Print Article Close Window

From: www.cio.com

Top 10 Ways to Blow Up an Agile Project

– David Taber, CIO

May 30, 2013

This article is the second in a series of excerpts from the second edition of David Taber's Salesforce.com Secrets of Success*, which is scheduled to be released later this year.*

In the spirit of David Letterman's "Top 10 Lists," it's time again for a list of worst practices that can make even the best agile team melt down. Here's David Taber-man's newest Top 10 List of things you can do to make that a reality. (Yes, I've seen specific examples of every one of these.)

10: Start a Crash Project With a Big Budget

The idea behind agile is to break down communication barriers and have software engineers work on what the users really need, rather than what somebody was willing to push through a committee and transcribe in a big document. Teams are small and tight.

How-to: 5 Ways to Send a Custom Software Project Off the Rails

In contrast, crash projects that throw all the resources in at the beginning don't allow anybody time to think or plan—yes, you do that in agile—so the team will tend to whipsaw for the first few weeks. If you really have a crash project, start it with a small team of architects and developers who've worked together successfully before and add team members only as they ask for them.

9: Demand Requirements Tomes, Data Dictionaries and Lots of Standards

These management artifacts are terrific for hardware projects or monolithic software applications, such as an application package or a network switching system, but those aren't the kind of projects where agile thrives. Most business apps are incremental adds and integrations with existing functionality. While agile projects involve careful thinking, it has no use for big documents or heavy processes such as Six Sigma, software lifecycle management, ITIL or ISO 9001. They only get in the way.

Case Study: Rapid Application Development the Zappos Way

The clean code folks take it one step further. They don't want any comments in the code, partly because comments are often be misleading, partly because commented code is an invitation for people who don't really understand the workings to break code by trying to improve it. User training cheat-sheets and recorded webinars are fine—but you'll notice that they, too, are short.

8: Demand Fixed Price, Fixed Schedule

Agile builds on an old joke: "Price, schedule, features and quality; pick three." By doing software in a fundamentally different way—one that's user-centered, conducts tests first and constantly negotiates and re-prioritizes requirements—agile dramatically reduces waste, because the stuff you really didn't need never gets started in the first place.

But an agile project with a fixed set of goals, complete with an inflexible budget and immovable schedule, is an oxymoron. The only behavior you'll get out of the team is friction.

7: Demand Monthly Waterfall Metrics

Agile has its own style of management, which is fairly inward-looking. It's based on the rugby scrum or a football huddle, and it's not focused on producing management reports. Furthermore, since outsider managers don't really know what's going on hour by hour, there's not a lot of value they can add to the efficiency or effectiveness of the agile project.

Commentary: Kicking Waterfall Project Management Habits Requires Agility

Of course, if you're the budget holder, you want to see dashboards and flowcharts. Try to resist this; preparing those translations of agile's native project mechanisms—cards, stories, integrations and burn-down rates—won't make the team any more efficient. In effect, making these demands just adds a tax to the project.

6: Don't Let Users Participate Until the Very End

Agile projects succeed when there's direct, continuous contact between the developers and the users. It's not just user-centered design; it's intimate connections with the business processes, team know-how, and continuous user testing that help avoid the bugs and the real-world "gotchas" that plague most software projects.

Related: How Giving Enterprise Software a Social Makeover Attracts Users

While some managers say that they can't afford to put a user on the team, doing so is like building a custom house without being willing to actually visit the site during construction. Nobody would do that with his own home. Why would you let that happen with your IT investment?

5: Accelerator-Brake, Accelerator-Brake

Agile lets developers and users innovate fast and become high-functioning teams. This depends upon continuity of effort and objectives. Sure, detailed priorities shift continuously—but if the global purpose of a project jumps around, the good effects of agile will never take hold.

Case Study: Formula One Racing Team Speeds to Agile Development

The same is true if there are interruptions of more than a couple of weeks, as the flywheel never really gets going. Remember: Continuity matters. Due to a number of human factors, restarting an Agile project that's been stalled for a month is almost worse than starting from scratch.

4: Don't Give Teams Access to Test Resources Until Beta

What could be duller than testing? Seriously, the continuous building and changing of agile software makes testing an endless chore for the users and a bore for the compliance guys. Besides, the test labs can't afford to be available all the time.

How-to: Adjust to the Changing Face of Software Testing

You'll hear these kinds of arguments. You have to overrule them. You don't have to be a card-carrying member of the test-first crowd to know the economics of software defects: Catch them early and fix them cheaply. Here's the agile answer: If you test throughout the project, you won't need a beta at all. The software will just migrate into production when it's ready.

3: Use Politics for Leverage

You know the drill. Find and emphasize Catch-22 situations, exploit optics and rhetoric, use budgetary threats, shuffle the team, promote and demote people randomly, install a Czar—preferably someone who knows nothing about software but is an I'm-in-charge-here, get-it-done, take-no-prisoners kind of guy—then lather, rinse and repeat. These steps work wonders to show your power, and to break down internal trust and external communication.

Commentary: Why CRM Implementation Is So Political

If you see yourself characterized in Dilbert cartoons, and you're not Dilbert, watch out.

2: Don't Put Your Best Players on the Team

One thing's for sure: The folks who conspired to write *The Agile Manifesto* are smarter than the average bear. Way smarter. Agile tends to work best when you have users who are articulate and know their business processes cold. Effective agile developers are usually wicked smart.

Case Study: Hotels.com Speeds App Development with Agile

It's not just soloist brain-power, though. The team members must respect each other. If they don't, communication will not be quick and smooth. In my experience, location matters, too. It's best if team members are within shouting distance, if not in the same room, while it's almost impossible to have great productivity if the team is spread across multiple time zones.

And the No. 1 way to blow up an agile project…

Use the Element of Surprise

The agile effect comes from trust (through a team that's deputized and empowered to make key decisions on its own), as well as close communication and relentless focus on the highest-priority goals. Surprise the team with new mandates, unstated top priorities, an uncertain budget, political cross-fire or just a simple lack of trust and it's hard to imagine anything more damaging to both morale and productivity. Surprise your team and watch it melt.

Thanks to Rich Mironov and Bo Laurent for their assistance with this article.

David Taber is the author of the Prentice Hall book, "Salesforce.com Secrets of Success" and is the CEO of SalesLogistix, a certified Salesforce.com consultancy focused on business process improvement through use of CRM systems. SalesLogistix clients are in North America, Europe, Israel and India. Taber has more than 25 years of experience in high tech, including 10 years at the VP level or above.

Follow everything from CIO.com on Twitter @CIOonline, Facebook, Google + and LinkedIn.