

# Get Into The Game With Scrum

Reeve Fritchman, PMP®, CSM®  
Ayers Rock Software, LLC  
1560 Parkview Drive NE  
Bainbridge Island, Washington 98110  
[www.agilitrans.com](http://www.agilitrans.com)  
[reeve.fritchman@agilitrans.com](mailto:reeve.fritchman@agilitrans.com)

## Get Into The Game With Scrum

### **Introduction**

In rugby, a “scrum” appears to be a mass of large sweaty bodies heaving and struggling; too often that’s an accurate description of many IT projects. In IT, Scrum is a lightweight software development methodology that can dramatically improve IT’s productivity and image by delivering results on a regular basis that’s short enough to impress senior management. In a typical Scrum project, useable software is publicly demonstrated every 30 days (or less), and interested managers can keep up with the project’s progress on a daily basis. Scrum provides visible results shortly after the project starts and offers an elegant control mechanism to adjust what’s being produced.

Scrum is “lightweight” because there are relatively few standards and regulations. By comparison, the Project Management Institute’s “Body of Knowledge” is highly structured and can provide very tight management control.

But building software is not the same as building an aircraft carrier, and it’s intuitive that different project management techniques should be applied. When building an aircraft carrier, the design is based upon known engineering principles related to physics, hydrodynamics, and mechanical engineering. In addition, both the buyer and builder of an aircraft carrier agree on requirements before the design starts, and those requirements are known (and accepted) to be extremely expensive to change in mid-construction. The construction process is completely separate from the design and engineering effort, and there are known labor standard for cutting and welding steel.

A software project almost always mixes design and construction unless the project is run in strict “waterfall” mode, where the architects design and pseudocode the system and then hand off their output to a coding staff that creates the source code.

### **BxUF**

Let’s step back for a moment and consider tradition approaches to quantifying the cost and duration of a systems development project. The lynchpin of the “Big Requirements Up Front” (BRUF) approach is a detailed set of requirements prepared before the start of the project; this is a classic waterfall process, where one stage (requirements gathering) is completed before the next (design) starts. The problem with BRUF is that most users’ requirements are inexact and incomplete. When high-level requirements are implemented as working code and presented to users near the end of the project, users invariably refine (read “change”) their requirements, thus invalidating the original requirements that may have been used to estimate the cost and duration of the project.

Another approach is “Big Design Up Front” (BDUF), where a comprehensive end-to-end master design is generated. While planning is always good, the primary problem with BDUF is that the design is completed at the beginning of the project, when the least is known about the project and the price of a design error is high. BDUF has another major weakness: it assumes the requirements are correct and complete.

BRUF and BDUF fail to acknowledge that fact that systems design is so complex that getting the design and requirements right the first time for anything other than a very simple system is very unlikely.

Corrections and adjustments are required as requirements are refined or as design limitations become apparent. Scrum embraces the reality of corrections and adjustments and provides a protocol managing them sooner (when the cost of change is less) rather than later (when the cost of change is much higher).

It's important to understand that Scrum doesn't magically make project schedules or cost estimates more accurate. It's more of a "pay as you go" process, where senior managers have the ability, on a more-or-less monthly basis, to reconsider the project by evaluating the value provided to date.

### **I can't believe the project requirements have changed!**

Of all software projects issues, managing changes is at the top of the list. Scrum, unlike most other methodologies, welcomes changing requirements. A fact of business life is that requirements, often driven by external forces, change. And business priorities also change, sometimes driven by external forces and sometimes by internal forces. Regardless of the source or reason, the ability to handle updated requirements gracefully is extremely important.

Sometimes changing requirements have a silver lining. When a well-reasoned set of requirements has been laid down, a change often represents a refinement of, and an improvement to, the original requirements. Such changes reflect a deeper understanding of both the business and the proposed solution.

### **Scrum details**

The Scrum process starts with defining the Product Backlog, a list of prioritized requirements. The Team decides what it's going to do in the next "Sprint" (a 30 calendar day period), does it, demonstrates the results to the Product Owner, and works with the Product Owner to modify the Product backlog before starting on the next Sprint.

The Scrum process starts with a "vision" from the Product Owner. In traditional project management terms, this artifact is the project charter. The Team, in conjunction with the Product Owner and in a four-hour meeting, selects the Product Backlog Items (PBI's) it can complete within the Sprint, a period of focused efforts to design, code, test, and document those requirements. The selection of the PBI's is the Team's commitment to the project, and this commitment is key to the success of the Sprint. The Team meets for four hours to plan how it will meet its commitment, and then the 30-day clock starts ticking.

Sprint has five defined, time-boxed meetings. There's a four-hour Sprint planning session before every Sprint. There's also a four-hour Sprint review with the Product Owner where the results of the Sprint are reviewed and the Product Backlog is updated and reprioritized. There's also a shorter Sprint "Retrospective" meeting where the ScrumMaster and the Team discuss what worked well and what needs improvement. Finally, there are daily "Scrums", a 15-minute meeting where each team member reviews what s/he's accomplished since the last Scrum, what s/he plans on accomplishing today, and what, if anything, is preventing him/her from accomplishing his/her goals. The daily Scrum is extremely important because it's a brief way for every team member to bring the Team up to date; the format doesn't allow the introduction of problems or issues (they're handled after the conclusion of the daily Scrum). While it might look like there are "too many" meetings, the tradeoff is that the Team is completely protected from all external influences during the Sprint.

There are several parties involved in a Scrum-managed project. The Product Owner, a single individual, is the “buyer” of the project and responsible for defining the requirements and prioritizing those requirements in descending order of value.

The ScrumMaster is the party responsible for coordinating (not managing) the project. This person’s responsibility is to make sure Scrum’s practices and procedures are followed and to facilitate the Team’s efforts. It’s important to understand that the ScrumMaster does not control or lead the Team but has a more important role: to protect the Team from external distractions. If anybody buttonholes a Team member in the hall and requests a quick fix, the ScrumMaster will advise the requesting party that Scrum does not allow any Team member to be distracted from meeting its commitment to the Product Owner. The ScrumMaster is also responsible for clearing any impediments to the Team’s progress.

The Team is made up of around seven individuals (plus or minus two) made up to include a cross-functional skill set. The Team will be self-organizing and self-managing, and this concept is often a huge mental roadblock for managers external to the project and to the Team members themselves. The ScrumMaster doesn’t manage the Team because the Team figures out how to organize itself, and it figures out how to manage itself. One of the most effective ways to get the Team to coalesce as a team is to get all the members out of offices and cubicles and into a common “war room”, where high-bandwidth person-to-person communication can take place.

No IT project is complete without a complement of farm animals, and Scrum has pigs and chickens. Think of a ham-and-egg breakfast: the chicken is involved but the pig is committed. The Product Owner, ScrumMaster, and Team are pigs; everybody else is a chicken. And chickens are silent: they do not have a voice in the project, and this rule extends all the way up the chain of command. Chickens may attend daily Scrums but they are not allowed to comment, pass notes, or make faces. Only the Product Owner can deal with the Team.

### **Conclusion**

Scrum is a refreshing new approach to development. There are Scrum coaches available to help out with the inevitable rough spots and there are certification courses in ScrumMaster-y and Scrum product ownership. The net result of a properly-executed Scrum project is much higher customer satisfaction and lower overall cost.

### **Useful references**

[www.scrumalliance.org](http://www.scrumalliance.org): The Scrum Alliance is the organization overseeing Scrum techniques and procedures.

[www.agilealliance.org](http://www.agilealliance.org): The Agile Alliance is the organization supporting the “Manifesto for Agile Software Development”.