Portia Crowe

Robert Cloutier

Case History

The U.S. Army updates its readiness reporting system using an "Agile" approach in a challenging environment

Portia Crowe, Chief Engineer of the U.S. Army Defense Readiness and Projection Systems (portia.crowe@us.army.mil); Robert Cloutier, Ph.D., Associate Professor, School of Systems and Enterprises, Stevens Institute of Technology (robert.cloutier@stevens.edu)

The United States Department of Defense (DoD) faces many challenges, especially when it comes to the development of software systems and programs. One critical challenge is how to share information for operational capabilities rapidly in complex environments within a system that bridges legacy and new technologies. The authors explain how a product development team within the DoD achieved this goal using an Agile approach to upgrade every aspect of one particular program throughout every phase of the program's life cycle while that program continued to be operational.

This article provides an example of how the U.S. Department of Defense has used Agile lifecycle processes in the development and upgrade of critical systems and programs. It will describe how our development team created an evolutionary acquisition strategy using Agile lifecycle processes. We are sharing our lessons learned in hopes of helping others do the same when working with similar challenges.

This project of upgrading the U.S. Army "readiness reporting system" took place in an environment where requirements are unforeseen and quickly changing. (See Box for definitions.) Under those circumstances, we needed our systems to be flexible and adaptable. Before 2006, it was clear that the existing reporting system was no longer meeting the needs of commanders to provide the timely and detailed data on resources (i.e. personnel, training level of completion and equipment) for making informed decisions. Commanders were operating in a more complex IT environment. The challenge we faced was to transform an old Army readiness system to meet current needs while also developing key functions currently needed by commanders—without losing existing capabilities and within a nine-month period.

"The challenge we faced was to transform an old Army readiness system to meet current needs while also developing key functions needed by commanders."

The Product Manager Strategic Battle Command worked with the Headquarters–Department of the Army G3/5/7 on this. The solution our team came up with—to modernize the legacy Army readiness application, PC ASORTS—was to create the Defense Readiness Reporting System-Army (DRRS-A).

The objective was to create the DRRS-A which, once completed, would align with Service Oriented Architecture (SOA) strategies and support 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. We selected a phased-approach strategy, which allowed the deployment of high-priority capabilities first and then subsequent capabilities using an incremental process.

Defining the objective in detail

Our 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. We used "rapid prototyping and deployment" to do this. Such a challenge meant many hurdles for our multiple contract teams who were beholden to many stakeholders, not to mention having to be 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 Exhibit 1 on page 14.

Rapid prototyping and deployment challenges included development of a new set of capabilities that didn't exist in the old system, and training and retraining a large user community while building

Definition of Terms Used by the U.S. Army for the Development of DRRS-A Program

- "Agile" methodology: Methods that encourage frequent, iterative, and constant adaption through teamwork and self-organization.
- Defense Readiness Reporting System-Army (DRRS-A): A secure web-based software system that allows soldiers to report their current readiness, registration, projection and mobilization status for higher commands.
- PC ASORTS: The legacy system in place before DRRS-A went into effect.
- Service Oriented Architecture (SOA): A set of principles of governing concepts used during development and integration that require loose coupling of services.

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 the generally long lead time in getting approvals, we planned for these issues in the beginning stages. The stakeholders treaded in unfamiliar territory by collaborating closely with users and by 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.

Preview of the success

We achieved our objective: The DRRS-A software system was first deployed in late 2006—after only nine months of development, and new capabilities were added incrementally as soon as two months later.[1] The DRSS-A program consists of secure Web-based capabilities such as unit status reporting which 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 of executing 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.

"Using an evolutionary strategy, the legacy application was a stepping stone for the development of new capabilities."

Exhibit 1: Emergent Agile Characteristics For Rapid Prototype and Development

Characteristics	**Comments**
Liberty to be dynamic	Agility needs dynamic processes while adhering to acquisition milestones
Non-linear; Cyclical and non-sequential	Lifecycle behavior 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 SW 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 tight user feedback loop
Lightweight phase attributes	Heavy process reduction such as milestone reviews, demonstrations, and risk management

SOURCE: U.S. Army

Details on our "Agile" approach

The DoD embraces change after a long history of "waterfall" software and single-step methods to full-capability approaches. By "waterfall" we mean that each iteration or upgrade was done separately. The goal of DoD's Evolutionary Acquisition policy was to provide operational capabilities to the soldier more quickly than than those types of traditional methods through what we refer to as "rapid incremental fielding"—upgrades and harmonization done in such a way that it can be continuous.[2]

The DRRS-A implementation plan included using the DoD Evolutionary Acquisition (EA) approach as a guideline for delivering capabilities in increments. Our evolutionary strategy required taking the existing readiness system and modernizing it in a phased approach, which included leveraging functionality inherent in the old system and translating it into usable functionality in a Service-oriented architecture (SOA) combined with serial guidance and directives issued by the Joint Chiefs of Staff. Today, 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).

Our methodology included integration of all aspects of program lifecycle phases using an "Agile" approach with rapid prototyping to ensure that the customer and user needs were met. We took a linear lifecycle approach and worked lifecycle 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.

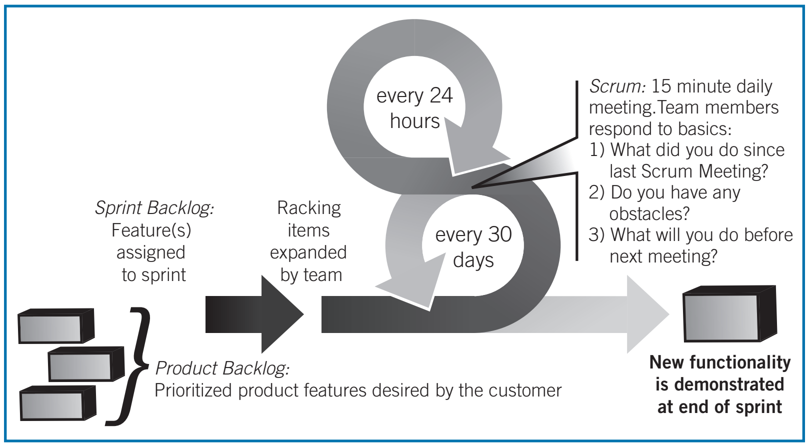
The rapid and iterative software development process included conducting continuous integration and testing on a daily basis through the use of checking in and out software code. "Scrum," our Agile process, was implemented in 30-day software-development sprints using a prioritized requirements list also 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 Exhibit 2 on page 15.

At the end of each development sprint, a team of six people would develop or edit training materials and user guides, and train the help desk on these features. The user community was 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 in line with priorities as a result of canvassing for feedback during biweekly iteration meetings, testing events, and surveys.[5]

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.

Exhibit 2: Example Agile process flow for the “Scrum” development process


every 24 hours
every 30 days
Scrum: 15 minute daily meeting.Team members respond to basics:
1) What did you do since last Scrum Meeting?
2) Do you have any obstacles?
3) What will you do before next meeting?
Sprint Backlog: Feature(s) assigned to sprint
Racking items expanded by team
Product Backlog: Prioritized product features desired by the customer
New functionality is demonstrated at end of sprint

SOURCE: “What Is Scrum?” Scrum: It’s About Common Sense <http://control chaos.com/about>.

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 out 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 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.

Conclusion

The DRRS-A program was successfully developed and fielded, including training and strong support, in 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 and ease of use, and decreasing manpower and manual input.

For example, in terms of cost savings, the USARC has predicted that savings from DRRS-A Web-based applications are expected to reach more than $1 million annually. The Army Reserve Medical Command has already saved an average of $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.

Program development and fielding brought the importance of collaboration, communication, and risk management to the forefront of the Agile development process. Here are a few of the lessons we learned:

- A tight collaboration of several contracting teams, stakeholders, and program offices is necessary to prove integration of the right requirements into the software.
- The setup of check points 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 which can hinder synchronization across the battle space.
- It is easy to concentrate on risk management in the software development stage since it is the meat of the program, but lessons learned have taught us that all the other lifecycle stages need to be risk-analyzed and evaluated often during rapid development. For example, training needs to be planned upfront and during development or else there won’t be ample time to train the user community. This, in turn, could lead to program failure.

During the project we proved—through cost and user acceptance of this system—that 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. This approach proved to be a critical aid in the creation of a positive knowledge-sharing environment among contractors, as well as in developing a close collaboration between 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. V

Endnotes

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. Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics. Washington, D.C, 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://control chaos.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>*.

NOTE: A more detailed version of this article appeared in CrossTalk, *The Journal of Defense Software Engineering*, May/June 2009 issue, Vol. 22, No. 4, pp 15-17.