Improving Agile Execution in the Federal Government

Committed Partner. Creating Results.

In December of 2010 the government introduced the **25 Point Implementation Plan to Reform Federal Information Technology Management,** now most often referred to as simply, the 25 Point Plan. The primary goal of the Plan is stated in the introduction;

"Government officials have been trying to adopt best practices for years – from the Raines Rules of the 1990s through the Clinger Cohen Act and the acquisition regulations that followed. But obstacles have always gotten in the way. This plan attempts to clear these obstacles, allowing agencies to leverage information technology to create a more efficient and effective government." [1]

The Office of Management and Budget, (OMB), has recognized the value of one of these "best practices", an incremental approach to software delivery, which it terms Modular Development in the 25 Point Plan:

"Modular development delivers functionality in shorter timeframes and has long been considered best practice in the private sector and in some areas of government; in fact, both Raines Rules and the Federal Acquisition Regulation (FAR) advise agencies to plan programs in this way. Successful organizations using modular development base releases on requirements they define at a high level and then refine through an iterative process, with extensive engagement and feedback from stakeholders."[1]

OMB subsequently provided a set of guidelines for Modular Development in the **Contracting Guidance to Support Modular Development** released in June of 2012.[2]

Outside of the federal government, the core principles of this Modular Development approach are most often categorized under the umbrella term 'Agile'. The most popular Agile method being implemented is Scrum. Scrum defines the management processes and roles in a software development project.

Recently the GAO released a report on Software Development called **Effective Practices and Federal Challenges in Applying Agile Methods**.[3] In it they discuss the key challenges to applying Scrum within agencies and the practices which help Scrum succeed.

Not Meeting Expectations

Many agencies have engaged in Agile software development via a top-down approach because their CIO or other high level management are following the guidance in the 25 Point Plan and OMB guidelines. They then have identified programs in their agency to apply Agile within. They may have even sent some of their staff to a couple of days of training (compared to many years of non-formal training in more sequential development methodologies such as Waterfall). In addition, they've awarded the work to contractors with some form of Agile software development credentials. This may mean the contractor has some staff with their certified Scrum Master certification (again, a couple days of course work). With this model the first few projects may or may not have been successful. Often many of the ideas of Scrum are not fully used and there is a large difference between what was discussed in their training and what is actually occurring during the projects. This is the realization that understanding Agile is easy; whereas, practicing it is difficult.

Scrum principle of always improving

Three of the most important principles of Scrum are Inspection, Adaptation and Transparency. These principles are the key to the idea of always improving. Retrospectives and the repeated planning activities of Scrum offer numerous opportunities to keep improving the process. The team has to inspect the process and outcomes. They then make changes (adapt) that will improve the process and outcomes. In order to do this the process and outcomes must be easily inspected (transparent).

Initially many of the improvements are made to external influences or involve personnel issues on the team. In order to keep improving Scrum the team will eventually need to dive deeply into how the work is being executed during the sprint. This is when issues of software quality, design quality, reliance on a single team member and other technical issues arise. All of these issues introduce risk to the project success. The next sections address how to improve the technical delivery skills of the Scrum team and allow them to reduce the risk associated with the rapid changes that occur in an Agile project.

Take Agile to the next level with AEP

How can an agency increase the chance of success with their Agile projects and become more Agile themselves? One approach is to bring in superstar, experienced Agile practitioners who know how to bring the academic lessons of Agile into the pragmatic practices of a project. Unfortunately there are only a limited number of these practitioners available. An alternate way to take your Agile projects to the next level is to introduce Agile engineering practices (AEP) from Extreme Programming (XP). While Scrum defines process management and roles, XP is a framework for the development techniques. Many Federal agencies are ready to start using Scrum with AEP. An Agile Assessment can help determine the readiness of an agency to add AEP into their Agile projects.

Delivering high quality software not just delivering on-time, on-budget

At many agencies, Agile projects are delivering working software on-time and on-budget. In addition, thanks to Scrum, the highest priority functionality is being built and the teams are improving their ability to adapt to change. However, there is room for improvement in both the quality of the delivered software as well as the agility of the teams.

By using Test-First Programming (TDD) practice in a rigorous manner teams can drastically reduce the number of defects found in the code. They can also achieve much better software design via TDD. There is a greater adherence to design by contract with TDD and the coded functionality reflects the functionality specified in the user stories. Refactoring is an important part of Test-First Programming. TDD gives one the confidence that improving the code via a refactoring is not breaking the functionality. (Just-in-time adaptation)

Increase transparency

Project transparency is greatly improved via Scrum. The Daily Scrum gives a view of progress on user stories. The backlog and burndown charts give a view into the progress on the project. Velocity charts show the team's speed. But what about individuals? Their level of work and quality of work can be hard to ascertain. (Knowledge hording)

The Pair Programming practice brings tremendous transparency to the skill and contribution levels of the individual team members. Those that are highly skilled and giving high levels of effort will be quickly exposed. Those that are not skilled and/or not working hard will also quickly be exposed. The Scrum concept of succeeding or failing as a team doesn't mean that Pair Programming should be used as a way to expose incompetence and discipline those that don't meet the standards. Instead it can be a way to bring out the need for further training or counseling.

Continuous Integration practice allows everyone knows where the quality of the product stands. The successful builds tell the whole team if the product is deliverable and meeting the definition of done. Failed builds let the team know if the product is going off course and needs to be corrected immediately. This transparent measure of quality is not achievable with any traditional method. In order to achieve Continuous Integration it is important to aim to implement the Ten-Minute Build practice as well.

Code and Tests as primary artifacts of the project can greatly increase transparency because unlike all other artifacts, the tests are guaranteed to reflect what is actually developed. They are always up to date and accurate.

Earliest feedback

In Scrum we use short iterations of batched work called sprints so that we can receive feedback on that work immediately after it is completed versus waiting to a large UAT effort at the end of the project. In sequential waterfall development each phase provides feedback to the previous phases. For example, the development phase may enlighten the team about a problem coming from the design phase. This may occur many weeks or months after the design phase is completed. The rework that occurs at this point can be very expensive as the cost curve increases over time. Thus it is highly desirable to receive feedback about work as early as possible. This is why all phases of software development (analysis, design, development, testing) happen during the sprint in Scrum. Several of the XP practices help move feedback up even earlier in the process. For example, Continuous Integration provides feedback on the quality of the code within 10 minutes or less. Test-First Programming provides feedback on the code within minutes. Pair Programming provides feedback on the code within seconds. Having such early feedback reduces the cost to adapt even within the sprint and increases the chance of successful delivery of functionality within the sprint.

Increase feedback

Scrum has many feedback loops built into the different processes that the team uses. Sprint reviews and retrospectives are examples. In XP, there are engineering feedback loops that constantly allow the team to inspect and adapt as they work.

The Continuous Integration practice allows the Whole Team to see if the work they have integrated is fulfilling the requirements as reflected in the test results and checks run by the Continuous Integration tool.

The Test-First Programming practice provides feedback on the teams understanding of the requirements. They won't be able to write the tests if they don't fully understand the requirements. The automated tests then provide immediate feedback to the team demonstrating whether the code meets the requirements under test.

The Pair Programming practice provides instant feedback to an individual team member about their work. It is a simultaneous, rigorous code review session. Design ideas are critiqued along with the implementation of those designs.

Improve estimates

XP practices such as Pair Programming, Shared Code, Whole Team, and Sit Together can improve the Scrum team's ability to make better estimates.

With Pair Programming the team will see many different methods of problem solving and technical approaches such that when they have a new story to estimate they will draw upon this knowledge to produce their numbers. It is important to correctly manage Pair Programming in order to maximize the value.

The Shared Code practice means that everyone on the Scrum team has knowledge of the complete code base. There are no silos in place that limit who can work on what part of the code. Therefore all user stories can be estimated by all members of the team. This improves the accuracy of the estimates as more input from different viewpoints helps pinpoint the LOE for the work.

The Whole Team practice in XP means that everyone who is needed to get the work done (based on the definition of done) will be taking place during the estimation process. This avoids the situation in which a developer may be making an estimate on a story in which they have no real experience or domain knowledge of the content. By having the whole team involved, the accuracy of the estimates can be improved.

The Sit Together practice benefits estimation because the team is constantly hearing others talk and engaging them in discussions that don't just involve the stories that they are working on at the time. This increases the knowledge of the domain and technical system such that their estimation improves.

Increase collaboration with co-location

One of the key values of the Agile Manifesto[4] is the value of customer collaboration over contract negotiation. In Scrum we try to reduce all forms of uncertainty simultaneously. Uncertainty can be around the features of the final product, the design and technologies used to implement the product, or even who the customer or market is for the product. In Scrum we address that uncertainty via collaboration. Via the XP practice of Sit Together all of the team is collocated so that they collaborate in the most effective way to eliminate the uncertainties as soon as they need to (last responsible moment decision). Nothing enhances collaboration as well as communicating with all our senses.

Co-location is an advanced practice and can be impractical for some agencies where they have their staff spread amongst various locations geographically or even within the same building. The downside is that the communication process is much more difficult and effective collaboration will be limited. Another advanced practice of XP derived from Lean thinking is that of Shrinking Teams. With Shrinking Teams the idea is to make one person as idle a possible and rather than easing the load on everybody. The ultimate outcome is to eventually reduce the team size. By using the Shrinking Teams practice, one can attempt to move toward co-location by reducing the work of those that are not collocated until you eventually shrink the team to just those members that are collocated.

Minimize risk

Scrum attempts to minimize risk by constantly adapting and inspecting as we build the software. XP is extremely risk adverse and addresses issues of quality in the code via such principles as Pair

Programming, Test-First Programming, Real Customer Involvement, Whole Team and others. Together they make a perfect pairing as Scrum allows the Product Owner to adjust priorities based on feedback during the project and XP allows the Scrum team to make the changes required to satisfy the Product Owner without undue technical risk to software quality.

Summary

The Agile Engineering Practices enable a Scrum team to take their performance to the next level. Each of the practices mentioned in this paper will increase the value delivered to the customer via Scrum. While using all of the practices is the most effective way of achieving ultimate team performance, many Scrum teams are not ready to make the leap. The GAO makes this point when discussing an Agency's Strategic planning:

"Allow for a gradual migration to Agile appropriate to your readiness"[3]

Therefore it is advisable to undergo an assessment of the team's Agile maturity and determine which of the practices can be instituted immediately and which ones the team can aspire to achieving down the road. Again the GAO reiterates this point:

"Combine Agile frameworks such as Scrum and XP if appropriate."[3]

CC pace can help with both AEP and bringing in experienced practitioners. We were born Agile. We've been successfully delivering Agile projects since 1999 and have trained over 10,000 individuals in Agile techniques. With our Agile Assessment, we can recommend the Agile Engineering practice to move to next. Whether a basic practice or advanced practice, we can guide you in the adoption of it and transform your Scrum projects to a higher level of achievement.

1. 25 POINT IMPLEMENTATION PLAN TO REFORM FEDERAL INFORMATION TECHNOLOGY MANAGEMENT - VIVEK KUNDRA U.S. CHIEF INFORMATION OFFICER DECEMBER 9, 2010

2. CONTRACTING GUIDANCE TO SUPPORT MODULAR DEVELOPMENT – OFFICE OF MANAGEMENT AND BUDGET, EXECUTIVE OFFICE OF THE PRESIDENT JUNE 14, 2012

3. SOFTWARE DEVELOPMENT, EFFECTIVE PRACTICES AND FEDERAL CHALLENGES IN APPLYING AGILE METHODS – UNITED STATES GENERAL ACCOUNTABILITY OFFICE (GAO-12-681) JUNE 14, 2012

4. http://agilemanifesto.org/

About the Author

David C. Patton, PhD
Director, Enterprise Solutions, Federal Practice
CC Pace Systems, Inc.