Dysfunctional Scrum: Making it work in a matrixed environment

Authors: Daniel Johnson & Ranata Johnson

Introduction

The Pacific Northwest National Laboratory (PNNL) is one of the U.S. Department of Energy's (DOE's) ten national laboratories. PNNL is managed by DOE's Office of Science and performs research and engineering for DOE as well as many other government agencies, universities, and industry to deliver breakthrough science and technology to meet today's key national needs. We have several thousand staff, and over 2,000 projects of all sizes, where each project has its own project manager that works directly with the funding client. Consequently, we have a very diverse workplace and work force where constraints, regulations, and guidelines are established for each project.

Furthermore, we have a highly matrixed organization that works across a large spectrum of science, engineering, and technology disciplines, and the majority of staff members are on more than one project at any given time.

We have several hundred staff responsible for the development and maintenance of a computing infrastructure to support PNNL's research agenda. We also have software engineers and computer scientists engaged in software development and are constantly looking to improve their software engineering skills and disciplines to produce ever higher-quality software.

This desire for improvement led a small group of staff in 2008 to attend an initial Agile Software Development with Scrum training course. This group began a grass roots movement to advocate for changing PNNL's software development culture and to implement Agile with Scrum.

Challenges

Software development is essential to the laboratory's mission, although it is viewed differently across the various organizations of the lab.

For research projects software development is often not the end product. We often develop new software applications in support of our research agenda where the project deliverables are data and scientific reports, not necessarily the software. The client may not see or use the software. Consequently, research dollars spent in support of improved software engineering are often scarce.

For engineering projects software development is often a part of a system engineering effort that delivers software, middleware, and hardware. The emphasis of the software is typically the integration of the various pieces of the system and the collection and retention of data and metadata. Consequently, the software development activity is not a standalone product but a part of the whole.

And lastly, we have projects where our clients hire us specifically to develop software applications that are very specific to their unique mission and must work within their specific workflow. On these projects, the software is the key deliverable.

A typical waterfall approach to software development includes significant lead time with an emphasis on establishing a set of requirements that are signed off and agreed upon. This represents a substantial risk to our projects which are subject to frequently changing requirements and the possibility of funding cuts on short notice. Therefore, a software engineering philosophy, such as Agile, with an emphasis on embracing change and frequent delivery of working software was very attractive.

Barriers

From the beginning, we understood that our corporate culture, or business model, would not provide for a typical implementation of Agile. We have several barriers in implementing textbook Agile with Scrum. These include:

- Our staff are located throughout our 600+-acre core Richland campus and in other cities. It is rare that teams are co-located.
- Our projects are the basic unit of work where every project is executed independently of every other project. While there are guidelines and best practices, no centralized authority exists to guarantee consistent implementation across projects.
- The majority of staff members work on more than one project at any given time. Typically, each team member commits a defined percentage of time, often 25 to 75% to each project. It is rare that an individual is 100% committed to a project. This team commitment approach is a corporate cultural phenomenon. It is a byproduct of the project being our basic unit of work. No project is immune from budget cuts or abrupt termination; therefore, having project staff spread their time over multiple projects is a necessity.
- Our true product owner frequently is not located at our facility and not committed to the project. Ultimately, our product owner is a PNNL staff member who is a liaison to our remote client and takes on the daunting task of trying to channel the client's vision of the product, effectively becoming a "proxy" product owner. In most cases, regular face-to-face communications with the product owner do not occur and are an ongoing challenge for each team. Therefore, the Scrum team is one additional degree removed from the product visionary.
- Flexible work schedules are also part of our corporate culture. The non-standard hours that many staff members keep can be challenging when you expect a team committed 100% and to be readily available.

We looked at the basic requirements for successful Agile teams and knew we would not meet some that are essential. Consequently, there was an immediate question as to whether we could, as an organization, be successful implementing Agile with Scrum. In hindsight, we were looking for a definition of what it would mean to be successful given our constraints. Other questions arose: Would the benefits of the rigor of Scrum and the accompanying Agile engineering practices still be sufficient to warrant the effort, given the inevitable compromises and/or adaptations of the process? Could we still

consider ourselves Agile software development capable given those adaptations? Would there be a positive return on our investment in time and resources?

As is the case with many organizations, we have experienced our share of failed software projects—projects that sustained budget overruns, projects that were unable to integrate software modules into a single viable application, software that failed to meet customer expectations in both quality and functionality. In spite of these barriers, we undertook the implementation of an Agile with Scrum environment, hoping to see the benefits of the process with an expectation of higher-quality products in an environment where we can sustain and replicate successful software development efforts.

Initial Implementations

Looking back, we feel there is something positive to be said for just jumping in. Sometimes, not knowing what you don't know results in progress. We were initially trained with an emphasis on the Scrum process, so our early adapters focused on process. Our initial product backlogs were good but not great. Our initial planning sessions were also good but not great. We did little to no roadmap definition or release planning. However, our initial projects consisted of small software development teams that knew enough about what their products should look like. These factors allowed us to jump right in and start implementing the process. We usually did a good job of defining goals for each sprint; we used Scrum boards hung up in empty offices, and the ScrumMasters took the responsibility of burning down task hours in a spreadsheet, which enabled us to produce the burn-down chart.

Those early implementations varied in style and team dynamics. Eventually, each project began to see the benefits of the process with the improved communications and the demonstration of working software to our product owners and stakeholders early and often during the project lifecycle. We were hooked.

We gained even more momentum when a couple of newly hired staff members who had Agile software experience with their previous employers were able to persuade their new project managers to implement Agile. They also exposed the process to new groups within PNNL. In this case, our highly matrixed organization paid benefits as staff who had experienced successful Agile implementations migrated onto other projects and became their new project's change agents.

Moving Forward

As more projects and staff learned about Agile, terms such as *Scrum*, *Iterative Development*, and others were frequently heard and often used interchangeably to describe various processes and practices that did not resemble Agile or Scrum as we had learned it. We had teams that started to claim they were Agile because they were performing an iterative development, running sprints, or having occasional "daily" Scrums. When the results of some of these projects were less than successful, the viability of Agile and Scrum was called into question; it started to get a bad reputation within some organizations across PNNL.

In response to these challenges, our small group of early adopters, along with representatives from several research organizations, formed an "Agile Steering Committee." Our goal is to educate both project teams and managers on what it means to be and to implement Agile and what it is not. We have management support (e.g., a small budget) to meet regularly. By this time, we had hosted a couple of two-day Scrum training courses and formed our Agile user group, which now has more than 100 staff members at PNNL. While only a handful of the people who have attended the Scrum training actually want to be ScrumMasters, we have described what it means to be Agile, and we have a means to further communicate how to implement it.

As word of Agile projects and their successes were becoming known, we started to see management interest and support to keep the movement alive. We had additional management support to help remove some of our barriers and impediments to help our teams be more successful. These included:

- The creation of a pair programming/test lab with systems committed to tools for implementing Scrum, continuous integration, and pair-programming stations.
- A Scrum room for multiple projects to share. We initially started with an empty single office where teams could hang their Scrum boards, and then moved to a larger office. Today we have a dedicated Scrum room available for multiple teams.
- Creation of a "Live Wall" display that provides local video conferencing with a shared touch screen window interface for displaying a team's Scrum board. This provides our non-co-located teams the opportunity to participate in the daily stand-ups in multiple locations and still have that face-to-face communication.
- Support and funding to maintain a web site that houses the Agile community of practices, where teams can share their practices and provide a forum for discussions.
- Invitations to and presentations by several Agile community guest speakers.

Improved Communications

Since our initial attempts at executing Agile with Scrum at PNNL, one consistent theme has emerged: improved communications. There are so many activities built into the process that are designed to increase and/or facilitate communications that every team we have interviewed or participated in has commented positively on the benefits of increased communications. Our experiences anecdotally support the concept that any team larger than three people typically fails to communicate effectively or to create the opportunities to discover the next great obstacle before it becomes too big to manage.

From facilitated conversations with the product owner to creating and grooming the product backlog, through the planning poker during sprint planning, and to the daily stand-ups, the process creates many opportunities to collaborate. The benefits of increased communications are everywhere. Nearly every stand-up we have run inevitably has led to side conversations that enable team members to calibrate their efforts and further define the product.

One ScrumMaster shared an example of a team member located 200 miles away at our Seattle office, who teleconferenced in every day for the stand-ups. Initially, he would say he had nothing new to

report and was still working on the same issue. After a few Scrums, the ScrumMaster started to ask more questions about what was going on with the work because the team member had accomplished little or nothing for several days. It turned out that the Seattle team member was dealing with a development environment issue. So we discussed this particular issue in the Scrum, and it became obvious that several other team members had already solved that problem. Soon, the Seattle team member was making steady progress and was much more effective as he had learned to leverage the work of his team. The entire team benefitted and it all traced back to the daily communications. This would not have happened without the rigor of the Scrum process.

Additionally, this was our first attempt at having a remote scrum teammate. At first those essential communications were not happening naturally as the team was normalizing and getting familiar with each other. In order for the team to remember the remote teammate, the ScrumMaster mounted his picture and placed it on the table next to the telephone.

Adaptations to Scrum

Project Initiation/Product Backlog Development

Our initial teams attempted to plan their sprints based directly on our training as ScrumMasters. We counted on the Product Owner to fully own the product backlog, to have it ready prior to planning, and then, in a four-hour planning session, we groomed stories and developed tasks. Many of these felt like marathon sessions that were often tedious and inefficient. We determined we were trying to plan our sprint with an underdeveloped product backlog; our Product Owners (even to this day) are often first-time Product Owners with minimal experience with Scrum and often do not truly own the product backlog to the extent necessary to develop it properly, especially at the beginning.

Today, our process typically begins with one to several meetings with the Product Owner, potential team members, the ScrumMaster, and possibly some stakeholders to iteratively develop a viable product backlog. In parallel, we begin the education of our new Product Owner, find available staff members that can commit some of their time to our new Scrum team, identify end users, and define or find our stakeholders. Because we rarely have everyone in place at the very beginning of a project, the creation of the product backlog and the identification of the people who will be significantly involved happens over time.

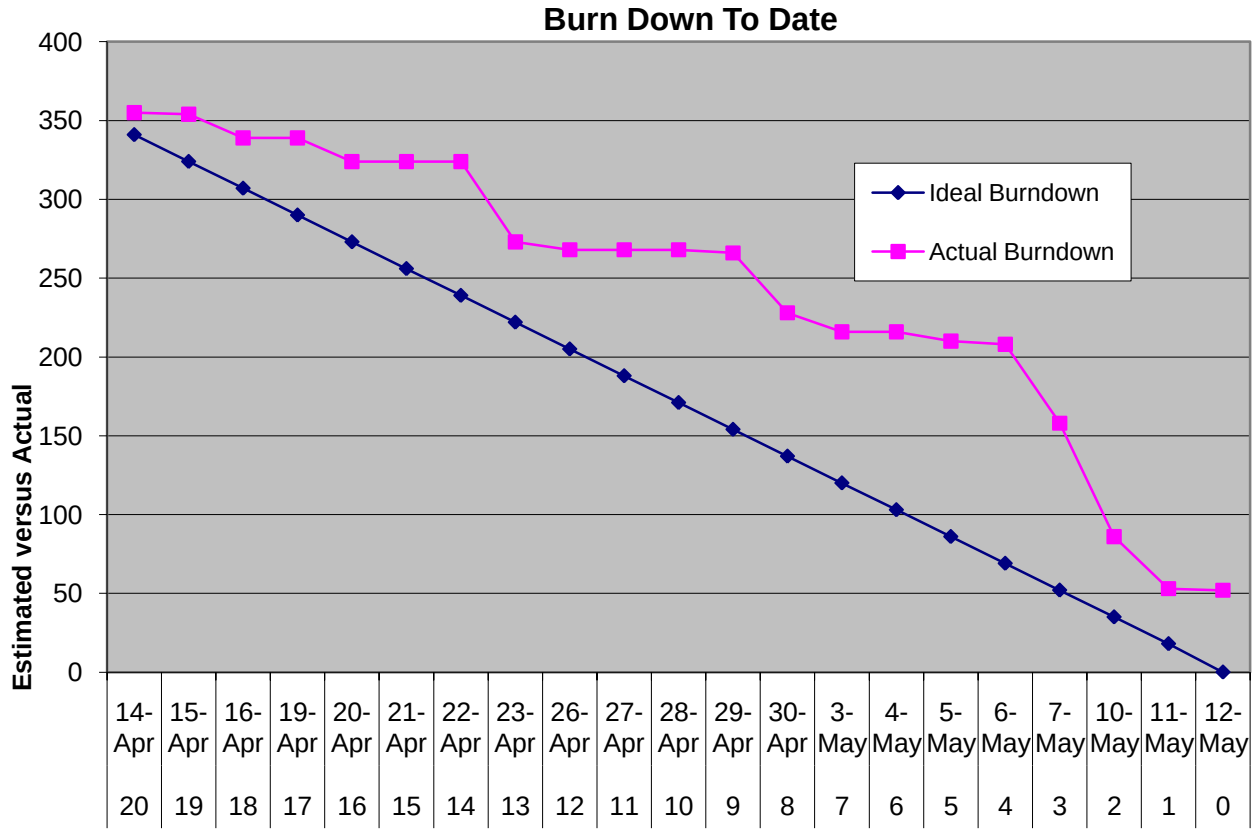
As we get more people involved, we host meetings to identify desired features, interview users to determine how they envision the new product, and/or meet with the stakeholders. Ultimately, we conduct one or more meetings to write our initial user stories in a group exercise.

This process is iterative and typically led by the ScrumMaster. When there is a new product for a new team, we usually start with the vision statement. However, when we have an existing product that is to be enhanced or refactored, we may start with a roadmap or an existing release plan. We may have a known set of features or even traditional requirements. In any case, we work through the creation of our initial product backlog until we have enough stories to execute a backlog grooming session.

Regardless of where we get our initial inputs (vision, roadmap, list of features), we are now usually successful when working with people who are new to the process and developing an initial set of stories that can be used to plan a sprint.

Sprint Grooming and Planning

We believe our backlog grooming and sprint planning sessions are typical of a Scrum team in any organization, with minor exceptions (see Capacity Calculation below). We have migrated over time to shorter sprints (typically two weeks) for most teams, enabling our grooming and planning sessions to run two hours or less. However, we didn't start out that way. Our teams were concerned about committing to two-week sprints because most of team members are matrixed and not committed 100%. The initial theory was that four weeks would be better because our capacity would be larger. Teams felt that having four weeks would allow them to balance their time better between projects and still meet commitments. Instead, staff tended to spend less time at the beginning of the sprint and focused on their other projects so they could be available at the end of the sprint (see Burndown Chart example). When this happened, our teams were unable to meet their definition of done; everything was getting completed at the end of the sprint. In an attempt to solve this problem, some teams tried two or three week sprints and then took a week off between sprints to focus on other projects.



We determined the shorter sprint length makes it easier to ensure our staff will be able to make their commitments to the Scrum team as they can more easily balance the commitments to their other projects. Once we kick off our first sprint, we have grooming and planning sessions on alternate weeks, preferably on the same day and time each week. This consistent routine tends to cut down on conflicts between projects.

Capacity Calculation

It has been difficult to accurately calculate capacity within our matrixed organization. We had to determine a method to calculate total hours for the team based on every individual's commitment. For example, a typical team in our environment might look like the following:

- Member 1 is available for 50% of his/her time for 10 of the 10 days in the two-week sprint
- Member 2 is available for 75% of his/her time for 8 of the 10 days in the two-week sprint
- Member 3 is available for 50% of his/her time for 7 of the 10 days in the two-week sprint
- Member 4 is available for 35% of his/her time for 10 of the 10 days in the two-week sprint

In our first attempt to calculate capacity, we used the following formula:

((% commitment *8 hours/day) * 70%) * Number of days in sprint = Individual capacity

We used 70% to indicate the amount of time actually spent working on tasks, assuming inefficiencies, interruptions, and context-switching between projects. Given that formula, Member 1 has 28 hours of capacity for the two week sprint:

((50% *8 hours) * .70) * 10 days = 28 hours total capacity

We calculate capacity for each team member and then add the individual totals to get a total team capacity for the sprint. We then use the total capacity during sprint planning.

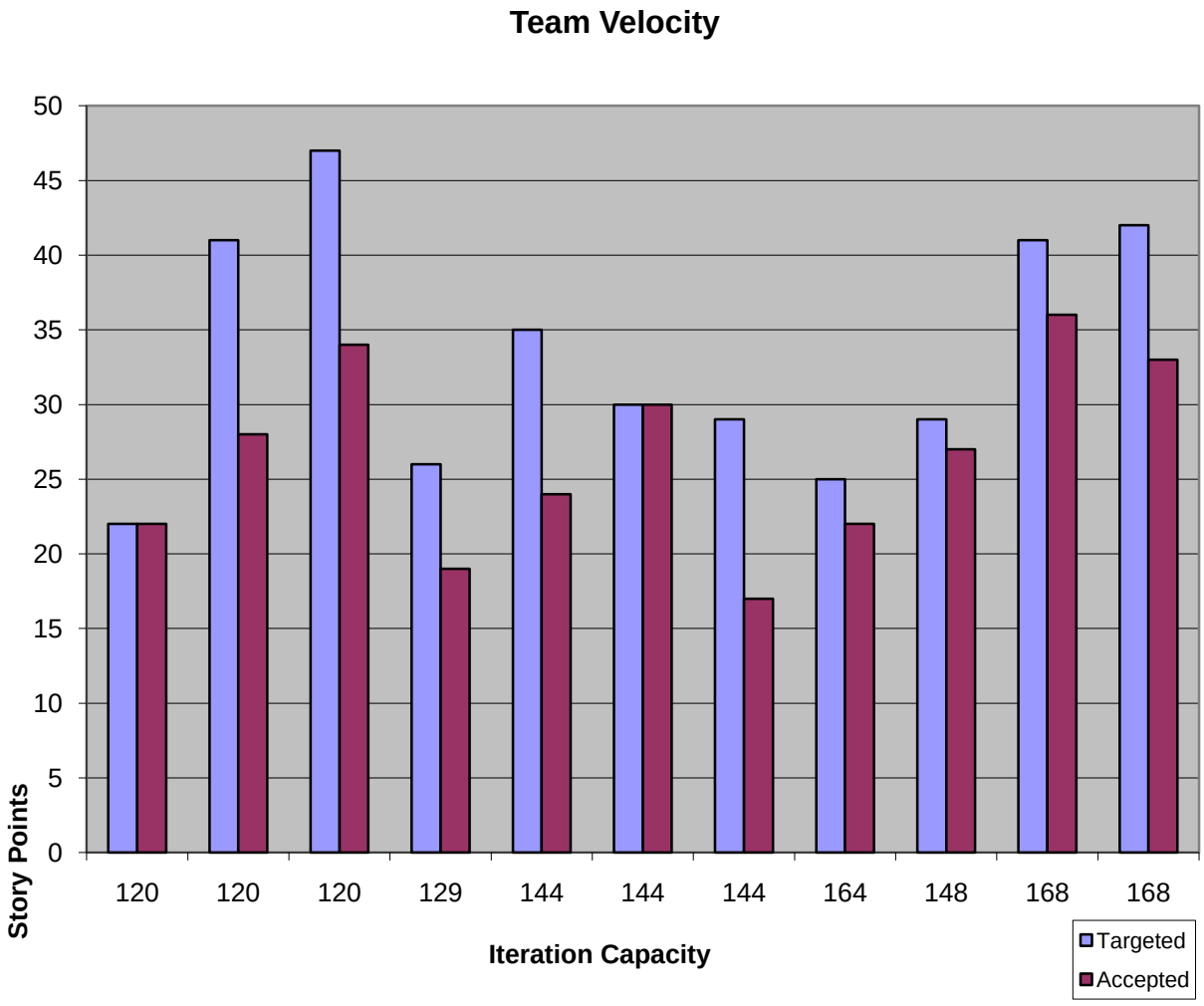
After a few attempts, we determined 70% was too optimistic and we are now using 60%. This seems to fit our organization better.

Velocity

We've had many debates on whether velocity is a useful or valuable measurement for our teams to capture. Does velocity tell us anything useful about our teams when commitments and team members change regularly? Because teams measure velocity for purposes of release planning, most have not tracked it; frequently, our sprint teams are formed with a single release in mind and an expectation the team will disband or move onto another project within a few months. Nevertheless, we encourage our ScrumMasters to capture velocity.

When we have longer-running projects, our teams do capture velocity for purposes of release planning. We measure velocity the way we learned in our Scrum training: it is based on the total story points from each sprint of successfully completed, demonstrated, and accepted user stories. In one case, we have a long-running team for which the team membership has not changed, although each individual's capacity

fluctuates between sprints. We found that as team commitments changed, the team delivered relatively the same amount of velocity during each sprint over time (see chart below).



The team's capacity changed consistently over the course of 11 sprints but the velocity was relatively the same over that time. We concluded that for longer-term projects, we can use the average velocity for planning purposes.

We have had several teams use velocity simply as a gauge during sprint planning to validate that they are not taking on too many user stories. These teams rely on capacity first and then check story points against the team's average velocity.

Daily Scrum

Team members who dedicated less than 100% to their Scrum team initially pushed back from the daily meetings, even for 15 minutes. One concern was the cost in time caused by the inevitable context switching from the person's other project to the daily standup, and back to the other project, plus the time to travel to the Scrum room from other buildings, as our teams are not co-located. Staff members

were concerned that they would not have anything to contribute if they had worked solely on their other projects since the last Scrum. As a compromise, some teams decided to meet for only two or three days a week. This idea seemed appealing and reasonable, but changed quickly. For example, staff from teams that met only on Tuesdays and Thursdays felt disconnected from the team when they missed a single Scrum because they had effectively missed an entire week. The door was opened for ScrumMasters to recommend daily meetings and to "just give it a try." In the beginning, we put no pressure on team members to come every day; however, some did and the feedback from the retrospectives of teams meeting daily was clear: team members saw the value in the daily meetings and that they directly affected team cohesion and performance for the better when held daily. So, as ScrumMasters, we began to push our teams to meet daily. Now, most teams have daily Scrums.

The improved communications at the daily Scrums have actually helped team members stay better connected, even when not working on the project every day. Teams see the value in knowing what others have done and what they should work on next. On new projects, some staff members who are new to the Scrum process object to the daily Scrum. The veteran Scrum team members advocate for doing it every day. We had to work through the assumptions and ideas on how to adapt to our matrixed environment. In this case, we have found the best practice is to meet daily, as prescribed, because the daily Scrum has shown tremendous value and benefits in keeping the team connected.

Impact on Staff

We have seen a positive impact on our staff members who have participated in this process. Although we have nothing measureable, team members have made some comments in various retrospectives that are worth sharing:

- Team members are happier and continually state they are having fun.
- The ability to self organize to accomplish the necessary work has empowered teams and they feel more in control.
- Team members have stated that the transparency this process brings to the work we do helps them to be more successful than they have been in the past, especially with our customers and clients.
- Team members feel more focused in spite of our environment.
- Team members feel better connected with the team and the customers they are working with.
- Individual team members routinely state in retrospectives that they feel great about delivering products that customers truly love and appreciate. The immediate feedback about the product from customers provides the team confidence in their ability to meet expectations.
- Some team members have stated they can't go back to doing it the "old" way. They prefer to have a product owner (not a project manager) that provides the vision and goals for the team and a team that decides the best approach to accomplish those goals.
- Many staff members who have been in a support role have stated they feel they are adding more value in the Scrum process than in traditional software development projects.

Summary

While we may be operating as dysfunctional Scrum teams at PNNL, where consistency of implementation will remain an ongoing goal, our experiences are showing a pattern of benefit from following the process as best we can. We have found value in socializing stories and user narratives to describe the user experience. We have teams that meet daily for 30 minutes and sit down while they do it, but it has proven valuable to them. We have had several successful teams deliver products even in our dysfunctional environment. Our ScrumMasters work multiple projects and strive for consistency but in our environment (R&D, engineering, and software development) we need to be adaptable and implement those aspects of the discipline that provide benefits to the teams and clients. Our implementations are not "by the book," but as long as teams and customers find value in it and our communication and quality of our software products are improved, then we expect to continue using Agile with Scrum.