ThoughtWorks®

To Estimate or Not to Estimate, Is that the Question?

By Matt Philip (@mattphilip), October 2017

Like many practices, NoEstimates has run the gamut of early adopter curiosity, zealous adherence, thoughtful usage, polite debate, skeptical inquiry and, of course, noisome trolling. In this article, I'll attempt a soft-edged, non-confrontational explanation of how I have come to appreciatively view NoEstimates through my experience coaching delivery teams and teaching various groups at conferences and private workshops.

What's the Problem We're Trying to Solve with Estimates?

Nearly always, the question that we're trying to answer when we estimate something, whether it's a batch of work or any particular single piece of work, is "When will it be done?" This is a reasonable question, since it often relates to planning and scheduling of staff, sequencing of work, timing of marketing campaigns, the cost of projects and entry to the marketplace.

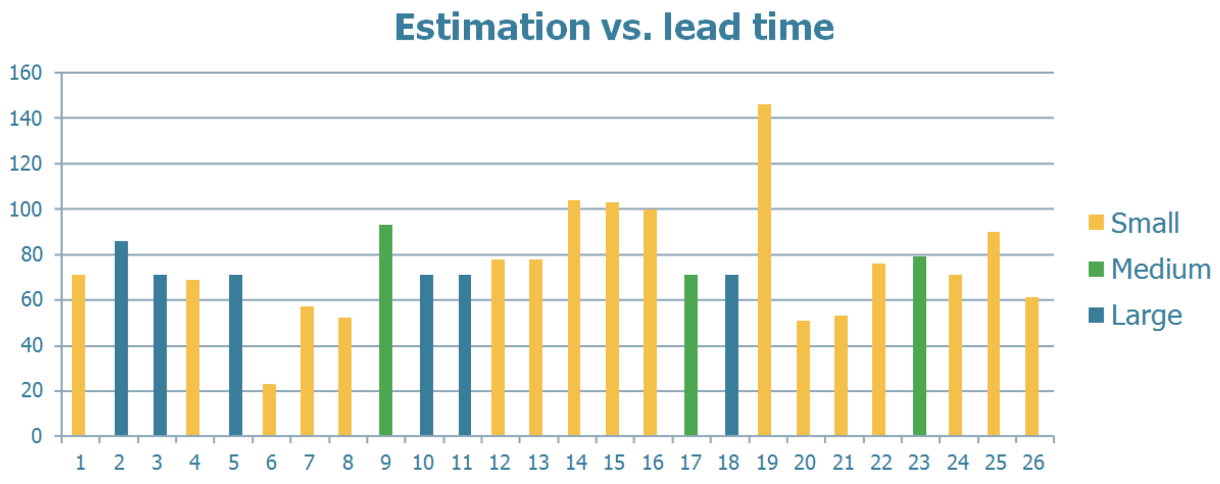
What's the Problem with the Solution?

So do our traditional ways of estimating work? When we ask some group or individual to estimate knowledge work — work that by its nature is complex and unique — the information about the work is very low. Although agile estimating practices (when done right) mitigate some of the pain of estimating — planning poker, for instance, is intended to reduce biases and time — it still requires time that we could spend on creating the software (did you ever consider that estimates are a form of comprehensive documentation?) and involves several human biases, such as recency bias. And we're still considering (and guessing at!) only a part of what accounts for time: effort, or essential complication — that is, the amount of time it takes to do the job itself.

Why do we base our estimates on effort? The implicit assumption, of course, is that we believe effort strongly correlates with time duration. But is that true?

The Myth of Correlation

Several people have studied the relationship between upfront estimates of size and the time it takes to deliver those items. (I refer to this simply as "delivery time," though people sometimes use terms like "cycle time," "lead time" or "time in process.") In his book *Real-World Kanban*, Mattias Skarin writes: "Is the initial sizing a good predictor for when you can get your stuff? In our case, the surprising truth was 'no.'"



— From *Real-World Kanban*

I, too, have conducted similar research with teams and found the same lack of correlation. One team even had a negative correlation between their initial estimates of feature sizes and the delivery times for those features! Think about that for a second: The "smaller" features actually took longer to deliver, meaning not only did the team completely waste their time estimating, they gave themselves (and their customer) misleading information. (In my workshops in which participants play a project simulation in which they're given crystal-ball-like perfect knowledge about effort required, teams still regularly have low correlations between effort and delivery time.)

What’s Going on?

Why is this? The reason is that the majority of time it takes to deliver software is spent waiting and in other waste. That might sound strange, but research shows that typical teams have a flow efficiency -- the percentage of value-added time to total time a work item takes to finish -- of only 5 to 15%! Troy Magennis sums it up well by saying that "Low process efficiency means that even if we nailed the effort estimates … we would be accurately predicting 5-15% of elapsed delivery time!” So our first lesson is: Effort is one of *many* sources of variation.

Sources of Variation

The other sources range from obvious to hard to see. Blockers and rework are examples of the obvious. But what about things like multitasking? Or what happens when you suddenly get a ticket to fix a production defect or a story “request” from a VP? That work usually takes precedence over other in-progress work, incurring “flow debt” for those other items. We call this “selection policy,” and if not made explicit it can create a particularly insidious drag on delivery time because it’s unseen. Teams with high or varying levels of work-in-progress find it especially hard to avoid flow debt. Which brings us to another source: work-in-progress, which we know from Little’s Law has a direct relationship with delivery time and throughput. Do you ever work outside your specialty in order to help the team finish a story? Along with WIP, your team’s collaboration policy is another key lever that Dimitar Badzharkiev found likely impacts delivery time as much as story size. Other sources:

- Technology/domain/product
- Team composition
- System dependencies
- Team dependencies
- Specialization
- Waiting for availability
- Rework
- Steps/handoffs in process
- Stages in team development (Tuckman)

The good news is that, while we have little control over the *essential* complication (aka effort) of our work, we do have some control over many of the other sources of variation. And since humans are not very good at estimating and likely to improve our skill in it only marginally, we should (as Strengths Finder tells us) spend less time trying to become better estimators and more of our time working to reduce sources of variation.

By the way, Magennis reminds us that “often system factors account for more of the elapsed delivery time than different story sizes,” which means that we don’t need to worry about all of our work items being the same “size” (a common myth about NoEstimates). Remember, after all, that upfront story size (guesses) have very little correlation with delivery times.

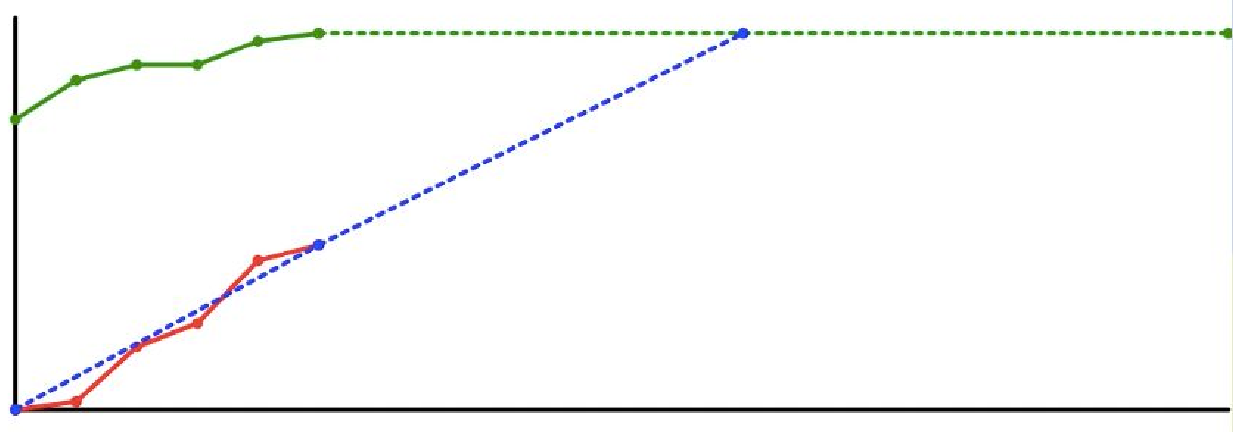
So then what?

If upfront estimates based primarily on effort are doomed, what are we to do? After admitting to ourselves — and our customers — that we can’t possibly answer very well the question about when something will be done without considering all sources of variation, we do happily have something that does: delivery-time data. Delivery time — the elapsed time from commitment (when we agree to start working on a story and no longer manage it as an option) to delivered (wherever you agree with your customer that that point is) encompasses all sources, obviating the need for us to apply unscientific “fudge factors” to “pad our estimates” with.

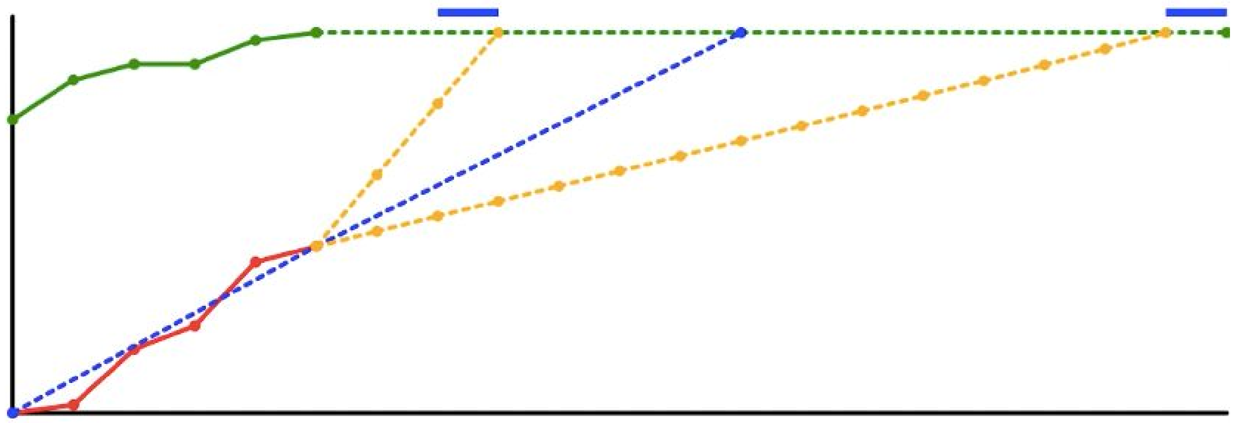
Deterministic vs. Probabilistic Thinking

Of course, using data isn’t foreign to agile planning methods. The venerable burnup chart uses throughput data (measured in either story points or count of stories) to project future delivery outcomes.

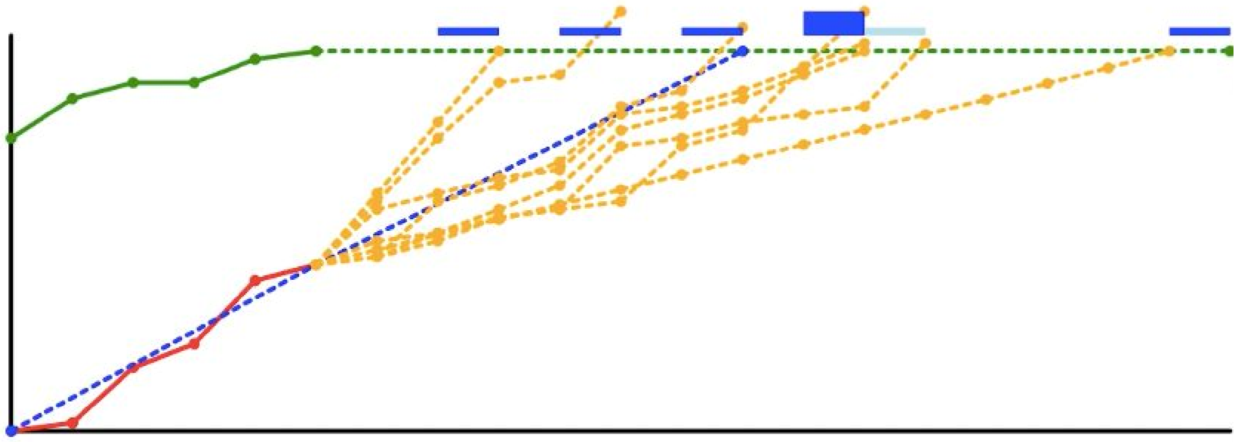
You’re familiar with the approach: After a few iterations, we have some data for a team’s throughput (plotted in red below). Using “yesterday’s weather” or even the entire history of data, we then create a straight-line average forecast (blue) that tells us some future date when we’ll complete the planned scope (green).



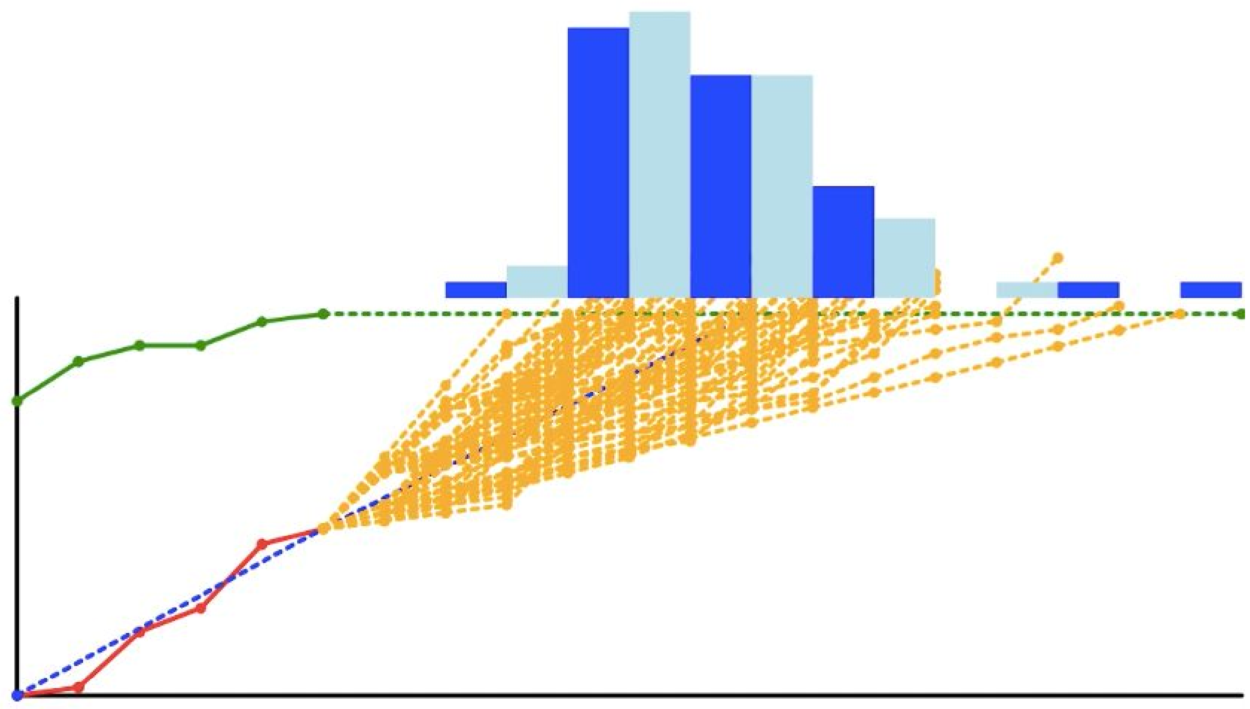
But the future isn’t deterministic in this way. Dan Vacanti, author of *When Will It Be Done?*, states what I call “Vacanti's Verity": "When making a forecast, you have to accept that there is *more than one possible outcome*." A straight-line forecast in our burnup chart has only one! So what do those multiple outcomes look like? Well, we occasionally acknowledge this with so-called optimistic and pessimistic lines, which we draw based on the best-case scenario — our future throughput matches our best past iterations — or worst-case — the future is comprised of the lowest-throughput iterations. Granted, those aren't likely, but they're *possible*, so we need to account for them, or we'd be lying to ourselves and our clients:



And of course, other outcomes — in-between the worst- and best-case scenarios — are possible. We typically don't draw those on a burnup, but if we did, they'd look something like this:



Depending on our past performance, some of those outcomes are likelier than others. With our data, we can start building a frequency chart (or histogram) of those outcomes, which create a distribution of all possible outcomes. Each bar represents the percentage chance or probability of the outcome occurring. For instance, the best-case scenario — the first little bar — may have only a 3% chance of occurring. The tallest might represent a 40% chance.



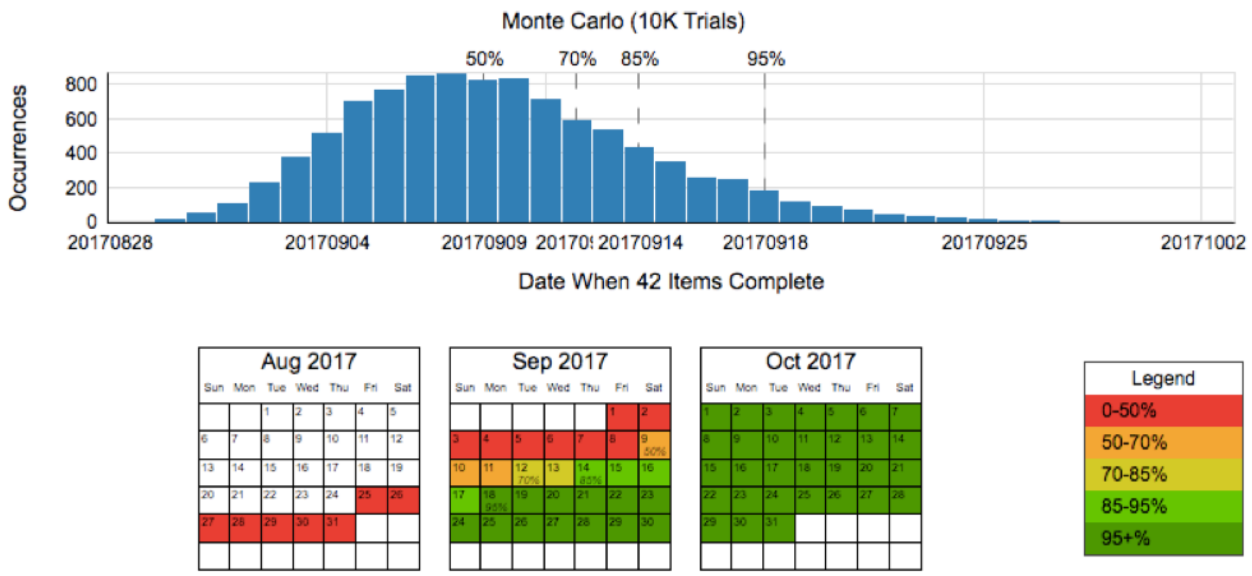
Now we've satisfied Vacanti's Verity: A forecast that includes both a *range* and a *probability* of that range occurring! This is the key difference between deterministic (one outcome) and probabilistic (multiple outcomes) thinking and is at the heart of NoEstimates.

(Special thanks to Larry Maccherone, who created this visualization at maccherone.com/lumenize.)

The Probabilistic Forecast

When we think probabilistically, we have real power — real as in the real thing, and not guesswork! — with which to plan and communicate with customers and others. Not only do we spend no time estimating, we can provide our customers with objective likely outcomes to make economic decisions with. We know at any given time the answer to the question "When will it be done?" by running a Monte Carlo simulation of future outcomes. This simulation (available in open source and pay options) basically uses our delivery-time data to generate all of those possible scenarios, from best to worst case and everything in between. Think of it as rolling two six-sided dice: occasionally you'll roll snake eyes, which you need to account for, but other combinations are more likely (do you know the most likely outcome? And how likely it is to occur?).

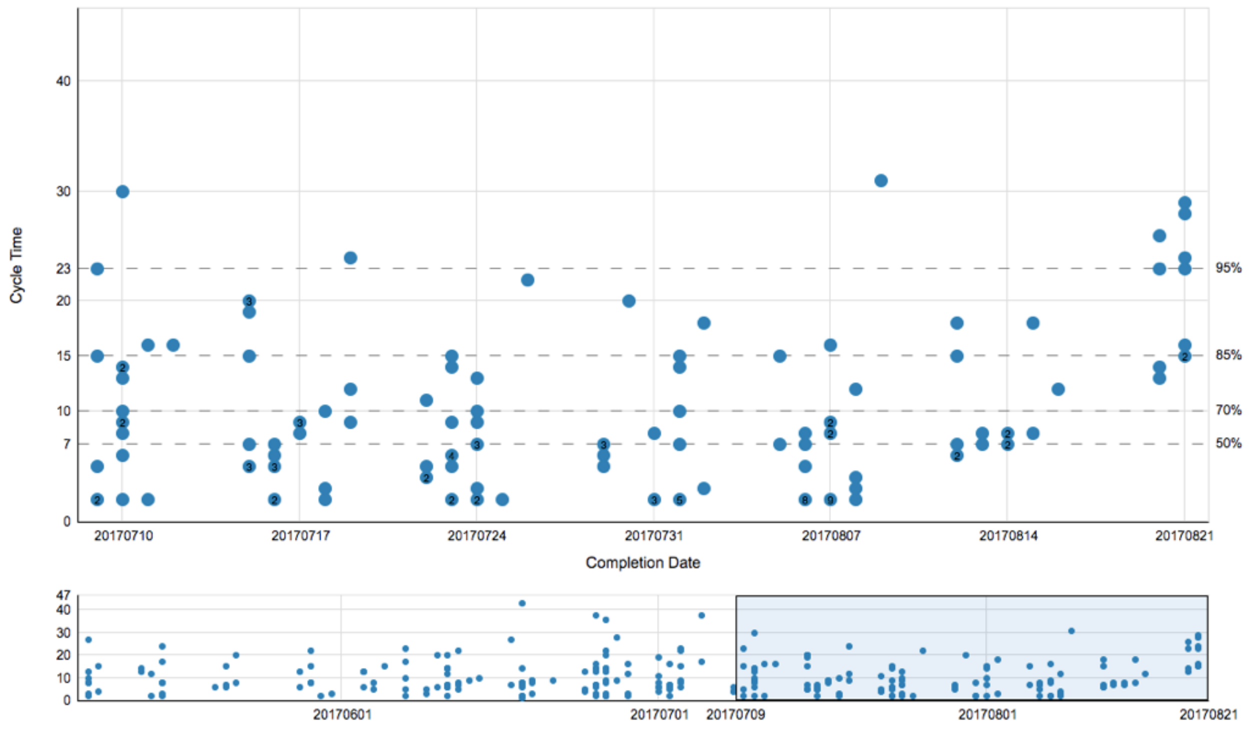
The result of a Monte Carlo simulation is a range of outcomes, with a percentage confidence at each. This allows us planning flexibility based on risk tolerance: For a very conservative, risk-averse need, we would use the 95th percentile, which tells us that 95% of the time, we expect to complete the given work (in the example below, 42 more stories) in 25 days or less. If we're waiting to launch a marketing campaign, we might choose this confidence interval (or perhaps 85%). But if we're heading toward a fixed-date event with high reward potential, say, Apple's next announcement of a new watch — it might be worth it to "roll the dice" at the 75th or even 50th percentile. The point is that the planning now becomes a business decision based on risk. And that is honest and reliable. Also, it reduces stress and harmful pressures that influence the delivery team to (consciously or unconsciously) estimate in ways that can negatively affect their behavior and/or a trustful relationship with the customer. Further, it helps avoid the problem of estimates tending to lock us into initial solutions and bias us against discarding them in favor of better ones.



— From *Kanbanize.com*

And you'll need surprisingly little data in order to forecast: Magennis suggests, and industry-leading tools (like Kanbanize.com) require, only about 10 to 12 data points to create a reasonable forecast.

Answering the question for any particular piece of work — "When will any new story be done?" — is similarly simple. We simply use our data to find our confidence intervals: For instance, we can graphically display this using a scatterplot chart showing delivery times for individual work items over time:



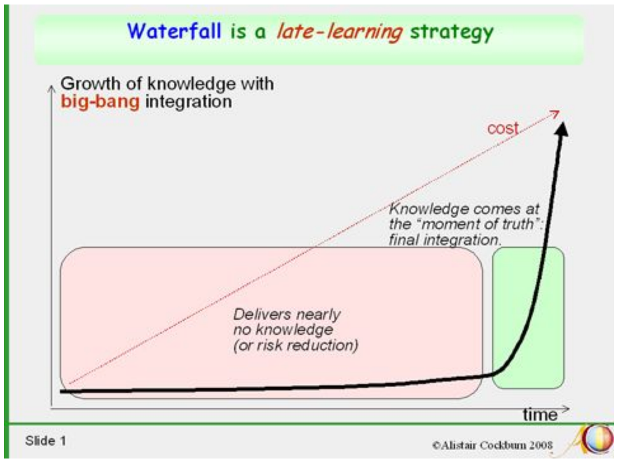
We then look at our confidence intervals — drawn at the points where 95% of our items are below the line, 85% of them, etc. — and make the confident claim that "any next story is 85% likely to finish in 15 days or less" (in the above example).

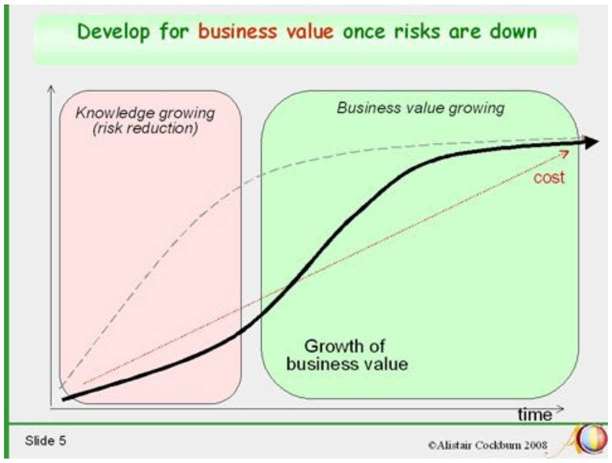
The Business and Better Questions to Ask

Again, asking the "when" question is often important. From a business perspective, it's often helpful to go one level above that, and not simply accept as *a priori* the need to know when.

If you're trying to engage in new work and/or with a new client, develop trust before trying to quote the work. Just like any good relationship, it's built on trust. Before getting married, you don't ask "How much will this marriage cost me?" Similarly, when we're trying to prefer the manifesto value of "customer collaboration" over "contract negotiation," we need to establish trust first. That is, "Is this a partnership that can collaboratively work together and grow, for mutual benefit?"

Given that the beginning of a project -- or even after some initial discovery -- is the time when we know the least about what is almost certainly a complex system, it's a fool's errand to try to "estimate the work." Rather, try to find a low-risk way to engage: Can we work for three months at a certain run rate per month and work toward an MVP? This initial "fixed-price, variable scope" approach reduces risk and affords flexibility as we progressively learn about the work together. After all, the value-proposition curve of working iteratively and incrementally is what business agility is all about:


Waterfall is a late-learning strategy
Growth of knowledge with big-bang integration
cost
Knowledge comes at the "moment of truth": final integration.
Delivers nearly no knowledge (or risk reduction)
time
Slide 1
©Alistair Cockburn 2008



-- Alistair Cockburn

Some other questions you might consider asking:

- In what context would estimates bring value, and what are we willing to do about it when they don't? – Woody Zuill
- How much time do we want to invest in this? – Matt Wynne
- What can you do to maximize value and reduce risk in planning and delivery? – Vasco Duarte
- Can we build a minimal set of functionality and then learn what else we must build?
- Would we/you not invest in this work? If not, at what order-of-magnitude estimate would we/you take that action?
- What actions would be different based on an estimate?

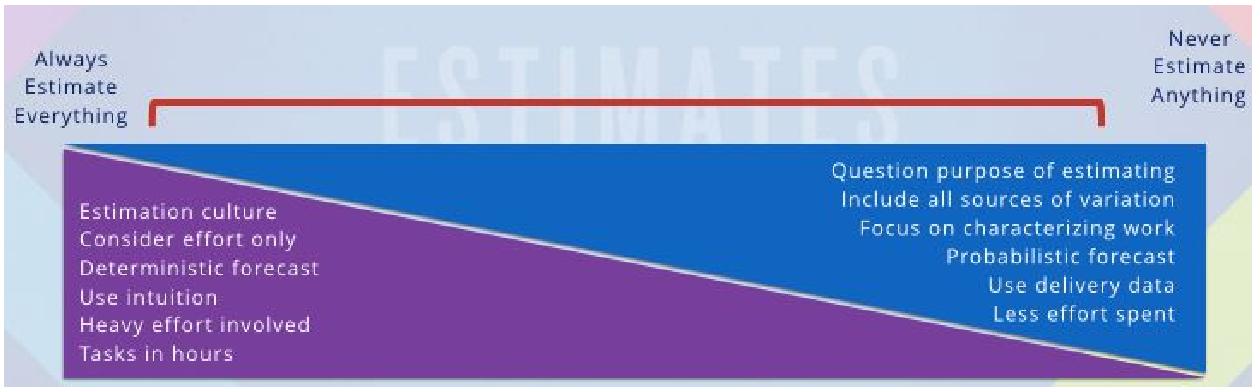
Many of these questions obviate the constraints of the traditional "iron triangle" of project management and shift the conversation toward outcomes and business value and away from cost, schedule and scope discussions, which focus our work on outputs. Douglas Hubbard has this sage advice, which can be a conversation point for clients and consultants early in a sales process:

> "Even in projects with very uncertain development costs, we haven't found that those costs have a significant information value for the investment decision… The single most important unknown is whether the project will be canceled. …The next most important variable is utilization of the system, including how quickly the system rolls out and whether some people will use it at all."

The Spectrum of Estimating

In reality, the question isn't a binary one between whether or not to estimate. Rather, it's a matter of finding the right location on a spectrum between those polar ends in which we're spending progressively more effort on things that have value -- working software -- while using more accurate and reliable means of forecasting. As Dimitar Bakardzhiev likes to say, NoEstimates is simply “estimating with minimal effort, and not basing our estimates on effort.”

The key is to thoughtfully “uncover better ways” to do it (as the Agile Manifesto instructs) while respecting the people in the delivery system.



How to Start

If these ideas seem helpful to you, here’s how you can start moving across that spectrum.

If you’re in the middle of a project:

1. Continue having value-adding discussions about the work to analyze it and break it into thin vertical slices.
2. Start tracking two pieces of data for each work item: commit date and delivery date.
3. After accumulating 10 data points, run a Monte Carlo simulation to provide a probabilistic forecast.
4. Begin answering the “When” question using data, either at the project level or individual work-item level.
5. See whether the probabilistic forecasts match what actually occurs.
6. If the forecasts are helpful, wean your team away from upfront estimating.
7. Shorten the tail of delivery-time distribution by focusing improvement efforts on reducing the sources of variation.

If you’re getting ready to start a project:

1. Ask the “better questions” above about the use and importance of estimates.
2. Use reference-class data (similar delivery times from other projects in similar tech stacks with similar teams) to provide an initial probabilistic forecast.
3. As soon as you accumulate 10 data points, run a Monte Carlo simulation to provide an updated probabilistic forecast.

If you’re trying to engage in new work and/or with a new client:

1. Develop trust before trying to quote the work.
2. Find a low-risk way to engage without giving an estimate. If necessary, use reference-class data (similar delivery times from other projects in similar tech stacks with similar teams) to provide an initial probabilistic forecast.
3. Ask "The Business and Better Questions to Ask" questions above.

In all cases, make short and long term forecasts with the understanding that shorter forecasts will be more accurate than longer ones. And continually reforecast when you get more information.

My NoEstimates Manifesto

Finally, in the spirit of the Agile Manifesto, here's my NoEstimates Manifesto.

... We have come to value:

- Probabilistic over Deterministic
- Delivery time over Development time
- MVP scope over Full scope
- Data over Intuition*
- Reducing sources of variation over Improving estimating

That is, while there is value in the items on the right, we value the items on the left more.

**Neil Killick uses "empiricism over guesswork"*

References and Further Exploration

Reading, Etc.

- noestimatesbook.com (Vasco Duarte)
- infoq.com/articles/noestimates-monte-carlo (Dimitar Bakardzhiev)
- priceonomics.com/why-are-projects-always-behind-schedule
- http://scrumandkanban.co.uk/estimation-meets-cynefin/
- ronjeffries.com
- Lizkeogh.com
- https://neilkillick.wordpress.com/
- http://zuill.us/WoodyZuill/
- mattphilip.wordpress.com/noestimates-game
- When Will It Be Done? (Dan Vacanti)
- http://alistair.cockburn.us/Design+as+Knowledge+Acquisition

Forecasting tools (including Monte Carlo simulations)

- focusedobjective.com (Troy Magennis)
- actionableagile.com (Dan Vacanti)
- Kanbanize.com