

Scrum at a Fortune 500 Manufacturing Company

Richard Moore, Kelly Reff, James Graham, Brian Hackerson
3M Company
rjmoore@mmm.com

Abstract

In a mature fortune 500 manufacturing company, such as 3M Company, many processes and procedures for developing products have been put in place over the years. Many of these tools and procedures are very formal with a focus on developing and releasing products that are to be mass manufactured worldwide by the thousands. Software development at 3M has traditionally been force fit into these processes with less than optimal success. This paper outlines our experience in introducing the Scrum technique of agile software development into such a formalized environment.

1. Introduction

3M is a manufacturer of many thousands of products, and software has historically been sold by 3M as part of a total system or embedded into hardware that they sell. Recently, 3M has begun a focus on providing customers with turn key systems, including major software components, especially in the rapidly growing RFID Track and Trace market. Throughout its history, 3M has had many internal and external process management systems and procedures, most recently Six Sigma. Software development to date has been waterfall oriented and has been force fit into the business process of the current time.

When 3M decided to create a common development platform for Track and Trace applications to lower the cost of software product development and speed the creation of new applications, we wanted to try a new software development methodology. This decision was a cooperative group one made by the lead technical developer, the program manager, and the management team. The major driver for this decision was a collective frustration with our current software development procedures. The dynamic nature of software development programs we were running was not being handled well by the waterfall process. We

also wanted to try to lead 3M into new areas of software development. After some research, we decided to use agile software development with the scrum methodology. A new team of six developers started the development of a core library using scrum. Team size has increased to eighteen developers, with software testers being considered a developer. There are four scrum teams with 4-5 developers per team. Two teams are focused on development utilizing the framework under development, and two teams are focused on the continued integration and creation of a common library platform.

Our move to an agile/scrum development environment has required many changes, a lot of learning, and several adjustments. Seeing how scrum creates cohesive teams that focus on developing functional software and allows us to be reactive to new customer requirements and priorities has been rewarding. The areas we will discuss in this paper are developer acceptance, work environment, management support, process, tools, testing, and corporate processes and politics.

2. Developer Acceptance

The successful adoption of agile begins with getting acceptance and buy-in from the developers. We wanted each developer to be 100 percent committed to making agile successful. To accomplish this, we met with the initial team of developers and described the scrum process and our desire to implement it on the new project. We gave our commitment to the scrum process and stated the requirement for developers to do the same. They were given the option of being moved to another more traditionally managed project, an option which none of them took. We gave the team the book "Agile Software Development with SCRUM" to read and discuss. After reading the book and discussion, all team members bought into the process and in June of 2006 we started our first scrum.

The developers were open in accepting the scrum process, although some have later admitted that they were unsure and skeptical about scrum in the early going (now they are some of the staunchest supporters of scrum), and none of them have ever expressed an interest in turning away from it. We attribute this to the focus on software development and minimizing outside distractions during a sprint. If software developers can do what they love to do they will enjoy the process. Integration of new team members has worked well and the high level of team interaction helps them come up to speed very quickly.

Were we scrum experts right away? No, we blindly implemented what we had read, and within a couple of sprints were doing the basic mechanics of scrum. Despite this, we had a feeling that we were missing things and lacking experience that would allow us to be more efficient. To help the team in becoming more efficient at scrum and agile, we had team members attend the Agile 2006 conference and held an on site three day agile/scrum training session. Four months later, it was apparent that we were floundering and still not being as successful in using scrum as we could be. To help guide us in a more thorough use of scrum we have engaged an onsite consultant to help refine and improve our scrum process. It has been beneficial to have somebody with numerous scrum experiences help us. To become more self-sufficient, we are training team members as scrum masters and in other areas of scrum to help run the day to day operation of our development process. We have one certified scrum master and will be seeking training for others. As we will discuss later, we are continually refining our process and tools to help the teams work more efficiently.

3. Work Environment

Creating a work environment to support scrum was quite challenging. The creation of a non-traditional work area better suited for team collaboration (and an easier implementation of scrum) required us to interact with several other groups within 3M, and that was not always an easy task. We co-located the team in an office area with a common open area around which low wall cubicles were placed. The low wall cubicles were a compromise over a truly open work area based on developer concerns about not having some minimal privacy, and facilities management about not using standard 3M materials for office areas. Design of the team area was led by management, but was a highly cooperative effort with considerable input from team members into the layout and office style. It took four

months and numerous meetings with facilities management to get the office area approved and designed.

To help create an atmosphere of a new and different work environment, we had the walls painted different colors than the generic beige found in all 3M buildings. The simple action of painting walls different colors took three weeks of bickering back and forth with facilities management until they agreed to let us pick from a small palette of other 3M approved colors.

In the end facilities management was not happy with the results; they considered the open middle “wasted space” where they could have placed 5-6 more cubicles. They were also not pleased with having to modify the standard 3M cubical walls for each office to create a more open feel. Management support in driving this non traditional office design was critical, since facilities management finally gave in to our management’s repeated requests and their willingness to pay for the extra work.

We have found an open area wonderful in encouraging impromptu collaboration. Team members can quickly see if other team members are available, unlike a typical row/column cubical office arrangement. Compared to existing cubical arrangements, the energy level and continuous collaboration are noticeably higher serving to energize the team. Designing a team room focused on collaboration has been beneficial to implementing a scrum work environment and has improved the focus and cohesiveness of the team.

Just off our common area is a meeting room that is reserved for the group's use for scrum, design, and planning meetings. This room cannot be used by other groups within 3M or officially scheduled, which takes some effort in a meeting intensive company. The focus of the room is for allowing spontaneous meetings by the team to address issues as needed. As our number of scrum groups using the room increases, there is not enough wall space to hold paper documentation of scrum backlog tasks, retrospectives, user stories, etc. We have begun the process of using electronic tools for the management of our scrum process, which is facilitated by having a projection system in the room. Electronic tools have been effective in streamlining our process, and have helped to integrate off-site programmers.

Due to rapid growth of the team we have run out office space in the team area, and have noticed that team members located outside of this space have a

harder time being part of the scrum process. They are less involved with the process and less likely to participate in impromptu design meetings and conversations. Work has started on finding a solution to co-locate the whole development team. Co-location had been delayed due to a planned building move in a year. Nobody wanted to spend money remodeling if we were going to move, but recently the move was called off. We have determined that co-locating scrum groups is important to the success of implementing scrum and are working hard to convert a space quickly.

Creating a work environment and space to facilitate scrum has been critical to our adoption of agile and greatly contributes to better team cohesiveness and collaboration. While the changes we undertook would not be considered radical by most software development companies, we had to fight resistance to change and perceived wastefulness at each step.

4. Management Support

One thing becomes obvious in trying to drive institutional change, without strong management support you are not going to make it very far. Fortunately, management support for our move to an agile software process has been very strong and encouraging. Our management team has consistently supported the non conventional room design and scrum methodology in the face of a 3M culture that does not take change easily and smoothly. A big area of support has been in their willingness to invest money in facilities, training, and tools.

This support has been a critical component in the success of the team in implementing scrum and we would not have achieved as much as we have and as quickly without it. Without this support and encouragement we would not have been able to create an atmosphere that encourages the collaboration needed to run agile software development.

Scrum has the advantage of being very visible and its focus on functional software shows results to management that they can see and get excited about. The use of an electronic tool for managing scrum allows management to see team progress on line any time they would like to. Sprint reviews keep management in the loop on the program and allow them to provide feedback on the project. Constant exposure to the work creates a level of excitement for management and in turn increases their support of the process.

5. Process

Much like the production lines 3M runs, scrum requires some tweaking of parameters to find the best match for a team and environment. The mechanics of scrum seem deceptively easy, but it took us some time to figure out the best scrum interval, scrum meetings, product backlog creation, sprint planning, and release planning.

Determining the best sprint interval has been a bit of an adventure. To start, three week sprints were selected with a three day gap for review and planning between the sprints. After a few sprints the teams felt that they were just getting started when the sprint was ending, so it was decided to try four week sprints with a three day gap for review and planning. The three day gap was an attempt to avoid developer burn out from sprinting so long. With a four week sprint the time to plan each sprint became very long and drawn out. During the third and fourth weeks developers were either running out of work or the planning was not complete enough. Product owners did not see waiting four weeks to start new high priority task as being very agile. One of our projects has a hardware development team that relies on the software team. A four week sprint did not allow us to support their changing requirements.

Moving to two week sprints with the last day being a sprint review and retrospective and the first day of a sprint being the planning with no gap between the sprints seems to be much better for many reasons. We can plan design tasks during a two week sprint and then more thoroughly plan the work to achieve the designed subsystem in the next sprint. Our ability to incorporate new feature requests is much more responsive, and the rhythm of a two week sprint seems to avoid the burnout and stress felt during a 3 or 4 week sprint. We are forced to break down larger user stories into smaller stories that can be worked on during a two week sprint.

Scrum meetings are meant to be quick meetings. Our scrums are held in the afternoon to allow developers time in the morning to concentrate on programming. One of our team norms is to not schedule meetings in the morning in an attempt to have a large block of non meeting time. The length of scrum meetings tends to increase over time, and the main culprits are late arrivals and expanded conversation around a topic or impediment. Starting a meeting on time is an issue at 3M; typical meetings start 10 minutes late. Our scrum meetings tend to be better than

that, but do tend to slip. We have not resorted to fines or other tricks to get meetings to start on time. Usually an email reminder and just starting the meeting on time no matter who is there seems to help. Engineers want to solve problems so there is always the temptation to expand on a subject and try to solve it during a scrum meeting. To avoid this we have a “pounding the table” signal to stop the conversation and leave it as a post scrum meeting topic so that only those interested need to be involved. Scrum masters are critical in getting a scrum meeting rolling and keeping the focus on the essentials. Any issues or impediments that come up are put on a Post-it® note and placed on a team parking lot or impediment area on a wall. To facilitate cross team communications we have an area to post messages in the scrum room and have a scrum of scrums every week with the scrum masters from our four teams.

Developing a good product backlog has been the hardest part of thoroughly implementing scrum. One tends to underestimate the time and effort involved in getting the backlog in place and developing the user stories for each backlog item. With a well fleshed out product backlog of user stories the planning process for each sprint goes much smoother. For many sprints our product backlog was not well defined and minimal. Much of our planning process was being spent defining product backlog items before we could break them down into sprint backlog tasks. To improve this process we have dedicated resources to continually refining our product backlog and expanding high priority user stories to have them ready for sprint planning. This has streamlined our sprint planning process considerably, allowing teams to easily plan the next sprint in a single day or less.

Our sprint planning day is split into two sessions. The high priority user stories are reviewed and refined as needed for clarity in the first session. Bugs and sprint retrospective issues are also reviewed to determine which are to be addressed during that sprint. Based on team capacity, a number of user stories, bugs, and retrospective issues are selected to work on during the sprint. This process usually lasts 3 hours in the morning, and at the end of the meeting each developer takes responsibility for creating sprint backlog items for one or more items. Sprint backlog items are created over an expanded lunch break, and the team meets a second time to finalize the sprint backlog items.

At first, sprint backlog items were written on Post-it® notes with somebody scribing them into Excel and then importing them into our sprint tracking system. Currently, developers enter the sprint backlog items into our electronic tool and the team reviews and

revises the tasks electronically. This allows the team to leave the meeting room with all sprint planning captured electronically, and helps our remote team members to follow along. The only disadvantage is the painfulness of watching somebody type, although luckily the revisions tend to be small.

6. Tools

Shifting to a scrum based software development process has required us to use electronic tools, as we have a dispersed development team, with people in other buildings on the 3M campus, one developer in Florida, and another in Wisconsin. In searching for a tool, we wanted one that would support scrum metrics and process flow, was integrated into our development environment (Microsoft Visual Studio 2005), and was accessible via a web browser for non developers.

Our search did not turn up any tools that do all of this in an integrated, one tool approach. The closest for our environment was the new Microsoft Team Foundation Server (TFS) with its scrum plug in. This tool is integrated with our development environment, provides an automated build framework, has some nice features available, but suffers from a typical version 1 incompleteness of features. Fortunately, TFS is a flexible architecture that allows users to modify the use of the product through templates and plug-ins. We decided to go forward with TFS and dedicated two developers for a month to create new templates to match the functionality we want from a tool for managing our scrum workflow.

An integrated tool has the advantage of having user stories, sprint tasks, bugs, issues, and retrospective information all stored in the same database. We can create links between related items to create traceability for each user story. Integration with our development environment forces developers to associate a sprint item to source code when it is checked in. While creating a tailored tool for managing our scrum process is not what we desired, it seemed to be a better alternative than purchasing tools from separate vendors and trying to tie them together ourselves.

The development team has covertly put a team Wiki into action to serve as a shared collaboration tool. The Wiki is maintained by the team and hosted on a local test machine. Covert action was required since the 3M IT organization has not embraced Wikis and recommends using Lotus Notes team rooms. Our experience with a Notes team room left a lot to be desired and we wanted a much more interactive tool.

For collaboration, the Wiki has been a huge benefit to the team and fits well into the scrum process. Developers can do a lot of design concepts and discussion on line to avoid having a lot of formal design meetings, and the design discussion is captured on line and is searchable. A year later the IT organization is just starting to test a new Notes based collaboration tool and not recommending Wikis. We will be sticking with our current Wiki tool.

8. Testing

Many aspects of agile development encourage better testing of software. Including the constant building of code, integration of software components for sprint reviews, unit testing, test driven development, and the integration of testers as part of scrum teams. We have implemented all but test driven development into our development process, but we are considering its use. The struggle has been in integrating the traceability that testing needs into the scrum processes we are using. We anticipate a potential for software audits in medical and other markets.

We are working to incorporate testing procedures and tasks into our sprint planning. Currently, we make sure that each product backlog item (user story) being worked on during a sprint has associated testing sprint backlog tasks. Problems discovered during testing are entered as bug items for that sprint and assigned to the appropriate developer to be fixed. Unit tests are created by developers and we create separate sprint tasks for developing unit tests. All testing tasks are linked to the parent product backlog being worked on to provide a way to trace what testing has been done to validate that product backlog. Many times testing requires acceptance test procedures; these documents are stored in our electronic system as attachments to the appropriate sprint backlog item.

We are working to create a method of traceability using our custom plug in to Team Foundation Server to extract a hierarchy of product backlog user stories, sprint backlog items, and bugs into document format. The ability to create the necessary documentation will help for any audits we may encounter.

9. Corporate Processes and Politics

A major challenge to incorporating scrum has been working with other groups within 3M that use more traditional tools and processes. Our two application teams have product owners that are outside our group

in a different business group. These product owners are very attached to their product requirements documents and formalized scheduling. We have heard the same request many times; could you get this into Microsoft Project and a Gantt chart so I can see the exact schedule? They see the agile/scrum process as being very out of control and not conforming to the 3M way of doing things.

Gantt charts or other formal project schedules and specifications are seen as a contract to budget against, even though these schedules and specifications are usually invalid after a single month of development. The team has worked on creating story boards to show product owners sprint timelines, and the product backlog items that are to be addressed during each sprint. These story boards are a simple 4x8 foot sheet of foam board with Post-it® notes that can be moved quickly and interactively based on product owner input to visually see schedule and feature trade offs. We are currently revisiting the story boards and creating release schedules and user stories for all our products under development. Including product owners in planning helps them achieve a higher level of confidence in scrum.

If requirements exist, they are in many different forms; product requirements documents, software requirements documents, feature spreadsheets, marketing request documents, or maybe no documents at all. We are not in a position to specify what format business units will use for specifying a product, and are taking the approach of translating these documents into product backlog items as user stories. Conversion to user stories is a valuable process that helps to flesh out a more complete specification of the desired product.

One of our development teams has been focused at developing software for a medical specimen tracking system. The original product owner had a list of product requirements in Excel format. We worked with the product owner to clarify these requirements and to estimate the relative effort for implementing the stories. To determine which features could be accomplished by what dates, the estimates were placed on a task planning board using Post-it® notes. After four months, 3M created a new business unit to sell this application (an expected event), and a new product manager was assigned to the program. The new product manager is very entrenched in waterfall software development, and wanted formal product and software requirements along with a detailed product time line. This difference has become a major issue, and has been enhanced with the addition of developers in the new business unit. The product manager wants

to run that team separately using traditional software methodology. We are working to rectify this situation and to have a user stories be the preferred method of capturing system requirements. Many meetings have been involved and the scrum team has skipped a scrum to work on refining the user stories. This issue has not been completely solved to date, and the solution may be to have our group of developers work using scrum (they are not willing to change) and the business unit's team using a traditional approach. As evangelists of scrum we are still working on a totally scrum solution by continuing to educate others on scrum.

10. Conclusion

Implementing scrum in a fortune 500 manufacturing company has challenges and frustrations, but the outcome has been valuable to our software development process. Although we do not have historical data to compare the performance of the team using scrum versus a more traditional waterfall process, the consensus of those involved on the team and in management is that the team is much more productive and is able to incorporate new features more efficiently by using scrum. Even if team productivity was only the same as more traditional software development methods, we believe that scrum provides many advantages to the software development cycle over waterfall that would make scrum the methodology of choice.

Much of the perceived increase in productivity comes from the focus that scrum places on delivering functional software. Developers are not caught in an endless cycle of creating the perfect solution that will do everything, handle every condition, and with many features that will never be used. With two week sprints, decisions must be made and cannot be deferred. Experience is showing that a fast bad decision is much better than a long drawn out decision (and potentially still bad). To make a decision and find out in two weeks or less that it is incorrect does not have the adverse effect many think it would, and what you learn from a poor decision may be valuable and allow you to quickly explore and eliminate ideas. Fast decision

making is counter to large company thinking, and many are uncomfortable making such decisions.

Quick reaction time is a benefit derived from using scrum. To be able to tell a customer that in two weeks or less you can start work on their newest high priority item seems to be a satisfactory answer that product owners can live with. Experience on other software development programs has shown that developer silos can be very counter productive and having a developer hole up for weeks or months at a time can lead to improper implementation and wasted time. The forced status update of daily scrum meetings does not allow a developer to get off track and reviewing progress with the product owner every two weeks keeps wasted effort to a minimum and allows for fast paced correction of effort. Another benefit is the increased customer interaction throughout the development process. This benefit is enjoyed by both product owners and teams leading to much more functional software.

There are other smaller benefits to using the scrum methodology in addition to the larger benefits just mentioned.

- Team energy and excitement
- Recruiting of new developers
- Communications, in team and out of team

From our standpoint the effort needed to bring scrum into the software development environment of a large manufacturing company has been well worth it. We continue to evangelize others in 3M, and also see many potential applications for scrum for other non software areas within 3M.

11. References

[1] Schwaber Ken and Beedle Mike, "*Agile Software Development with SCRUM*", Prentice Hall, Upper Saddle River, NJ, October 15, 2001.