



Building an Agile Team

Bruce Taylor, Senior Agile Coach
ManagingProgrammers.com

I've spent a great deal of time working in and with software development teams, and I've learned that team dynamics contribute at least as much to productivity as does talent or organization. I've been part of agile teams long before "agile" became a recognized term, and I've seen teams that were nominally agile become paralyzed by their own internal dissension. So, when I start working with a team to introduce agile development processes, I spend as much time working on their behavior and coherence as I do on training them in formal agile methods. This paper is a reflection on my process for helping teams into the fluid world of agile development.

A Shared Vision

Teams collaborate best when they share a vision: when all team members understand and accept the same goals and priorities. This coherence isn't accidental, and it can't be imposed as a dictate from management; it has to be built deliberately when the team forms, it has to come from the team members, and it has to be nurtured to survive times of stress. Agile training is really part of this process: when the team trains as a unit in the same agile process, they begin to use the same terminology, they practice collaborating, and they come to a common vision of the product they are building.

So when I do agile training I don't simply teach, "Here's how to do Scrum." Instead, I teach the theory of agility and introduce agile practices that have helped other teams, and invite the team to design its own agile practices. Then we practice their agile process on realistic problems, with realistic deadlines and setbacks, and tune it until everyone on the team understands his or her role and the team has developed its own implicit rules. Typically I start the training in the role of a teacher and gradually transition into the role of a coach, so that the team feels invested in the agile process that it has chosen.

Agile Leadership

As a team is converting to an agile development approach, I think that managers are asked to make the biggest transition. In my experience, most software development organizations function in "command and control" mode: goals are handed down from executives to managers, and managers convert the goals into plans and allocate tasks to developers. In the agile world this relationship is inverted: the team members (assisted by the scrum master) decide on the sprint goals and largely run the sprint themselves, so what role does the manager play? Good agile managers play a supporting role - they act as the interface between the team and the rest of the organization, they negotiate for required resources, they maintain the agile process itself, and, most important, they maintain the coherence of the team in the face of development pressures and stresses.

When I am helping a team adopt an agile process, I usually spend extra time with the team manager, because he or she has the most significant role change to make. I spend some of this time training them in agile methods, communications skills, and "people management" skills, but I also help them understand that their new role is *not* a demotion - that they are even more important to the team's success than when they played the role of ringmaster.

Mutual Trust

It is in the nature of software development that dependencies define the schedule: employee information page can't be demonstrated until the logon page and process are implemented. And because sprints are short, each team member's success depends on the talent and dedication of the rest of the team. In short, each team member must display integrity and the rest of the team has to rely on that integrity to be successful. This does *not* mean that all team members need to be equally experienced - in fact a mix of senior and junior team members is healthy - but it does mean that backlog stories are assigned appropriately and that more experienced developers take time to mentor their less experienced colleagues.

When I coach agile teams, I try to instill the notion that, "it isn't a sin to underestimate and miss a deadline, but it *is* a sin to cover it up." If team members report early that they are behind schedule, then the scrum master or manager can often compensate by reassigning other stories to minimize the problem, but if a developer waits until the last day of the sprint to break the news, then the demo is likely to be embarrassing.

Collaboration as a Way of Life

All variations of agile development are based on the notion that collaborative development is the normal way of doing business, and that the team wins or loses as a whole. I try to reinforce this by teaching quality techniques like development standards, code reviews and pair programming, and by discouraging over-specialization.

Especially, I try to limit the pernicious notion that "this is my code, don't touch it without asking me." If a team member doesn't understand the code that she needs to modify, her best approach is to ask the author to explain it to her and show her the best way to make the modification; and the original author should make time to show her around the code so she can make the next change without having to ask. When a team reaches equilibrium, every member should know the basics of all the components of the product, and should know who to ask about perplexing problems.

Not every team is ready or willing to adopt pair programming, and I don't try to force it on teams that aren't ready, but I do teach the notion of "pair inspection," and it seems to be quite popular. When a developer is stuck on a difficult bit of code or an obscure bug, she should be able to ask a colleague, "Could you give me minute to look over this code and help me spot whatever I'm missing?" without feeling embarrassed. And her colleague should feel pleased to be asked and be willing to help her out as soon as his schedule permits.

About Bruce Taylor



Bruce Taylor is the principal of ManagingProgrammers.com, a management coaching firm located near Boston, Massachusetts. Bruce helps software organizations of all sizes to create low-stress, supportive, adaptable working environments, so that the engineers, leaders, and managers can work as effectively as possible. He provides executive coaching for senior managers who are creating superior organizations, management coaching for technical leaders who are adapting to new agile practices, and individual coaching for engineers who are upgrading their skills. Bruce has a Masters in Computer Science from Duke University, a Masters in Community Psychology, and a Certificate in Job Stress and Healthy Workplace Design, both from the University of Massachusetts.