Harvard Business Review

PROJECT MANAGEMENT

Your Agile Project Needs a Budget, Not an Estimate

by Debbie Madden

DECEMBER 29, 2014

Nearly every software development project starts with one question: How much is this going to cost? Perhaps it's a stakeholder who asks the question: a CEO, Board member, VC, or boss. Maybe it's been asked another way — How long will this take? How many stories can we get into this next sprint? How many people do we need to hire to get this done? How much money do I we need to raise?

These are all variations on the same question — how much effort is this going to cost in terms of time and/or money?

There are two common replies to this question:

1. We don't know;
2. Let us estimate and get back to you.

Stakeholders and decision-makers don't like the first reply because they desperately need an answer to their question and they don't have the knowledge to answer it themselves. Technical teams don't like the second answer, because estimating takes a ton of time and it's often abused — stakeholders sometimes turn the "estimates" into maximums and get upset if the team exceeds them.

Yet the truth is: It's the responsibility of the technical team to answer the question "How much is this going to cost?" because technical teams are the ones that have the most relevant knowledge to answer it. If I ask you, "How much does it cost to build a ballpark?" only those of you who have first-hand experience building a ballpark can truly answer. The rest of us are taking a stab in the dark.

Here's the thing: "How much is this ballpark going to cost?" is a fair question. Do I need to buy a shovel so I can build a sandlot? Or do I need to raise $100 million dollars to build Yankee Stadium? Stakeholders have decisions to make and deserve to have an idea of what their investment is going to cost and if it's worth pursing.

A recent Harvard Business Review article revealed that one in six IT projects has a cost overrun of 200%. That's a pretty high rate of failure for estimation. To minimize the risk of having your next technical project go awry, stop estimating and start budgeting.

For most strategic decisions, estimating is too precise. Estimating breaks down a software project into granular, 1-to-3 day chunks. For a $100,000 project, that's a lot of work. For a $1 million project or bigger, that's inordinate. If you attempt to break an entire project into estimates at the beginning of the project, you are truly wasting weeks of your time. Why? Because there is no way that you are going to get estimation at a granular level correct at the beginning of a project.

Instead, here's a four-step, tactical approach to budgeting. It takes 20% less time than estimating, and it's easily updated throughout the life of your project.

Step 1: Identify Decisions

Before you even begin to think about budgeting, or estimating for that matter, it is critical to know what decisions you are trying to make. What will you do once you have the data? What are you trying to learn?

Some examples of decisions you might be trying to make are:

- Should we start this project or kill it?
- Hire more people or outsource?
- Start marketing this feature?
- Build this project next or deprioritize it?
- Launch a startup?
- Allocate budget this quarter?
- Code this set of stories next?

Do not proceed to steps 2, 3 and 4 without truly knowing what decision you are trying to make. If you can't identify which decision you are making, estimating *and* budgeting are both a waste of time.

Step 2: Match Precision to Decision

Once you identify the decision you are trying to make, your next task is to match the tool to the job. For example:

- If you are making a detailed decision (i.e. How much will we get done this iteration?), then estimating *is* a good tool for the job because its precision matches the granular decision you are making.
- Otherwise, for more strategic decisions like many of the ones listed above, budgeting is a more appropriate tool.

Step 3: Budget

When the decision you are making is strategic, budget using a top-down approach, and get only as granular as you need to in order to have enough information to make your decision.

To illustrate, let's say we are building an online bookstore. What high-level functionality might we need? Maybe something like these:

Shopping Cart
Browse Books
Search Books
Manage Inventory
Preview Inside of Book

Do we have enough information to answer "How Much Is this Going to Cost?" Probably not.

Getting more granular is warranted. Let's break down "Search Books" into more detail.

Maybe we want to search books by
Title
Author
Within the Book

Let's assume someone on our team has built a search component before, so we can say that search by "Title" will take 1-2 week, by "Author" 1-2 weeks, and "Within the Book" 4-16 weeks.

Do we now have enough information to answer "How Much Is this Going to Cost?" Probably not, so breaking it down further is warranted. If we take a ballpark guess as to the range of time each of the 5 topics will take, and associate that with a cost, we might wind up with:

SAMPLE BUDGET FOR AN ONLINE BOOKSTORE

TOPIC	TIME (weeks)	MONEY
Shopping cart	4–8	$100k–200k
Browse books	1–2	25k–50k
Search books	6–26	150k–450k
Manage inventory	12–96	300k–2.4M
Preview inside of book	8–48	200k–1.2M
TOTAL	31–180	$775k–4.3M

SOURCE DEBBIE MADDEN HBR.ORG

To sum up, our data so far is:

- The decision we aim to make is: Should we build this Online Book Store?
- We expect the project to cost 775k – 4.3M

Do we have enough info to answer "How Much Is this Going to Cost?" Maybe.

- If our budget is $500k, then we do have enough information. The answer is no, we can't afford it.
- If our budget is $5M, then we do have enough information. The answer is yes, we can afford it.
- If our budget is $2.5M, then we do *not* have enough information.

Step 4: Ranges and Confidence Levels

If our budget is $2.5M then more information is needed. We must now assign confidence levels to each topic. To do this, we:

1. Prioritize the topics
2. Identify which topics are "Required" vs "Nice to Haves"

In our online bookstore example, we come up with:

UPDATED SAMPLE BUDGET FOR AN ONLINE BOOKSTORE

TOPIC	TIME (weeks)	MONEY	PRIORITY	REQUIRED?
Shopping cart	4–8	$100k–200k	1	Yes
Browse books	1–2	25k–50k	2	Yes
Search books	6–26	150k–450k	4	
Author				Yes
Title				Yes
Within book				No
Manage inventory	12–96	300k–2.4M	3	Yes
Preview inside of book	8–48	200k–1.2M	5	Yes

SOURCE DEBBIE MADDEN HBR.ORG

At this point, we've spent maybe 2-4 hours. If we had attempted to estimate this accurately, it would have taken 2-3 days.

A note here on Minimum Viable Products (MVP): If you are using a Lean Startup approach to building your software and are building an MVP, have two options. First, you can create one budget for the entire project and list the topics to include in the MVP as "Required" and everything else as not required. Or alternatively, you can create a budget solely for the MVP and leave the rest of the product intentionally vague. The beauty of a top-down approach to budgeting is that it's so much quicker than estimating that you can easily run a budget multiple times throughout the project as you learn more.

Step 4 is difficult to do manually. To get accurate confidence levels, you need mathematical simulations. The best tool I've seen for this for software projects is Ballpark. (Disclosure: it's an application developed by my coworkers. It's free, and we don't make money off of it.) It's a budgeting tool that takes the input from a few hours of budgeting (as done above) and simulates thousands of possible outcomes using complex mathematical calculations, and in a matter of seconds spits out confidence levels that you can use to have a powerful conversation with your stakeholder to answer the "How much is this going to cost?" question.

In this case, Ballpark tells us:

Name	Mandatory	Optimistic Effort (weeks)	Pessimistic Effort (weeks)	Likelihood of Finishing within $2,500,000.00 budget
Shopping Cart	●	4	8	100.000%
Browse Books	●	1	2	100.000%
Manage Inventory	●	12	96	86.500%
▼ Search Books	●			59.000%
Author	●	1	2	
Title	●	1	2	
Within Book	○	8	26	
Preview Inside of Book	●	8	48	23.000%

- If we have a $2.5M budget, we have 100% confidence that we'll finish our top 2 priorities within our budget.
- 86.5% likelihood of finishing "Manage Inventory"
- 59% likelihood of finishing "Search Books"
- And 23% likelihood of finishing "Preview Inside Books"

Note that we input that "Within Books" isn't mandatory. We can then rerun the simulation without "Within Book" and learn that if we only look at our mandatory topics, we get –

▼ Search Books	●			81.000%
Author	●	1	2	
Title	●	1	2	
Preview Inside of Book	●	8	48	45.000%

- 81% likelihood of finishing "Search Books" within our budget
- 45% likelihood of finishing "Preview Inside of Book"

Ballpark also calculates percentiles of the likelihood of each budget range. In the Online Bookstore example, we see there's a 50% chance we'll get the entire project done for $2.75M:

Percentile	Effort (weeks)	Budget
5%	71	$1780K
10%	76	$1900K
20%	84	$2100K
50%	110	$2750K
80%	134	$3340K
90%	149	$3730K
95%	161	$4010K

Ultimately, it is the tech team's responsibility to give the stakeholder the answer to the cost question. Estimating is often a time-sink and not worth the effort this early on, with such a high-level question. So next time, try budgeting instead.

Debbie Madden has built 5 companies from the ground up and has been CEO of three of them. She is currently the CEO of Stride, an Agile software development consultancy in NYC. Prior to Stride, Debbie was the CEO of Cyrus Innovation, which she ran for 10 years, grew into a 5-time Inc 5000 winner, and Crain's NY Best Place to Work.