Is It Bigger than a Breadbox? Getting Started with Release Estimation

It's becoming clear to organizations adopting Agile methods that one still needs to estimate how long a project or a release of a product will take. It won't suffice for businesses to simply take guesses or accept unreasonable constraints. We must be able to derive credible estimates, based on a history of similar projects. But how can we estimate a project in advance, while still maintaining the ability to manage the backlog in an Agile manner?

In this article, we'll answer that question, compare release-level estimation to the techniques used for iteration estimation, and give some pointers on getting started with release estimation in an Agile environment.

If the Release Backlog Will Change, How Can I Estimate in Advance?

Arguably the biggest benefit of Agile methods is right in the name: Agile teams can respond quickly to changing priorities and conditions. But if our backlog—an organized list of project requirements to achieve a successful final product—changes throughout the course of our release, does it still help to estimate based on what we know in advance? Absolutely.

We can estimate in advance because it isn't necessary to know all the details of our backlog in order to develop an accurate prediction. **The major driver for a release estimate is the overall size of the release, not the individual items on the backlog.** Let's look at the ways backlogs change during the course of a project to see why our release estimates will remain viable.

1. The priority order of the backlog will change. (This is perhaps the most important aspect of staying Agile.)
2. The level of detail on the backlog will change. (We'll break down some of the high-level features or epics into user stories and other "developer-sized bites" that can be produced in an iteration.)
3. The specific items will change. (Some individual stories will be added and some removed.)
4. There may be a major change in the business, and some major features will be added or removed.

For change variables 1-3 above, we can expect a backlog to change frequently over the course of a project. However, these variables will not drastically change the overall size of the release, so an estimate based on overall size remains viable.

If a major change to the business occurs, our estimate may have to change, as will likely the entire justification for the project). Fortunately, while changes of this magnitude can occur quickly, without warning, they don't occur often and aren't major contributors to the changes that Agile teams typically respond to.

So barring major disruptions to the business, we can estimate our overall release while still maintaining the level of flexibility that Agile methods make possible.

Estimating Size: Differences Between Release and Iteration Planning

To measure the size of our release backlog, we can borrow techniques from Agile iteration planning.[1] However, there are several differences between release and iteration planning, such as:

- Release backlog items are less uniform in size than stories refined for an iteration.

- Detailed comparison of size among the backlog items is not as important for release estimation
- Consistency among multiple teams and multiple projects is more important for release estimation.

Let's look at each of these differences.

Non-Uniform Sizing

Many Agile experts suggest counting stories as the simplest way to estimate size during iteration planning. At this point, a team has refined the part of the backlog it needs to estimate into stories that are roughly comparable in size, so the size differences average out. For release planning, we'll need to group these items into "size buckets." For example, we might count totally new epics, new stories that enhance existing features, and modifications to previous stories, or—even more simply—count big, medium and small backlog items. Using just a few buckets makes it simple to count. To compare projects, we'll need to combine the bucket counts into a weighted count. We may decide that an epic is about the same size as five new stories or 15 modifications to previous stories, and then "weigh" projects according to this formula.

Detailed Comparison of Size

With techniques such as "planning poker," Agile teams compare backlog items that are targeted for a particular iteration in detail. But for release estimation, fine differences between items are not important; enough will change by the time these items are developed that the current details are immaterial. (If you're starting to ask questions like, "Is this item worth six story points instead of five, since it's slightly bigger than the last item we gave five points to?" you're getting too detailed and spending too much time.)

Consistency Among Multiple Teams

When measuring iterations using story points, it's important that a team is consistent from iteration to iteration. When measuring for release planning, our goal is to compare the size of one project to another. Therefore, multiple teams must measure consistently across projects. This is sometimes called "normalizing story points." This is an example of what experimental researchers call "inter-rater reliability." Simply put, we want to ensure that different team members rate most backlog items the same way.

Getting Started

"The relationship among size, duration, and effort is complex. Larry Putnam, Sr., the founder of QSM, codified this in "The Software Equation."[2] What's needed is a history of comparable projects to use for determining the effort/duration tradeoffs for a project of a particular size. You can collect this history for your own organization, or you can use industry-wide data."

While collecting data for your projects, you'll need to work on establishing consistent measurement methods (inter-rater reliability), so you can compare the size of projects. Here are some possible steps:

Build Consensus

Build consensus among your teams on the technique and scale you will use to measure size. The specifics of the technique are much less important than applying it uniformly, so keep the debate about "favorite technique" to a minimum. You may decide on a standard set of "buckets" to count ("feature, new story, enhancement, throw-in"; "huge, big, medium, small"; or some other set that fits your style). You may decide to measure rather than count, comparing items to each other. If so, be sure to set a consistent scale (e.g., story points using Fibonacci numbers, powers of two, or another unit or scale).

Practice Estimating

Ask multiple teams to estimate each other's backlogs using the chosen technique, and see if the ratings of the items are comparable. Discuss major differences to gain consensus on the meaning of the buckets or scale. Certain differences may arise because of a different understanding of the backlog items (for example, one rater might be more familiar with the project than another); that's a consequence of the practice exercise, so you can ignore these differences. But through the discussion, you can build consensus on the meaning of the buckets or measurement scale, and improve comparisons for different projects.

Once you can get consistent size estimates, you're on your way to use these to estimate project durations and costs. Then, you can compare projects directly and make informed decisions for the best chance for success.

References:

1. *Agile Estimation and Planning* by Mike Cohn
2. *Measures for Excellence: Reliable Software On Time, Within Budget* by Lawrence Putnam and Ware Myers