Costing Extreme Programming

In our efforts to drive development in an ExtremeProgramming (or possibly more generally Agile) direction, we continually run into a seeming catch-22 with regards to costing/budgeting the project. Many companies today rely on a budget-then-build process to manage their IT spending. In this environment, a business solution is suggested, a cost for development of the system is estimated, and funding is either granted or denied. How is it possible to reconcile this process with the PlanningGame required by XP? If we are required to estimate the entire cost/time of the project without any development, we have no ProjectVelocity on which to base our estimates. In addition, we require a fairly complete set of UserStories/Functionality in order to estimate the resources/time required. Trying to determine these things begins to look a lot like BigDesignUpFront.

I realize that it would be simpler if management would simply buy into the PlanningGame, but our experience has been that unless funding can be secured, then (at least in the minds of the business) there is no project to run the planning game for.

How do we reconcile this?

See OptionalScopeContracts.

Answers:

It is possible that it will require an old-fashioned ParadigmShift. It would be super of management were to buy into the ideas. The need to do lots of up-front estimation and planning could be replaced by actual, running (albeit small) systems. These systems could be put to actual use. As a side effect, this would provide feedback that could help you determine which features are worth the expense. These decisions could be made on a granular basis giving more power to the business people (and more fun for the programmers).

But we're getting ahead of ourselves.

Until you are able to prove the worthiness of your extreme ideas, you could continue using your current estimating techniques. The advantage of also doing XP is this: you'll know how accurate your original estimate is much sooner than usual.

Ask for a smaller budget.

Ask for a small budget to demonstrate the worthiness of the project. Make the time frame small, like two or three iterations worth (one to two months). After that time, have an evaluation session to determine the worth of continuing the project.

If your management is resistant to this, encourage them to try it out at least once. Don't screw up! (*) Argue that it's not going to cost a lot to try this for one test project, and it may be worth evaluating this budgeting strategy as one that could potentially reduce costs when rolled out. After all, catching useless projects or useless teams early is a big cost savings than waiting two years for a failure.

In general, this is really the better way to go anyway. It's counter XP to request a year long budget if it's conceivable that the project will be canceled in a week. The real problem comes from the budgeting process that has its own cycles and processes. You will have to play into that because you have nearly no hope in Hades of changing it. If project budgets must be done at the beginning of fiscal year, then you will need a large project budget. But don't allocate the money to a specific team or project, but to a class of projects. i.e. Don't allocate it to the "Thrompwhistle 3000", but to the general class of developing the whistle line of products.

In my experience, many medium sized or smaller companies are flexible to such creative budgeting.

(*) Maybe it's better to screw up on purpose to demonstrate how valuable it is to be able to cancel a project early on.

-- SunirShah

I'm experiencing this pattern right now as a start up freelancer.

Another approach is to ignore XP, and cost it however you would have costed it before. In theory, they will end up getting more sooner (i.e., more for less) with XP than without, so you should end up with a happy customer. -- KyleCordes

Not even in theory. In theory, you won't know your project velocity until the end of the first iteration. And this is likely to change as the project progresses. If you estimate the 'old fashioned way', you'll likely underestimate the 'old fashioned way', and get screwed anyway. ProgrammersAreCompulsiveOptimists, they always underestimate (as a rule of thumb). Even XP can't save you in that situation.

My assumption was the that 'old way' would have had to generate a rather generous estimate, to accommodate for the 'old way' attempting to guess at the beginning what the whole thing would cost. After a few times over the waterfall (so to speak), organizations learn to estimate high enough, in spite of the optimism you mention. High enough for whatever you were doing before should be at least high enough for an XP approach. - KyleCordes

If ProgrammersAreCompulsiveOptimists, the SystemAnalyst or whoever sits in that position should double or triple the estimates with the proviso than any extra funds be rolled over into the next project. Voila.... after a few projects you (the development department) should have the extra funds to demonstrate costing the XP way.

Wouldn't you be concerned that seriously inflated estimates would a) stand the risk of losing the contract to a more optimistic competitor, and b) lead to inflation of expectations?

Yes and yes.

I was suggesting that the SystemAnalyst specify the higher estimates in the hope that budgeting will bargain itself into XP. Ie: Budgeting decides to come back and say, "Your pessimistic cost is too high, can you do it in X days at Y cost." You already have the optimistic cost and the ability to suggest XP guidelines be used as the only way to meet the schedule and toss in for outside customers that this allows incremental billing (ohhh.... wouldn't financing love that turn of events) with near-immediate customer ROI and the ability to terminate the project at any step X with cash in hand. If you run over the new lower estimate, you've already used the old CYA adage and are a step ahead.

Um, why not just do a ReleasePlan, estimate your velocity based on your last project, and use that number?

That's just the same thing as OldFashionedEstimation, and still prone to error and all the other problems. I don't think the issue is "How can we fit XP into the OldFashionedMethod?", but "How can we change the OldFashionedMethod to work with XP?"

What if ReleasePlan is *not* "just the same thing"?

I meant that "use that number" is "just the same thing", not the ReleasePlan. Using any number to fit into a budget is anti XP if that budget is rigidly fixed

Using any number to fit into a budget is anti-XP if that budget is rigidly fixed as it is in most organizations today. I don't think it is possible to do BookXp with a fixed budget of any kind. The closest thing I've seen is FixedPriceContractsWithChangeControl.

Or, call the first scrum/deliverable the "architecture", call the second the "proof of concept", call the third the "alpha version", etc.

In this discussion I don't hear any one talking about why the old-fashioned budget-build process is used in the first place... In my opinion, it is because management needs:

- A cost-indicator for a longer period of time
- To be able to steer the company
- To make sure they can cover the cost (let alone justify it)

They also need a long term global indication of where the project (after x iterations) will be heading, to be able to determine:

- If it fits in the overall company strategy
- If it is in line with other projects
- Etc.

In that way "the old fashion way" can be sufficient; it can be complementary to the short term techniques suggested for XP. You just need to be sure that you tell, with your overall budget, all the assumptions and disclaimers that you made to reach your overall number. And that's the hard part, and difficult to master... but it's doable.

-- BasDeBaar

Which makes me wonder why companies need a long-term indication of where things are going. The way the world is, your business context will change two months from now anyway. Imagine doing all the estimates and plans for a one-year iteration, and then starting development on September 10, 2001.

This is the curious thing about agile methods: They promise to make our software development processes as nimble as possible, but they end up largely exposing the inertia of the customer. What was it GerryWeinberg said? Something about how when you solve your number one problem, you only promote your number two problem ...

As a freelance contractor, I've also found the best approach to dealing with fiscal-year budget estimations is to ask for a large block up-front to handle a class of projects. The principle problem with that approach is that you need a close relationship with someone with budget control - someone who understands you're working in their best interest and can afford to block such a large amount. To get that far, you usually need to have proven your worth with smaller, more trivial projects in the past, or else have a *very* convincing argument.

On small scale projects, it works fine - essentially, if you're planning by a 'Cost Driven Commitment' (which, working hourly, has a tendency to equate to a Date Driven Commitment - I've asked about this in PlanningGame), you figure that you'll want to schedule, say, a half dozen $5000 projects, so you ask them to block in a budget of $30,000 for the year. But for a larger-scope XP project? I don't see how it would work, unless you just do an OldFashionedEstimation and multiply it a few times over in the hopes that one or more XP projects (of varying scope) can squeeze into that window.

Or is there another way around it? How, in particular, do you cost an XP project for, say, a VP of Marketing (I do web-based application design) who has to turn in a fixed budget once each year? Irrelevant of whether or not that's a *good* way to budget (also, FYI, I think stock prices and investors factor into their need for long-term predictability), it's an inevitability. And XP is all about flexibility, eh?

- Joseph Riesen

I do not believe there is any difference in preparing a budget for an Extreme Programming project versus a traditional project. The majority of the budget comes down to number of programmers times length of time. In a yearly budgeting process, this length of time is either one year or less than one year. Going in, someone makes his best guess about how many programmers will be needed and how much money can be allocated. There is a period of negotiation where funding is increased and number of personnel is decreased and then the budget is set, although a mid-year adjustment may occur. XP does not magically add or remove bodies as the project progresses, nor does traditional project management, in either case, one just makes do with the people at hand. The advantage of XP is that if the budget prediction is low, one can deliver a partial solution. With a traditional Waterfall, it is all or nothing. --WayneMac

---

Last edit May 2, 2014, See github about remodeling.