Evolutionary Capabilities Developed and Fielded in Nine Months

Portia Crowe
U.S. Army, Program Executive Office C3T

Dr. Robert Cloutier
Stevens Institute of Technology

The DoD is facing challenges to rapidly deploy operational capabilities in complex environments where bridging legacy and new technologies are key to success. The challenges arise as a result of diminishing budgets and the need for new capabilities to operate in war environments, including the global war on terrorism. To balance this imperative need with rapid response, we found that our developed Agile life-cycle paradigm was a viable solution to meet challenges brought about by changes in the environment. This article demonstrates how a DoD program used an Agile approach, throughout every phase of the program's life cycle, to rapidly field capabilities.

Our intent with this article is to give DoD programs another successful data point for implementing an evolutionary acquisition strategy using Agile life cycle processes, and share our learned tenets of this process in the hopes of helping others rapidly field capabilities. In an environment where requirements are unforeseen and quickly changing, we need our systems to be flexible and adaptable to meet these growing challenges. Before 2006, the U.S. Army readiness reporting system had no longer met the needs of commanders in providing timely and detailed data to make informed decisions. Complex environments and service-oriented architecture (SOA) changed the landscape of operation and the systems that were operating in it. The challenge was to, within nine months, transform an old Army readiness system to meet current needs without losing existing capabilities while also developing key functions currently needed by commanders.

The solution was for the Product Manager Strategic Battle Command together with the Headquarters–Department of the Army G3/5/7 to modernize the legacy Army readiness application, PC ASORTS, by creating the Defense Readiness Reporting System-Army (DRRS-A). The DRRS-A aligns with SOA strategies and supports the demands of new requirements, capabilities, and modifications in the areas of force registration, force readiness, force projection, and mobilization. The DRRS-A team included about 60 people from the government and multiple contracting teams. The strategy was a phased approach allowing for the deployment of high-priority capabilities first and then subsequent capabilities using an incremental process. The DRRS-A software system first deployed in late 2006 after only nine months of development and has fielded new capabilities incrementally in as soon as two months [1]. The program consists of secure Web-based capabilities such as unit status reporting that details mission-critical information including personnel levels, training status, equipment availability, and equipment serviceability. It is used as a commander's assessment tool as it reports a unit's capability to execute missions. Using an evolutionary strategy, the legacy application was a stepping stone for the development of new capabilities and rapid, yet disciplined, transition of processes.

Approach

The DoD embraces change after a long history of Waterfall software methods and single-step to full-capability approaches. The goal of the DoD's Evolutionary Acquisition (EA) policy is to provide operational capabilities to the warfighter—quicker than traditional methods—through rapid incremental fielding, building to full-objective capability [2]. The DRRS-A implementation plan included using the DoD EA approach as a guideline for delivering capabilities in increments. Our evolutionary strategy was to take the existing readiness system and perform system modernization in a phased approach, which included leveraging functionality inherent in the old system and translating it into usable functionality in an SOA, combined with serial guidance and directives issued by the Joint Chief of Staff. The DRRS-A currently has as many as 5,000 users including Army Commands, the National Guard Bureau, Army Forces Command, and the United States Army Reserve Command (USARC).

> *"In an environment where requirements are unforeseen and quickly changing, we need our systems to be flexible and adaptable."*

Our system methodology was to integrate all aspects of program life-cycle phases using an Agile approach with rapid prototyping to ensure that the customer and user needs were met. We took a linear life-cycle approach and worked life-cycle phases in parallel and often at the same time. Working within an aggressive schedule, we carried out continuous facilitation of the following phases:

- Concept refinement, requirements, and architecture analysis and design.
- Capability and software development.
- Integration, testing, and demonstrations.
- Production and deployment.
- Operations, support, and training.

The user community consistently worked with the developers to refine concepts and requirements to be developed. In a month, the team could get as many as five new requirements and enhancement requests from various sources, such as the readiness community and the Joint Chiefs of Staff. New requirements can range anywhere from new calculations to new information required from the user. During testing time, the team mainly focuses on fixes and performance of the applications. The rapid and iterative software development process included conducting continuous integration and testing on a daily basis by checking in and out software code. Scrum, our Agile process, was implemented in 30-day software development sprints using a prioritized requirements list also known as a *backlog*. It helped keep the focus on user needs with demonstrations at the end of each interval [3]. In a month, the development and test teams can work through as many as 15 requirements. The Scrum development process [4] is shown in Figure 1 (see next page).

Characteristic	Comments
Liberty to be dynamic	Agility needs dynamic processes while adhering to acquisition milestones.
Non-linear; cyclical and non-sequential	The life-cycle behavior was not like traditional waterfall models or linear frameworks; decreasing cycle times.
Adaptive	Conform to changes, such as capability and environment.
Simultaneous development of phase components	Rapid fielding time may not lend to traditional phase containment (i.e., training and software development together).
Ease of change	Culture shift to support change neutrality; ease of modification built into architecture and design.
Short iterations	Prototyping, demonstrating, and testing can be done in short iterative cycles with a tight user feedback loop.
Lightweight phase attributes	Heavy process reduction, such as milestone reviews, demonstrations, and risk management.

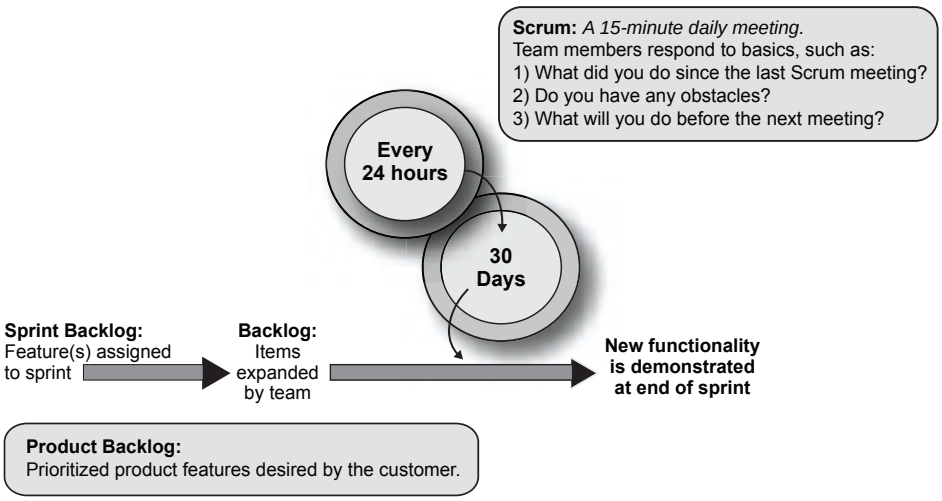
Table 1: *Emergent Agile Characteristics for Rapid Prototype and Development*

At the end of each development sprint (estimated as every 30 days), a training team of six people would develop or edit training materials, user guides, and train the help desk on these features. The user community is trained on the new or edited features via remote, computer-based, and/or face-to-face training. The imposed user feedback loop gave us greater confidence that we were building to expectations and user requirements. We were inline with priorities as a result of canvassing for feedback during bi-weekly iteration meetings, testing events, and surveys [5].

The key to an Agile life cycle is to keep everyone informed so that functions can be done in parallel. To ensure continuous coordination, all of the functional leads (i.e., system engineers, development, logistics, and requirement proponents) and key users were at every sprint review. This helped to keep the team in sync on new application features, training needs, integration and test events, and priorities of requirements.

Interoperability and integration testing was frequently conducted not only at the subsystems level but for system of systems and external dependencies. Prior to major releases, we found three to be the *magic number* of user dress rehearsals or test events. These allowed users to test functions of the applications and engineers to get a good read on the performance of the system. Participants (usually 20 to 30 users) were chosen by the functions or applications being tested during that event and the location of participants (such as Iraq or Afghanistan), which gave us good performance data. These testing events, each lasting three days, usually started three months prior to a big release and were about three to four weeks apart. Coordination with participants, training, and test procedures distribution occurred prior to each event. An online survey was prepared for the users to track their issues and concerns during the event. We recently added performance questions to track how fast the functions were loaded and displayed. After each day of the test event, a configuration control board met to discuss the feedback and developers began fixing or discussing problems with the test event participants. This rapid test approach allowed us to get feedback and work problems right away. Fixes were incorporated into the next test event. Performance test cases were conducted with up to 50 simultaneous users doing the same functions on the force registration application. During this process, we found memory leaks to be causing significant delays and were able to be proactive in optimizing performance before the application went *into the field*. Testing resulted in force registration, the application for unit registration data, performing five times faster when users access the most popular functions of the application. Further application testing led to response time improvements of up to three times. Performance enhancements were made where applications may have many simultaneous users in the United States and abroad.

Figure 1: *Example Agile Process Flow for the Scrum Development Process*



Learned Tenets

Our rapid prototyping and deployment challenge was to take an Army readiness reporting system that had existed for 12 years and modernize, deploy, and support its new capabilities within nine months. This brought about challenges for our multiple contract teams who were beholden to many stakeholders, not to mention being rigidly adherent to changing requirements in a complex wartime environment. These Agile and rapid fielding initiatives led us to several revelations of the rapid prototyping and development approach. We found some key characteristics that are outlined in Table 1.

Rapid prototyping and deployment challenges included development of a new set of capabilities that didn't exist in the old system, and training and re-training a large user community in parallel of building software and testing. New functionality included embedded workflow process, identity management, and Web access using the Secure Internet Protocol Router network. We also had to gain security certificates and authority to operate; because of their generally long lead time, we planned for these issues in the beginning stages. The stakeholders treaded in unfamiliar territory by collaborating closely with users and knowledge sharing with other contractors. Risk management was a collaborative effort that emphasized the software development phase. The greatest challenge was to develop, train, and conduct integration testing with multiple contractors in nine months.

Success

The initial fielding of DRRS-A capabili-

ties were successfully developed, taught to users, fielded, and supported within nine months by using incremental and Agile methodologies [6]. Subsequent releases have been just as successful adding on existing capabilities and deploying new ones. The success of DRRS-A and its net-centric capabilities include improving user accuracy, ease of use, and decreasing manpower and manual input. As an example of cost realization, the USARC cited DRRS-A Web-based applications savings that are expected to be more than $1 million annually, and the Army Reserve Medical Command has saved what averages to be $118,933 per month. The DRRS-A capabilities are part of the Defense Information System Agency's Net-Enabled Command Capability program and readiness model for the U.S. Air Force and Marine Corps.

Conclusion

Lessons learned during program development and fielding have brought the importance of collaboration, communication, and risk management to the forefront of the Agile development process:

- A tight collaboration of several contracting teams, stakeholders, and program offices is necessary to prove integration of the right requirements into the software.
- The set-up of checkpoints during the process is crucial to ensure that development was meeting the customers' needs.
- Without the communication and involvement of stakeholders and customers, there would be limited sharing and transfer of knowledge that can hinder synchronization across the battlespace.
- It is easy to concentrate on risk management in the software development stage since it is the meat of the program, but lessons learned has taught us that all of the other life-cycle stages need to be risk-analyzed and evaluated often during rapid development. For example, training needs to be planned up-front and during development or else there won't be ample time to train the user community. This, in turn, could lead to program failure.

As stated in the introduction, our intent was to give a current successful data point to the DoD's evolutionary acquisition strategy using an Agile life-cycle approach. We have also proven—through cost and user acceptance of this system—that DoD life cycles can be developed and maintained using Agile methodologies. With an aggressive schedule and high program visibility, we broke traditional developmental and cultural barriers by implementing an Agile and evolutionary approach to rapid prototyping, development, and fielding. It was critical to implore a knowledge-sharing environment between contractors and a close collaboration between the functional proponents and users, which in turn laid the groundwork for success. Our Agile and flexible approach to systems and software engineering allowed us to capture the true essence of rapid prototyping and capability deployment while still meeting budgetary, schedule, and customer satisfaction goals.◆

References

1. "Army to Modernize its Unit Status Reporting Processes." Army Stand-To! 10 Aug. 2006 <http://lists.army.mil/pipermail/stand-to/2006-August/000136.html>.
2. DoD. Department of Defense Directive 5000.1: The Defense Acquisition System. Washington, D.C.: Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics, 2003.
3. Boehm, Barry, and Richard Turner. Balancing Agility and Discipline: A Guide for the Perplexed. Boston: Addison-Wesley, 2004.
4. "What Is Scrum?" Scrum: It's About Common Sense <http://controlchaos.com/about>.
5. Hansen, W.J., et al. Spiral Development and Evolutionary Acquisition. The SEI-CSE Workshop Special Report. SEI, Carnegie Mellon University. May 2001 <www.sei.cmu.edu/pub/documents/01.reports/pdf/01sr005.pdf>.
6. "New System Gauges Military Readiness." AFCEA Signal. 16 Oct. 2006 <www.imakenews.com/signal/e_000144589000039843.cfm?x=b8fBhsr,b5kS6S>.

About the Authors

Robert Cloutier, Ph.D., is an associate professor of systems engineering in the School of Systems and Enterprises at the Stevens Institute of Technology. He has more than 20 years experience in systems engineering and architecting, software engineering, and project management in both commercial and defense industries. Industry roles include lead avionics engineer, chief enterprise architect, lead software engineer, and system architect on a number of efforts and proposals. His research interests include model-based systems engineering and systems architecting using Unified and Systems Modeling Languages, reference architectures, systems engineering patterns, and architecture management. Cloutier has a bachelor's degree from the U.S. Naval Academy, an MBA from Eastern College, and a doctorate in systems engineering from the Stevens Institute of Technology.

School of Systems and Enterprises
Stevens Institute of Technology
Hoboken, NJ 07030
Phone: (201) 216-5378
E-mail: robert.cloutier@stevens.edu

Portia Crowe is currently working for the Army, Program Executive Office C3T-PM Battle Command as chief engineer of the Army Defense Readiness and Projection Systems. She is conducting research in integrating risk management in Agile systems engineering. Crowe has received two Commander's Awards for Civilian Service: one for successful implementation of the DRRS-A program, and the other for excellence in research and development of technology objectives. She has a bachelor's degree in computer science from Rutgers University, a master's degree in engineering management from the New Jersey Institute of Technology, and is currently a doctoral student in the School of Systems Engineering at the Stevens Institute of Technology.

U.S. Army, PEO C3T-PM
Battle Command
SFAE C3T BC
BLDG 2525-Bay 3
Ft. Monmouth, NJ 07703
Phone: (732) 427-5757
E-mail: portia.crowe@us.army.mil