Over the Waterfall in a Barrel – MSIT Adventures in Scrum

Jeanne Lewis  
Jeanle@microsoft.com  
Senior Program Manager  
Microsoft Corporation

Kevin Neher  
kneher@microsoft.com  
Senior Program Manager  
Microsoft Corporation

Abstract
This paper describes how a large, traditional & globally distributed IT organization is responding to the grassroots introduction of agile methodologies into a strongly waterfall based development culture.

1. Introduction

1.1 Who are we?

We are not the people that bring you Windows, Office or any of the other familiar Microsoft products. We are Microsoft IT. An organization over 4000 strong, which develop, deliver, and maintain trustworthy technology solutions that protect corporate resources, increase employee productivity, and showcase the value of running Microsoft products to our customers.

With more than 106,000 users and 300,000 devices on a worldwide network spanning 98 countries, Microsoft depends on IT teams worldwide to provide basic IT services, which include the corporate network, e-mail, and Helpdesk.

IT workers around the globe also build and support many internal business applications as well as applications that interface with SAP and Siebel. MS IT engineering is engaged in 300 development projects building new applications, systems and platforms as well as enhancements to legacy systems.

Microsoft IT has another mission that makes it unique among global enterprises. In addition to running IT services, we play a strategic role as Microsoft’s first and best customer. All enterprise products must run stably in our environment before we approve them, signalling that the software is commercially ready. We also push next generation feature, usability and performance recommendations and requirements back to the product development teams.

2. Challenges

In this section we will describe some of the challenges we are facing both as teams attempting to utilize agile in this environment and as a management organization attempting to support the movement and to grow best practices.

2.1 Business as Usual

Several years ago, MS IT formally adopted a standardized Software Development Life Cycle (SDLC) methodology based on waterfall and Project Management Institute (PMI) style project management methodologies. This SDLC is required by Microsoft corporate policy to be followed when developing applications that support Microsoft’s business functions. It is within this structured and somewhat inflexible environment that a grassroots agile movement has organically grown and taken root among developers.

2.2 Becoming Agile

In 2006, LPO, a team within MS IT, decided they wanted to try Scrum. But, how could they when the current corporate policy required that all teams use the SDLC, a methodology born of waterfall? The obvious first hurdle became getting Scrum adopted alongside waterfall in the SDLC. This would require the team to petition the SDLC governance body.

The first task became mapping waterfall terms with Scrum terms. Pre-Baseline became one of the “Phase-end reviews” and Business Requirements Document (BRD) became Product Backlog, captured at the start of each Sprint.

Once terms were defined, an Agile SDLC Guide was written. This document describes Scrum in terms common to the SDLC and included such topics as “What is Agile?”, “What is Scrum?” and “How does the MS IT SDLC differ from Scrum?”

Explaining Scrum in this way allowed the SDLC governance body to quickly become familiar with Agile techniques and overcome the assumption that agile (Scrum) was a pseudonym for chaotic and uncontrolled.

LPO was successful in their efforts to gain formal acknowledgement of Scrum and is now into their 11th sprint. The team now actively evangelizes
Scrum across MS IT and helps other teams begin using Scrum on their projects. Other teams in MS IT have also adopted agile methodologies, but not all with the same level of adherence to corporate policy. This has presented another challenge that will be described in greater detail in the “Next Steps” section to follow.

2.3 Working with the PMO

We made two primary changes to our project management approach: schedule management and phase reviews. Both changes had the goals of minimizing change to upper management while utilizing standard Scrum artifacts to adhere to the Scrum process.

Our project schedules are managed using Gantt charts in MS Project. The Engineering schedule is managed as a sub-project within the master project schedule. Since our initial implementation of Scrum replaces the Engineering SDLC design/dev/test phases with sprints, it became easy to discard the detailed Gantt chart and replace it with the Sprint Backlog.

For a Scrum project, we have replaced the phase rollups in the master project schedule with the sprint dates. All further detail, usually found in the Engineering Gantt chart is now in the Sprint Backlog.

This message has been received well by our management and customers. They continue to allow the team to be self-managing at the task level.

The other primary change is with periodic management reviews. Under our Waterfall SDLC process, there are 3 management project reviews: the Pre-Baseline Review happens at the end of the Requirements phase; the Baseline Review happens at the end of the Design Phase; and the Go No-Go Review happens at the end of the UAT phase. These reviews are an extremely important part of MS IT management’s ability to steer our project portfolio to successful releases.

It is critical for a team rolling out agile techniques to create a well-defined process that provides an equal amount of visibility and oversight to that provided before adoption. The LPO Agile Rollout Team created a Scrum version of the SDLC process described earlier and had it approved by our SDLC group, with similar control points. They continue to have an Agile Pre-Baseline Review, which occurs before Sprint 1; the key difference is that under the Scrum model, they do not need a Product Backlog that is 100% complete and final, since we acknowledge that Scrum works via dynamic scope management. This allows us to launch projects sooner. The Scrum SDLC replaces the single Baseline Review with multiple monthly Sprint Review meetings. As such, it provides better visibility and control than the Waterfall SDLC. The team maintains a standard Go No-Go Review at UAT complete.

By minimizing the changes to the top-down project management model and providing equivalent or better visibility and control, it was much easier to get management buy-in to try an agile approach.

2.4 Choosing an Agile Methodology

We have learned that the first decision for a development team converting to an agile methodology is to select an agile methodology that is compatible with corporate needs and dynamics.

Many MS IT teams have adopted Scrum. This is for two primary reasons:

1) Scrum is a process framework that doesn’t radically change basic tasks such as coding and testing, and

2) Scrum sprints can be easily used as a plug-in replacement for the classic waterfall design/dev/test phases within the Engineering group.

In particular, our user acceptance testing (UAT) process can be quite complex, involving many co-dependent applications and using testers around the world.

While leaving out the UAT phase is not “pure Scrum”, as a pilot approach and a stepping stone, this can be a path to minimal impact outside the engineering organization.

2.5 Stay the Course

Universally MS IT Scrum teams have told us that the first staying focused and on track during the few sprints of the pilot project are critical. There can be a temptation for some members of the team to modify or customize the Scrum process, including suggestions to drop some Scrum practices seen as “not valuable” or “improvable” by some. For instance, some did not see the value in talking about what they worked on yesterday, one of the 3 components of the daily Scrum meeting. Some felt
that we should be creating the design work one sprint ahead of the sprint where it would coded and tested.

However, two prior ad hoc attempts within the LPO IT group to try agile-like processes had failed due to deviation from agile processes and philosophies; these attempts had been labeled “fragile” by some. The LPO Scrum Coach provided great advice at this juncture: adhering to the process, typically for at least a year, was necessary to change our mindsets and habits. Looking back, several team members have commented that this was an important key to the success of our pilot.

This isn’t to say that we didn’t have to make some choices. As described above, due to some constraints, we had to start by piloting Scrum without the user testing function within each sprint. However, we view our approach as a stepping stone to the final goal, and our success with Scrum has allowed us to now have open and encouraging discussions with the UAT team for future projects.

### 2.6 Find the Right Team Members

We have discovered that a key success points in launching pilot projects has been finding the right team members – ones willing to discard drilled-in preconceptions and give Scrum a chance. The understanding of Scrum and the benefits of the agile approach unfold over time. After a few sprints some team members may find that they were not comfortable with the approach. When this happened to the LPO team they quickly found more compatible replacements and the new team began to gel over the following sprints.

One of the key dynamics we often ignore in team formation is the like-mindedness of the individuals when it comes to process. One common input from new MS IT Scrum teams is that having team members that were resistant to Scrum was the biggest challenge to their early roll out. They report time and productivity losses due in part to addressing the objections arising from skeptical team members. It also have been a hit to the general moral.

In general, MS IT’s primary criteria for including a member onto a team usually involve either technology skills and/or expertise with the application or application business domain. Rarely in our industry do we form teams based first on process style and beliefs. Yet this is so critical to team dynamics.

We recommend that when forming an agile team; “agile-mindedness” should be added to the usual selection criteria.

### 2.7 Adapting to Changes in Role

The majority of significant role changes occurred within the engineering team and only resulted in a few challenges. The Scrum concept of a “flat” organization required some adjustment for team members that previously held a lead or manager role. Such team members must be willing to produce as an individual contributor. They must also be able to wear two hats by separating their duties as an HR manager with responsibilities for the career development of their reports on and off the agile team while performing as a peer to the same employee in the Scrum team.

The only role outside the engineering organization that is affected in a major way is the product/project manager role which was described in the PMO section of this paper. But it is important to point of the value in finding a Product Manager who was already “agile-minded” and willing to try the Scrum process.

### 2.8 Having the Right Coach

It is important for the fledgling agile team to have access to the “right” coach. It is very easy to for a new Scrum team to get off track due to team dynamics (among other reasons) or for the novice ScrumMaster to lose confidence in their grasp of the technique.

Another MS IT agile team, working on a project codenamed “Rome”, believes that uncertainty about their methodology had a negative impact on the early productively of their pilot and that they would have appreciated having a coach. Having an impartial third party to fall back on for guidance is believed by the LPO team to be a key success factor in their pilot project.

Initially, agile consultants have been hired to perform this role by some, but now that a few of our Scrum teams have matured, members have begun to evangelize Scrum to other interested teams, to seed new agile teams or to take on a formal mentoring role when it is appropriate.

### 2.9 Cohabitation

Microsoft software development groups are typically organized in personal offices. Furthermore many senior Microsoft employees are hostile toward the concept of a cube or bullpen, as it represents the workspace traditionally allocated to a vendor or contract employee. Also, within some teams, the
offices are grouped primarily by functional discipline (e.g., testers in one area, developers in another area). Sometimes they are on the same floor, sometimes on different floors. In the worse cases they may be in different building or different campuses.

Within Microsoft, if you are in the same building, you are considered “co-located”. This is a clash with the Scrum concept of co-location, where the entire team is in a single room. Since “co-location” already has a meaning within Microsoft, we have informally taken to calling the ideal Scrum state “cohabitation”.

In many cases the two challenges are: moving team members to be next to each other and tearing down the office walls. As a pilot project, most teams have not had these luxuries. Additionally, the number of available conference rooms is limited in most buildings, so it is not possible to “inhabit one”. To reduce this impediment, the LPO pilot team instituted two temporary workarounds. They reserved a conference room for a full hour for the daily Scrum, allowing the team 45 minutes of guaranteed team and room availability for key JAD sessions and other discussions. At critical times, they would block out work space in conference rooms. Because some team members do not have laptops, they have also had to find creative ways of being mobile. From a pilot perspective, the LPO team has found acceptable ways to overcome these obstacles.

As Scrum becomes more accepted and common within MS IT groups, we hope to find more sustainable long-term options. Several non-IT Microsoft teams, including Agile Alliance member Paul Hammond’s London based R & D (Research and Development) Team are setting the future vision with a complete workspace restructuring designed by the Scrum teams.

While some of our teams are co-located, MS IT has moved aggressively in recent years to globalize our teams. The “Shasta” Project Scrum Team had a group of vendors from China joining the team during the fifth sprint. Initially all member of the off shore team was invited to attend their daily Scrum but this proved unproductive. After moving the meeting time to 6PM PST and limiting the invitation to the vendor development manager the work of the off shore team began to fit into the flow of the method. This is not meant to say that on & off shore teams cannot be productive as part of a virtual team, rather that accommodation for culture, time zones and process also should be considered.

Some teams are split by waterfall phase, such as design/development in the U.S. and test in India. This will represent new challenges for the organization as we roll out Scrum to more applications.

2.10 Mixing Scrum and Waterfall

Many groups manage 50+ applications and at any given time, over half are in some stage of development or modification. With complex technical, scheduling, and requirements interdependencies, implementing Scrum beyond a limited pilot present other challenges. Today, it is not feasible to stop all systems to retool our processes. We are essentially forced to “change the tires on a moving car”.

We are currently examining various models for dealing with what we call “WaterScrum” - simultaneously managing co-dependent waterfall and SCRUM projects together1. WaterScrum is like trying to play two sports on the same field at the same time. Extra coordination is necessary to make it work.

The LPO team gained some limited experience with WaterScrum during a pilot Scrum project, which had a sister project using the Waterfall SDLC. In this case, the Scrum project was the supplier and the waterfall project was the consumer.

In the beginning, this worked very well in LPO’s favor. The Scrum project was able to start sooner and have several sprints completed before the waterfall project was at the point of consumption. However, one key issue arose that was not fully anticipated: the waterfall project would need to complete its design phase before LPO had completed all of their sprints, and because the Scrum project was incremental, some design aspects for later sprints would not be complete until after the waterfall project design complete milestone. This mixture caused some temporary grief.

In the future, WaterScrum projects will need to consider timing dependencies as part of the Product Backlog order and prioritization plan. Similarly, teams need to examine timing aspects for Scrum consumption from a Waterfall project.

2.11 Working with multiple test environments

Over the years, the use of traditional waterfall-based methodology has leads to role specialization

---

1 “WaterScrum” should not to be confused with what is sometimes called ”Scrummerfall”; the practice of running a waterfall process inside a Scrum sprint
and sequential code stabilization that goes through several test environments.

This is particularly true of client/server-based solutions typically found in enterprise applications. A typical project may advance through up to 5-6 environments.

In a waterfall process, code is iterated and each build is deployed to a development unit test (DUT) environment. This environment is primarily used by developers to test their own code.

When the code has been determined as stabilized, a build is deployed into another environment called the development integration test (DIT) environment. This is the environment where component testing often begins with build verification and unit tests being run.

When a build it deemed “ready” it is iterated and deployed into a system integration test environment (SIT) where end-to-end and system testing such as performance is done.

The last stop for a build is the user acceptance test environment (UAT) where user and other types of testing is done.

Every major build from release candidates to the final build will eventually have gone through each environment. This model while effective in a waterfall method does not adapt easily to the Scrum Model.

On the other end of the spectrum, the most minimalist environment approach would be a single integration environment that everyone shares. But, his may not be the most efficient method as resource crunches are likely, such as needing to run verification tests and performance tests like soak and stress concurrently.

The LPO team is currently recommending two environment models for environments using a Scrum projects – a simple and advanced environment model.

The Simple Environment Model is currently the recommended model. This model defines two environments, one for creation of new code (DUT) and one where the code is integrated into the system (SIT).

The Dev Unit Test environment may be as simple as a personal workstation or may include a set of shared servers that multiple developers are accessing concurrently, even though the code base on those servers may be changing dynamically. The second environment is a System Integration environment, where only finished code is accepted into the daily build.

The Advanced Model defines three environments used during the sprints, with a fourth UAT environment used after the sprints are complete. (This model matches the current definition of SDLC-Scrum; if SDLC-Scrum is updated to incorporate UAT into each sprint, then the Advanced Model would add a fourth environment). This model is still under evaluation for its efficacy as compared to the Simple Model. To reduce the total time spent getting code to the final environment, it is recommended that the team select a reasonable automated test suite to validate the intermediate stages (e.g., Dev Integration Test) to reduce the delay from a day to a few hours or less.

3. What’s Next

As the number of MS IT teams adopting agile methodologies and Scrum continues to grow, we must begin to consider the future and attempting to glean what will be required to keep this grassroots movement vital.

How can we institute best practices while maintaining the vitality and viral nature of the movement? How to mentor, but not manage agile teams? And most of all, how to better integrate agile methods into complex systems with complex dependencies?

3.1 Distributed Scrum

One of our biggest challenges will be incorporating Scrum into our world-wide project structure. Over the last several years, the LPO IT team has moved to a global resourcing model, with people in Redmond, Fargo, Ireland, and India.

At times we have maximized our waterfall model by being able to pick and choose different locations to execute each phase of the waterfall model, relying on heavy documentation as needed. Additionally, our product management function resides in Redmond, which is 12 hours off from India. As we move to a Scrum model, we will have to examine various distributed Scrum models to ensure we can maximize the benefits while overcoming communication-related issues from not being co-located.

3.2 Complex Dependencies

As we move from a single pilot application with limited cross-application dependencies to using Scrum across the organization, we will be faced with complex issues. One of our most pressing problems is that our applications have become more and more
entwined to support integrated end-to-end usage scenarios, providing users with great interoperability across multiple systems. However, each customer base views the system only via their process flow. We have multiple competing customers crossing multiple co-dependent applications, including dependencies on applications outside of our organization; multiple funding sources; and a mixture of Waterfall and Scrum projects.

For our software teams to be able to execute efficiently, we will need to push the complexity and prioritization upstream, scaling teams to meet demand while maintaining clean, simple prioritized Product Backlogs for each product.

3.3 When is a Duck NOT a Duck?

Due to the viral nature of agile adoption complicated by the sheer size and diffusion of our organization, a current challenge of our management organization is to identify who has adopted agile techniques as well as to determine if they are using Scrum as defined in the agile SDLC described above. We have described this problem as “When is a duck NOT a duck?”.

For instance, someone may declare “I’m a duck!”... “Sure I have no wings, feathers, bill or webbed feet, but I’m still a duck.” Similarly a team may say they are using Scrum but do not have a daily standup or Product Backlog.

In a recent “quick and dirty” survey (four questions) of MS IT development managers, we have been able to determine that nine development teams have adopted agile techniques and most of them self identify as using Scrum. The challenge remains to determine which of these teams are truly doing enough “Scrum” to be considered at “duck”.

A small management team has been formed to create a more detailed survey that will probe into techniques and training. Individuals identified as agile experts will also be interviewed. This information will be used to determine next steps that may include additional training and enhanced communication (i.e. distribution list & bulletin boards) to support the movements to maintain momentum.

3.4 Closing Thoughts

This is not the first attempt to alter and improve the development process at MS IT. The “Team Software Process” (TSP)\(^2\) was greeted with interest and enthusiasm. Some successful pilots were executed and interest grew. But, this movement which held much promise has begun to stall. One of the participants compared it to small campfires separated by great distances that have now begun to flicker out - one by one.

Our company culture may be a factor in the TSP phenomenon; Microsoft is a fast moving dynamic place, always interested in trying something new. This is why we are now embracing Scrum and other agile methods, but it is equally why in spite of our success we may be tempted to abandon it for the next new thing.

Successful pilots may not be enough to spread our agile movement. Our showcase team, LPO has well documented success; but other teams in the group have not yet adopted Scrum and we can only speculate why this is so.

Will our agile movement suffer TSPs apparent fate? What will be the future effect of changes in personnel, weak management or simple inattention? These are only some of the questions we must answer if we wish to keep our movement alive and growing.

While it will not be easy, MS IT’s agile teams are strong and committed. Time will tell if we will be successful in our quest to unseat waterfall as the dominant development method in MS IT.

---

\(^2\) The Team Software Process (TSP) from the Software Engineering Institute at Carnegie Mellon has been used at Microsoft across 100 projects in the last five years. TSP is a scripted process with the goal to enable a multi-discipline team to development, test and production support to deliver extremely high quality deliverables on time.