Keywords: *Agile, Systems Engineering, Information Technology (IT), DoD Agile IT Acquisition, IT Box, Scrum*

Inserting Agility in System Development

Matthew R. Kennedy and Lt Col Dan Ward, USAF

With the fast-paced nature of technology, rapidly fielding systems has never been more important. Success depends on well-defined requirements and the ability to rapidly respond to change during and after deployment. The inability to rapidly respond may cause the system to become obsolete before initial fielding. Creating a structure where processes allow for changes during system development requires restructuring system development values and principles at all levels. This article addresses progress toward agility and defines agile values and principles being used by agile organizations in the Business, System, and Software Aspects. It also defines operationally effective agile practices being utilized to implement those values and principles that provide a starting point for inserting agility into the system development process.

image designed by Diane Fleischer »

With the fast-paced nature of technology, the need to rapidly field systems has never been more important. Success does not just depend on well-defined requirements, but also on one's ability to respond to change during development, deployment, and post-deployment. The inability to rapidly respond to change may cause the system to become obsolete before initial fielding. Creating a structure where processes allow for changes to occur during system development requires a restructuring of system development values and principles at all levels.

Three Aspects of a Software Intensive System Development

Software Intensive System (SIS) development can be understood as having three aspects: Business, System, and Software. Although the three aspects sometimes overlap one another, general responsibilities can be attributed to each. The Business Aspect is responsible for the overall acquisition of the system, including contracting, funding, operational requirements, and overall system delivery structure. Next, the System Aspect is responsible for the technical and technical management aspects of the system, and serves as the interface between management and engineers. The Software Aspect is responsible for the software items contained in the SIS. Viewing SIS development through the lens of these aspects helps highlight components of the work that are often neglected.

Agility is "the speed of operations within an organization and speed in responding to customers (reduced cycle times)" (Massachusetts Institute of Technology, n.d.). It must be incorporated into each aspect. The degree of agility when developing an Information Technology (IT) system determines the organization's ability to respond to change.

Currently, each aspect is at a different maturity in terms of the agile frameworks and methodologies available. However, the speed at which changes can be made during development is held captive by the aspect that is most resistant to change. This article addresses each aspect and its progress toward agility, and defines the agile values and principles being used by agile organizations in both the Business and Software Aspects. It defines agile practices being utilized to implement these values and principles to provide a starting point for inserting agility into the system development process.

Business Aspect

The Business Aspect is where operational requirements are realized and the strategy for overall system development is identified. Currently, the Department of Defense (DoD) uses DoD Instruction (DoDI) 5000.02 to manage how it will perform the acquisition of weapon systems, services, and Automated Information Systems (AIS) (DoD, 2008).

Recognizing that the current DoDI 5000.02 was not responsive to the changing needs of technology, Congress signed the Fiscal Year 2010 National Defense Authorization Act (NDAA), which directed the Secretary of Defense to "develop and implement a new acquisition process for information technology systems" (NDAA, 2009). This new Defense Acquisition System process must include:

- early and continual involvement of the user;
- multiple, rapidly executed increments or releases of capability;
- early, successive prototyping to support an evolutionary approach; and
- a modular, open-systems approach (NDAA, 2009).

Moreover, this process should be based on the March 2009 *Report of the Defense Science Board (DSB) Task Force on Department of Defense Policies and Procedures for the Acquisition of Information Technology* (NDAA, 2009). The DSB report concluded that "the conventional DoD acquisition process is too long and too cumbersome to fit the needs of the many IT systems that require continuous changes and upgrades" (DSB, 2009). The report also noted that an agile acquisition approach would increase IT capability and program predictability, reduce cost, and decrease cycle time.

The DSB has developed an Agile Business Aspect framework, which is divided into four phases: Business Case Analysis and Development, Architectural Development and Risk Reduction, Development and Demonstration, and Operations and Support (DSB, 2009).

Figure 1 depicts the four phases of an Agile Business Aspect Framework (DSB, 2009). A brief description of each phase follows:

- **Business Case Analysis and Development:** "Establish the need for the proposed capability and develop the concept for the proposed solution and perform a cost-benefit analysis to quantify the benefits of the solution."

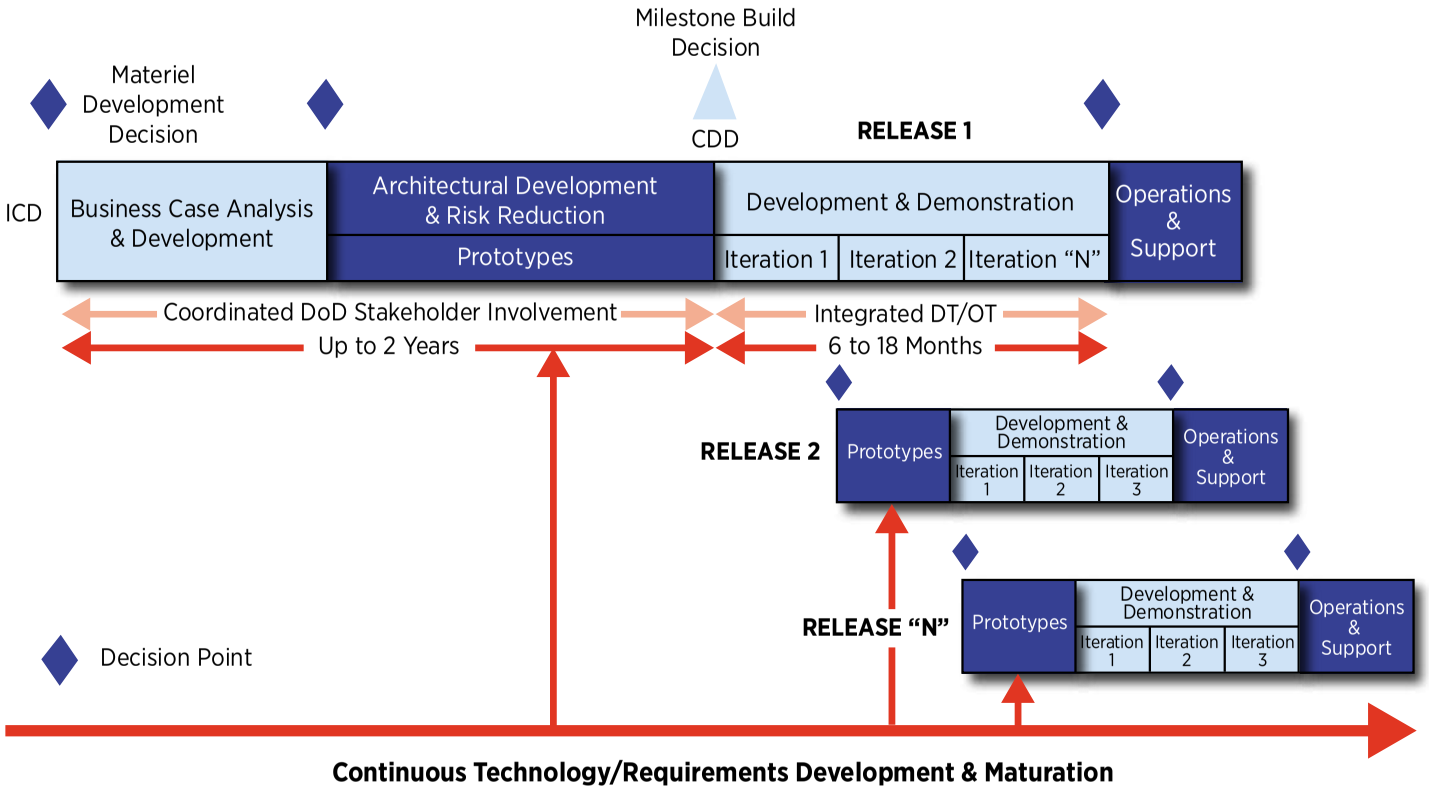
- **Architectural Development and Risk Reduction:** "The core architecture is built and architecturally significant features demonstrated. Prototyping begins during this phase and continues throughout the acquisition life cycle to assess the viability of technologies and minimize high-risk features."

- **Development and Demonstration:** "The period when operational capability is built and delivered for a discrete number of releases. Capabilities are prioritized and parsed into groupings to establish release baselines for the subprograms. Includes development of training programs and testing in realistic environments to ensure successful fielding of new capabilities."

- **Operations and Support:** "Provides materiel readiness, user training, and operational support over the total program life cycle."

In addition to the emerging IT Acquisition framework, the DoD developed an agile requirements process for IT systems called the "IT Box" (Wells, 2009). The Joint Requirements Oversight Council Memorandum 008-08 stated, "IT programs are dynamic in nature and have, on average, produced improvements in performance every 12–18 months" (Joint Requirements Oversight Council, 2009). Recognizing the need for performance improvements, the "IT Box" allows IT programs the flexibility to incorporate evolving technologies. This allows for greater agility in the current DoD requirements process.

FIGURE 1. AGILE BUSINESS ASPECT MODEL



Note. CDD = Capabilities Development Document; DT = Direct Test; ICD = Initial Capability Document; OT = Operational Test

To be used in conjunction with the framework is a guiding value set called FIST (Fast, Inexpensive, Simple, Tiny), which may be utilized throughout the process (Ward, 2010). The FIST approach identifies a set of priorities and preferences that should be employed by project leaders during the development process to streamline, accelerate, and simplify (Ward, 2010). These values are declared in the FIST manifesto as:

> Talent *trumps* process.
>
> Teamwork *trumps* paperwork.
>
> Leadership *trumps* management.
>
> Trust *trumps* oversight. (Ward, 2010)

The FIST Manifesto also contains a series of principles and implementation guidelines, which can be applied to all three aspects of development (System, Software, and Business). These principles follow:

- Fixed funding and floating requirements are better than fixed requirements and floating funding.
- Complexity is cost.
- Simplicity scales. Complexity does not.

The implementation guidelines include:

- Minimize team size and maximize team talent.
- Incentivize and reward underruns.
- Requirements must be achievable within short time horizons. (Ward, 2010)

The FIST approach describes a particular pattern of decision making that has been successfully used on various DoD programs. Recent examples include the Marine Corps "Harvest Hawk," which incorporated a gunship modification onto a C-130 airframe. This modification was fielded just 18 months after the program was announced (Axe, 2010). Similarly, the U.S. Air Force's new intelligence, surveillance, and recon-

naissance aircraft—the MC-12W—flew its first combat mission just 6 months after the contract was signed. This is a divergence from the typical decade-long weapons system program and shows the DoD can deliver inexpensive systems on short timelines.

In addition to rapidly delivering inexpensive systems, capabilities produced by using the FIST approach tend to outperform more expensive, complex systems when actually fielded. Examples include the Air Force's Condor Cluster supercomputer, which was developed for one-tenth the cost of a traditional supercomputer and uses one-tenth the electricity of comparable systems. It operates at 500 TFLOPS (**T**era **FL**oating point **OP**erations per **S**econd), making it the fastest supercomputer in the entire DoD.

The Agile Business Aspect framework and the FIST approach are examples of how the Business Aspect is making advancements toward becoming more agile and adaptive to changing requirements, which is required to keep pace with today's rapidly changing environment.

System Aspect

The System Aspect addresses the technical and technical management pieces of the system and serves as the interface between management and engineers. Utilizing various systems engineering standards and guides, operational requirements are decomposed into technical requirements. The System Aspect holds the overall responsibility for the development of the system given the contractual, schedule, and fiscal constraints of the Business Aspect.

Though the systems engineering process is generally portrayed in a waterfall-like fashion, the systems engineering community has moved toward an incremental delivery approach. The (DAG) identifies incremental development as a capability that *Defense Acquisition Guidebook* that "is developed and fielded in increments with each successive increment building upon earlier increments to achieve an overall capability". This incremental approach relies heavily on prototyping and allows for technology maturation in subsequent releases (DAU, 2010). The move toward an incremental delivery allows the systems engineering process to better adapt to change than the waterfall-like implementation. However, with the rapid rate of change, the incorporation of an incremental model alone may not be enough. Currently, no agile systems engineering frameworks, principles, or values are in place to guide the System Aspect.

2

Software Aspect

The Software Aspect addresses the software items contained in the SIS. Provided a set of requirements from the System Aspect, the Software Aspect creates the software items required for the system.

Software development has been on a continuous process improvement track for decades. Initially, the waterfall software development methodology was used, where software was developed in one long release cycle (Royce, 1970, pp. 1–9), although this approach was described as "risky and invites failure." The waterfall software development methodology provides the fundamental steps required to develop software. However, it has one major flaw in that it assumes that once the requirements process is complete, the requirements will remain unchanged throughout the development life cycle. This assumption rarely holds true in practice as change is inevitable in all large software projects (Sommerville, 2004).

Long waterfall-like development cycles do not allow for requirements changes, a flaw identified by Royce in his original paper. Breaking software development cycles into a series of increments allows one to better adapt to changing requirements. In the incremental model, an increment is a potentially shippable piece of functionality. Incremental delivery allows the user to gain value from a portion of the system prior to the entire system being released.

Agile Software Development

Though seen as an improvement over the waterfall software development methodology, the incremental approach has several disadvantages; namely, the majority of requirements must still be known up-front (U.S. Air Force, 2003). Agile processes have emerged to match the pace in which change is encountered during software development.

Agile software development is a broad term used to describe development methodologies that adhere to a set of values and principles defined by the Agile Manifesto (Beedle et al., 2001). The Agile Manifesto was formed when a group of 12 people calling themselves the Agile Alliance gathered to find an alternative to the current documentation-driven, heavyweight software development process (Beedle et al., 2001). Through this effort, they framed the following set of values to improve the way software is developed (Beedle et al., 2001):

- 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.

The Agile Manifesto also defines the following principles, which are used to separate agile practices from their heavyweight counterparts (Martin & Martin, 2006):

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Simplicity—the art of maximizing the amount of work not done—is essential.

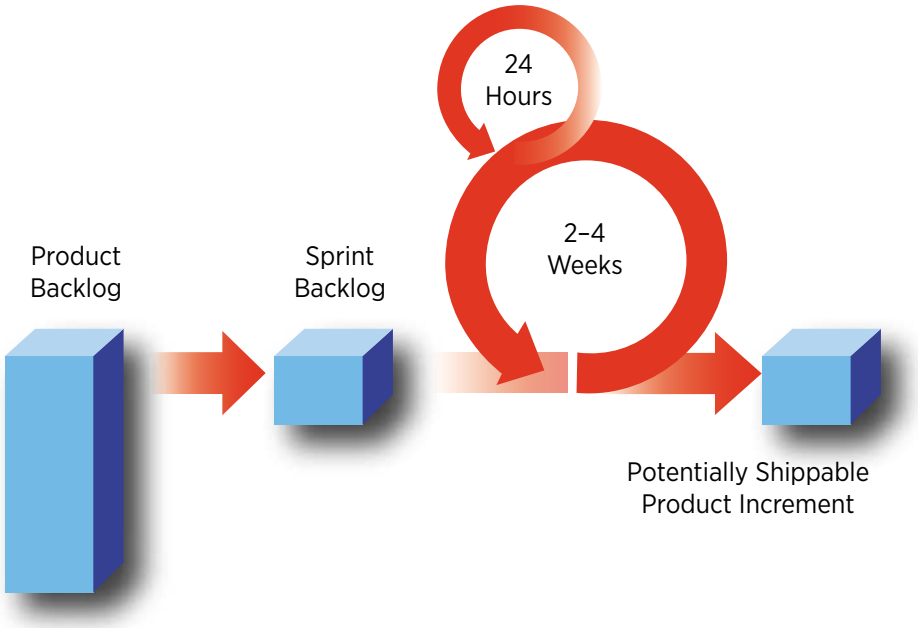
The application of these principles varies in practice as no predetermined number of principles must be utilized for a development methodology to be deemed "agile." Several development methodologies are in use today; however, a survey conducted by VersionOne, which included almost 1,700 individuals and 71 countries, found Scrum and eXtreme Programming to be the most widely followed methodologies (VersionOne, 2007). Other common methodologies include Crystal, Dynamic Systems Development Methodology, and Lean Software Development.

2

Scrum

Scrum is a framework used for project management, which is designed for projects where it is difficult to look ahead (Brede Moe, Dingsøyr, & Dybå, 2008, pp. 76–85). It provides a framework with which these activities will be executed (Figure 2). Scrum comprises self-organizing and self-managing teams that release a potentially shippable product in sprints (increments) of 2–4 weeks.

FIGURE 2. SCRUM FRAMEWORK



Note. Adapted from *The SCRUM process/SCRUM framework* [Web page], by Expert Program Management (n.d.) at http://www.expertprogrammanagement.com/2010/08/the-scrum-process/.

The process starts with a product backlog (requirements) that is prioritized by the user prior to the start of each sprint. The team then selects what can be accomplished within the designated sprint duration; however, the team must select the requirements in the order specified by the user. These selected requirements then become the sprint backlog. The items on the sprint backlog are what will be delivered to the customer at the end of the sprint.

eXtreme Programming

Whereas Scrum is a process to manage a product, eXtreme Programming (XP) is an agile development methodology focused on software development as a whole. XP is one of the most well-documented agile methodologies, and it consists of the following 12 rules (Cohen, Lindvall, & Costa, 2003):

1. The Planning Game	2. Small Releases	3. System Metaphor	4. Simple Design
5. Continuous Testing	6. Refactoring	7. Pair Programming	8. Collective Code Ownership
9. Continuous Integration	10. 40-Hour Work Week	11. On-site Customer	12. Coding Standards

No set number of rules need be practiced by a team to claim they are doing XP (Wolak, 2001). However, the strength of XP is in the combination of the rules and not implementing a single rule alone (Cohen, Lindvall, & Costa, 2003).

The Software Aspect has a greater selection of agile methodologies to utilize during development, allowing for valuable resources when inserting agility within the Software Aspect.

Maintaining Agility Between Aspects

With the growing complexity of today's systems, the systems engineering effort becomes increasingly important to success. Currently, both the Business and Software Aspects have an agile framework and a proven set of agile values to help guide development. However, traversing from the Business Aspect to the Software Aspect requires passing through the System Aspect, which could hinder the agile advances made in the other aspects. The System Aspect's ability to respond to the agile processes developed within the Business Aspect, as well as fostering the agile processes in the Software Aspect, could play a pivotal role in overall system success.

Agile Principles

When combining the FIST implementation guidelines and principles and comparing them against similar principles, much constancy is evident.

Though no one-to-one relationship exists between the FIST principles/guidelines and the Agile Manifesto principles, they all remain important complementary principles while developing a complete agile organization.

Agile Practices

Agile projects use various practices to implement the Agile Values and Principles identified. When considering both the Software and Business Aspects, a common set of practices emerges. These practices are:

Incremental Development	Small Teams
Iterative Development	Time Boxing
Short Time-lines	Lean Initiatives
Retrospectives (Lessons Learned)	Prototyping
Empowered/Self-organizing/ Managing Teams	Continuous User Involvement
Prioritized Product Backlog (Requirements)	Co-located Teams

Implementation of these practices varies greatly from project to project. Using co-located teams as an example, a large program retrofitting military aircraft may be structured in a way to have the teams located on the same installation so that the contracting, development, and testing activities are located on the same installation. This contrasts with software development teams, which implement the practice of co-located teams by having the development team work in the same room.

These practices are well-documented and demonstrated and offer great promise for helping deliver affordable systems that are available when needed and effective when used. By implementing these proven practices, we can increase agility with the Systems Aspect.

What to Expect from Implementing Agile Practices

Studies have been conducted over the last decade documenting the results when utilizing agile practices. Rally Software Development Corporation found an average 37 percent decrease in time-to-market and a 16 percent increase in productivity (Software Engineering Institute,

n.d.). Findings from seven individual studies found a benefit-to-cost, productivity, and quality ranging from 14 percent to 93 percent (Rico, 2008). The averages from the study can be found below:

67 percent, average increase in productivity,

65 percent average increase in quality, and

49 percent improvement in cost (Rico, 2008).

Conclusions

More than ever, military technology programs need to rapidly field systems within tight budget constraints and still maintain an ability to respond to change. The Agile approach provides a useful starting point to achieve these objectives of speed, thrift, and agility.

Inserting agility within an organization is a journey, not a destination. Agile practices that work for one organization may not be as effective when implemented at another organization. Conversely, agile practices found effective within an organization last year may no longer be as effective as their initial implementation due to external, internal, or personnel changes. These changes may require periodic modification or even removal of practices to remain competitive in today's fast-paced world of IT. It is not a single practice that makes an organization agile, but a combination of practices.

Author Biographies

Professor Matthew R. Kennedy is a professor of Software Engineering at the Defense Acquisition University. He served in the U.S. Air Force as a network intelligence analyst and has more than 10 years of experience in Information Technology. Professor Kennedy holds a bachelor's degree from Northern Illinois University and a master's degree from the University of Illinois, both in Computer Science.

(E-mail address: Matthew.Kennedy@dau.mil)

Lt Col Dan Ward, USAF, is currently serving as chief, Acquisition Innovation at the Pentagon. He holds a bachelor's in Electrical Engineering from Clarkson University, a master's in Engineering Management from Western New England College, and a master's in Systems Engineering from the Air Force Institute of Technology and is Level III certified in two acquisition career fields: Program Management, and Systems Planning, Research, Development, and Engineering.

(E-mail address: daniel.ward@us.af.mil)

References

Axe, D. (2010). Marines' instant gunship blasts Taliban, Pentagon bureaucracy. *Wired*. Retrieved from http://www.wired.com/dangerroom/2010/11/marines-instant-gunship-blasts-taliban-pentagon-bureaucracy/

Beedle, M., Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Highsmith, J., ... Thomas, D. (2001). *Manifesto for agile software development.* Retrieved from http://agilemanifesto.org/

Brede Moe, N., Dingsøyr, T., & Dybå, T. (2008). *Understanding self-organizing teams in agile software development.* Presentation at 19th Australian Software Engineering Conference, Perth, Australia, March 25–28.

Cohen, D., Lindvall, M., & Costa, P. (2003). *Agile software development.* New York: Data and Analysis Center for Software.

Defense Acquisition University. (2010). *Defense acquisition guidebook.* Retrieved from https://dap.dau.mil/Pages/Default.aspx

Defense Science Board Task Force. (2009). *Department of Defense policies and procedures for the acquisition of information technology.* Washington, DC: Office of the Under Secretary of Defense for Acquisition, Technology and Logistics.

Department of Defense. (2008). *Operation of the defense acquisition system.* Washington, DC: Office of the Under Secretary of Defense for Acquisition, Technology and Logistics.

Martin, R. C., & Martin, M. (2006). *Agile principles, patterns, and practices in C#.* Boston, MA: Prentice Hall.

National Defense Authorization Act for Fiscal Year 2010, Pub. L. 111–84 (2009).

Rico, D. F. (2008). *What is the Return on Investment (ROI) of agile methods?* Retrieved from http://www.afei.org/WorkingGroups/ADAPT/Documents/rico08a[1].pdf

Royce, W. W. (1970). *Managing the development of large software systems.* Retrieved from http://leadinganswers.typepad.com/leading_answers/files/original_waterfall_paper_winston_royce.pdfIEEE

Software Engineering Institute–Carnegie Mellon. (n.d.) *Brief history of CMMI.* Retrieved from http://www.sei.cmu.edu/library/assets/cmmihistory.pdf

Sommerville, I. (2004). *Software engineering 7/E.* Boston: Addison-Wesley.

U.S. Air Force. (2003). *Guidelines for successful acquisition and management of software-intensive systems.* Ogden, UT: Software Technology Support Center.

VersionOne. (2007). *2nd annual survey: The state of agile development.* Retrieved from http://www.versionone.com/pdf/StateOfAgileDevelopment2_Summary.pdf

Ward, D. (2010, November-December). The FIST manifesto. *Defense AT&L, 39*(6), 31–32.

Wells, C. (2009). *Information technology requirements oversight and management (The JCIDS "IT box")* [PowerPoint slides]. Leveraging technology evolution for information technology systems (JROCM 008-08). Retrieved from https://acc.dau.mil/adl/enUS/421037/file/55578/IT%20Box%20Overview.pdf

Wolak, C. M. (2001). *Extreme programming (XP) uncovered.* Ft. Lauderdale, FL: Graduate School of Computer and Information Sciences, Nova Southeastern University.