

## Agile Processes and Self-Organization

*This paper discusses the use of self-organizing teams in agile processes. Scrum is used as the model agile process during this paper.*

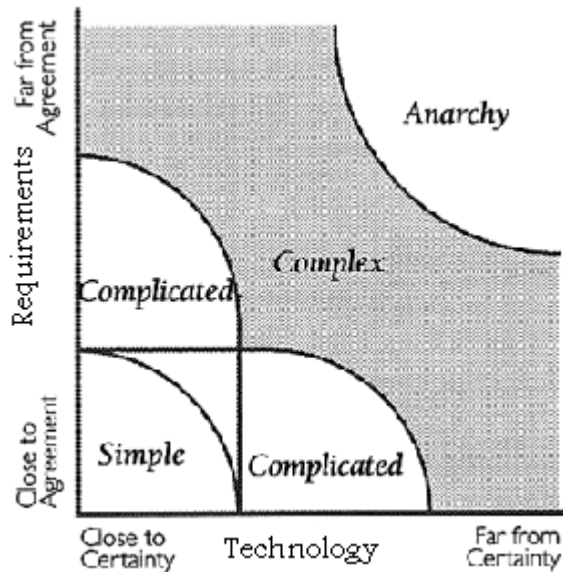
Management is now faced with how to reach feature consensus and technology utility rapidly and iteratively, to capture and retain market advantage. This need translates into the requirement for a development process that both regularly delivers working products and that constantly adapts to business needs and technology availability and reliability.

Building advanced technology, competitive systems is complicated. As shown in the requirements/technology figure<sup>1</sup>, the degree of complexity increases as the requirements are less known and the technology is less certain.

Agile processes employ self-organizing teams to handle the complexity inherent in systems development projects. A team of individuals is formed. They organize themselves into a team in response to the pressure of a deadline, reminding me of the saying, “Nothing focuses the mind like a noose!” The pressure cooker of the deadline produces cooperation and creativity that otherwise is rare. This may seem inhumane, but compared with non-agile practices for dealing with complexity, self-organization is a breath of fresh air.

In most work circumstances, management imposes a deadline and tells the workers what to complete by the deadline. This violates the rule of common sense, “You can tell me what to do or how to do it, but you can’t tell me both.” Within agile processes, the length of the iteration imposes a deadline. Scrum iterations (Sprints) are always thirty calendar days. The team selects how much work it believes it can perform within the iteration, and the team commits to the work. Nothing demotivates a team as much as someone else making commitments for it. Nothing motivates a team as much as accepting the responsibility for fulfilling commitments that it made itself.

Management forms a team by selecting the available people most able to transform the requirements into a working system. Scrum recommends teams of seven plus or minus



<sup>1</sup> Adapted from *Strategic Management and Organisational Dynamics, The Challenge of Complexity* - 3rd Edition, Ralph D. Stacey, Prentice Hall, Feb. 2000

two people<sup>2</sup>. My experience has been that complexity of the work and self-organization overwhelms larger teams. If the team is too small, the benefits and productivity of team collaboration and self-organization are limited. Maximize the team size almost to the breaking point to maximize the productivity

The team collectively selects requirements from a prioritized list, selecting only as much as it believes it can turn into an increment of working product functionality during the next Sprint. The only constraints on the team are any existing organizational standards, conventions, and previously constructed product functionality. The team commits to management that it will turn these requirements into working product functionality by the end of the Sprint. The team is left alone to do so for the duration of the Sprint.

Left alone! No one to tell the team what to do? No methodology to tell the team how to transform the requirements into functionality? No one to blame if the team fails? No one to grab the glory if the team succeeds? That's right, left alone. It is solely and utterly the team's responsibility to figure out what to do, and to do it. The old saying, "Be careful what you ask for, because you might get it!" describes the dilemma and opportunity of the team. In my experience, every team is at first shocked by this responsibility. However, as the team realizes that it has the full authority to do whatever it deems necessary, a sense of liberation and empowerment (that usually trite phrase) occurs. The team starts talking, drawing designs on whiteboards, figuring out what work needs to be done. People start defining what work they'll do, and what help they need to do it. Some people ask to do work that they've always wanted to learn, signing up for other additional work to offset their learning curve. The team collectively simmers, brainstorms, and works to meet its commitment. The team self-organizes.

Teams are cross functional and without assigned roles. Team members may have job titles, training and experience, but that doesn't entitle them. Instead, the team self-organizes based on its strengths and weaknesses to do the work at hand. If the testers are overwhelmed, developers may have to help test. If the tech writer has extra time, he or she can help write test cases. Everyone on the team, each, creates the product, contributing whatever he or she has to what is needed.

Each individual on the team has varying skills to apply to the problem and technology domain. Each individual also has intelligence, determination, and focus with which they will apply their skills. At any moment, on any day, the intersection of the problem domain, technology stability, skills available, and presence of determination, focus and intelligence is hugely variable, but the individual has to "self-organize" himself or herself to figure out what to do and how to do it. Every day, everyone on the team must coordinate his or her own individual self-organization with the rest of the team. To facilitate this, Scrum and xP have daily synchronization and status exchange meetings. Scrum's daily meetings (Daily Scrums) are quite formal, ensuring that specific

---

<sup>2</sup>"The Magical Number Seven, Plus or Minus Two" by George Miller, *Psychology Review*, 1956

synchronization occurs. xP's daily meetings are less formal and are driven by need rather than format.

The Daily Scrums last no more than fifteen minutes. During the meeting, everyone on the team answers three questions:

1. What have you done since the last Daily Scrum?
2. What will you do between now and the next Daily Scrum?
3. What's getting in the way of you doing your work?

As team members answer these questions, the other team members are adjusting and adapting their own work in response. They may think, "Oh, now I don't need to do that.", or, "Sounds like Tom's successfully using that new library; I'll check it out with him this afternoon." As the team members commit to what they will do over the next 24 hours, team self-organization occurs. When they report what they were able to do over the last 24 hours, they help everyone adjust from previous assumptions and take into account what really happened.

When a team member reports things that got in the way, he or she is reporting impediments to self-organization. He or she is saying, "I was trying to do this, to organize myself to get this done, and I couldn't because of x." If yeast in bread batter could talk, it might also report impediments, "This bread will be a lot better if you raise the heat 10 degrees, and I'm just yeast so I can't do it by myself." Similarly, the team member is asking for help to make things better. He or she is reporting what is needed to stay productive. Since management attends every Daily Scrum, its job is to immediately remove these impediments. The Daily Scrum foments this self-organization, keeping it invisibly churning.

At the end of the Sprint, a team demonstrates the executable product increment that it built. The team selected requirements at the beginning of the iteration. It wrestled with the technology and requirements to build functionality that delivers business value. Now the team demonstrates the functionality to the customer. Pre-knowledge of this demonstration focuses the team's attention. It knows that at the end of the Sprint, it will demonstrate the system to the customer and management. No excuses, no one to point at. The team is on the hook to do what it said it could do! The sense of determination and pride within such teams is palpable.

I've heard, "If you want me to take the responsibility, give me the authority." Scrum does so, facilitating the self-organization necessary for teams to focus on the complexity of modern software development projects.