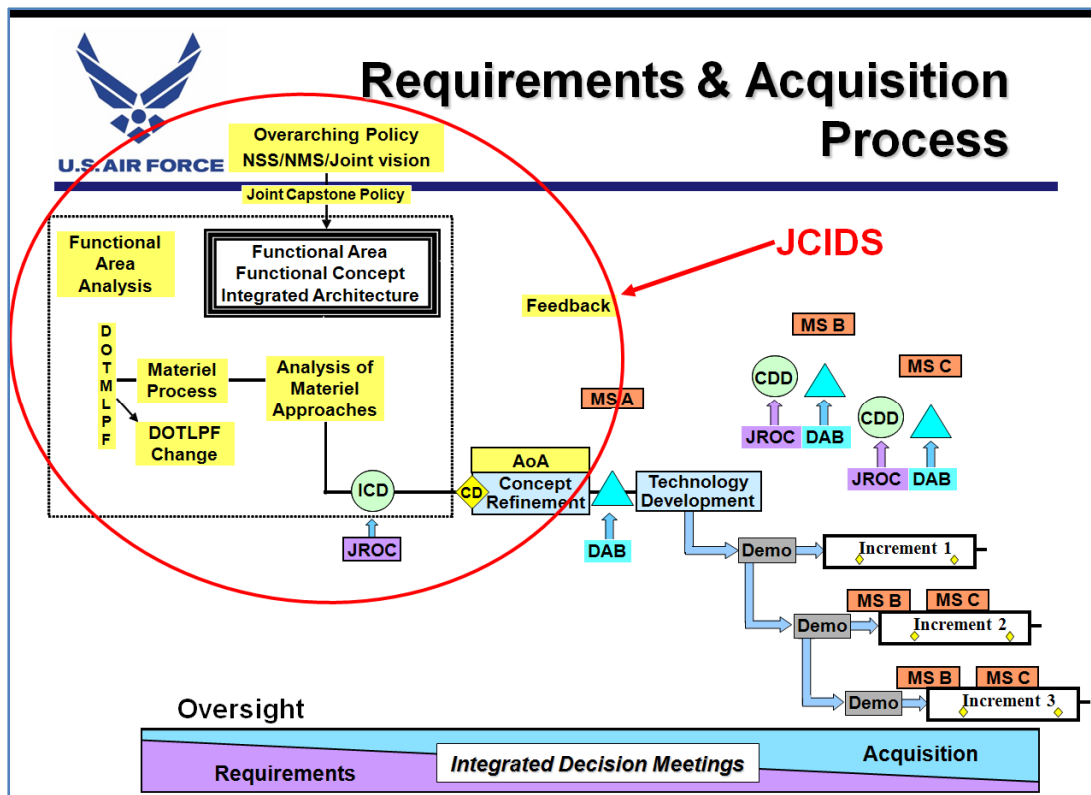


Figure 18 - DoD Architecture, Requirements & Acquisition Process [34]

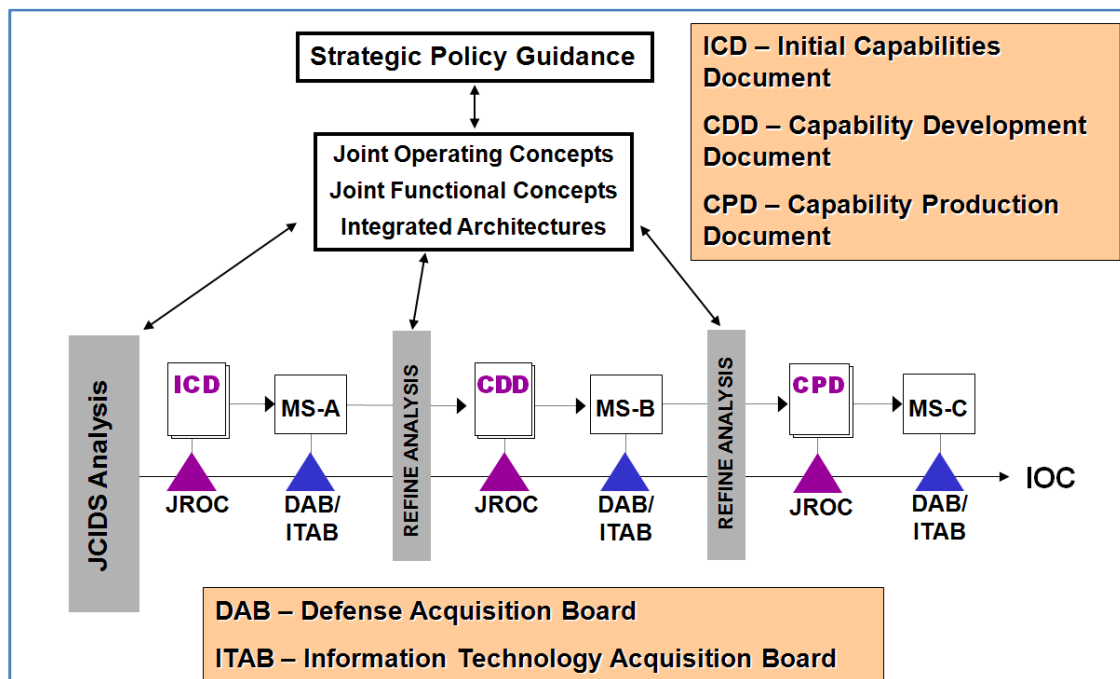


Figure 19 – JCIDS and the Defense Acquisition System [34]

Colonel John R. Boyd

The late Colonel John R. Boyd was a USAF fighter pilot that developed the energy-maneuverability theory for aerial combat that is still the core philosophy for fighter combat training today. Colonel Boyd was soon recognized as a strategic visionary and went to the Pentagon where he continued with a reputation as a shrewd and confrontational maverick, resurrecting the F-X project which became the F-15. He followed this effort with the light-weight fighter program that led to the F-16 and F-18 fighter jets. Colonel Boyd was discouraged by the Pentagon's emphasis on machines over people and ideas, with our best warriors and commanders leaving in unprecedented numbers. "People, ideas, hardware – in that order", Boyd would often shout. [1] Colonel Boyd's advice was relied upon and credited for helping form the winning strategies used in Operation Desert Storm.

Perhaps most famously, Colonel Boyd is known for the OODA loop (see Figure 20), used by military and business organizations alike, to provide a framework for analyzing the various parts of the organization's performance. It is important to note the OODA loop is much more faceted than a single simple loop — there are multiple feedback loops within the larger OODA loop, feed-forward loops, and decision points. Operating the OODA loop faster than the enemy's OODA loop yields an advantage, but 'faster' never meant skipping any of the steps:

- When you're doing OODA "loops" right, accuracy and speed improve together; they don't trade off.
- A primary function of management is to build an organization that gets better and better at these things.

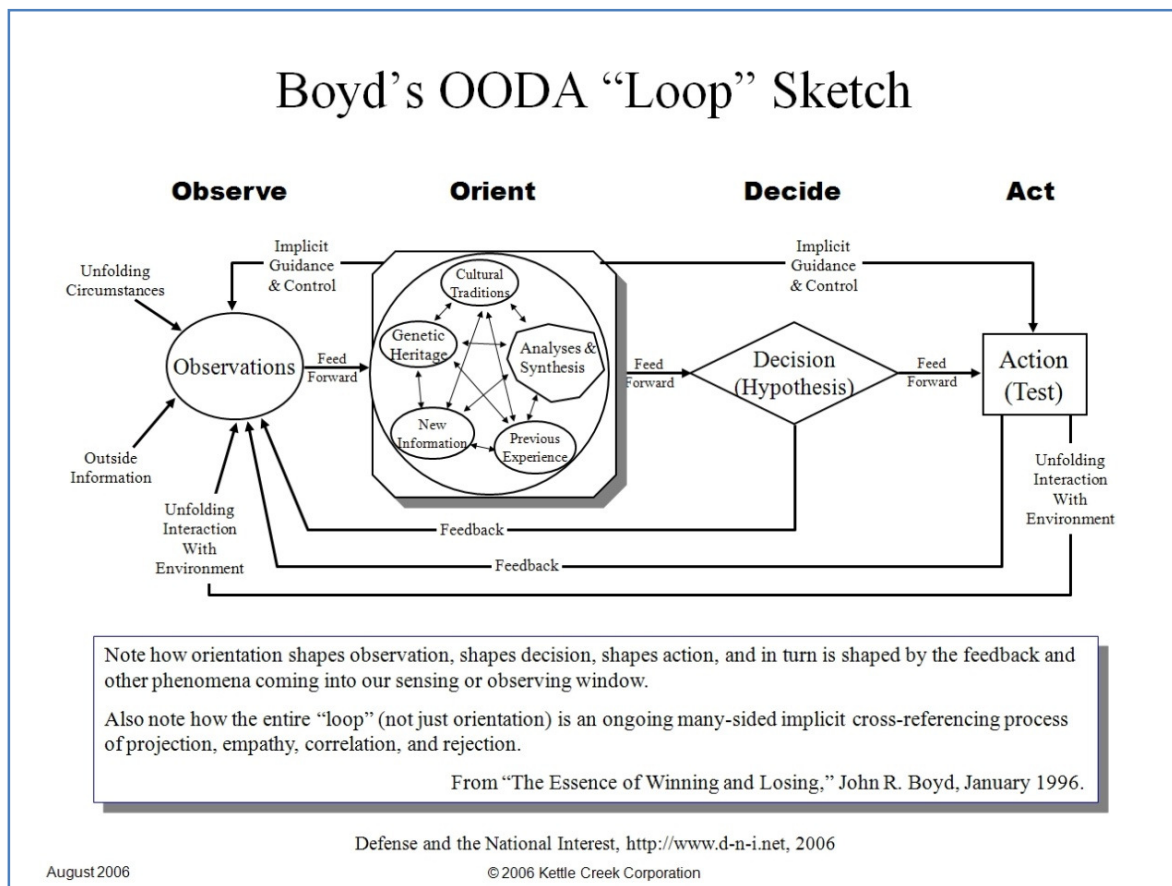


Figure 20 - The One and Only OODA Loop [35]

I think the best source for further information is the *Defense and the National Interest* online web site, <http://www.d-n-i.net/dni/john-r-boyd/> [35]. This site contains many of Boyd's unfinished works, including "A Discourse on Winning and Losing", which contains "Patterns of Conflict", "Strategic Game of ? and ?", and "Organic Design for Command and Control".

There are also several books which discuss Colonel Boyd, such as "Boy: The Fighter Pilot Who Changed the Art of War", by Robert Coram, and another one of my favorites "The Pentagon Wars: Reformers Challenge the Old Guard", by James G. Burton. Both are essential reading for acquisition mavericks out there in the world [36].

Law of Competing Motives

Throughout my experiences managing programs through the acquisitions process I observed that, although acquisitions efforts were supposed to be primarily requirements driven, they tended to ebb and flow between competing motivations. Depending on the focus of any particular meeting or working group, the meeting's priorities and conclusions tended to favor one motive at the exclusion of others. I tried to postulate and extract what exactly should always be our primary motivation — the main thing we should try to make most efficient and most effective.

Given my operational flying background, I initially advocated that the mission should be the dominant motive, since it seemed that we derive the other motives from the mission or task to be accomplished. After all, we shouldn't be creating missions to justify the need for more people — it is the other way around, thus we have people only because we need them to perform tasks and missions.

Realizing my own operational bias, I considered other perspectives and concluded that none of the motivations should be dominant. Furthermore, I observed that the more out of balance a particular acquisition program's competing motives were, the more unstable, costly, unpredictable, and uncontrollable the program became — regardless of what the deviant motive was.

Thus, I developed the Law of Competing Motives (see Figure 21) as follows:

- 1) Proper leadership is the influence which brings the competing motives into balance — that is, toward the center of the color wheel. The correct leadership for a given situation isn't always the center of the color circle, but it isn't at the edge either. The correct leadership depends on assessing which motive or motives are relatively over-dominant or under-dominant and

counter-adjusting resources and priorities accordingly, (i.e. one 'unit' of leftward deviation needs one 'unit' of rightward leadership correction).

- 2) Stability of leadership is essential for maintaining consistent performance. If the commander or director changes their emphasis too often, the performance will circle about the center of the color wheel in a sort of hysteresis loop eddy current. This hysteresis loss represents inefficiency in the use of resources; the bigger the area encircled, the bigger the efficiency loss.
- 3) Allowing one or more motives to remain over-dominant drives relative inefficiency in all other areas. The difficulty, of course, is properly assessing and comparing qualitative factors with quantitative measures. Thus inefficiency in one area may be an indicator of incorrect measurement or assessment rather than incorrect leadership or correction.

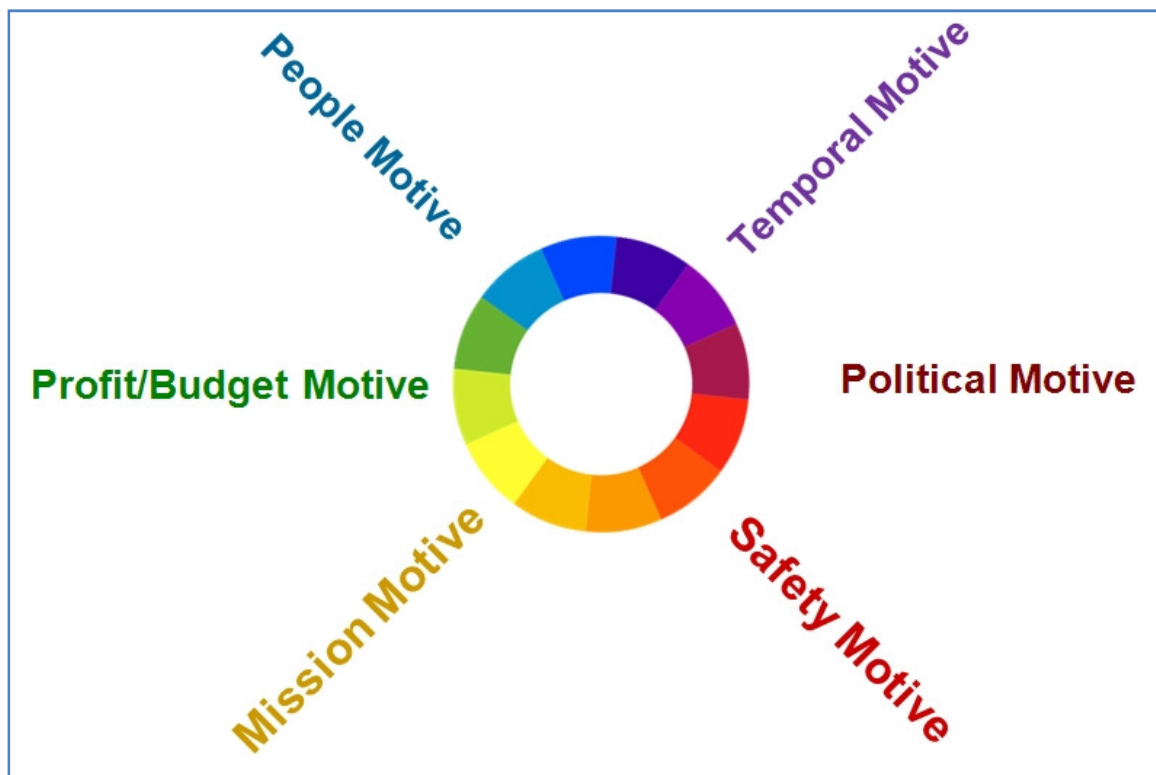


Figure 21 - Law of Competing Motives, Example Color Wheel

Note: The number of competing motives can range anywhere from two to infinity. The competing motives may be modeled in a circular or spherical fashion. Determining which particular motives are most opposite other motives is keystone to properly modeling the process.

Bibliography

- [1] R. Coram, *"Boyd: The Fighter Pilot Who Changed the Art of War."* New York, New York: Back Bay Books/Little, Brown and Company, 2002.
- [2] D. F. Frank. (2002, Jun.) Defense Acquisition University. [Online]. http://www.dau.mil/conferences/presentations/2002/briefings/S1T4TheoreticalConsid_Frank.pdf
- [3] C. H. Hanks, E. I. Axelband, S. Lindsay, M. R. Malik, and B. D. Steele, "Reexamining Military Acquisition Reform: Are We There Yet?," RAND Corporation MG-291, 2005.
- [4] Chairman of the Joint Chiefs of Staff/J-8, Capabilities and Acquisition Division. (2005, Nov.) JCIDS Overview. [Online]. http://www.dau.mil/performance_support/docs/Nov_2005_JCIDS_Overview.ppt
- [5] Defense Acquisition University. (2008) Defense Acquisition Guidebook. [Online]. <https://akss.dau.mil/DAG/>
- [6] Joint Rapid Action Cell. (2005, Jul.) Defense Acquisition University. [Online]. <https://acc.dau.mil/GetAttachment.aspx?id=22117&pname=file&lang=en-US&aid=2111>
- [7] Chairman of the Joint Chiefs of Staff, "RAPID VALIDATION AND RESOURCING OF JOINT URGENT OPERATIONAL NEEDS (JUONS) IN THE YEAR OF EXECUTION," Chairman of the Joint Chiefs of Staff Instruction, 2005.
- [8] J. T. Bennett, "Joint Staff officials will 'tweak' JCIDS to better address urgent needs," *Inside the Air Force*, Feb. 2006.
- [9] Program Manager Magazine, "Colleen A. Preston on Acquisition Reform," *Program Manager Magazine*, vol. 26, no. 1, Jan. 1997.
- [10] A. Boessenkool. (2009, Mar.) DefenseNews.com. [Online]. <http://www.defensenews.com/story.php?i=3975151>
- [11] P. ". Rustan. (2005, Sep.) Congressional Testimony: US Defense Space Acquisition Problems and Potential Solutions. [Online]. <http://www.library.dau.mil/Testimony%20Pedro%20L%20Rustan.pdf>
- [12] G. Grant. (2009, Apr.) www.DoDBuzz.com. [Online]. <http://www.dodbuzz.com/2009/04/10/dod-identifies-key-themes-for-qdr/>
- [13] T. Christie. (2006, Mar.) Military.com. [Online]. <http://www.military.com/forums/0,15240,90349,00.html>
- [14] Rizzo. (2007, Mar.) Air Force Acquisition Officer 101. [Online]. [Formerly available at https://www.safus.hq.af.mil/usamtraining/](https://www.safus.hq.af.mil/usamtraining/).
- [15] D. Best. (2009, Jan.) Best Vantage Inc.. [Online]. <http://www.bestvantageinc.com/bvlibrary.html>
- [16] P. McNamara. (2008, Sep.) PCWorld.com. [Online]. <http://www.pcworld.com/article/151659/>
- [17] PM Magazine, "Acquisition Center of Excellence Will Drive New Capabilities to the Warfighter," *Program Manager*, vol. 31, no. 2 (DAU 167), p. 118, Mar. 2002.
- [18] S. Yegge. (2006, Sep.) Stevie's Blog Rants. [Online]. http://steve-yegge.blogspot.com/2006/09/good-agile-bad-agile_27.html

- [19] I. McFarland and I.). (Pivotal Computer Systems. (2006) Agile Practices on Real-World Projects. [Online].
<http://javamug.org/mainpages/presentations/AgileDevelopmentatGoogle-DallasJUG.pdf>
- [20] P. Hodgetts. (2006, Jul.) The Agile Product Owner And Customer Field Guide. [Online].
http://www.agilelogic.com/files/TU36_TheAgileProductOwnerAndCustomerFieldGuide.pdf
- [21] HashRocket.com. (2009, Jan.) HashRocket =>. [Online].
<http://www.hashrocket.com/>
- [22] R. Baskerville, B. Ramesh, L. Levine, J. Pries-Heje, and S. Slaughter, "Is Internet-Speed Software Development Different?," *IEEE Software*, pp. 70-77, Nov. 2003.
- [23] P. Barry. (2008, Aug.) PaulBarry.com. [Online].
<http://paulbarry.com/articles/2008/08/15/my-guest-appearance-on-a-hashrocket-3-2-1-project>
- [24] D. S. J. Hutchison, "Reinventing IT Test and Evaluation," *Military Information Technology*, vol. 3, no. 13, Apr. 2009.
- [25] S. Adolph. (2006) "What Lessons Can the Agile Community Learn from A Maverick Fighter Pilot?". [Online]. http://www.d-n-i.net/fcs/pdf/adolph_2006_agile%20lessons_final.pdf
- [26] Deputy Under Secretary of Defense, "ACTD Helps Warfighters Get High Altitude, Accurate Parachute Resupplies", " 2007.
- [27] E. M. Roth, et al., "Designing Work-Centered Support for Dynamic Multi-Mission Synchronization," Roth Cognitive Engineering, BBN Technologies, Northrup Grumman IT, C5T Corporation, Air Force Research Lab, 2006.
- [28] E. M. Roth, et al., "Work-Centered Design and Evaluation of a C2 Visualization Aid. Proceedings of the Human Factors and Ergonomics Society 50th Annual Meeting (pp. 255-259).," Human Factors and Ergonomics Society, 2006.
- [29] S. Spielberg and G. Lucas. (1981) Raiders of the Lost Ark. Film.
- [30] G. E. Moore, "Cramming more components onto integrated circuits," *Electronics*, vol. 38, no. 8, Apr. 1965.
- [31] Intel Corporation. (2009, May) Gordon Moore and Moore's Law. [Online].
http://www.intel.com/museum/archives/history_docs/moore.htm
- [32] Wgsimon. (2008, Nov.) Wikipedia. [Online].
<http://commons.wikimedia.org/wiki/User:Wgsimon>
- [33] P. E. Ross, "5 Commandments: The rules engineers live by weren't always set in stone," *IEEE SPECTRUM*, pp. 30-35, Dec. 2003.
- [34] Office of the Department of Defense. (2003) DODAF and JCIDS. PowerPoint Briefings.
- [35] Defense and the National Interest. (2006, Aug.) Defense and the National Interest. [Online]. http://www.d-n-i.net/richards/boyds_ooda_loop.ppt
- [36] J. G. Burton, *The Pentagon Wars: Reformers Challenge the Old Guard*, First Printing ed. Annapolis, USA: Naval Institute Press, 1993.

Vita

Major Matt Larkowski entered the United States Air Force through the Reserve Officer Training Corps program at the Embry-Riddle Aeronautical University, Prescott, Arizona, where earned a B. S. in Electrical Engineering, Mathematics minor, and was commissioned in December 1994. He subsequently earned his Navigator rating and flew the B-52H, as an Electronic Warfare Officer, then earned his Pilot rating and flew the KC-135R/T as an Instructor and Aircraft Commander. He is Level I certified, by the Defense Acquisition University, for Acquisitions Program Management; Test and Evaluation Engineering; and Systems Planning, Research, Development, and Engineering (SPRDE-SE). Matt also taught introductory and advanced computer programming as an associate professor of computer science at the North Dakota State College of Science.

Major Larkowski has served in operational and staff positions at the squadron, group, deployed, and major command levels. His duties have included Flight Commander; Chief of Squadron Scheduling; Deputy Chief of Group Mobility Operations; Deployed Assistant Director of Operations; Global Operations Director and Command & Control Systems Coordinator, Tanker Airlift Control Center; and Requirements Development for the KC-X Tanker and for Mission Planning Systems Branches of the HQ AMC Strategic Plans, Requirements, and Programs Division. He has over 3000 military and civilian flying hours — including 300+ combat hours on 50+ sorties in Iraq and Afghanistan.

In 2008 Matt was selected to attend AFIT and is currently completing the Cyber Warfare Intermediate Developmental Education program. Upon graduation he will attend the Defense Advanced Research Projects Agency (DARPA) Service Chiefs' Air Force Fellowship Program.

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 074-0188	
The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to an penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.					
1. REPORT DATE (DD-MM-YYYY) 18-06-2009		2. REPORT TYPE Graduate Research Paper		3. DATES COVERED (From – To) May 2008 – June 2009	
4. TITLE AND SUBTITLE The Cyberspace Development Dogfight: Tightening the Acquisitions Turn Circle				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) Matthew P. Larkowski, Maj, USAF				5d. PROJECT NUMBER ENS 09-153	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way WPAFB OH 45433-7765				8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/ICW/ENG/09-03	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Headquarters United States Air Force, Office of the Vice Chief of Staff, Quadrennial Defense Review Section Attn: Maj William F. Dobbs 1670 Air Force Pentagon Washington DC 20330 DSN: 425-8586 e-mail: william.dobbs@pentagon.af.mil				10. SPONSOR/MONITOR'S ACRONYM(S) HAF/CVAQ	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION IS UNLIMITED.					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT The purpose of this research was to assess the ability for DoD software development to keep up with the increasing rate of technological change, then propose avenues for improvement. Specifically, this research attempts to answer fundamental questions based on the concerns for the 2010 Quadrennial Defense Review. In general, how do we adapt software acquisitions strategy to cope with the increasing rate of technological change? The following conclusions were reached: (a) software projects must be scoped and scheduled for development cycles on the order of months, not years, and use open architecture, Agile Development methods, and scalable designs with modular code; (b) budgets must be stabilized for long-term integrity, with a software development working capital fund reserved for JUONS-like urgent IT needs; (c) increased use of MAJCOM- or AOC-level business centers must be encouraged and funded to produce tailored software modules that interface with larger agile programs built to accept these modules; (d) we must take advantage of ATCD and ATD efforts from research laboratories, giving MAJCOM and AOC business centers budget authority to “pull” a limited amount of ATDs, ACTDs, and JACTDs from the labs, through the appropriate System Program Office, to produce and field operational software (by default, not by exception); and (e) periodic software development working groups and conferences should be continued, but with emphasis on standardization and sharing of lessons learned between services, MAJCOMs, and AOCs.					
15. SUBJECT TERMS Software Development, Acquisitions Reform, JCIDS, JACTD, ACTD, ATD, JUONS, JUON, Agile Acquisitions, Evolutionary Acquisitions, Thin Client, Rapid Prototyping, Boyd, Burton, Moore's Law, Rock's Law, Larkowski, Law of Competing Motives					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT UU	18. NUMBER OF PAGES 83	19a. NAME OF RESPONSIBLE PERSON Robert F. Mills, PhD (ENG)
REPORT U	ABSTRACT U	c. THIS PAGE U			19b. TELEPHONE NUMBER (Include area code) (937) 257-3636, ext 4527; e-mail: robert.mills@afit.edu

Standard Form 298 (Rev: 8-98)

Prescribed by ANSI Std. Z39-18