The Three Software Project Development Traps (And How to Avoid Them)

Why do software projects fail?

Such a wide open question invites a multitude of potential answers. Perhaps, in a rush of zeal and excitement (not to mention the desire to satisfy hungry shareholders), developers jump into their projects without truly accurate perceptions of what it will take to complete their work. Maybe their approach ends up being too inflexible, and they fail to make necessary adjustments as requirements and demands change. Possibly, managers, fearing that their projects are running overschedule, make the mistake of ramping up staff unnecessarily, resulting in cross communication, mixed messages, and even greater time and cost overruns.

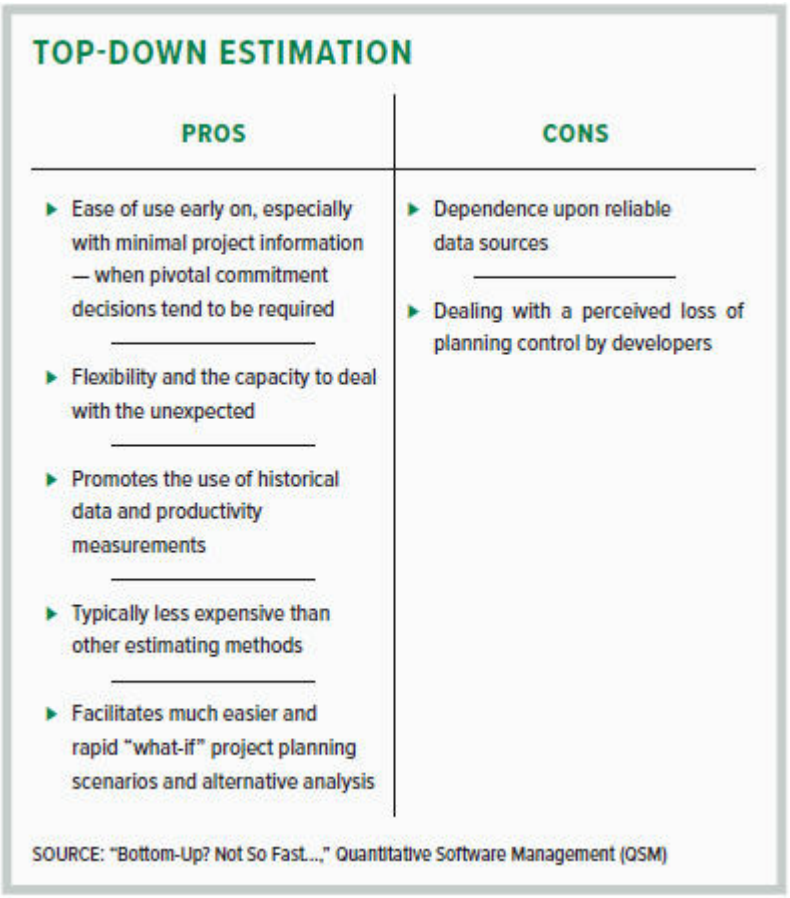
More likely than not, the answer is “all of the above.” Software development can be enormously challenging in the best of circumstances, but it can be downright impossible when projects are rushed, inflexible, or plagued by inaccurate scopes and estimates. These are serious pitfalls that can scuttle projects before they even begin. Avoiding them is critically important for software companies that want to get out of the costly cycle of jumping into development projects, struggling through challenges, running over time and budget, failing — and starting the whole process all over again.

To sidestep this costly fate, consider the following strategies. These can help you avoid some of the biggest software project development traps and keep your projects running smoothly.

Create Up-Front Plans Based on Scope-Based Estimates

Traditional bottom-up approaches to software development can lead to trouble. These approaches typically involve having each team member estimate how long it will take them to complete specific tasks and managers compiling their input into a spreadsheet to get a sense of how many hours will be involved. In reality, this process only gives the illusion of control, as it is based on incomplete information and guesswork. Indeed, poor or inaccurate planning can be one of the biggest culprits behind missed deadlines, additional costs, unhappy stakeholders, and failed projects.

A top-down, scope-based approach can be far more effective. Scope-based estimation involves accurately determining the entire scope of the project at the outset. It takes into consideration how big the project will be, the efficiency of the team, and the complexity of the work.



Proper scope-based estimation incorporates historical data into the up-front planning process. This information can be gleaned from past projects the team has worked on, but it can also be taken from outside the company by looking at industry trends via software project databases. Through these sources, managers can compare impending development projects to projects that have already been successfully completed. They can set accurate deadlines and compile pinpoint cost estimates, plan for anomalies, and set their teams up for success before a single piece of code is written.

Leave Room for Reforecasting for Agile Response to Changes

While some see careful up-front planning as anathema to agile development, the two can go together like hand-in-glove.

True to their name, estimates can easily fluctuate and be easily adjusted based on the needs of the development process. For example, a team using scope-based estimation may find very early on that a particular time frame may not be achievable unless certain priorities are shifted or even removed completely. They can adjust the scope accordingly to create a more realistic and adaptable plan for completion.

This adaptability can help teams meet their agile goals of delivering “minimum marketable products” — solutions that provide the bare bones minimum value that their customers are expecting. It allows teams to focus on core pieces of functionality and adjust work schedules to prioritize creation and delivery of those components.

This can take place throughout the development life cycle. Longer projects might be more prone to changes and require managers to stop development and reestimate at various points in the project. This recalibration is both acceptable and encouraged. Reevaluating allows managers to explore different scenarios, adjust their scope, and look at various options to get their projects back on track.

Focus on Smaller Rather than Larger Teams

When projects do go off track, often the first reaction is to throw more bodies at the problem. The consensus is that the more people working to solve the issue, the faster the issue will be resolved and the project right-sized.

In fact, evidence suggests the opposite will occur. Noted computer scientist and author Fred Brooks once wrote, “adding manpower to a late software project makes it later.” Overstaffing can lead to greater complexity, increasing the chances of miscommunication, defects, and rework. Adding additional personnel will also take resources away from development as developers take time to train new people.

Managers can use the historical data and trends cultivated during the scope-based planning process to find the right balance of human resources. The planning stage will allow them to more accurately match the number of team members assigned to a project based on estimated demand and capacity. In some instances teams may need to be larger, but smaller teams will likely be more efficient and productive and better able to respond to changes as the project moves along.

While there are many reasons why software projects fail, there really is no reason they should fail, at least as far as planning goes. Developing up-front plans that incorporate scope-based estimation can help developers avoid the traps that routinely cause software projects to falter, resulting in a smooth, agile, and successful path toward completion.