

Listen to *"What is SCRUM?"* podcast at <http://budurl.com/iTunescast> (+ *"Why SCRUM?"*)  
Hosted by: Robin Borough | Speaker: David Sheriff

## SCRUM

### What is Scrum?

Scrum is the most widely used, state-of-the-art path to world-class software development. Scrum defines an environment where developers thrive and can be maximally productive. That's the bottom line. It's about the developers. With creativity and imagination, developers turn concepts into code. Scrum strips developers of constraints. Consequently they turn out better code faster.

A Scrum development process becomes very responsive to changing requirements. You can start producing working software even when requirements are incomplete. Software releases can follow each other frequently.

Finally, Scrum is empirical. Scrum implementations generally improve over time. Scrum reinforces what works, and changes what doesn't.

### Why has Scrum become the most popular agile practice?

Oddly enough, I think Scrum is popular for what it does not define. It does just enough to be empirical and challenges you to figure out how to maximize various relationships, processes and tools yourself. Scrum is no shortcut; the transition to Scrum is very hard work on everyone's part.

Perhaps more importantly, the business gains advantages in shorter time-to-market, and the ability to better to respond to moving targets. Scrum also scales well to hundreds of parallel development teams working across many time zones.

Scrum is being used at Google, IBM, Microsoft, Yahoo, virtually everywhere on the front lines of software development. Productivity gains of 500 to 1000 percent have been reported over 1990's style waterfall processes, but improvement depends on where you start from.

### Where did Scrum come from?

Scrum condensed from many of the best ideas floating around in the 1980's and 90's. Scrum has been described as what happens automatically in every small startup. Scrum is what we all do when our backs are up against the wall. The ideas in scrum are not original; the elegant combination of them by Schwaber and Sutherland is what's original. Lots of people remember doing scrum-like practices in the past, before they were called Scrum.

I subscribe to the general philosophy that times are always pregnant with new ideas. Instances of nearly simultaneous invention abound. The Calculus, Manned flight, the vacuum tube, the telephone, the automobile. We seem to have a need to ascribe invention to single individuals, but I think at the time any number of people could have done it.

### Is Scrum just another fad?

I don't think so. Scrum has been modified slightly over the last fifteen years but it's still the most popular Agile organizational practice. Because Sprint empirically modifies itself in each instance, it adapts to the prevailing circumstances. Sprint purposely does not define HOW to write high-quality code because those practices may evolve.

---

Listen to *“What is SCRUM?”* podcast at <http://budurl.com/iTunescast> (+ *“Why SCRUM?”*)  
Hosted by: Robin Borough | Speaker: David Sheriff

Some tweaks have appeared recently which further specify what happens during the sprint. Kanban and Lean, two terms borrowed from defined process manufacturing, are examples.

At some point Scrum's ideas may be recast under another Name, just as Scrum itself is a selection of ideas common in the 1980's and 90's under different names. But it works and it's self-optimizing. Scrum is not going away.

## AGILE

### How do Agile and Scrum relate to each other?

Scores of Agile processes have been developed. They all share respect for the software professional, value collaboration and live communication. Agile processes are responsive to requirements change. Late changes can be customer competitive advantages. Agile groups deliver high-quality software frequently. Agile handles vague and evolving requirements and produces high-quality code most quickly, inexpensively and efficiently.

Look at the definition of Agility: moving quickly, easily and nimbly. Agile is a good word to sum up the general approach.

Scrum is the leading Agile software development framework. Scrum most effectively describes how to organize the development process. Scrum isn't a recipe, but it defines sensible bounds so collaboration occurs and the process is self-inspecting and self-correcting.

Another Agile practice, XP, best describes how to produce high-quality code. XP often runs in parallel with Scrum. You cannot succeed with Scrum unless you write and refactor until you have simple, virtually error free code.

[Refactor: to rewrite and improve software without changing its function]

### How are these Agile methods different from the processes most popular in the 1990's?

Fundamentally, all Agile methods view software development as creative, professional work. As such, a lot of trust is placed in developers. Developers are trusted to do their best, trusted to get the job done. Management supports the needs of developers, and isn't required to tell them what to do every day.

A group of leading software professionals published a Manifesto for Agile Software Development in 2001. The manifesto summarized their consensus on better ways of developing software. The manifesto is arranged as four strong preferences: for Individuals and interactions over processes, Working software over plans, Customer collaboration over contract negotiation and responding to change instead of fighting it. 12 principles expand on them. For example: One principle states that the best architectures, requirements and designs emerge from self-organizing teams. Another posits that simplicity – the art of maximizing the amount of work not done – is essential.

### What are some other Agile characteristics?

Teams deliver software frequently so progress is easy to monitor. High quality, working software receives special stress because, well, it's easier to live with long term. It's a lot less expensive to get it right up front.

---

Listen to *"What is SCRUM?"* podcast at <http://budurl.com/iTunescast> (+ *"Why SCRUM?"*)  
Hosted by: Robin Borough | Speaker: David Sheriff

Both Scrum and XP hold that having a high-bandwidth conversation works better than anything using paper or computers. Collaboration and real-time face-to-face communication are key in Agile. . . communication and collaboration between developers and customers; Communication and collaboration within and between development teams.

### **Where do you see Scrum and Agile trends headed?**

A crystal ball would be nice, but I don't have one. Software once was the most difficult high-tech process to manage principally because we were trying to apply production line management methods to knowledge work. Agile and Scrum are a fundamental re-orientation recognizing that you cannot define the perfect process, the silver bullet, for software. You must be empirical, you have to adapt, you must continuously improve. Every development team is different.

As we become more sophisticated in how we view creative work and motivation, I think we will get better. But the big step has been taken. Once you are accustomed to continuous improvement, everything comes in small steps.

## **Skeptics**

### **A number of customers I've talked to have tried Scrum and didn't see the benefit.**

The only promise in Scrum is that you will find what isn't working in your development process very quickly. It's up to you to fix that.

A lot of organizations have difficulty completing the transition to Scrum because it usually provokes serious organizational change. Many people pick and choose among Scrum's ideas and end up with some very sub-optimal combination. They declare failure and go back to what they were doing.

It's not immediately obvious which Scrum practice produces what result. You don't know what a high performing team feels like until you've been there. Some groups need a lot of help sorting that out.

Scrum isn't magic. In fact it's not a "method" at all in the cookbook sense. Scrum gives you the bones, the framework, of a good empirical development process. What's not working is almost immediately apparent. Then you periodically modify things so they work better.

### **How could smart, well intentioned people tend to get Scrum wrong?**

Scrum is the polar opposite command and control management. That makes some people see it as dangerous. Scrum quickly exposes process and organizational dysfunctionality. It can take strong top management commitment to complete the necessary changes.

Some people try to bite off too much at once. Some don't let a Scrum pilot run long enough to get results. Scrum doesn't work without something very much like XP to ensure quality code. These are big changes for developers.

Scrum is incompatible with micromanaging. It takes time for people to figure out how to work this way. No one has to lose professionally, but sometime it looks like that at first.

---

Listen to *“What is SCRUM?”* podcast at <http://budurl.com/iTunescast> (+ *“Why SCRUM?”*)  
Hosted by: Robin Borough | Speaker: David Sheriff

## Scrum Details

### Describe what happens in Scrum.

Yes. Unless you understand why Scrum is a good idea, you tend to get lost in terms and details before you ever find out. I would be asking you to pay attention for two days of training, for example, without telling you why you should.

Scrum is summarized in various guides and primers, usually in less than 25 pages. The Scrum Alliance is the place to look for details. That's SCRUMALLIANCE.ORG

My partner Scott Dunn is conducting a Scrum Master training at Omnikron University on May 13 and 14. The training gives you the basics for functioning as a Scrum Master on a Scrum team. It's also a good source of answers and information for anyone curious about Scrum.

### Two minute description of how scrum works

In Scrum, a development team consists of five to eight software professionals. As a team, they cover all the expertise required from architecture through Quality.

The team also has a Scrum Master and a Product owner. The Scrum Master's responsibility is to help everyone understand and implement scrum. The scrum master also works to remove any obstacles to progress identified by the team. The Product Owner represents the interests of the business and the customer.

The team works in cycles, called “sprints” where a limited number of software requirements are realized in tested, deliverable software over a period of weeks.

A release planning meeting takes place at the beginning of the whole process. With the Product Owner, the team deconstructs the functionality to be built during a series of sprints. Individual features are prioritized and estimated relative to each other.

Before each sprint, the team selects the highest priority or riskiest features to work on during that sprint. Then they plan together how to implement them.

Every day during the sprint, the team meets for a brief stand-up meeting where they formally report progress, immediate goals and obstacles to each other.

At the end of the sprint, the team demonstrates the new functionality to the product owner and other interested parties. They discuss how “lessons learned” during the sprint should be incorporated into the rest of the release.

Finally, the team holds a retrospective meeting to review how the sprint went regarding people, relationships, processes and tools. They decide what to change to improve the process.

Sprints run end to end until the project is completed or until the business decides the product is good enough to release.

*These FAQ's were authored by David Sheriff – Embedded Scrum Coach – in preparation for the April 26<sup>th</sup> OmniUniCast – based on experience and previous conversations with Robin Borough, EVP for Omnikron Systems Inc.*

---

Listen to *“What is SCRUM?”* podcast at <http://budurl.com/iTunescast> (+ *“Why SCRUM?”*)  
Hosted by: Robin Borough | Speaker: David Sheriff

## Miscellaneous

### How did you get into Scrum Coaching?

I became aware of Scrum several years ago. The more I looked at it, the more I discovered that it embraced the values and principles I learned over my management career. I needed a change in direction and Scrum seemed a good way to practice my professional values while helping both individuals and organizations become more effective. So I trained, studied, partnered with another Scrum professional and here we are.

### Explain the concept of embedded coaching.

Scrum can seem simple, but it requires changing some of a company's culture and practices. Usually, going through that transformation is a lot of work for everyone. An embedded coach works with the Scrum teams for months, helping the teams and the company figure out how to do their work within the Scrum framework. It's a coaching role rather than that of a mentor. The coach's job is to help the team come up with answers from the team's combined experience and intelligence rather than telling them what to do. You try to live the Socratic Method. You pull the best ideas out of people with questions that stimulate them to think.

### Isn't Scrum is a rugby expression?

Scrum is purposefully filled with unusual names: sprints, time boxes, burn-down charts and Scrum Masters. The authors of Scrum, Jeff Sutherland and Ken Schwaber, wanted to make a distinction between aspects of Scrum and similar terms in vogue. Yes, “scrum” is a rugby term for a process of getting the ball back into play. Players form a tight circle, and that's where the analogy ends. The daily 15 minute team meeting where everyone stands in a circle sort of looks like a scrum formation.

For instance, a Product Owner in Scrum funnels prioritized software requirements to the development team. It's a role that looks somewhat like product management and product marketing. A product manager may play the product owner role in a Scrum team, but her interface to the team is pretty tightly defined. Her role is to keep the team working on the most valuable software features during every sprint, or development iteration. Exactly how she does that is not defined.

### Don't you have to pretty well define a product up front?

Well, you need a shared vision of where you are going to be sure, a “big picture.” However, by developing a project in cycles or “sprints,” you work on what seems to be the most important or riskiest features first. Then you see how they actually work. Often requirements change as you learn more. Because you don't try to work on everything at once, you get to make mid-course corrections. Sometimes you discover the product won't work, but at least you find out quickly.

### Can you detail what “empirical” means in this context?

Empirical processes apply in complicated situations where fixed, production processes don't work. These are processes where states never exactly repeat. There is no pattern. Empirical processes must be transparent (everyone has access to all the data) and you periodically inspect the process to find what works and what doesn't. You change what you are doing and then repeat the cycle of inspect and adapt. Driving your car is an example of an empirical process.