Dr. Dobb's Portal
The World of Software Development

10 Mistakes in Transitioning to Agile

Levent explores the 10 most common mistakes in the transition from legacy development methodologies to Agile.

By Levent Gurses, Dr. Dobb's Journal
Nov 01, 2006
URL:http://www.ddj.com/dept/architect/193402902

Levent is a technology consultant and cofounder of Jacoozi. He can be contacted at levent.gurses@jacoozi.com.

While agile software development may seem straightforward at first, the transition period can be particularly challenging. In this transition, many companies face unique challenges and make mistakes. Most of the available literature concentrates on what steps to apply for a successful transition, yet little has been said on what to avoid. In this article, I explore the 10 most common mistakes in the transition from legacy development methodologies to agile. Like land mines, these mistakes are easy to make and are things that every software-development manager should try to avoid to achieve the greatest return on this important investment.

Mistake 1: Go All In

Months of discussion and e-mail threads are finally gone and it's been decided that agile is the right thing for your organization. Be bold. Do not start with a pilot project; that's for people who have all the time in the world. You run multiple projects and you have no time—pick a big and risky project or force this new methodology to all projects simultaneously. If you find yourself looking for a good disaster candidate, try to find the one with a new technology and/or a tight deadline. Because you won't have time to learn from mistakes, something pilot projects are specifically conducted for, you'll multiply them in all teams putting your agile initiative in life support.

Mistake 2: Go Fast to Go Fast

Do not take the time to get your group and other groups within your organization to understand agile development. Especially if you're also growing fast and creating new departments, do not slow down and throw them into the mix. For example, say you just established a new User Experience group. Before you even have time to define the new group's roles, throw them into the agile initiative.

Make sure you don't clearly define the terminology, such as the similarities and differences between use cases and user stories. This way, everybody will have a different idea of what is being created and that will contribute to the overall confusion.

Mistake 3: Ignore the Corporate Culture

What is corporate culture? Things such as:

- The way projects are managed (estimation, project planning and monitoring, scope management, and so on).
- Teamwork versus individual heroism.
- Employee attitude. Brave and aggressive, innovative, and risk-taking or conservative and protective.
- Departmental relations and politics.
- Fun and learning.
- Pace of development.

Agile development is based on teamwork and cross-functional teams. Many organizations are composed of departments and sometimes walls exist between them. While forming the agile team, make sure to ignore or overrule the existing departmental boundaries. This helps you alienate department heads and quickly take their support away.

Mistake 4: Fail to Identify the Sponsor

This is different than project stakeholders. This is more like the person who pays the bill and, at the end of the day, needs to be convinced that agile works. Fail to identify and engage them early and often and you guarantee the day they'll come and ask for the return of their investment. For example, you have a body of use cases produced over a period of a year. You find out that they are not good enough and direct the team to either rewrite a portion or convert them to user stories. Because this is a time-consuming process, do not communicate that to the sponsors so that when they ask what happened to the old use cases, you have nothing to say.

Mistake 5: Fail to Define Roles Within the Agile Team

Do not define the roles of each team member. This creates confusion within and outside the team that keeps everybody messed up for the time to come. This also helps in the rise of agile supercoaches, a new breed of Swiss-Army-knife-type people that, among other things, can do project management, coding, architectural design, training, and more.

Mistake 6: Do Not Create a Project Plan Before the First Two Iterations

You have your user stories and you assigned them points. It already took you seven weeks to rework and you are now at Iteration 0. When the boss asks you for a rough project plan, just tell him that you'll have one in six weeks.

Mistake 7: Overdo the Team-Room Concept

Let's say that the business architecture team is finalizing the business model and clarifying requirements. You get a sense that it could last for several weeks, if not months. Move your team to the team room before you are ready. Make sure to put the entire team of engineers, including the architect, in the team room and only then decide what to have them do. This not only ensures that they stay underutilized, but you'll also bore the team and contribute to their disengagement from the project. Resource underutilization and disgruntled employees are two good reasons to kill your agile initiative.

Do not allow or encourage remote work, because it is against the team-room spirit. Make sure people are in the room all the time.

Mistake 8: Trash All Computer-Based Project Management and UML Tools

Do away with all project management and UML tools completely. Do not use project management tools such as XPlanner or VersionOne. Do not use any UML tools such as Rational Rose or TogetherSoft. Use whiteboards only and take digital snapshots every time you work on an architectural design. Make sure the team has to rework the design completely every time something changes, just by looking at digital pictures. Do not create a domain model. Do not setup a Wiki or another versioning system for tracking your user stories. Lose the deltas of the use cases or user stories and just get the latest version. Since you can't really get back to any date, forget what you did and why. This surely is a good way to lose track of fast-changing requirements.

Mistake 9: Choose Your Key People Poorly

This is valid for all projects, but in the case of agile development where teamwork is so important, it is even more accentuated. If you have only one employee that has done any agile work in the past, regardless of competence and attitude, put him/her in a leadership position. Especially if this employee has a polarizing character, it quickly cools the team off the project and eventually shows itself in the final product. That's if the project ever delivers one.

Similarly, if you have a budget to hire consultants to help, make sure to go with the cheapest. That's generally going to ensure less leadership quality and dedication, which can't be bad if you want to kill a project. Choose an architect that strongly believes in "big" design and is afraid of not knowing what's going to happen after the requirements change. Someone who has been around for a long time doing many designs, mostly for tomorrow, would be ideal. He/she will play the small design game poorly and will eventually want to over-design every part of the system, which leads to a slower team velocity and architectural design anxiety.

Mistake 10: Make Agile the New Religion

Evangelize this "new" way of doing things at every chance and make sure to promise great things. Be passionate to the degree of pushing. The time comes when someone asks for all the promises and that alone is a great opportunity to lose personal credibility and faith in the process. In addition, this is a good way to make agile the scapegoat of a failed leadership or project.

Divide and be conquered. Classify people as either "agile believers" or "heretics." Camps are always good when it comes to making a case against something, in this case agile development.

Conclusion

There are certain things that you can do to ensure a flawless transition to agile. I've focused on 10 things you should try to avoid. Some are applicable to most software-development organizations, regardless of industry and size, while others are based on my personal experience, mostly with clients, and may not apply universally. One thing to remember is that while books and articles can teach a lot, a little up-front analysis and constant real-time monitoring are key factors in every agile transition. A wise man once said "slow down to go fast."