Scrum

In Sebastian Junger's *The Perfect Storm*, a fishing boat out of Gloucester, Massachusetts encounters a storm—not just any storm, but "the perfect storm"—and ends up meeting a tragic end. This "perfect storm" was the result of a combination of various meteorological phenomena. Together, these phenomena made it impossible for the fishermen to do their job and bring swordfish back to port.

In reading *The Perfect Storm* and watching the movie based on the book, I felt a strange sense of déja vu. I eventually realized that the environment I work in is very similar to the one that the boat's crew was working in. After all, the software industry is nearly as chaotic and unpredictable as the open sea. Those of us in the software industry often feel as though the world is conspiring to smash our projects to smithereens and ruin all of our well-laid plans.

I have managed a great number of software development projects. Some of them have succeeded, and some of them have failed, but all have been undertaken under dangerous and complicated circumstances. The main difference between the crew of the *Andrea Gail* and me is that I've lived to learn from my mistakes.

During the course of my career in the software industry I've learned that software cannot be developed without flexibility, adapability, and initiative. This is especially the case given the ever-changing nature of today's economical and technological requirements. I've learned to eschew detailed, one-size-fits-all, cookbook processes and methodologies. Though they may work in fair weather, projects that rely on them will only founder and capsize in rough seas.

I recently attended a business meeting at a hot software startup in which the company's Chief Executive Officer excitedly described how he felt seeing so much new business rolling in: it was, he said, something like "standing on a beach and seeing a 40 foot wave head toward shore." The company's leaders were preparing to take drastic measures to survive the impact of the huge wave of business that was about to hit them.

In the context of our "Perfect Storm" metaphor, this company was a fishing boat with the motor out for repairs with a hurricane, tidal wave, and a cyclone all approaching. Fortunately, this company uses Scrum and therefore stands a good chance of surviving all of this. People at this company will neither panic nor cling desperately to obsolete plans and useless structures. Instead, they will respond intelligently and coherently to the challenges that they encounter. The company will keep its bow into the wind and perform as well as it can under the circumstances.

The History of Scrum

If an organization fails to achieve its objectives, it isn't necessarily because its members haven't worked hard enough. More often, an organization fails to achieve its goals because it has been distracted by other factors and has lost its focus. The corporate world is a chaotic world, and the number of things that can distract a team and knock it off course are almost innumerable. The primary purpose of Scrum is to help teams to focus on their objectives and to help them avoid getting thrown off track by other less important concerns.

At the request of the Object Management Group, Jeff Sutherland and I sat down to summarize what we know about product development. Jeff Sutherland and I have both had a great deal of experience managing the development of commercial software products, and so we had a great deal to say on the subject. As we worked together to summarize what we have learned over the years, we realized that methods we have found to be the most effective together comprise a coherent methodology. We named this methodology Scrum.

Since its introduction in 1995, Scrum has been used to build a wide range of products, including data warehouses, Internet portals, voice recognition systems, telemedicine products, Y2K initiatives, middleware, and even DWFM optical laser hardware. In this article I relate some of the experiences I've had introducing various corporations to Scrum.

The Nature of Scrum

> Jim Coplien of AT&T Labs once said, "Scrum is getting the best compliment possible: people are saying, 'Oh, that's what I already do.'"

If you learn only one thing from this article, I want you to learn what I have learned during the years I have spent implementing Scrum: common sense, simple practices, and paying attention yield tremendous benefits. In our industry, we tend to rely on the complicated, the aggregated, the sophisticated, the hard-to-do and intellectually challenging. Sometimes we have to make an effort to keep things simple. We ought to keep Okham's Razor in mind: the simplest solution is usually the best solution.

Scrum has been categorized as such a lightweight methodology, or minimalMethodology. I view this categorization as a compliment. After all, a minimalist methodology is, in essence, a relatively natural and unobtrusive methodology. Minimalist methodologies change only what *needs* to be changed. They leave everything else alone. Scrum is in essence no more than a collection of reasonable practices that are woven together with a few rules, a couple of expressions, and a good dose of common sense.

Scrum Primer SIDEBAR

Definitions

Scrum team: A cross-functional team of 5-8 people who focus on building a product or part of a product
Scrum meeting: A 15 minute daily status meeting where the scrum team shares status and identifies obstacles for management to remove
Sprint: A development iteration, usually lasting 30 calendar days, during which the team incrementally and demonstrably adds to product functionality

Backlog binds these three elements are all held together:
Product development is driven by a product backlog. The backlog consists of a list of the features, facilities, content, and technology that developers plan to incorporate into the product. The product manager maintains this list and determines the priority of its various items. The priority of each item is important, as items of higher priority will be tackled before those of lower priority.

Before a Sprint, the team meets with product manager and other interested parties. Based on current product functionality and emerging market requirements, the customer and team select from the product backlog the items that the team believes it can accomplish during the next sprint. The product manager has last word on which items to tackle, and the team has the last word on how many items to tackle.

The Daily Scrum: The Meeting to End All Meetings

Captain Billy Tyne was best able to assess the condition of his ship and the situation it was in by looking out his bridge windows. His Global Positioning System (GPS, a satellite triangulation system), for example, could tell him his geographical location. His barometer measured the air pressure. And his fuel gauges indicated the amount of fuel he had left. However, none of these tools measurements was of any use to him. All the Cap'n had to do to determine how bad of a fix he was in was to look out the bridge windows. The bad news is that the *Andrea Gail* sank during the perfect storm. The good news is that Billy Tyne didn't have to read any reports to know that they had capsized; they had first hand experience to rely upon.

Most managers don't get feedback as direct as the feedback Billy Tyne got during the storm. Checking "Project Dashboards," reading status reports, and attending weekly review meetings are all mere substitutes for real involvement in project, and all too many project leaders rely on them. Scrum offers simple ways to get direct feedback and renders obsolete all other managerial gadgetry. I find that the best way to comprehend the complex interaction of people, technology, needs, competing interests, and satisfaction is to observe that interaction first-hand.

The word "Scrum" refers both to a methodology and to one of the primary features of that methodology. The Daily Scrum is a quick meeting at which team members report what they're doing and what obstacles they are encountering. Daily Scrums provide direct, first-hand feedback and status checking to a project team and management. You can learn as much about a project by attending a Scrum as Captain Billy Tyne learned about the *Andrea Gail* by looking out his bridge windows. Daily Scrums are a way to get direct evidence of the progress being made on a project and to get an idea of what the chances are that the project will succeed. The best way to begin implementing Scrum is to establish daily Scrum status meetings.

The Power of the Daily Scrum

Other Scrum practitioners and I have been astounded by the power of the daily Scrum. Here is testimony from Linda Rising and Norman S. Janoff, who implemented Scrum at AG Communication Systems:

> "The team began to cooperate almost immediately. The changes happened before our eyes. One plausible explanation is that our developers are superb engineers; they love to solve problems. When one team member shares an obstacle in the Scrum meetings, the entire team's resources come together to bear on that problem. Because the team is working together toward a shared goal, every team member must cooperate to reach that goal. The entire team immediately owns any one individual's problems."[1]

Scrum makes visible to all the inner workings of a given project. Transparency is a good thing in and of itself, but it also has some surprising side benefits. One of these benefits is the degree to which transparency reduces the amount of time project leaders are forced to waste trying to get everyone on the same page. Scrum makes clear to all team members what the project's goals are and what their individual goals within the context of the project are. Another benefit of transparency is the degree to which it reduces the amount of time wasted on politics.

Scrum also brings team members together and thus encourages collaboration. Scientists and engineers are by nature solitary creatures: though willing to be social, they would prefer to stick to themselves. Daily Scrums overcome this tendency by requiring team members to sit in the same room and discuss their work with each other. The benefits of this simple interaction can be absolutely astounding.

[1] L. Rising and N.S. Janoff, "The Scrum Software Development Process for Small Teams," IEEE Software, July/August 2000, pp. 26-32.

Team members begin to see not just their own piece of the pie, but also the pie as a whole. They gain an understanding of the context in which they are working, which is absolutely invaluable. Collaboration drastically increases the strength of both the team as a whole and the individuals who comprise it. So does communication, which also increases. As team members describe the work they do, the team as a whole helps its individual members to work better.

The daily Scrum can transform a team from a group of individual engineers working away on seemingly disparate problems into a team of engineers who help each other to accomplish a goal. The team eventually becomes completely self-organizing, self-regulating and begins to own the projects that it works on.

Without daily Scrums, perhaps the biggest obstacle facing a team is the preponderance of meetings that its members must attend. People from all over the company, from manufacturing to marketing, call engineers into "status review" meetings so that they can hear about the progress the engineers are making. Such meetings typically require the presence of the entire team and usually last for two to three hours. Ironically, calling upon a team to describe its progress will only impede it from making progress.

However, daily Scrums meet at the same place and time every day. Anyone who wants to see how the project is going only has to come to the meeting and listen. Also, when problems arise, people can determine whom they need to work with to resolve them during the daily Scrum meeting. Not only are technological problems and obstacles to productivity spotted sooner because of Daily Scrums, but they are also more efficiently resolved.

Incremental Delivery through Sprints

Aboard the *Andrea Gail,* Captain Billy Tyne faced a series of challenges. More literally, he faced a series of enormous waves. Tyne tackled one wave at a time rather than tackling the storm all at once. This approach helped him to last through as much of the storm as he did. Any other approach—say, tackling the entire storm at once—would have been too much for Tyne and his crew. Tyne's approach is like that of Scrum in that it is fundamentally incremental and iterative.

On the open seas and in the software industry, an incremental and iterative approach allows progress to be made in a thoroughly chaotic environment. Tyne reduced the storm to a manageable dimension by viewing it as a series of waves. Similarly, Scrum reduces a project to a series of what I call "sprints." A sprint is period of time during which the team incrementally and demonstrably adds to the product's functionality. That is, the team further develops the product.

I've found that teams are capable of performing heroically. But being heroic can really wear a team out, so Scrum constrains the heroics to 30-day sprints. Every organization that uses Scrum wants to alter the duration of a Sprint. Doing so could be compared to taking a Julia Child recipe and throwing in some additional spices. Sutherland and I initially chose thirty days as the duration of a Sprint for the following reasons:

- During thirty days developers can get their arms around the design, coding, and testing of some major functionality
- The marketplace changes so rapidly that product redirection is desirable as frequently as possible

Some development organizations need to prove to themselves that Scrum works and need to prove to their users that they can build products. I've started such organizations out on 15 days Sprints, just to establish credibility and to set a rhythm for the development work they do. The gains the organization makes in trust and credibility are worth the initial loss of productivity.

Sprint Reviews

The crew of the *Andrea Gail* and other fishing vessels hung out at the Crow's Nest, a bar in Gloucester, Massachusetts. Having risen to the rank of Chief Mate in the Merchant Marine after years of work on freighters and research ships, I know bars like the Crow's Nest well. At the bar you figure out where you stand and what you ought to do next. Crew members make plans for the future on the basis of the relative success of their past trips.

Software product development is in some ways very similar to long line fishing for swordfish as Junger describes it in *The Perfect Storm*. With fishing, you don't know how the weather will affect your voyage, whether if you'll find the fish, whether the ship will hold up, and what is the market value of your catch until you return to port. When you build software products, you don't know how much you'll get done until you try, you don't know how the users or market will respond until they see what you've built, and you don't know how technical advances, the competition, and the economy will affect the fate of your product.

Considering that the most valuable software is built on the cusp of technology advances, the crew of the *Andrea Gail* had it easy. They at least knew how to use their gear. By comparison, in a software project that uses advanced technology, everyone is a beginner to varying degrees and how well the technology will work is unknown until someone tries to use it.

At the end of the Sprint, team members and other interested parties gather together. These other interested parties are referred to as users, and are usually members of marketing, management, and alpha customers. At these meetings, the team relates its successes and failures, the opportunities that they see, and the risks they've come to understand. The users then discuss the market, the needs, the possibilities, and the demands. I've seen end-of-Sprint demonstrations turn into brainstorming sessions that last all day. The team and the product owners then determine what to do during the next thirty days.

I once worked at a financial services company that was considering how to improve a product that was both used and developed internally. Everyone was fed up with the product and desperately wanted to either fix or replace it. We instituted monthly Sprint review meetings at which the users and developers could discuss their needs and plot a chart for further development of the product.

The developers showed the users where they were in their development of the product and discussed with them the successes they'd had and the problems they'd encountered. The users then told the developers what was most important to them and explained what aspects of the product were causing them difficulty. Contrary to the expectations of the developers, the users ended up supporting technical innovation. And contrary to the expectations of the users, the developers were able to get a lot of important functionality incorporated into the product relatively quickly. The product ended up being totally revamped.

Sprint reviews get people talking. They share their frustrations, hopes, and successes. They then chart the path they will take in the future, keeping in mind that they will be able to review and edit that chart every thirty days. The product evolves Sprint by Sprint. Review meetings are the rhythm, the heartbeat of that development. The product is periodically released, with releases timed according to the state of the product and the state of the market. You can usually tell a product is going to be released when you hear product management saying things like, "Wow, this really is ready for prime time, we can get some traction with this." The Sprint that immediately precedes the release is spent readying the product for market, testing and packaging it for shipment and use.

Conclusions

Scrum humanizes product development by introducing regular communication of successes and failures and by helping teams of people commit to shared goals. The mechanisms of Scrum are easy to understand; after all, they're just common sense. I have found that organizations that start using Scrum become more fun and challenging places to work within two to three months. These places become more grounded and resilient. They are eventually able to look any circumstance in the face and deal with it honestly and openly. What more can you ask for?

Team Rooms and Open Space: The End of Housing Projects SIDEBAR

Many of my customers use cubicles. Scott Adams has said most everything there is to say about cubicles, but most organizations still think that they're ok and offer a modicum of privacy. I think that cubicles are the greatest scam since junk bonds. Also, I don't know if everyone else noticed, but in the crew quarters of the *Andrea Gail*, there were **no** cubicles.

When I enter a cubicle environment, there is a silence, an absence of interaction (except maybe two people in some cubes conversing). The energy of people sharing, communicating, working toward something just isn't there.

I've visited some very attractive alternative recently, such as Zefer, Scient, Cambridge Incubator, and eLogic. Representative of their approach, and if I were starting another software company, I'd gut the space, put in wood or concrete floors, cover the walls with whiteboards, and intersperse telephone and network connections. Then I'd issue to everyone a rolling desk, a rolling file cabinet, and a cart with their computer and monitor. I'd let them form their own work groups, clusters of desks, etc. that would form based on who was working with whom at the time.

When I walk into one of these companies, the hum of activity is palpable. You know good things are happening. You can feel the energy.

I worked at a software company once that had tasteful pictures in every conference room. To initiate Scrum, I start daily Scrum meetings. These are always held in the same place at the same time, making it easy for everyone to plan around them. So, one of my first requests is for a permanent "scrum room" where the team can conduct their daily scrums and then conduct follow-on meetings.

So, we're holding the daily scrum meeting and there is nothing on which to record issues, risks, and other team related notes. We took the pictures down and starting writing on the walls with markers. It worked great and reminded everyone that the purpose of the room was to help us build software, not to impress us with how tasteful the designer was. Everyone got the point, and we soon had whiteboards covering up our wall writing.

At a cubicle company, we gutted several offices and turned them into team rooms. We installed shelving, everyone brought in his or her computer, and the team really started producing.

I don't know how we've gotten to a point where most offices are structured to impair teamwork and communications rather than to facilitate it.