U.S. DoD vs. Amazon—18 Architectural Principles to Build Fighter Jets Like Amazon Web Services Using DevOps

Lean and agile thinking, values, principles, practices, tools, and technologies are now a global phenomenon. Originally conceived in the 1940s and 1950s, lean and agile thinking was directly injected into Western firms in the 1980s and 1990s. Lean and agile principles gained a strong foothold as a paradigm for managing the development of information systems around 2000. Although, these principles originated in manufacturing, many falsely believe they originated from computer programmers who were too lazy to use traditional project management, systems engineering, and software engineering principles. At first, computer programmers applied these principles to small teams. Eventually, they began scaling these principles upwards to projects, programs, portfolios, and the enterprises themselves (*including government agencies*). Western philosophers also began transferring these practices from manufacturing to administrative business functions as well. These included strategic planning, organizational change, marketing, legal functions, finance, accounting, contracts, and many others. Between the manufacturing community and the computer programmers, business enterprises and government agencies were getting squeezed in from all sides. Today, many of these initiatives are known as lean enterprises, business agility, just-in-time supply chain management, etc. Information systems initiatives come with monikers like "*Continuous Integration*," "*Continuous Delivery*," and "*Development Operations*" (*DevOps*).

A key and often forgotten goal of lean and agile principles is the notion of speed, cycle time, time-to-market, and competing on Internet time (*from the information systems community*). Management scientists began studying the fundamental differences between Eastern (*Japanese*) and Western (*American and European*) styles of manufacturing. One of their discoveries was that Eastern manufacturers were about two to three times faster than Western firms. That is, the Japanese could bring a new product from concept (*drawing board*) to showroom (*salesfloor*) twice as fast as Americans and three times faster than Western Europeans. This discovery from the early 1990s set off a firestorm of new textbooks, management fads, and cottage industries of management consulting tools to help Western firms compete or speed up their cycle times. Even the U.S. government became infected with the notion of becoming "*better, faster, and cheaper*!" However, as Newton discovered in the 1600s, "*every action has an equal and opposite reaction*!" (*fierce, elongated resistance*). It takes an idea about 40 years to go from concept to general practice, and traditional project management, systems engineering, and software engineering were in their golden age during the 1990s. That is, they were about in their 40th year, and in the vernacular of Everett Roger's and Geoffrey Moore's five stage model (*innovators, early adopters, early majority, late majority, and laggards*), traditional paradigms were in their last two stages.

It's rather difficult to speed up a traditional manufacturing firm and detractors of lean and agile principles just weren't seeing the payoff of this latest management fad. As alluded to earlier, the lean and agile paradigm constituted a large organizational change requiring at least 40 years to take hold. It's kind of hard to say when the clock exactly started, but let's say 1985 or 1990 in the Western Hemisphere. Elisabeth Kubler-Ross created her five-stage model in 1969 (*denial, anger, bargaining, depression, and acceptance*) and Virginia Satir documented her seven-stage model in 1990 (*old status quo, foreign element, resistance, chaos, transforming idea, integration, and new status quo*). Together with Everett Roger's model, these models show that when an "*old idea*" is in its golden age (*early and late majorities*), any "*new idea*" will meet with fierce resistance and chaos over an extended period of time. Geoffrey Moore penned a bestselling book in 1990 called, "*Crossing the Chasm*" to describe the period between Roger's "*Early Adopters*" and "*Early Majority*." Traditional paradigms are fraught with pitfalls including predictive methods (*pushing out unwanted products and services on the unsuspecting masses*), large batches (*complex product and service architectures requiring millions of dollars of upfront investments and decades of production time*), and complex production systems consisting of millions of dollars in capital investments (*in buildings, manufacturing equipment, processes, documentation, people, etc*.).

There's actually a lean and agile term for the bureaucratic morass associated with traditional paradigms and it's called "*Work in Process* (*WIP*)" or "*Waste*." That is, the larger the problem, product, service, architecture, or solution, the more by-product of waste it requires to develop it. This is what Eiiji Toyoda (*Toyota*) saw when he visited Ford's factories in Dearborn, Michigan in the early 1950s (i.e., *large piles of waste resulting as a byproduct of a few automobiles*). Toyoda went on to create the infamous "*Toyota Production System*" (*TPS*) from which contemporary lean and agile principles sprung forth. But, Toyoda wasn't the first, as Ronald Coase called these "*transaction costs*" in his infamous 1937 textbook called "*The Nature of the Firm*" (*for which he was awarded a Nobel Prize*). Ronald Coase, a British economist with a bent towards Marxism, came to America to study the success of its economy. As a result, he discovered that numerous "*transactions costs*" arose as a result of the simple exchange of goods and services, which created jobs (*and sort of had a spillover or trickle-down economic effect spurring on more economic activity*). So, what have we learned from this minor history lesson? We've learned that large product and service architectures or batches create waste, WIP, and transaction costs that create jobs and spur on economic activity (*traditional methods*). However, the downside is that productivity, quality, cycle time, and competitiveness greatly suffer under these constraints in the 21st century.

Today's global economy presents new challenges for traditional paradigms. Four of the world's seven billion inhabitants are interconnected on the Web. There are five billion active mobile phone subscriptions generating trillions of communication packets of information. Everyone on the planet is considered a customer, competitor, or a dangerous adversary. Information is exchanged between the world's inhabitants at the speed-of-light. Fast cycle times and extremely small batches (*resembling network packets*) are required to do business in today's marketplace. Suppliers must be able to develop and deliver ultra-reliable products and services to billions of users in fractions of a second. Furthermore, they must recall and reset previous product and service baselines if they make a mistake (*also in fractions of a second to billions of global devices using the Internet*). Most important of all, enterprises must be able to effect large-scale organizational and global change without incurring the debilitating effects of resistance-to-change described by Everett Rogers, Elisabeth Kubler-Ross, Virginia Satir, and Geoffrey Moore. In 1965, Gordon Moore of Intel predicted a semiconductor would halve in size and double in speed every 18 months. Conversely, in 1999, Ray Kurzweil predicted the rate of change was exponential and would become fractional in the early 21st century (*Singularity*). What's the bottom line? The time is up for traditional, large-batch paradigms, and manual lean and agile paradigms with marginal gains.

A handful of Internet giants rose to the challenge of the bold new 21st century global landscape (*Amazon, Google, Yahoo, EBay, Facebook, Apple, Microsoft, Samsung, etc*.). Early predictions were that only these handful of firms had the vision, insight, technologies, and resources to do business on the Internet (*and all other should simply stay away*). In other words, these firms would monopolize and dominate the global Internet landscape to the peril of everyone else. Perhaps, this was just another simplification of the change models which held that large scale change was simply impossible for the masses (*remember it takes about 40 years for an idea to go from concept to mass public acceptance*). It's not surprising that it was information systems firms that jumped out quickly on the 21st century electronic marketplace. Even manufacturing proponents realized that firms needed not only lean and agile principles, but flexible product technologies as well to speed up production to unprecedented levels. By 2010 Amazon and Google were producing hundreds of thousands of NEW value-adding, ultrareliable products and services per day (*and pushing them out to billions of Internet users in fractions of a second*). Furthermore, even traditional manufacturing equipment, medical devices, airplanes, spacecraft, computers, mobile phones, etc. were basically personal computers with loads of software (*that could be redesigned, validated, and loaded on millions of Internetworked devices in fractions of a second*).

Some universities pushed their curriculums onto the Internet and few retailers pushed their products into electronic commerce websites. By 2006, the Internet 500 (i.e., *electronic commerce firms*) were outperforming Fortune 500 brick-and-mortar firms in terms of sales, revenues, profits, and growth. In spite of these early gains, Harvard economist Michael Porter warned traditional firms to "*stay away*" from the Internet in his 2001 article, "*Strategy and the Internet*." The technical terms for "*fear of change*" are "*metathesiophobia*, *xenophobia, technophobia, or Ludditism*" and Porter warned that competing on speed and price would undermine value. The biggest surprise of the new era was "*Open Source Software*!" Thousands of programmers created billions of lines of ultra-reliable software in fractions of a second and gave it away for free. This phenomenon was based on yet another theory of change by Dan Pink called "*Drive*," which held that the ultimate human motivator was enjoyment (*vs. profit*). That is, world wide programmers were creating software for the "*pure fun of it*!" So, part of the secret sauce lay in creating software (*vs. hardware*), making small incremental changes, using lean and agile principles, and being fast. However, in the vernacular of Everett Rogers, a few die-hard "*laggards*" insisted on large capital-intensive big-box footprints, traditional paradigms, large product architectures and batches, hardware-intensive designs, voluminous job-creating waste, and multi-decade long projects.

Okay, let's examine a few of these so-called lean and agile principles in greater detail. There are almost as many variations of lean principles as there are management consultants. According to James Womack, the five major principles of lean thinking are "*identify value, map the value stream, create flow, establish pull, and seek perfection*." Although fairly ethereal in-nature, from these we get that speed and quality are important, but Womack says nothing about scaling, size, and technology. In fact, his theory was based on incremental change of traditional brick-and-mortar manufacturers and did not have Amazon or Google in-mind. Don Reinertsen (*a specialist in manufacturing mathematics such as "queuing theory"*) defined eight slightly more rigorous (*mathematical*) lean principles, "*economic view, manage queues, exploit variability, reduce batch size, apply WIP constraints, control flow, fast feedback, and decentralize control*." Now, the picture comes into slightly better focus, with an emphasis on smaller queues (*simplicity vs. complexity*), smaller batches (*simpler designs*), WIP constraints (*less waste and bureaucracy*), and flow (*increased time-to-market*). Lean principles differentiate between "*lead time*" and "*cycle time*," where the former clock begins when the customer or market requirement is identified and the latter when product development begins. What's the bottom line? Lean principles "*clearly*" place time-to-market, cycle-time, speed, and productivity at center stage (*it's always been about speed*).

Lean and agile principles are "*kissing cousins*" (*more like the Hatfields and McCoys*). Whereas lean thinking is deeply rooted in rigorous (*mathematical*) manufacturing theory (*benefits of small batches*), agile principles are rooted in evolutionary biology, organismic biology, systems theory, systems dynamics, chaos theory, adaptive theory, systems thinking, adaptive organizations, emergence, etc. To quote Charles Darwin (*1809*), "*It is not the strongest of the species that survives, nor the most intelligent, but the one most responsive to change*." Numerous theories of business, manufacturing, and information systems agility were created since the 1990s. The "*Agile Manifesto*" from information systems is certainly the most memorable, which defines agility in terms of four broad values, "*individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan*." From these we learn teamwork is important (*two heads are better than one*), the end-product is critical (*vs. WIP, waste, and bureaucracy*), talking with customers face-to-face is necessary (*knowledge is implicit vs. explicit*), and continuous improvement is absolutely necessary (*on a daily basis*). Many argue that "*incrementalism*" is the key to information systems agility (*which supports notions of flow, cycle time, and speed*), while dismissing a perhaps more important principle of "*daily process improvement*" to the detriment of this community.

A key differentiator of agility was a relentless focus on the end-product (*vs. process WIP*). Whereas lean principles began with the current organizational WIP legacy (*value stream mapping*) and then incrementally chipped away at it over a period of years, agility began with no WIP at all. Agility quickly drew sharp criticism from every corner. Traditionalists grew to value job-creating WIP at the expense of market competitiveness, while the lean community grew to value the architecture of 20th century brick-and-mortar organizations (*including their WIP*). In other words, the lean community struggled to save 20th century manufacturing firms by incrementally improving their efficiency, while agilists strove to replace them all together in one-fell swoop with 21st century notions of information systems speed and agility. Agilists did not value one iota of WIP in the value stream and strove instead to completely automate it away. The earliest attempt to eliminate WIP came in the form of automated unit testing. This quickly grew into notions of Continuous Integration and Behavior Driven Development (*completely automated testing*). Remember, during the heyday of Microsoft in the 1990s when Windows 95 hit the mainstream, 80% of Microsoft's 40,000 developers were testers (*non-value adding WIP, waste, etc*.)? Queuing theory demonstrates that inordinate complexity, WIP, and waste causes infinite delays and extremely poor quality, which was certainly the case in terms of Microsoft Windows (*and most of its other product portfolios*).

Once again, agility came under attack from the lean world for sub-optimizing information systems (*Empire Strikes Back*), while ignoring the larger brick-and-mortar organization (*which they sought to incrementally improve*). It is this criticism that betrayed their lack of vision, because they sought to resurrect 20th century manufacturing (*much like Michael Porter*), while agilists (*Rebels*) were busy trailblazing a bold new vision of 21st century electronic businesses. However, agilists continued their vision of automating the rest of the enterprise. This came in the form of "*Continuous Delivery*" (*2010*) and "*Development Operations*" or DevOps for short (*2012*). Whereas Continuous Integration sought to automate testing, Continuous Delivery aimed to automate the process of completely building and delivering products to deployment. It's a subtle and misunderstood variation of Continuous Integration. The breakthrough came with DevOps, which automated the process of delivering the final product created by Continuous Delivery to the customers themselves (*all four or five billion of them instantaneously if necessary*). If you're Amazon or Google, one merely has to update the servers with the latest release. If one is Apple or Samsung, one has to update four or five billion mobile phones in fractions of a second. If you remember, 80% of Microsoft's assets were tied up in non-value added testing WIP and waste. Continuous Delivery and DevOps completely free up these resources to capitalize on new markets (*zero WIP*).

A key contributor to unraveling the mystery of 21st century electronic firms is certainly Martin Fowler. He helped create "*Extreme Programming*" (*XP*), a blockbuster new agile method from the 1990s. He was one of the 14 original apostles who created the "*Agile Manifesto*" at Snowbird, Utah in 2001. He also pioneered "*Automated JUnit Testing*," "*Test Driven Development*" (*TDD*), "*Continuous Integration*" (*CI*), and "*Refactoring*" (*eliminating 80% to 90% of code waste or "dead code"*). However, perhaps, his greatest contributions, if these weren't great enough already, was the notion of evolutionary, emergent, continuous, collaborative, just-enough, and lean architecture and design principles (*which would prove to be the keys to global enterprise agility*). You see, the cornerstone of 20th century systems and software engineering was a multi-million dollar, multi-year investment in large enterprise and product architectures to last decades (*WIP, waste, etc*.). Conversely, Fowler believed that large upfront, extended investments in "*Big Design Up Front*" (*BDUF*) were not necessary at all. In fact, he believed that system architectures and designs should "*emerge*" from iteration to iteration, sprint to sprint, and requirement to requirement. Given that Fowler was a computer programmer (*not an armchair quarterback*), all of his concepts were firmly grounded in principles of applied computer science. The breakthrough came in 2004, when he devised perhaps his greatest creation of all, "*The Strangler Application*!"

Why monolithic, large, and big design up front (*BDUFs*) became so pervasive early 21st century is a bit unclear? Perhaps, it was the maturity of traditional paradigms that had been around for at least 40 years and penetrated the psyches, consciousness, and psychological culture of the late majority of practitioners (*not to mention Carnegie Mellon University's Software Engineering Institute and the Systems and Software Productivity Consortium had been pushing it like gangbusters for nearly two decades with its multi-billion dollar endowments from the U.S. DoD*)? Furthermore, John Zachman of IBM created a framework for "*Enterprise Architecture*" espoused by the early majority in the mid-2000s as well (*and U.S. DoD agencies were investing billions of dollars in their own Zachman variation called the "DoD Architecture Framework"*). Object oriented principles were pioneered in the 1950s, object oriented computer programming languages were created in the 1960s, and "*modularized*" software became a foundational computer science principle by 1972. All of these paradigms were based upon decomposing monolithic computer programs into smaller, more manageable modules. Principles of "*high-cohesion*" and "*loose-coupling*" undergirded the notion of object-oriented design and were theorized to be a cure for the "*software crisis*" of the 1960s (*free up the morass and low productivity caused by frozen queues when creating monolithic computer programs so the U.S. DoD could create new weapon systems more quickly*).

While lean and agile principles sought to free up organizational and process queues caused by excessive WIP, systems and software engineering organizations continued investing in large product and service architectures (*BDUF*). It was as though the lessons learned from the 1950s to the 1980s concerning modularization, object-oriented principles, high-cohesion, and loose coupling had never occurred at all. It was a clash of value systems as organizations sought to be lean and agile, while freezing their organizational queues with BDUF-based architectures. It was like running a marathon with an aircraft carrier strapped to your back. Finally, organizations sought to disassemble their monolithic architectures into smaller and more manageable pieces once again. Notions of Web Services, Service Oriented Architectures (*SOAs*), Enterprise Application Integration (*EAI*), Enterprise Service Buses (*ESBs*), and Asynchronous Messaging Services (*AMSs*) came into being. They were great ideas for creating asynchronously cooperating application services, but did not solve the problem of tight-coupling (*and only exacerbated the fallacy of investing in monolithic enterprise architectures that froze productivity queues*). In some cases, organizations applied lean and agile principles to create systems and software architectures one requirement and one iteration or sprint at a time until their designs became so large all productivity came to a screeching halt due to the immense complexity of their architectures.

In a last-ditch effort to free up frozen organizational productivity queues resulting from BDUF, as well as evolutionary architecture and design, organizations began to untangle, break apart, and decompose their monolithic product and service architectures into small modules. That is, for the first time in nearly 40 years, organizations finally seemed to understand the value of high-cohesion and (*completely*) loose coupling. Using "*Refactoring*" principles, organizations such as Amazon, Google, Yahoo, Facebook, Etsy, Github, and Blackboard "*reexamined*" their legacy code bases and carved them up into completely independent (*loosely coupled*) modules. Furthermore, lean and agile proponents advocated the use of top-down concepts such as "*Minimum Marketable Features*" (*MMFs*), "*Minimum Viable Products*" (*MVPs*), "*Minimum Viable Architectures*" (*MVAs*), "*Story Maps*," "*Impact Maps*," and "*Lean Canvasses*" to create highly-cohesive and loosely-coupled (*modularized*) product and service architectures from the "*get-go*." However, the biggest breakthrough probably came from "*Microservices*" (*Containerization*), which were not unlike their earlier lean and agile cousins called MMFs, MVPs, MVAs (*except Microservices were even "smaller" and more "microscopically-scoped"*). Microservices were about 1,000 lines of code, could be created by a couple of programmers in one or two days, and could be grouped into highly-cohesive ecosystems that performed functions of larger architectures using "*Continuous Design*."

Furthermore, the entire end-to-end process of building, testing, deploying, and delivering Microservices from the fingertips of programmers to billions of global users in fractions of a second could be completely automated using DevOps principles. It wasn't unusual for U.S. DoD fighter jet simulators to take up to three decades to create one release. Fighter jet mission computer software also took two or three decades to create basic fly-by-wire skeletons that would take decades more to mature and stabilize. The average length of a U.S. DoD information system project took 81 months (*going downhill with the wind at its back*). These projects were anything but lean and agile, certainly weren't very speedy, and were chock full of waste and monolithic queue freezing BDUF-based architectures that bogged them down in bureaucracy. Ironically, Frederick Brooks compared computer programming projects to dinosaurs inexorably trapped in tar pits in his 1975 blockbuster book entitled, "*The Mythical Man Month*!" Basic agile projects moved about 10 times faster than traditional ones when starting from the ground up. Lean projects moved about 10% to 30% faster than agile ones using Kanban principles. Projects using "*Continuous Integration*" moved 100 times faster than their traditional counterparts and those using DevOps moved thousands of times faster than the average U.S. DoD projects (*especially when they applied* "*highly-cohesive*" *and* "*loosely-coupled*" *architectures along with end-to-end DevOps automation*).

Okay, so now we've solved the mystery of organizational, portfolio, program, project, and team success, right? The trick is to create highly-cohesive and loosely-coupled product and service architectures from the top-down using Microservices or bottom-up using the Strangler Application? Furthermore, lean and agile principles must be applied to manage queues, limit WIP, evolve designs, and automate the testing, deployment, and delivery process? All of these principles should be applied to eliminate what Edward Yourdon called the "*Death March*" caused by immense complexity, over-scoped and under-resourced computer projects, and other unrealistic constraints? In fact, Amazon delivers over 136,000 fully-tested Microservices to billions of users each day using these principles? Google performs hundreds of millions of automated tests each day to ensure its Microservices are delivered to billions of users each day? Best-in-class information systems firms such as Facebook, Github, Etsy, Blackboard, and many others deliver hundreds of fully tested Microservices to millions and billions of users each day as well? Okay, so all is well in Camelot, right? "*What does this have to do with the price of tea in China*" (*U.S. DoD*)? Fighter jets, air craft carriers, nuclear attack submarines, intercontinental ballistic missiles, and satellite constellations still take decades and trillions of dollars to produce! How can one compare Amazon to U.S. DoD weapon systems? Well, let's damn the torpedoes and give it a try for God's sake!

- AVOID CONWAY'S LAW. In 1968, Dr. Melvin E. Conway, published a paper entitled, "*How Committees Invent*?" The thesis or essence of his theory became known as "*Conway's Law*," quoted in its original form here: "*Organizations which design systems are constrained to produce designs, which are copies of the communication structures of these organizations*." Translated into English, Conway's Law basically means "*people will create a new system based upon the materials they have at-hand, rather than conceiving a superior design using materials they do not have*." Think of a cook who simply makes dinner from the ingredients in the pantry and the pots and pans on-hand, rather than shopping for new food supplies and cooking instruments to try a bold new recipe. Conway's article was initially submitted to and rejected by the Harvard Business Review (*an opinion magazine*) for lack of scientific rigor? Datamation picked up the article, and the rest as they say is history. Conway was merely documenting an instinctual human malady known as cognitive blindness (*a common decision-making error*). Information systems designers immediately began automating the functions of their existing bureaucracies (*including Coasian transaction costs*), instead of replacing them with bold new structures (*as Amazon has done for the last two decades*). **Bottom line**: *The U.S. DoD needs to cure its acute case of "Conway's Law," stop codifying its degenerate acquisition, systems engineering, and product architecture practices using modern information technologies, and begin conceiving bold new structures as Amazon has done* (*remember Michael Hammer's old adage, "Obliterate it, don't automate it"—Don't just automate the bureaucracy*).
- THINK OUT-OF-THE-BOX. A closely-interrelated concept to Conway's Law is thinking-out-of-the-box. Humans, especially the U.S. DoD are subject to groupthink, which is "*a psychological phenomenon that occurs within a group of people in which the desire for harmony or conformity in the group results in an irrational or dysfunctional decision-making outcome*" (*Wikipedia*). American firms have been subject to groupthink for decades and groupthink is a primary contributor to the decline of American manufacturing industries from 1945 to the present time (*while the Pacific Rim continues to reach new heights well into the 21st century*). Historically, the executives of Fortune 500 firms were all the same race, color, ethnicity, language, gender, religion, education, experience, and mindset. This led to a variety of decision-making errors known as the "*Four Villains of Decision-Making*" (i.e., *narrow framing, confirmation bias, short-term emotion, and overconfidence*). The U.S. DoD, in particular, is guilty, not only of groupthink, cognitive blindness, and the so-called Four Villains of Decision-Making, but simply cannot think-out-of-the-box when it comes to decision-making. They've been making ships, tanks, aircraft, missiles, artillery, guns, bombs, and bullets for so long, they don't know how to conceive, design, develop, or do anything else. If a product or service architecture doesn't fall into this pattern, they don't want anything to do with it. **Bottom line**: *The U.S. DoD needs to bring fresh new blood into the decision-making process, engage international partners, and start thinking-out-of-the-box to create bold new structures as Amazon has done (Amazon and Google use UAVs to deliver packages and provide global Internet service)*.
- TIGHTLY-SCOPED, SINGLE PURPOSE PRODUCT AND SERVICE ARCHITECTURES. While lean and agile proponents believed their methods were good enough to tackle any size of problem, they quickly learned that the key to success was tightly-scoped product and service architectures. This gave rise to a myriad of ideas such as "*Minimum Marketable Feature*" (*MMF*), "*Minimum Viable Product*" (*MVP*), "*Minimum Viable Architecture*" (*MVA*), "*Lean Canvas*," "*Story Maps*," "*Impact Maps*," etc. They were not unlike the "*Metaphors*" of early agile methods (*Extreme Programming*). Rather than create a never-ending systems architecture loaded with a multi-million dollar, multi-year upfront investment, the key was to identify a single vertical functional requirement the system must perform on a near-term schedule with measurable properties. These included market or mission value, end-user functional needs, and key non-functional needs like volume, capacity, speed, reliability, security, usability, etc. All-in-all, the scope could be completed in three, six, or nine months by a small team of cracker-jack engineers, and the end-user, customer, and supplier could reap some near-term rewards without breaking the bank, over-scoping the project with unrealistic risks, and, of course, technological obsolescence. BDUF-based product architectures were a by-product of traditional project and systems engineering paradigms and were not a problem during the early jet age. Skunkworks aircraft manufacturers created enduring platforms such as U-2, SR-71, F-117, and many others using tightly-scoped designs. **Bottom Line**: *The U.S. DoD needs to go*

back to creating tightly-scoped, single-purpose designs instead of gold-fleecing acquisitions with job-creating WIP and waste.

- USE SOFTWARE VS. HARDWARE. Since the inception of the electronic computer in 1946 (*ENIAC*), the U.S. DoD has recognized the strategic importance of computer programming as a means of harnessing this technology for the creation of bold new 20th and 21st century weapon systems. This gave rise to the development of human readable computer programming languages at a feverish pace, not unlike the Manhattan Project that led to the creation of the world's first atomic bomb (*signaling the end of World War II*). These included Formula Translation (*FORTRAN*), Common Business Oriented Language (*COBOL*), and literally hundreds of computer programming languages throughout the 1950s and 1960s. With the advent and stabilization of mainframe computer technologies in the 1950s and 1960s, these gave rise to the creation of large computer programming projects with infinite complexity, extremely low productivity and quality, and, inevitably "*frozen queues*!" In 1968, at a North Atlantic Treaty Organization (*NATO*) conference, two major terms emerged, "*software crisis*" to characterize the strategic nature of this unmanageable new technology and "*software engineering*" to describe the solution to slaying the "*software werewolf*!" Even Harvard (*Stephen F. Thomke*) begged U.S. manufacturers to compete on software vs. hardware in his 2003 classic, "*Experimentation Matters*!" The 1968 NATO mandate was "*crystal clear*" to all those who paid attention, "*win the Cold War with software technology and use software engineering principles to construct reusable software modules in order to do it*!" **Bottom line**: *The U.S. DoD needs to go back to fundamentals and start winning 21st century battles with software vs. hardware*.
- HIGH-COHESION AND LOOSE-COUPLING. The notion of high-cohesion and loose-coupling also goes back to the advent of software engineering in 1968. (*when "modularization" and "software reuse" were also coined as underpinning principles to the newfound software engineering discipline*). Prior to the popularity of object-oriented analysis and design, concepts such as "*structured analysis*" and "*structured design*" emerged from the minds of Larry Constantine, Edsger Dijkstra, David Lorge Parnas, and many others. Cohesion refers to the degree to which two software modules are related (i.e., *they should serve one another*). In other words, software developers should create closely-related families of software modules. Loose-coupling refers to the degree to which to software modules are architecturally independent or separated from one another (i.e., *they should not be physically connected*). The NATO visionaries were using the advent of integrated circuits and electronic components as a metaphor for software development (i.e., *computer programmers should create a large supply of reusable software components one could order from a catalog to rapidly compose new weapon systems*). The term "*object*" was actually coined in the 1950s and the first object-oriented languages were created by 1960 (*LISP, Simula, etc*.). However, it wasn't until about 1980 that the DoD decided to go all-in on creating their own unified computer programming language called "*Ada*" (*and these two principles were embodied and combined into an object-oriented programming principle*). **Bottom Line**: *The U.S. DoD needs to realize the bold new vision of the 1968 NATO conference of creating highly-cohesive and loosely-coupled software components*.
- 80% TO 90% OF FUNCTIONS IN SOFTWARE. A subtle shift in technologies occurred by the late 20th century. This was caused by the creation of the F-22 and F-35 fighter jets. Engineers noted that 80% to 90% of the functional systems requirements were allocated to software vs. hardware. Their software was on the order of 10 million lines of code and was attributed to most of the cost, schedule, and technical performance resources and risk. Prior generations of fighter jets such as F-4, F-14, F-15, F-16, and F-18 paled in comparison (*in spite of their previously unprecedented volumes of software*). Surprisingly, the F-22 and F-35 relied more on hardware than alternative software-intensive designs that would have optimized fuel-efficiency, radar-cross section, and overall mission efficiency and effectiveness. However, the U.S. DoD chose more conventional hardware-intensive designs, because they realized their cost, risk, and ability to productively realize weapon systems software functions had not improved since 1968. Weapon systems integrators turned to lean and agile principles to help make up lost time in order to increase productivity and accelerate lost cycle time. However, it was too little, too late, because the U.S. DoD was locked into Cold War era hardware-intensive fighter jet designs and manufacturers (*while rejecting firms with greater software prowess*). This caused the cost of the F-22 and F-35 to reach $400 billion by 2010 and their final estimated cost to be on the order of $2 trillion. **Bottom Line**: *The U.S. DoD needs to consider alternative weapon systems, software-intensive systems, and modern software development methodologies instead of building $20 trillion hardware-intensive fighter jets in the 21st century*.
- PREPARE FOR NEXT GENERATION FIGHTERS WITH VOLUMINOUS SOFTWARE. Visionaries now realize that next-generation U.S. DoD fighter jets beyond F-22 and F-35 would require 10 times more software than their 20th century counterparts (*perhaps 100 million lines of code to fulfill their functional requirements*). Therefore, next generation fighters could be on the order of $20 trillion vs. $2 trillion for F-22 and F-35, while the GDP of the U.S. economy is projected to decrease over the same period of time. Even with marginal gains in productivity and quality associated with the application of 20th century lean and agile principles, 21st century fighter jet mission computers and avionics are not possible with today's technologies (*or are they*?). Once again, the U.S. DoD's programming productivity has not increased since the 1960s and 1970s. Also, remember that Gordon Moore claimed technologies would be obsolete in 18 months, while Ray Kurzweil said they would be obsolete in seconds by the early 21st century. So, any attempt at a next-generation fighter-jet with 100 million lines of code would simply not be possible even with basic lean and agile principles (*and certainly not with traditional 20th century project management and systems engineering paradigms*). However, while the U.S. DoD languished in 1960s era mainframe programming methods, the productivity of U.S. Internet firms skyrocketed to at least 2,000 times above the U.S. DoD average by 2010 using Internet-era principles for creating software-intensive systems. **Bottom Line**: *The U.S. DoD needs to abandon mainframe era computer programming principles characterized by Tayloristic capability maturity models and embrace Internet-era paradigms*.
- DO AWAY WITH BIG-UP-FRONT ARCHITECTURES AND DESIGNS. Simply do away and dispense with creating multi-billion dollar weapon system architectures to last 100 years. This is actually a pervasive myth of 20th century systems and software engineering (i.e., *weapons systems must be infinitely complex, reliable, maintainable, and expensive*). This was the reason that traditional project management, systems engineering, and software engineering disciplines were created in the 1950s. The electronic computer gave rise to the myth of the complex system requiring whole new disciplines of administrative and technical management. This was merely a misplaced application of Tayloristic Scientific Management Principles, the American System of Manufacture, Fordism, and Mass Production. In other words, wasteful, big-batch designs that Eiiji Toyoda disassembled in the 1940s. Ironically, U.S. DoD planners extended outdated manufacturing principles from the 19th century into the 20th century jet age (*while Japan pioneered lean and agile thinking*). Of course, Japan had unique constraints the U.S. did not (i.e., *no natural resources to waste and a culture of perfecting foreign ideas*). Eventually U.S. mathematicians decoded the mathematics of Japanese manufacturing principles (i.e., *queueing theory shows that system complexity freezes queues, slows production, and*

decreases quality). However, it was the U.S. information systems industry that discovered the ultimate unit of design was no more than a few lines of code that could be designed, tested, and globally deployed to billions of end-user devices in fractions of a second. **Bottom Line**: *The U.S. DoD must dispense with big up front multi-billion dollar designs to last 100 years*.

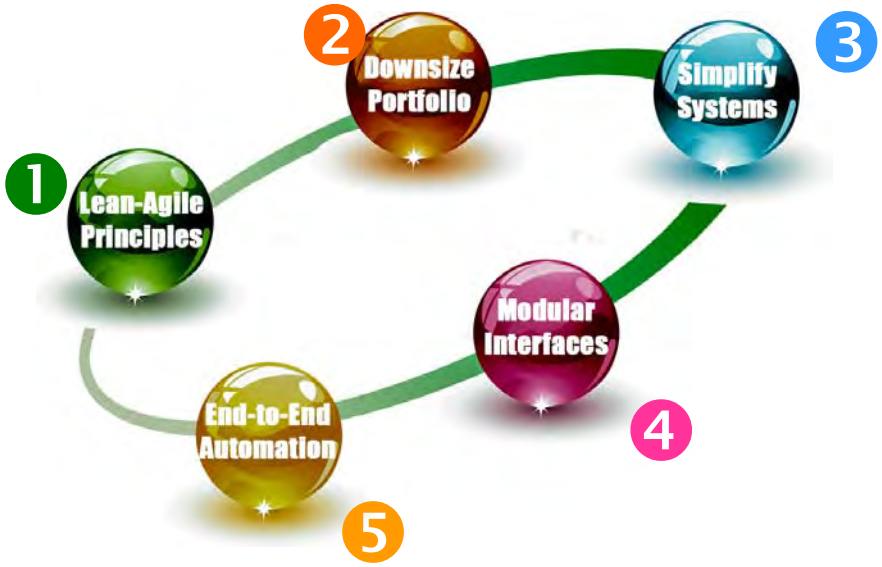
- SOFTWARE-INTENSIVE MICROSERVICE-BASED WEAPONS SYSTEMS. So, how exactly do U.S. Internet firms reach productivity levels at 2,000 times that of hardware-intensive U.S. DoD weapons systems? They do so through a triad of software-intensive products and services, lean and agile principles powered by automation-intensive DevOps, and, most importantly of all, by creating highly-cohesive and loosely-coupled ecosystems of (*Containerized*) Microservices. What? That is, thousands of programmers create 1,000 line-of-code Microservices that perform useful, value adding functions one-at-a-time using lean and agile incrementalism, and push them out independently of one another to billions of global devices using automation-intensive DevOps pipelines. In doing so, they can push out hundreds of thousands of ultrareliable Microservices to billions of end-users at the speed-of-light, recoup bureaucratic Coasian transaction costs associated with traditional project management, systems engineering, and software engineering mainframe paradigms from the 1960s, and reallocate these previously wasted resources to capturing new markets, products, services, revenues, sales, profits, customer trust, and loyalty. Furthermore, they do this with a smaller, more talented scientific workforce that enjoys their jobs and are willing to do this for the pure excitement of doing so. This has multiple implications for the future of the U.S. DoD, including the use of Microservices to undergird the very fabric of weapons systems as well as completely replace hardware-intensive platforms with software-intensive Microservices. **Bottom Line**: *The U.S. DoD needs to capitalize upon software-intensive Microservices for a sharp 21st century strategic advantage*.
- BILLIONS & TRILLIONS OF GLOBALLY INTERCONNECTED APPLIANCES (*IOT*). The Internet was originally created by the U.S. DoD during the 1950s for military communications purposes. However, it grew into a commercial instrument during the late 1980s and early 1990s in the form of the World Wide Web (*WWW*). By the early 1990s, the number of interconnected computer hosts (*servers*) grew to around 10 million (*from a few thousand prior to 1990*). This was a rather large exponential gain. By around 2005, the number of hosts grew to one billion and approached one trillion by 2015. Furthermore, the definition of host expanded from a computer server to the end-user devices themselves such as personal computers, laptops, tablets, mobile phones, and other Internet-enabled devices. Eventually, hosts would include every imaginable household and personally-worn Internet-enabled device (*gas meter, water heater, furnace, stove, refrigerator, television, thermostat, doorbell, smartwatch, fitbit, treadmill, pacemaker, defibrillator, MRI machine, x-ray, manufacturing cell, satellite, radio, etc*.). That is, Internet hosts shrunk from hundred million dollar mainframe computers the size of buildings requiring a thousand people to operate, to miniature devices almost as powerful as supercomputers freely distributed for the price of a content service subscription. Hundreds of trillions of packets now circle the globe at the speed-of-light and the new battlefield consists of penetrating, controlling, and gathering intelligence from these devices to thwart the battle before it begins. **Bottom Line**: *The U.S. DoD needs to invest in mastering low-cost commercial miniaturized software-intensive Internet-enabled devices instead of trillion-dollar fighter jets*.
- USE UNMANNED AERIAL VEHICLES FOR IN-THEATER MISSIONS (*when necessary*). Much of the cost, schedule, and technical risk of weapon systems such as F-22 and F-35 has to do with the creation of traditional human operated aircraft. These systems must ease the risk of operating multi-hundred million dollar platforms with as much automation as possible (*increasing the size and sophistication of mission computer software*). These systems must meet ultra-stringent levels of human-rated safety, reliability, quality, etc. They must be armor plated, fly high and fast, have long ranges, carry large volumes of ordinance, and have sophisticated armor plating and defensive systems. Their avionics are replete with advanced radars, communications, heads up displays, maneuverability, landing gear, stabilization systems, ejection seats, and other advanced fly-by-wire controls. Our brightest pilots from the top military academies are recruited, trained, and conditioned to the point of having their humanity stripped from them so that they can operate an expensive aircraft with machine-like precision. They must fly tens of thousands of miles without getting tired in a cockpit the size of a kindergarten chair, drop a laser-guided 500-pound bomb on a high-value target without missing or getting shot down, and then return the expensive aircraft to home-base without so much as batting-an-eyelash. The cost of this systems of systems is in the hundreds of billions of dollars for research, design, development, and evaluation (*and the operations and maintenance costs range into the trillions of dollars*). **Bottom Line**: *The U.S. DoD must think-out-of-the-box and use lower-cost unmanned aerial vehicles when in-theater operations are absolutely necessary*.
- USE LOW-COST COMMERCIAL DISPOSABLE PLATFORMS (*when necessary*). Invest in low-cost, throwaway fire-and-forget UAVs instead of expensive reusable RQ-11 Ravens, Wasps, RQ-20 Pumas, RQ-16 T-Hawks, MQ-1 Predators, MQ-1C Grey Eagles, MQ-9 Reapers, RQ-7 Shadows, RQ-4 Global Hawks, and other weapon systems. The goal is to mimic commercial Internet giants vs. traditional paradigms hell-bent on resurrecting Scientific Management principles to engineer multi-billion dollar, century long weapons systems. Think of devices such as Apple iPhones, Samsung Galaxies, laptops, desktop PCs, and handheld tablets. These devices are manufactured to last about 90 to 180 days. Within a few months of buying the latest and greatest gadget, manufacturers preannounce the new version obsoleting whatever toy you have pained yourself to acquire. By the time you've loaded your contacts, downloaded you favorite apps (*Microservices*), and learned to maneuver the annoyingly subtle nuances of your new device, it's time to replace it with a completely new generation chock full of crazy new bells and whistles. These might include higher resolution screens and cameras, more memory for photos and music, longer lasting batteries and reliability, and, best of all, more throughput, bandwidth, and microprocessor speed. So, if your average global high-technology firm creates a brand new generation of Internet-enabled devices about every 90 days with no intention of maintaining them for 100 years, why does the U.S. DoD insist upon doing so? **Bottom Line**: *The U.S. DoD must transition its culture from 100 year systems to 90-day disposable weapon systems (as our most dangerous adversaries are doing)*.
- PURCHASE OR LEASE COMMERCIAL MICRO-MINIATURE TECHNOLOGIES AND ROBOTICS. Many Internet giants such as Amazon and Google are designing, developing, and dispatching millions of miniaturized technologies and robots to help them conduct global business. Amazon is dispatching an army of UAVs to help them deliver packages to the global community cost-effectively. These UAVs will even help with package returns. Google is dispatching fleets of UAVs to remote jungles, deserts, and mountain ranges to extend Internet service where it was not previously possible. If it's possible for these firms to reach the world's 7 billion inhabitants with new products and services, it is also possible for the U.S. DoD to do the same (*and even leasing these services being developed by Amazon, Google, and every international commercial firm on the planet*). Furthermore, instead of building custom miniaturized devices, why not just purchase them outright or lease their services for global security purposes. In the very least, one can purchase or lease their telemetry streams, data, and open source

intelligence. Most of these devices are cheap throwaway prototypes that won't break the bank. Furthermore, the U.S. DoD can purchase or lease commercial data centers, communication products and services, and the end-user or customer-facing devices themselves. The entire supply chain is now openly available to the commercial marketplace. Even the security, reliability, utility, and encryption of commercial mobile phones now meets or exceeds the levels of most military standards. **Bottom Line**: *The U.S. DoD should purchase or lease as many commercial, miniaturized products and services as possible*.

- TRANSFER RISKS FROM IN THEATER MANNED MISSIONS. In-theater missions, be it unmanned or manned, carry a high degree of cost, schedule, and technical risk (*not to mention geo-political risk*). The cost of some unmanned aerial vehicles (*UAVs*) rivals that of U.S. DoD fighter jets. While there are some lower-cost military-grade UAVs, these don't seem to be the platforms-of-choice for U.S. DoD. Is this merely guilt by the U.S. DoD for building low-cost, low-risk UAVs (i.e., *continuing to pump billions of dollars into military industrial complex for failing to engage large defense contractors with large airframe contracts*)? Worse yet, reconnaissance aircraft from the 1950s (*U-2*) continue to outperform late 20th century UAVs costing 10 to 100 times more to produce in terms of reliability, maintainability, maneuverability, and overall mission effectiveness (*Predator*). However, perhaps the greatest risk of all is political in-nature. The U.S. DoD flew an unprecedented volume of UAV missions from 2005 to 2015 in order to reduce burden on manned airstrikes and "*boot-on-the-ground*" (*creating a veritable international uproar questioning the humanitarian nature of these missions*). Of course, it helped that F-22 and F-35 were simply not ready for operational missions and the U.S. DoD had to rely on F-18s developed in the 1970s for manned missions. However, when possible, the U.S. DoD should not only reduce their reliance upon traditional fighter jets, but UAVs as well. **Bottom Line**: *The U.S. DoD should increase the use of satellite reconnaissance, Internet-based intelligence collection and analysis, and even the use of commercial capabilities for gathering data (and even prosecuting missions when possible)*.
- PERFORM REAL-TIME TELEMETRY UPGRADES USING DEVOPS. Amazon, Google, Apple, Samsung, Verizon, and just about every global Internet and telecommunication firm uses DevOps principles. They zap thousands of Microservices updates to billions of global devices every day. This includes TVs, servers, PCs, tablets, mobile phones, and every conceivable Internet-enabled device. Around 2006, many theorized that only the big two or three Internet giants would survive into the 21st century and all others would perish or be doomed to brick-and-mortar retailing. Some referred to this lopsided monopoly on global Internet commerce as a dinosaur-killing, extinction-level event. That is, massively-parallel Cloud Computing high-performance clusters built on commodity-grade motherboards, switches, and routers combined with DevOps to make lightning fast, bureaucracy and WIP-free innovations would simply wipe out the competition (*much like a meteor killing off dinosaurs 65 million years ago*). In fact, many 20th century manufacturing firms have perished at the hands of global Internet firms (*as well as a myriad of electronic firms during the Internet Bubble or Dot Com meltdown of 1999*). Studies as late as 2014 illustrated that less than 10% of information systems firms embraced DevOps at all in favor of manually-intensive lean and agile principles from the late 20th century. However, we seem to have reached a tipping point and the number of firms using DevOps principles to engage billions of global users simultaneously increased exponentially throughout 2015 and beyond. **Bottom Line**: *The U.S. DoD should embrace DevOps principles to push out military-grade Microservices to billions of miniaturized weapons systems*.
- TRANSFER ACQUISITION RISK FROM DEVELOPMENT TO COMMERCIAL LEASING. Internet giants now dominate the complete telecommunications spectrum. They do so with satellites, communication towers, digital switches, data centers with Petabytes of storage, and state-of-the-art end user devices such as PCs, tablets, mobile phones, household appliances, and wearable devices. These Internet-enabled devices often have more power, capabilities, capacity, and performance than 1970s-era military-grade super computers. Furthermore, telecommunication firms often give the latest hardware devices away for the cost of a service subscription to push their advertising content directly into your field of view with little or no direct cost to end-users. Given that 21st century firms have blanketed the global landscape with an ultra-reliable, state-of-the-art high-performance fabric or grid, it is not necessary for the U.S. DoD to attempt to duplicate this at the needless expense of taxpayer dollars. The U.S. DoD should simply lease these products and services (*much like mobile phone subscribers*), have these firms push out the latest software upgrades using DevOps principles, and simply trade up to the latest technology every few months (*as our adversaries are also doing*). If our adversaries are beating us on time-to-market and fast cycle times using commercial capabilities to conduct military operations against us, it's fruitless to build multi-decade long fighter jets at the expense of GDP. We'll simply fall victim to the fate of other information systems firms who became extinct during the Internet wars of the 1990s. **Bottom Line**: *The U.S. DoD should transfer fighter jet acquisition risk to the leasing of state-of-the-art Internet enabled devices*.
- TRANSFER BUDGETS FROM OPERATIONS AND MAINTENANCE TO R&D. Save a few mainframe firms from the 1960s, most information systems firms emerging in the 1970s allocated most of their resources to research and development. Few of these allocated any of their resources to planning, architecture, design, testing, documentation, or maintenance. However, as the 1980s bled into the 1990s, more and more information systems firms transferred the lion's share of their development resources into non-value-adding waste such as project plans, BDUF-based architectures, documentation, testing, and quality assurance. Management consultants from the 1960s urged these firms to allocate 80% to 90% of their resources to upfront planning with the hope of stemming the tide of cost-prohibitive operations and maintenance expenses. This became known as the "*Cost of Quality*" (*CoQ*) or the expense of "*doing it right the first time*!" Documentation became the rallying cry of Western firms and domestic and international information systems documentation standards mushroomed exponentially (*MIL-STD-1521b, DoD-STD-2167a, MIL-STD-498a, J-STD-016, ISO-STD-12207, ISO-STD-15288, CMMI, etc*.). However, the goal of Continuous Integration, Continuous Delivery, and DevOps was always to increase speed and reliability, while decreasing non-value adding waste and WIP to nearly zero. Firms using DevOps are able to flip CoQ upside down and transfer post development resources back into R&D and New Product Development (*NPD*). **Bottom Line**: *The U.S. DoD should transfer O&M budgets back into R&D activities using DevOps principles instead of the WIP and waste associated with its antiquated acquisition system*.
- EXPAND MISSION OPERATIONS EXPONENTIALLY. Firms like Amazon, Google, Apple, Samsung, and many other Internet and telecommunications giants have recouped the CoQ associated with traditional paradigms to reduce the cost, schedule, and technical performance risk of development. Furthermore, we've discovered that these Internet giants can zap thousands of Microservices updates back and forth across the globe in fractions of a second (*at the speed of light with ease and grace*). Most importantly, these firms grow their market share, firm size, and global footprint (*not only by focusing on creating value vs. wasteful CoQ activities*), but acquiring and gobbling up value-creating firms as well. It is no wonder why the top 5 or 10 Internet giants have monopolized the global landscape and decimated the competition (*as our adversaries do*). Using DevOps, they can

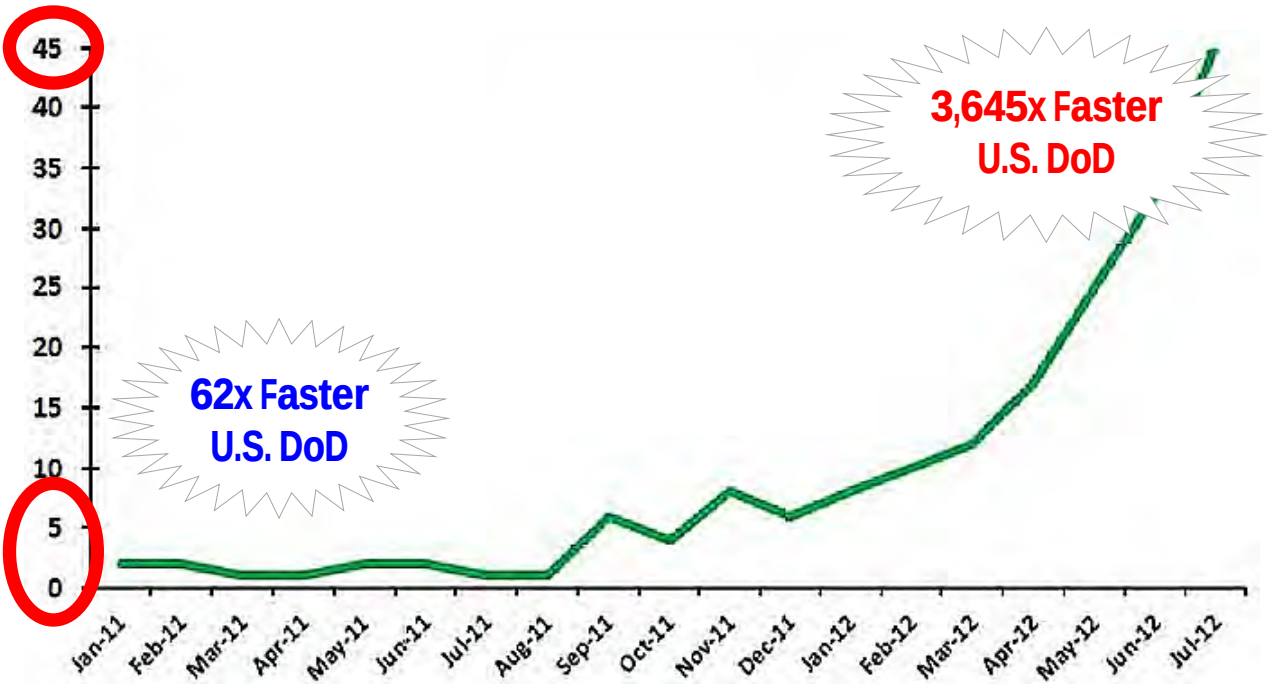
focus on total world domination vs. BDUF-architectures, documentation, manual testing, and preserving the organizational bureaucracies and non-value adding information systems resulting from the instinctual human desire to promulgate "*Conway's Law*!" Therefore, the goal is not simply to eliminate WIP and waste, or expand design resources, but to transfer resources to strategic planning, marketing, business development, innovation, and creativity-enhancing activities (*as well as quality-of-life*). Today, the sky-is-the-limit to the potential market growth, market capitalization, and value creation of global Internet and telecommunication firms. **Bottom Line**: *The U.S. DoD should expand global military mission operations exponentially like our adversaries, while decreasing acquisition costs and consumption of GDP, taxpayer resources, and other unneeded expenses*.

Why has the U.S. DoD fallen so far down since those fateful days when the ENIAC (*1946*), FORTRAN (*1957*), LISP (*1962*), SIMULA (*1965*), Object-Oriented Programming (*1967*), Software Engineering (*1968*), Software Reuse (1968), Object-Oriented Design (*1982*), etc. were created? "*Remember therefore how far you have fallen*" (*Revelation 5:2a*)? When did we stop wanting to be computer scientists and software engineers and start wanting to be project managers, systems engineers, and hardware manufacturers (*and why*)? Was it because computer programming was beneath us? Was it because electrical, mechanical, aeronautical, and civil engineering rose to such great heights (*and software paled in comparison to their glamorous light*)? Was it because systems engineering (*conceived in 1958*) finally matriculated, entered the late majority, and became a commonly accepted discipline by 1998? In the 1980s and 1990s, the first bachelors, masters, and doctoral degrees in software engineering itself first emerged. By the close of the 20th century these academic programs were all but obsolete and replaced with systems engineering degrees. Exactly what happened? Why did Amazon, Google, Yahoo, Ebay, Facebook, Etsy, Github, and Microsoft cling to the creation of software products and services by computer programmers (*while the U.S. DoD strove to create trillion-dollar fighter jets consuming the entire GDP of the Western Hemisphere*)? Oh well, there's no sense in crying over spilt milk!



Putting it all together, the integrated elements of architectural principles for all 21st century organizations are rather simple (*including the U.S. DoD*). These include "*Utilize lean and agile principles to streamline WIP and waste and maximize speed, time-to-market, and cycle time*," "*downsize your portfolio to unfreeze backlog queues*," "*simplify your products and services with tightly-scoped Microservices architectures*," "*modularize your Microservices ecosystems with highly-cohesive and loosely-coupled interfaces*," and "*utilize end-to-end automation using principles of Continuous Integration, Continuous Delivery, and DevOps to eliminate non-value adding WIP, waste, and CoQ*." You see it's the confluence and synergy of these contemporary best-practices that enable 21st century global Internet giants such as Amazon and Google to innovate at the speed-of-light and scale to distribute their entire portfolios of products and services to all of the world's inhabitants (*without incurring 20th century Coasian economic transaction costs*). Sub-optimizing any single one of these elemental principles will not enable 21st century technology-intensive organizations to unleash their full market and mission potential. Calling 21st century organizations "*technology-intensive*" is a bit of an oxymoron, since the fundamental characteristic of conducting global business in the 21st century involves being "*technology-intensive*" (*"software" technology, in particular, as the U.S. DoD seems to have forgotten in the last 40 to 60 years*).

Ironically, it was the lean community that accused agility of sub-optimizing information systems to the detriment of the enterprise. This was both true and false, as optimizing information systems delivery was the fundamental definition of 21st century agile organizations. Once again, it's been the goal of the lean community to incrementally squeeze efficiency and effectiveness out of 20th century hardware-intensive manufacturing firms so they can keep pace with Japan (*which is no longer a global player in manufacturing*). Furthermore, the lean community favors late 20th century Ludditism, in that they do not acknowledge the strategic nature of information systems technology (i.e., *tightly-scoped software Microservices and DevOps infrastructure automation as a competitive edge*). However, the information systems community spun-off their own agile methods in the late 1990s as a direct result of the Western flexible manufacturing phenomenon which was entering its golden age. In particular, methods such as Extreme Programming were specifically aimed at tightly-scoped software-intensive Microservices and laid the groundwork for the early 21st century infrastructure automation DevOps phenomenon. In other words, the information systems agility community laid the ground work for the five major architectural elements and principles (i.e., *lean and agile principles, downsize portfolio, simplify systems, modular interfaces, and end-to-end automation*). It was not the lean and agile community that devised this framework.



Remember, it took over 30 years to produce an operational fighter jet such as F-22 and F-35. It takes the U.S. DoD an average of 81 months to produce an information system on a good day (*which is an overly optimistic figure*). Information systems projects typically escalate in resources during the course of their life cycles due to over-scoping from the get-go, Conway's Law, increasing scope (*scope creep*), frozen queues, large numbers of human communication paths, glacial governance structures, overwhelming processes and documentation, and, of course, overburdening CoQ. This gave rise to the 1975 bestselling book, "*The Mythical Man Month*" where Fred Brooks compared information systems projects to pathetic dinosaurs inexorably caught in tar pits. That is, the so-called "*man-month*" was a hopelessly optimistic estimate of the effort it took to produce an information system due to lack of productivity. This was also partly due to "*Parkinson's Law*," which held "*work expands so as to fill the time available for its completion*" (*1955*). In fact, most medium to large-scale U.S. commercial and DoD information systems projects have a success rate of less than 5%, because of their hundreds of thousands and even millions of moving parts (*software source line of code statements*). Conversely, a 21st century ultra-reliable1,000 line of code Microservice can be coded, tested, and delivered to billions of global devices in fractions of a second using DevOps principles (*which is millions of times faster than an F-22 or F-35*).

So, what precisely have we learned in this treatise on 21st century information systems development principles? Well, we've learned that Tayloristic Scientific Management-manufacturing principles from the 19th and 20th century in the form of traditional project management, systems engineering, and software engineering paradigms are a non-starter when it comes to 21st century business success. This is true even when crude, manually-intensive lean and agile principles are applied to product and service development. We've learned that a key ingredient to 21st century business strategy is competing based on software vs. hardware-intensive products and services. We've also learned that portfolios must be dramatically downsized and product and service architectures must be tightly-scoped to single purpose-functions of immediate market impact and value. We've discovered that the optimal size and complexity of a 21st century product and service payload is about the size of a 1,000 line of code Microservice. Finally, we've described an underlying fabric consisting of end-to-end enterprise automation called DevOps, which does away with the degenerative effects of Coasian economic transaction costs and CoQ. **Bottom Line**: *The U.S. DoD must dispense with Conway's Law and begin acting and behaving more like Amazon and Google and stop acting like F-22 fighter jet manufacturers*.

References

- Duvall, P., Matyas, S., & Glover, A. (2006). *Continuous integration*. Boston, MA: Addison-Wesley.
- Erder, M., & Pureur, P. (2016). *Continuous architecture*: *Sustainable architecture in an agile and cloud-centric world*. Waltham, MA: Morgan-Kaufmann.
- Fowler, S. J. (2017). *Production-ready microservices*: *Building standardized systems across an engineering organization*. Sebastopol, CA: O'Reilly Media.
- Gruver, G., & Mouser, T. (2015). *Leading the transformation*: *Applying agile and devops at scale*. Portland, OR: IT Revolution Press.
- Gruver, G., Young, M., & Fulghum, P. (2013). *A practical approach to large-scale agile development*. Upper Saddle River, NJ: Pearson Education.
- Humble, J., & Farley, D. (2011). *Continuous delivery*. Boston, MA: Pearson Education.
- Humble, J., Molesky, J., & O'Reilly, B. (2015). *Lean enterprise*: *How high performance organizations innovate at scale*. Sebastopol, CA: O'Reilly Media.
- Kim, G., Debois, P., Willis, J., & Humble, J. (2016). *The devops handbook*: *How to create world-class agility, reliability, and security in technology organizations*. Portland, OR: IT Revolution Press.
- Krause, L. (2014). *Microservices*: *Patterns and applications*. Paris, France: Lucas Krause.
- Nadareishvili, I., Mitra, R., McLarty, M., & Amundsen, M. (2016). *Microservice architecture*: *Aligning principles, practice, and culture*. Sebastopol, CA: O'Reilly.
- Newman, S. (2015). *Building microservices*: *Designing find-grained systems*. Sebastopol, CA: O'Reilly Media.
- Ohara, D. (2012). *Continuous delivery and the world of devops*. San Francisco, CA: GigaOM Pro.
- Reinertsen, D. G. (2009). *The principles of product development flow*: *Second generation lean product development*. New York, NY: Celeritas.
- Rico, D. F. (2014). *18 reasons why agile cost of quality* (*CoQ*) *is a fraction of traditional methods*. Retrieved June 25, 2014, from http://davidfrico.com/agile-vs-trad-coq.pdf
- Rico, D. F. (2016). *Business value, ROI, and cost of quality* (*CoQ*) *for devops*. Retrieved June 1, 2016, from http://davidfrico.com/rico-devops-roi.pdf
- Singleton, A. (2014). *Unblock*: *A guide to the new continuous agile*. Needham, MA: Assembla, Inc.
- Womack, J. P., & Jones, D. T. (1996). *Lean thinking*: *Banish waste and create wealth in your corporation*. New York, NY: Free Press.

Case Studies in the Application of Lean & Agile Principles to U.S. DoD Aircraft

- Mid-20th Century U.S. DoD Exemplars. *Early innovators like Thomas Edison and the Wright Brothers were "tinkerers" and discovered their inventions through "trial-and-error." This culture of invention dominated the early American jet age following World War II and the beginning of the Cold War. Engineers and scientists conceived bold new designs, rapidly constructed inexpensive prototypes, and stretched the boundaries of aircraft through rapid prototyping. Time and materials were cheap and fun, curiosity, and the excitement of new discovery fueled their motivation. These included the Bell X-1 and X-2, Douglas X-3 Stiletto, Northrop X-4 Bantam, Convair X-6, Lockheed X-7 Flying Stovepipe, etc. None were more remarkable and enduring than the U-2 Dragon Lady, SR-71 Blackbird, and F-117A Nighthawk produced by Lockheed's Skunkworks (1950-1980). Many of these experimental airframes were tightly-scoped to perform a single mission. For instance, the U-2 was an extremely simple architecture designed to take high-altitude reconnaissance photographs of the Soviet Union without getting shot down by early surface-to-air missiles (SAMs). It had a service ceiling of 70,000 feet, range of nearly 2,000 miles, and speed of only 500 mph. Its primary requirement was high-altitude photographs. Soviet SAMs were specifically reengineered for greater speeds requiring a new tightly-scoped, single-purpose, Minimum Viable Architecture (MVA). Thus, the SR-71 was conceived. Its architecture was also designed for high-altitude reconnaissance, but with five times of the speed of 3,500 mph, an altitude of 85,000 feet, and a range of 2,500 miles. Finally, due to some physics breakthroughs in the diffusion of RADAR waves by the Soviet Union itself, the U.S. DoD discovered that the key to military aircraft was not speed and altitude, but rather low RADAR cross section. That is, if you can't see it coming or detect it in flight, then in-theater operations are more successful and it makes it difficult to shoot down. So, once again, a new tightly-scoped MVA was devised for the F-117A, consisting of near-RADAR invisibility, a 40,000-foot service ceiling, 500 mpg speed, and 1,000 mile range. These MVAs didn't have many bells and whistles, and often had no RADARs, defensive shielding or weapons, communication systems, ejection seats, and other safety-critical functions typically found in $400 billion fighter jets. The F-117A itself was the single-most valuable weapon system during Desert Storm and single-handedly decimating Iraq's infrastructure. The Skunkworks founder (Kelly Johnson) devised 15 principles of aircraft design not unlike the Agile Manifesto affectionately called, "Kelly's Rules!"*
- Late 20th Century U.S. DoD Exemplars. *While the U.S. DoD continued its love affair with F-22 and F-35 fighter jets, it quickly turned its attention to the use of Unmanned Combat Aerial Vehicles (UCAVs) by the late 20th century. Although UAVs had been in use for military reconnaissance since the mid-19th century, they had never come into widespread use to perform combat missions. The U.S. DoD reinvigorated its use of UAVs for military reconnaissance purposes during the Cold War, because of the increasing danger of Soviet SAMs on U-2 pilots such as Gary Powers. Vietnam proved to be a challenge to U.S. combat aircraft as the Soviets perfected SAMs for Vietnam. The U.S. conducted 4,000 drone missions in Vietnam largely for reconnaissance purposes. By this time, the U.S. DoD lost interest in the use of UAVs for military purposes of any kind. This was due in-part to the increasing use of space-based satellites and even manned-spaceflight for military reconnaissance purposes. Of course, throughout the Cold War, the U.S. was heavily vested in the use of short, medium, and long range missiles in the event on non-conventional warfare, as well as manned aircraft for conventional purposes. It was Israel that used UAVs for close-combat support to successfully defeat the Syrian Air Force in 1982 that brought a new focus on the use of UAVs for military combat (in addition to reconnaissance missions). Like Japan's necessity to develop lean principles for lack of resources following World War II, Israel also suffered from extremely limited resources and constant military conflicts with its immediate neighbors. Thus, they were forced to quickly press tightly-scoped UAVs based on MVAs into immediate service. From 1970 to 2000, the U.S. DoD remained focused on building trillion-dollar manned fighter jets such as F-22 and F-35, short of an all out nuclear war, which diminished with the fall of the Berlin Wall in 1989. However, it was the events of 9/11 that spurred the U.S. Intelligence Community to consider the use of UCAVs for conducting military strikes on high-value military targets half way around the world for fractions of the cost of fighter jets. In fact, it was the delay of F-22 and F-35 due to their immense complexity, WIP, waste, and use of traditional paradigms that pressed UCAVs into service. Donald Rumsfeld himself demanded the increased use of UCAVs to fight the war on terror resulting in the emergence of dozens of UCAV variations numbering in the thousands. UCAV pilots performed thousands of individual missions on hundreds of high-value targets using tightly-scoped MVAs at a fraction of the cost of conventional fighter jets such as F-22 and F-35.*
- Early 21st Century U.S. DoD Exemplars. *A great example of an early 21st century UAV with a tightly-scoped MVA is the X-37 Orbital Test Vehicle (OTV) built by Boeing's Phantomworks R&D facility in St. Louis, Missouri. A direct offshoot of the Cold War era U.S. DoD experimental aircraft, the X-37 was designed to replace NASA's Space Shuttle. The Space Shuttle was conceived in the late 1960s by the Nixon administration to replace the waning Apollo program. Started in 1972, the Shuttle flew its first space-based mission in 1981. 135 Shuttle flights were launched into space from 1981 to 2011, two of which were lost (one during launch and the other during reentry). Its total cost was over $200 billion and the U.S. DoD sensed a need to replace it for military reconnaissance and combat purposes. The U.S. DoD had been attempting to create manned space planes since the 1950s and 1960s to no avail. The Space Shuttle was conceived not only for civilian scientific purposes, but military ones as well (and many Shuttle flights contained military payloads of various sorts). The U.S. DoD had become so dependent upon the Shuttle, that its own dedicated military replacement began in earnest in the early 2000s. Beginning in 2004, the X-37 OTV reached orbit in 2010 only one year before the Space Shuttle's final retirement in 2011. No doubt, NASA was not allowed to retire the Shuttle until the X-37 had become operational, pending some sort of global military conflict. Unlike the over scoped Space Shuttle, which was originally conceived to take off like an airplane from a runway, but was dumbed down into a shuttlebus piggybacked on 1950s-era intercontinental ballistic missiles, the X-37 OTV was a tightly-scoped MVA with a single purpose (i.e., launch advanced tactical and reconnaissance military payloads into space). It broke all records for orbital operations, often staying in-space for two to three years at a time. While the exact purpose of these extended stays is not known, many believe the X-37 line of OTVs had native reconnaissance capabilities (in addition to launching small and inexpensive military satellites that were based on tightly-scoped MVAs themselves). Other purposes for the extended orbital operations included furthering the boundaries of military science (i.e., effects of long-term radiation on military space vehicles) and disabling adversarial capabilities? Perhaps, the U.S. DoD will conduct manned spaceflights for military purposes based on low-cost MVAs (known as Spikes in the agile community). The X-37 OTVs were developed at less than 1% of the cost, time, and technical resources required by NASA's over scoped Shuttle using traditional paradigms.*