I Can’t Seem to Get My Team to Become Agile

Date:	Oct 9, 2011
Title:	I Can’t Seem to Get My Team to Become Agile
Code:	AR-NOA

Abstract: The software industry has definitely become more “agile” and contrary to many other technology industries, software development has the unique characteristic of extreme flexibility and adaptability. Enforcing a strict sequential, waterfall approach for software projects just doesn’t work any longer. With Scrum being one of the most popular of agile frameworks being adopted for software project development, now even the Project Management Institute (PMI®) has launched a PMI Agile Certified Practitioner (PMI-ACP)℠ certification.

But just being an expert on agile and focusing on delivery of “working software” doesn’t necessarily guarantee success. This article shows tips and techniques for those of you struggling with getting your team and your company to adopt agile.

Version: 1.0 (2011-10-09) Original

Misconceptions About Agile

Senior management doesn’t really have to care about (or understand) agile.

In Doug DeCarlo’s insightful book *Extreme Project Management*, he identifies reasons why extreme [agile] project managers fail. The first one on his list is due to “Not having the right project sponsor, one who is a champion and barrier buster.”

Senior management should provide the leadership and drive to set the direction and to help guide the team to achieve corporate goals. (Some folks think that executives are really there just to punish.) Bearing in mind that the executive staff can have a positive influence, the more comfortable they are with how you’re building products, the more supportive they will be.

You can do a lot to help in “executive speak,” and as Sliger and Broderick state in *The Software Project Manager’s Bridge to Agility*, stay away from terms like “agile” and “extreme” with upper management. Use simple, non-technical terms or analogies that resonate with the audience.

Old-style programmers won’t have to change their behavior.

Resistance? Really??? Yep. Even the most technically brilliant engineers may be resistant to any change – especially to a new methodology. Sliger and Broderick have great advice to not fight it and avoid direct confrontations. “So do your agile in stealth mode.”

For long-term success in adopting any process methodology, you need believers and the more stubborn the team is, the more likely they won’t cooperate. You’re going to have to make an impact so that they’ll recognize first and foremost that there is a problem that needs to be solved. In the opening of this article, the senior team members didn't acknowledge that there was any problem to solve. In the next section, we'll cover some techniques that you can use to hopefully get around that obstacle.

Project managers don't really want to lose control.

We've witnessed project managers who are in perpetual stress preparing and presenting detailed reports and never-ending status reviews. These same folks tend to believe that their most important role is to host long-winded meetings that are designed to "punish" the team (not really, of course, but if you ask *any* programmer that has to endure these meetings…).

Some project managers firmly believe that they must practice strict command and control in order to complete a project and to direct the team. Otherwise, chaos will be result (and they'll lose their job). Getting past command and control to become agile can be tough hurdle since it does require a different mindset of letting the customer dictate what is most important while empowering the team to translate those customer needs into meaningful work.

Agile provides a way for teams to bypass any structure.

Sorry, even being agile won't relieve imposed structure on the team. On the contrary, the structure should become so intuitive and implicit, that you don't even have to think much about it.

Most software engineers would rather work alone and not have to be bothered by schedule pressure. For those organizations where profits and satisfying customer needs aren't that important, then maybe you can get away with that. For the rest of us, our livelihood depends on a team to be unified and schedules to be met. The unfortunate phrase "herding cats" would be the result if a project had no structure or process to ensure that focus and dependencies (among team members) weren't in the forefront of every project decision being made.

Having some form of organized approach, no matter what it is, should benefit you and your team. The trick is selecting the approach (or methodology) for a process that fits the project and the team.

If you pick a process that contradicts what your project and team needs, force fitting to the wrong process or even trying to accommodate everyone's favorite process methodology can be just as dangerous as having no process at all.

You don't need to do any upfront design work, detailed requirements, or role setting.

Anytime I present topics about agile project management, there is at least one attendee who emphatically states, "I don't see how that will work. If I made decisions on product architecture too late, the results would be disastrous."

Rightly said. The perception is that at the beginning of each agile iteration (every two weeks for Scrum, for example) architectural design work can commence, requirements defined, and team roles can be assigned.

It is very important that key architecture issues, key requirements, and roles and responsibilities are decided up front before the project starts up as long as it reduces the risk of project delivery. In the case of software architecture, you may even wish to perform a research mini-project prior to the real project in order to "nail down" the basic architecture.

Michael Krigsman's article "Agile: Anatomy of a Failed Project" really says it all. This sobering case study summarizes a major IT agile project that looked good until two months before planned delivery. What happened represents your worst nightmare where the concept "delivery of working software" along the way is not enough to guarantee project success:

1. Basic architecture issues were not discussed until a later iteration.
2. Requirements were too ambiguous and open for interpretation.
3. Roles changed continuously.

Positive Steps You Can Take

I know of no *single* approach to successfully transition an organization to agile since every organization is going to have a different purpose, receptivity, and culture. However, not carefully transitioning an organization to agile can result in a confused team with inconclusive results. In fact, if the transition is forced, you may have to try the transition again (and again). You don't want that!

I've found these simple steps to work:

1. Make sure you take it slow. I would set an expectation that any transition to agile may take 6 to 18 months. Also, be transparent with the team why you're putting them through this process and what the next step in the transition process is.

2. Find out what doesn't work (why project delivery isn't successful). Without even breathing the word "agile" out loud, get your team together and brainstorm what isn't working well in project delivery. Take special care to not turn this into a post mortem or blame tossing session. Just have everybody list what isn't working well and then prioritize the list.

   Host a similar meeting with your peers (in sales and product management) and senior management with the intent to produce their own prioritized list of their frustrations with product development processes. I would imagine the lists should overlap.

3. Let the team decide on improvements that would make a difference.

   After a short period of time after the meeting in step 2 (a week should be enough time), get the team back together and communicate their prioritized list of what doesn't work along with the second list from your peers and the senior management. Now, brainstorm solutions for key items in both prioritized lists.

4. Match the agile approach to what doesn't work and what does work.

   Now the fun begins (and this will take some planning). Take the proposed resolutions and weave them into a presentation of how agile can assist the team to address those inherent problems. There is excellent material available on the Web and there are some excellent books on the subject (see the Bibliography at the end of this article).

   By associating the pitfalls of the current process with benefits of an agile approach using the team's proposed resolutions as the basis for change, you should at least get initial team acceptance. Then, let the team decide on an upcoming project to try agile on. Adjourn the meeting and communicate the decisions and the presentation to the team, your peers, and senior management. Inform everyone that a test project will employ agile and set expectations that it will most likely take multiple projects to "get it right."

5. Test drive agile on a project. Dive right in! One software architect didn't like the Scrum poker game planning and thought it was childish, so he wouldn't really participate. What was interesting is that the more the team enjoyed Scrum activities; he eventually yelled out, "give me those cards!" Don't expect everyone to jump on board from the beginning – most engineers will sign up, once they witness the benefits.

6. Assess the results and fine tune the project management and do it again.

The above steps may seem obvious, but I was pleased to see that Sliger and Broderick emphasizing similar approaches. Whereas some processes in the *PMBOK*® *Guide* appear outdated and bureaucratic, you really want the agile approach you choose to be intuitive and fun resulting in successful projects. If your team includes those familiar PMI endorsed best practices, you may wish to refer specific *PMBOK*® *Guide* knowledge area processes that can help provide guidance to actions identified in step 4.

A Final Word

I'd like to leave you with a couple of tidbits that even experienced agile practitioners can take to heart every day on every project:

1. **Enhance your communication:** To keep a project moving forward in a positive way, you (as the project manager or ScrumMaster) need to ensure that obstacles to deliver on time don't fester. You may have noticed that a fair amount of this article emphasizes communication on your part. In fact, the *PMBOK® Guide* clearly indicates that about 90% of your time should be spent performing some form of communication.
2. **Let go of the reins:** Contrary to a typical and historical command and control approach to project management, agile transforms that ownership to the team with the project manager/ScrumMaster becoming more of a facilitator and obstacle remover. This different approach is a fundamental shift from those endless report-writing, meeting-centric way of running a project. The worst thing you can do is to bridge the gap and attempt to blend traditional and agile approaches into one to not "make waves." It will confuse the team and an "almost-agile" methodology will almost certainly end in disaster.

Bibliography

Christiansen, David. “Agile Software Development Tutorial: How to Transition to Agile.” (http://searchsoftwarequality.techtarget.com/tutorial/Agile-software-development-tutorial-How-to-transition-to-agile).

DeCarlo, Doug. *Extreme Project Management: Using Leadership, Principles, and Tools to Deliver Value in the Face of Volatility*. San Francisco: Jossey-Bass, 2004.

Krigsman, Michael. “Agile: Anatomy of a Failed Project” (http://www.zdnet.com/blog/projectfailures/agile-anatomy-of-a-failed-project/1100).

Project Management Institute, Inc. *A Guide to the Project Management Body of Knowledge: PMBOK® Guide, 4th Edition*. Newton Square, PA: Project Management Institute, 2008.

Project Management Institute, Inc. “PMI Agile Certified Practitioner (PMI-ACP).” (http://www.pmi.org/en/Certification/New-PMI-Agile-Certification.aspx).

Sliger, Michele and Stacia Broderick. *The Software Project Manager’s Bridge to Agility*. Boston: Addison Wesley, 2008.

Whitaker, Ken. *Principles of Software Development Leadership: Applying Project Management Principles to Agile Software Development*. Boston: Course Technology PTR, 2009.

Bio

Ken Whitaker of Leading Software Maniacs™ (LSM) has more than twenty-five years of software development executive leadership and training experience in a variety of technology roles and industries. He has led commercial software teams at Software Publishing (remember Harvard Graphics?), Data General, embedded systems software companies, and enterprise software suppliers. Ken is an active PMI® member, Project Management Professional (PMP)® certified, and a Certified ScrumMaster (CSM). Sources for LSM’s presentations come from case studies, personal leadership experience, the PMI *Project Management Book of Knowledge* (*PMBOK® Guide*), and Ken’s leadership books: *Managing Software Maniacs*, *Principles of Software Development Leadership*, and *I’m Not God, I’m Just a Project Manager*.

Leading Software Maniacs is proudly associated with:

PMI Registered Education Provider Project Management Institute