3 Ways to Meet Your Software Project Deadline

Software development is inherently unpredictable. Many dollars have been spent on training, methodologies, and consultants in the attempt to guarantee software delivery dates. These have been largely in vain. Although there are valid techniques that can improve your estimates, sometimes, sadly, those estimates will be wrong.

As a proponent of Agile methods, I believe that agility is about discovering how wrong you are as quickly as possible. Failing quickly gives you time to adjust, scale back, or just cancel a project before throwing good money after bad. It gives the business information about the true nature of a project, not an artificial sense of well being just before the explosion.

Agile methods use iterations, or short, time-boxed development cycles. During an iteration, developers work on the highest priority features. They plan to complete whatever they committed to by the end of the iteration.

Due to the unpredictability of software development, you can be sure that sometimes the team will have too much to do (and less frequently, not enough to do) if they are estimating honestly.

When there is too much to do, the best option is to cut scope, since changing the iteration date undermines the primary advantage of time-boxed iterations - quick technical and business feedback.

There are three ways I know of to cut scope, each of which has different tradeoffs. Here's a rundown:

Simplify Over-engineered Designs

On any software project, there are many technical decisions that can complicate the implementation of a feature. Sometimes these are unavoidable, such as the need to preserve legacy behavior, or support multiple variations. More often, though, a team will decide to use an approach that makes good technical sense, but adds on a good deal more complexity.

Make sure that you are doing the simplest thing that meets the requirements for today's functionality, testability, and ease of maintenance. Planning too far ahead for the future can hamstring a development effort, and force project delays.

For example, many teams get caught up in "architecture", which often means building complex application frameworks and middleware, some of which may never be used. This is especially dangerous early on in a project, when details are still fuzzy. Over-building of infrastructure is often a sign that the development team doesn't have enough details about what features to implement, and their priority.

Simplify Features

Sometimes the communication between business users and developers can be challenging, even with an ideal on-site customer (as proposed by XP). Many times, a specific feature request can be solved by alternative means that still addresses the business problem. This is a great way to reduce scope while still addressing critical business needs.

So if Joe Customer asks for a automatic notification whenever any order is placed in the system, maybe he actually would be fine with a daily or hourly email with an order summary report. The latter might be a matter of minutes to implement with a simple database report, while the former might mean intrusive, risky changes to the order processing logic in the system.

Cut Low Priority Features

As a last resort, you may need to eliminate the lowest priority features to make a deadline. This is why Agile methods stress that business priorities should be clearly understood before each iteration. If you run

out of time working on something technically difficult, but relatively unimportant, it's hard to explain why the ten simple, higher priority features won't make the release.

Communicate with your customer about the proposed cuts as soon as possible to allow them to re-negotiate priority if necessary. You build trust and confidence by involving the stakeholders in tough decisions while there's still time to react.

So the next time you run up against a deadline, don't just cut features, first look for ways to cut effort while still delivering value. Your customers will appreciate it.

About The Author

This article was written by David Churchville, founder of ExtremePlanner Software (http://www.extremeplanner.com). David has over 15 years of experience in software development and project management, including over five years working with Agile development methods. ExtremePlanner Software develops software products for empowering Agile software teams to work together more effectively.