# Patterns of Agile Adoption

by Mike Cohn • 0 Comments • originally published in TechWell on 01/04/2008

## Start Small or Go All In?

Conventional, long-standing advice regarding transitioning to the agile process has been to start with a pilot project, learn from it, and then spread agile throughout the organization. This approach is the frequently used *Start Small* pattern in which an organization selects typically one to three teams (of five to ten people each), gets them successful, and then expands agile from there. A

s agile spreads through the organization new teams benefit from the lessons learned by the teams that have gone before. There are many variations of Start Small, usually because of how many people the organization wants to transition to agile and how quickly they want to do it. Start Small can also be applied differently based on how risk-averse or uncertain about the transition the organization is. For example, in some cases the first team or teams will finish their projects before a second set of teams even begins. Other organizations will take an overlapping approach, where the second set of teams starts only one or two iterations after the first.

The Start Small pattern isn't for everyone. In fact, Salesforce.com followed the opposite pattern. I remember answering my phone on October 3, 2006 and hearing Steve and Chris from Salesforce.com tell me that they had just converted thirty-five teams to Scrum overnight. They asked if I'd like to help. My initial thought was that they needed a psychiatrist more than an agile consultant. Not one to shrink from a challenge, though, I agreed to help, packed a copy of Freud alongside my laptop, and set off for San Francisco. Part of what I saw there wasn't entirely unexpected - teams and individuals in an uproar over such a sudden, far-reaching change - but I also saw other things that helped this large-scale, rapid adoption succeed.

Salesforce.com was pursuing the *All In* pattern, which draws its name from a poker player who bets all of his chips on one hand. Salesforce.com has a hard-driving, aggressive, achievement-

driven culture that would not have been a good fit for a cautious Start Small approach. Once key executives were presented with a proposal to adopt agile, they were convinced. They felt that if agile was worth doing for one team, it was worth doing for all teams, so they chose go All In.

## Advantages and Disadvantages of Start Small

**The Start Small approach offers the following advantages:**

- The cost of mistakes is minimized. Making a mistake with one team is far less costly than making that same mistake with 100 teams.
- You can stack the deck in your favor by selecting the team and project most likely to be successful. This isn't cheating. Having made the decision to transition to agile you are not conducting some double-blind, randomized clinical trial. You are trying to help your organization succeed. Often, an early success can be vital to gaining buy-in from skeptics or fence-sitters.
- It seeds the organization with people who can be the ambassadors and mentors for others as they begin to make the move.

**There are of course drawbacks to the Start Small approach:**

- Conclusions may not be compelling. Success on a small project staffed by handpicked people does not necessarily mean success will follow on every other project.
- It obviously takes longer to Start Small and scale than it does to take the plunge and go All In.
- It may be seen as a small, tentative step, indicative that the company is not fully committed to agile. In some environments this will bring out sufficient additional resistance to squash even this first step toward transitioning.

## Advantages and Disadvantages of Going All In

**The advantages to the All In approach include:**

It demonstrates management's commitment to agile and to overcoming the pain of the transition.

- It's over quickly. Transitioning an entire software development team to agile all at once is

like quickly pulling a bandage off a wound. It may hurt a lot but it probably won't hurt for long.
- There is no organizational dissonance from two processes in use concurrently. Chris Fry and Steve Greene of Salesforce.com report that, “The key factor driving us toward a big-bang rollout was to avoid organizational dissonance and a desire for decisive action. Everyone would be doing the same thing at the same time.”
- Resistance is often reduced if skeptics are not allowed to hold out hope that things will revert to normal once the “agile experiment” is over. Like Cortes burning his boats at Vera Cruz to prove his resolve to his soldiers, an organization that chooses to go All In is demonstrating that it is committed to the new process and that it will not turn back. This level of visible commitment to the change can be beneficial in helping the change succeed.

**There are of course drawbacks to All In:**

- An all-at-once transition can be very risky. The cost of small mistakes will be magnified across the entire transition effort.
- Transitioning all at once is costly, not only due to the increased cost of any mistakes but because successful All In transitions are more likely to rely on outside coaches and trainers.
- It's unlikely you can transition an entire organization of any size all at once without reorganizing some teams and reporting relationships.
- An All In transition can create a lot of stress on an organization. If the organization is already under stress for other reasons, a sudden switch could be the proverbial straw that breaks the camel's back.

-

## Technical Practices First or Iterative First?

A view most closely aligned with Extreme Programming (XP) is that becoming agile begins with becoming good at certain technical practices. If a team is using the right technical practices-simple design, automated testing, pair programming, refactoring, and so on-then agility will be the natural result. Based on the broad popularity of XP, the *Technical Practices First* is one of

the most common patterns of agile adoption.

When following this pattern of agile adoption, a team typically starts by introducing the standard XP practices of simple design, short iterations, test-driven development, pair programming, refactoring, continuous integration, a strong emphasis on automated testing, and so on. As other teams see the productivity and job satisfaction gains of the initial team, they begin to emulate their practices. As the rest of the company sees the improvements, their trust of the development teams increases. This creates a virtuous cycle: the team improves, creating more business-side trust of the team, which in turn allows them to improve further, leading to more trust and an increasingly collaborative relationship.

Almost the direct opposite of the Technical Practices First pattern is the *Iterative First* pattern. The initial focus of this approach is solely on getting a team to work iteratively. Unlike Technical Practices First, there is little concern for the specific engineering practices of the team, except where those practices impede the team's ability to work iteratively. The Iterative First pattern is based on the notion that moving to an iterative process is often much easier than moving a team all the way to agile. Once a team has begun to work iteratively, the shift to agile is much easier.

## Advantages and Disadvantages of Technical Practices First

**There are a variety of advantages to the Technical Practices First approach:**

- Because teams don't debate which practices to begin with-they are supposed to all be adopted-very rapid improvements are possible.
- The transition can be very quick, especially in teams that are already predisposed towards adopting these agile practices.

**The disadvantages to this approach include:**

- The technical practices often support each other in subtle ways. Introducing practices piecemeal and in the wrong order may lead to undesirable results.
- Introducing some of the technical practices will likely present some of the more difficult of all of the challenges of transitioning. Many individuals are extremely reluctant to try such things as simple design, pair programming, and test-driven development.
- Some of the technical practices require significant new learning, which will likely require

outside trainers or coaches, increasing the cost of the transition.
- By focusing so strongly on improving their technical practices some teams move further away from the user-centric thinking they should be adopting as central to becoming agile.

## Advantages and Disadvantages of Iterative First

**There are certain strengths to the Iterative First approach:**

- It's easy to start. In some ways the Iterative First approach can be started in the same way we used to teach a kid to swim-throw him in the lake and let him figure it out. A team can be started by doing as little as saying, “Four weeks from now I'd like you to have a new version of your product in a shippable state.” Having never done this before, the team will panic, but they'll also likely figure out how to do it.
- It's hard to argue against the benefits of working iteratively. Some of the agile technical practices such as test-driven development and pair programming can evoke fierce opposition from team members. Most team members are willing to work iteratively; the most common resistance will be only to the length of the iterations.
- The primary drawback to the Iterative First approach is that the team may not choose to go any further and may not adopt any of the other agile engineering practices. They may feel that having become iterative they are now “agile enough.” They aren't; they've only just begun.

## Stealth Mode or Public Display of Agility

The final choice to make is whether to publicize your transition. One option is to operate in *Stealth Mode*, or in other words use agile methods but keep that fact private until the project is complete. This happened at one of my client sites. On my first visit there, I spoke with Sarah, the director of the company's project management office. She told me that the transition to agile was underway and had begun after I had delivered a two-day training class to many developers at their headquarters. Sarah outlined a well thought out plan to introduce agile across her company's more than 200 developers.

Sarah's plan showed four initial pilot teams, each of which had been selected for very specific reasons. One team was chosen for their willingness to relocate into a shared team space very

different from the dedicated cubicle environment in use at the time. Another team was chosen because they would be one of the first to use a new technology in which the company was making a significant investment. The other two teams were selected to be part of the pilot for equally good reasons. Sarah's plan was great because it was going to maximize the learning created right at the outset of this transition effort.

I left Sarah's office planning to visit each of the four teams so I could get their perspective on how things were going. Strangely, though, I didn't find four teams-I found five. When I figured out which of the five wasn't one that Sarah had told me about, I went back and talked with that team some more. I discovered that while to any outside observer they looked exactly like any other agile team, they were not an officially sanctioned part of Sarah's pilot project.

They had noticed one of the official teams, liked what they saw, and decided to try it themselves. They had a vague sense that they probably shouldn't be doing what they were doing and had placed their wall-hanging task board and burndown chart well inside a labyrinth of cube walls. I had only stumbled across it because I was unfamiliar with the building and had gotten lost looking for one of the official teams. This team was transitioning using the Stealth Mode pattern. They were using an agile approach, but were keeping their activities to themselves until the project was complete.

In contrast to a Stealth Mode transition is the *Public Display of Agility*. In this approach the team or organization announces with great fanfare that they are transitioning to agile. Depending on the scope and significance of the transition, the announcements may range from lunch room comments to other teams all the way up to press releases in the national media.

## Advantages and Disadvantages of Stealth Mode

**The advantages to a Stealth Mode transition include:**

- When operating in Stealth Mode you can wait until the project is successful before indicating that the project was run in a different way. Or, if the project fails, you can adjust your agile process, try again, and only tell people you're using an agile development process after you've figured how to do it successfully.
- If you start quietly such that no one but the individuals involved know about the change, there's no one who can tell you to stop. This fits with the old adage of it being easier to ask

forgiveness than permission.

**There are of course drawbacks to Stealth Mode:**

- You won't have any organizational support. If you need organizational support for some of the necessary changes you may not be able to get it.
- Announcing at the end of a project that the project was successful in part because of agile is much less compelling to skeptics than telling them upfront. Of all Babe Ruth's home runs the most famous is the 1932 “called shot.” With a count of two balls and two strikes Ruth pointed to the centerfield fence and hit the next pitch into the centerfield bleachers. Saying what you'll do and then doing it is more powerful than announcing your goal after it's been achieved.

## Advantages and Disadvantages of a Public Display of Agility

**The advantages to the Public Display of Agility include:**

- Everyone knows you're doing it so you're more likely to stick with it.
- Publicly proclaiming your intent provides an opportunity to create thought and discussion around the goal. With the intent out in the open, team members will feel comfortable talking about the transition with those outside the team. This hopefully will build support among some individuals and get the resistance of others out in the open for discussion.
- Unlike a Stealth Mode transition there's no backing away from the Public Display of Agility. This makes a very powerful statement: not only does the organization plan to initiate the transition, it plans to succeed.
- Like the All In approach, the Public Display of Agility certainly demonstrates a high level of commitment to agile and the transition process.

**There are of course drawbacks to the Public Display of Agility:**

- Announcing something before you do it can make you look foolish. Pointing to centerfield and then turning the next pitch into a home run over the centerfield fence is one thing; pointing to centerfield and then striking out on the next pitch is another.
- Announcing your intentions before having moved very far toward them is a sure way to bring out the naysayers and their objections.

Organizations benefit from making deliberate decisions between Start Small and All In, Technical Practices First and Iterative First, and between Stealth Mode and Public Display of Agility. While I've seen all eight combinations result in successful transitions to agile, some combinations of patterns are used more frequently than others. Going All In, for example, is most commonly combined with Iterative First because of the complexity that would be involved in changing the technical practices of an organization all at once. By consciously considering these patterns of agile adoption and choosing carefully you will be able to improve your chances of a successful transition.

---

Posted: January 4, 2008

**Tagged:** teams, management, programming, testing, project management, transitioning to agile, agile practices

---