# AGILE DEVELOPMENT:
# LESSONS LEARNED FROM THE FIRST SCRUM

**By Dr. Jeff Sutherland**

**October 2004**

In 1993, at Easel Corporation in 1993, we we first applied the Scrum process to software development teams when we built the first object-oriented design and analysis (OOAD) tool that incorporated round-trip engineering. In a Smalltalk development environment, code was autogenerated from a graphic design tool, and any changes to the code from the Smalltalk integrated development environment (IDE) were immediately reflected back into design.

Since the product was directed toward enterprise software development, we spent a lot of time analyzing best practices in software development methodologies.

## REVIEWING SOFTWARE DEVELOPMENT PROCESSES

We realized we needed a development process that fit a enhanced version of rapid application development, where visualization of design could result immediately in working code. This led to an extensive review of both the literature and the real experience of leaders of hundreds of software development projects.

Some key factors influenced the introduction of Scrum at Easel Corporation. In their book *Wicked Problems, Righteous Solutions,* authors Peter DeGrace and Leslie Hulet Stahl reviewed the reasons for failure of the waterfall approach to software development [5]:

- Requirements are not fully understood before the project begins.
- User know what they want only after they see an initial version of the software.
- Requirements change often during the software construction process.
- New tools and technologies make implementation strategies unpredictable.

In addition, DeGrace and Stahl reviewed “all-at-once” models of software development that uniquely fit object-oriented implementation of software and help to resolve these challenges.

All-at-once models assume that the creation of software is done by simultaneously working on requirements, analysis, design, coding, and testing, then delivering the entire system all at once. The simplest all-at-once model is a single super-programmer creating and delivering an application from beginning to end. All aspects of the development process reside in one person’s head. This is the fastest way to deliver a product that has good internal architectural consistency and is the “hacker” model of implementation. For example, in a project prior to the first Scrum, an individual spent two years wrting every line of code for the Matisse object database server used to drive US $10 billion nuclear reprocessing plants worldwide. At less than 50,000 lines of code, the nuclear engineers

said it was the fastest and most reliable database ever benchmarked for nuclear plants. Frederick Brooks has documented a variant of this approach called the "surgical team," which IBM has shown to be its most productive software development process [3].

The surgeon on superprogrammer approach has a fatal flaw in that, even in a large company, at most one or two individuals can execute this model. For example, it took years for a leading team of developers to understand the conceptual elegance of the Matisse object server technology enough to maintain it. The single-programmer model does not scale well for large projects.

The next level of all-at-once development is handcuffing two programmers together, as in pair programming in the Extreme Programming paradigm [1]. Here two developers working at the same terminal deliver a component of the system together. This has been demonstrated to deliver better code (in terms of usability, maintainability, flexibility, and extendibility) faster than two developers working individually [11]. The challenge is to achieve a similar productivity effect with more than two people. What is the best way to work with multiple teams of people on large projects?

Our scalable, team-based all-at-once model was motivated by the Japanese approach to new product development. We were already using an iterative and incremental approach to building software [8]. It was implemented in slices in which an entire piece of fully integrated functionality worked at the end of an iteration. What intrigued us was Hirotaka Takeuchi and Ikujiro Nonaka's description of the team-building process for setting up and managing a SCRUM [10]. The idea of building a self-empowered team where everyone had the global view of the product on a daily basis seemed to be the right one. The approach to managing the team which had been so successful at Honda, Canon, and Fujitsu also resonated with the systems thinking approach promoted by Professor Peter Senge at MIT [9].

After reading James Coplien's paper on Borland's development of Quattro Pro for Windows, we were persuaded into setting up the first Scrum meeting [4]. The Quattro team delivered one million lines of C++ code in 31 months with a four-person staff that later grew to eight. This was about a 1000 lines of deliverable code per person per week, the most productive software project ever documented. The team attained this level of productivity by intensive interaction in daily meetings with project management, product management, developers, documenters, and QA staff.

## WHY THE CEO SUPPORTED THE FIRST SCRUM

The primary driver for beginning the first SCRUM was absolute commitment to a date, where failure would break the company. The task: guaranteed delivery of an innovative product to the market that would achieve rapid adoption.

In a meeting with the CEO, I noted that for years he had received project plans that were supported by Gantt charts. The CEO agreed that no plan had ever delivered the required functionality on time. Many delays had been extensive and hurt the company financially.

Forcasted revenue on a major new product upgrade was millions of dollars a month, so every month that a project was late cost the company millions in revenue. As the company would operated at a loss for a quarter or more and damage to the stock price would be significant, we could not afford to repeat this cycle.

Further, I pointed out that the CEO had no view of the status of the software by the middle of the project. He had Gantt charts and reports that looked solid on paper but failed to deliver the software on time. He had never seen a promised delivery date met, and worse, he rarely discovered slippage until it was too late to reforecast company revnenue.

I told the CEO that in adopting Scrum, we set the objectives at the beginning of what Scrum refers to as a sprint. It is the teams responsibility to determine how to best meet those objectives. During the sprint, no one can bother team members with requests. At the end of a sprint, I added, working code that will be demonstrated, so you can see the progress made. You can decide to ship anytime or do another Sprint to get more functionality. Visible working code provides more confidence than extensive documentation with no operational system.

In the case of this project, the date was six months out, and we established six sprints. The CEO agreed to proceed with the first software development Scrum.

**SCRUM BASICS**

The first Scrum started with a half day planning session that outlined the feature set we wanted to achieve in a six month period. We then broke it into six pieces which were achievable in 30 day sprints. This was the product backlog. For the first sprint, the product backlog was transformed into development tasks that could be done in less than a day.

Daily meetings allowed everyone on the project team to see the status of all aspects of the project in real time. This allowed the collective neural networks of the team's mind to fine-tune or redirect efforts on a daily basis to maximize throughput. The result was radical alteration of the software development process by allowing sharing of software resources. Development tasks thought to take days could often be accomplished in hours using someone else's code as a starting point.

At Easel, daily meetings were disciplined in the way we that we now understand as the Scrum pattern [2]. The most interesting effect of Scrum on Easel's development environment was an observed "punctuated equilibrium" effect. This occurs in biological evolution when a species is stable for long periods of time and then undergoes a sudden jump in capability [7]. During the long period of apparent stability, many internal changes in the organism are reconfigured that cannot be observed externally. When all pieces are in place to allow a significant jump in functionality, external change occurs suddenly. A fully integrated component design environment leads to unexpected, rapid evolution of a software system with emergent, adaptive properties resembling the process

of punctuated equilibrium observed in biological species. Sudden leaps in functionality resulted in earlier than expected delivery of software in the first Scrum.

The meetings were kept short, typically under 30 minutes and discussion was restricted to the three SCRUM questions:

1. What did you do yesterday?
2. What will you do today?
3. What obstacles got in your way?

By having every member of the team see every day what every other team member was doing, we could make progress by identifying work that could be improved by others' work. We received comments from one developer, for example, that if he changed a few lines of code, he could eliminate days of work for another developer. This effect was so dramatic that the project accelerated to the point at which it had to be slowed down by outnumbering developers with documentation and testing engineers. This hyperproductive state was seen in a several subsequent Scrums, although never as dramatically as the first at Easel. It was a combination of (1) the skill level of the team, (2) the flexibility of a Smalltalk development environment, and (3) the way we approached production prototypes that rapidly evolved into a deliverable product.

For example, a key to entering a hyperproductive state was not just the Scrum organizational pattern. We did constant component testing of topic areas, integration of packages, refactoring of selected parts of the system, and multiple builds per day. These activities have become key features of eXtreme Programming [6].

Every Friday during the first Scrum, we held a demo and brought in development experts from other companies in to look at the product. As a result our developers had to do the demo for their peers in other companies. This was one of the best accelerators I have seen in software development. An outside expert would say, "That's terrible; look at Borland's Product X to see how it should be done" or "How could you possible have a dumb bug like that?" As a result of this outside input, all problems or bugs would be fixed the following week. Developers refused to be embarrassed a second time in front of their peers.

At the end of each month, the CEO got his demo. He could use the software himself and see it work. We then gave the software to the consulting group to use in prototyping consulting projects. This provided an incredible amount of feedback to incorporate into the Scrum product backlog: a list of desirable features to include in the software. At the beginning of each sprint, product backlog is reprioritized before transformation into development tasks. The Scrum adaptability to change enabled the CEO to steer product development more effectively than other project management techniques.

**SCRUM RESULTS**

The CEO saw significant, step by step progress in each increment and he agreed that the product was ready to ship in the fifth increment. It had more functionality than expected

in some areas and less in others. The sixth increment was primarily a packaging increment. We shipped on the day it was scheduled to be shipped.

We gave a money back guarantee to all clients that purchased the product, stating that this new design software would double client developer productivity in the first month of use. It sold well until the Smalltalk market started to hit the wall in the mid-1990's and was a model for Rational Rose development. The ScrumMaster went on to lead the Rational Rose development team a few years later.

Everyone agreed that first, Scrum could meet a deadline; second, more functionality was achieved than expected; and third, there would never be a return to waterfall type mentality because (1) the waterfall method could not predict, (2) it could not deliver on time, (3) it produced less functionality per developer unit of time, and (4) user satisfaction was terrible when the product was delivered, since waterfall approaches did not lend themselves to customer involvement or alteration of specifications required by rapidly changing market conditions.

Over the past decade, Scrum has emerged from humble beginnings to a movement involving tens of thousands of projects in hundreds of the leading software development companies worldwide. And during the past two years, more than 1,000 new certified ScrumMasters have been trained in the U.S. and Europe.

(The article was published by the Cutter Agile Project Management Advisory Service. Executive Update, Vol. 5, No. 20. Contact service@cutter.com for reprints.)

1. Beck, K., *Extreme Programming Explained: Embrace Change*. The XP Series. 1999: Addision Wesley.
2. Beedle, M., et al., *SCRUM: A Pattern Language for Hyperproductive Software Development*, in *Pattern Languages of Program Design*, N. Harrison, Editor. 1999, Addison-Wesley. p. 637-651.
3. Brooks, F.P., *The Mythical Man Month: Essays on Software Engineering*. 1995: Addison-Wesley.
4. Coplien, J.O. *Borland Software Craftsmanship: A New Look at Process, Quality and Productivity*. in *5th Annual Borland International Conference*. 1994. Orlando, FL.
5. DeGrace, P. and L.H. Stahl, *Wicked Problems, Righteous Solutions: a Catalogue of Modern Software Engineering Paradigms*. Yourdon Press, 1990.
6. Fowler, M., *Is Design Dead?* Software Development, 2001. **9**(4).
7. Gould, S.J., *The structure of evolutionary theory*. 2002, Cambridge, Mass.: Belknap Press of Harvard University Press. xxii, 1433 p.
8. Larman, C., *Agile & Iterative Development: A Manager's Guide*. Agile Software Development, ed. A. Cockburn and J. Highsmith. 2004, Boston: Addison-Wesley.
9. Senge, P.M., *The Fifth Discipline: the Art and Practice of the Learning Organization*. Currency. 1990.
10. Takeuchi, H. and I. Nonaka, *The New New Product Development Game.* Harvard Business Review, 1986(January-February).

11. Wood, W.A. and W.L. Kleb, *Exploring XP for Scientific Research.* IEEE Software, 2003. **20**(3): p. 30-36.

## ABOUT THE AUTHOR

Dr. Jeff Sutherland has been the development leader for nine software product companies, introducing Scrum to five of them. As CTO of PatientKeeper, he most recently built the leading software platform for mobile/wireless healthcare applications that enables physicians to improve patient care. His latest techniques for Scrum project management and reporting reduce project administration to 10 minutes a day and developer administrative overhead to one minute a day, an order of magnitude more efficient that traditional approaches to product management. He can be reached at jeff.sutherland@computer.org.