(/way_of_the_web_tester)

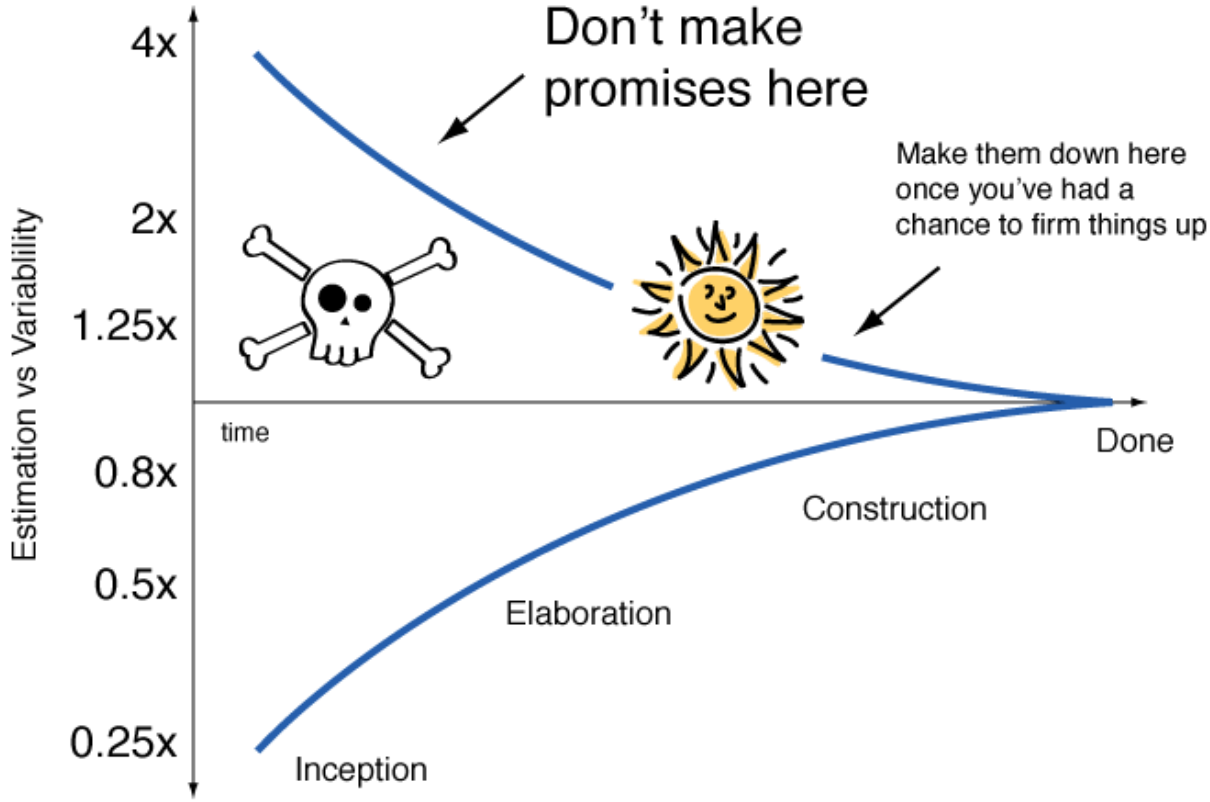
)out)

Getting Started (/) Videos (/videos) Courses (/bootcamp)

Learn More

Cone of Uncertainty

The Cone of Uncertainty (http://www.construx.com/Thought_Leadership/Books/The_Cone_of_Uncertainty/), described by Steve McConnel, shows what any experienced software professional knows. Which is at the beginning of any project we don't know exactly how long a project is going to take.



The reasons for this are many. No two ever projects have:

1. The same requirements.
2. The same people.
3. The same business context.
4. The same technology.
5. The same priorities & constraints.

Each is unique. Every line of code is hand crafted. And knowledge work involving smart creative people doesn't lend itself to precision the way ditch digging does.

Yet precision is what is asked for. Sponsors want to know exactly when the project will be done, and how much it will cost.

Dealing with this conundrum is almost as old as time itself. Here are a few ways teams and companies are deal with this uncertainty.

Dealing with the cone

Pad the estimate

After feeling the sting of underestimating, one common reaction is to double or triple the estimate the next time round. This definitely lowers the upfront risk, but padding the numbers is harder than it sounds.

Give too big a number, and sponsors will balk and not approve your project. Give too low a number and you risk running out of money. This gets doubly dicey when you are bidding on fixed bid contracts where there is even more pressure to keep the numbers down.

Most projects add some kind of padding onto the final numbers to give themselves sufficient wiggle room. Another things teams can do is compare this project with others.

Size the project relatively

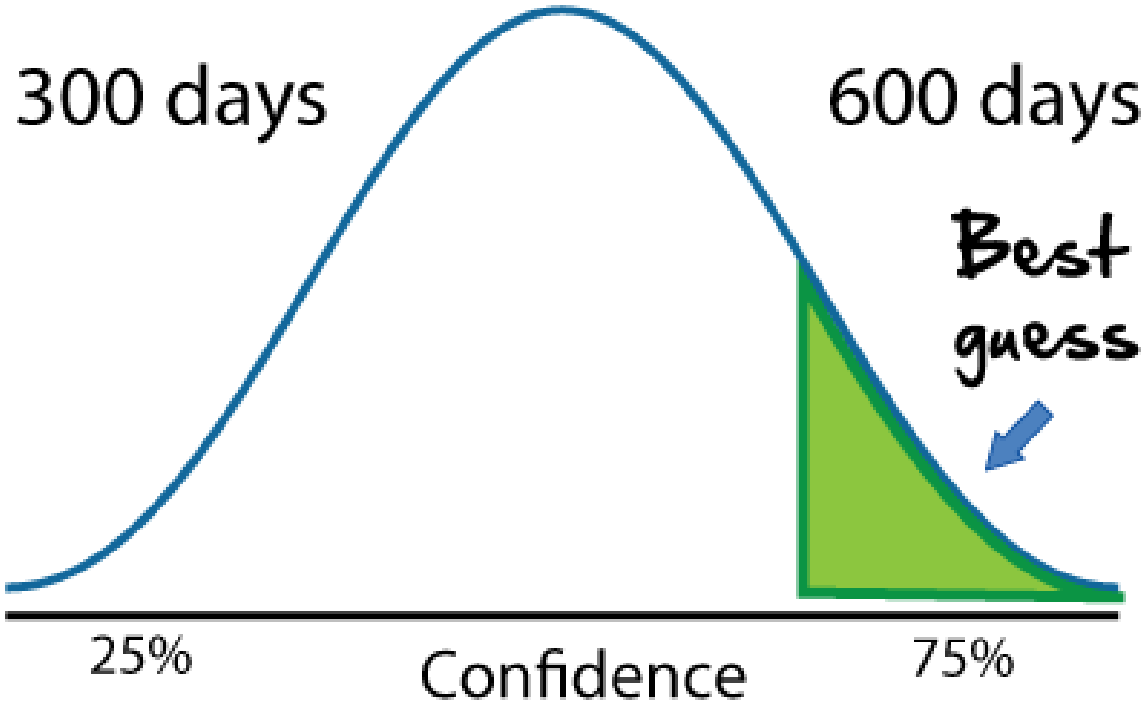
Humans are really good at sizing things relatively. We can't tell you precisely how big a rock is. But we can tell you how big it is compared to something else. We can use this when sizing projects too.

Be upfront and honest

Agile's default mode of operations is transparency and visibility. So it should be no surprise that the Agile way of dealing with the cone is to be upfront and honest.

> Look. We don’t know how long this is going to take. This is our best guess. But if you give us a couple iterations, we can build something, measure how long that takes, and then tell give you a much better sense of how big this thing is.

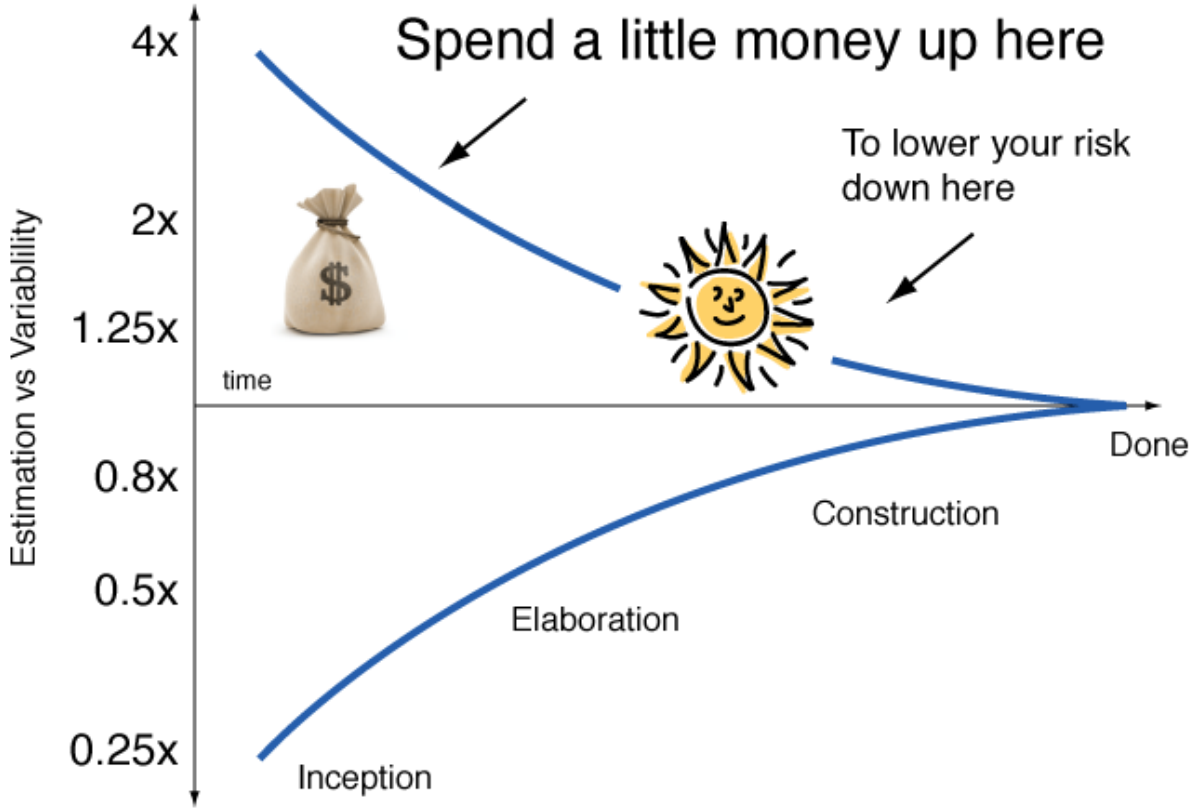
Another approach some Agile teams will take to help communicate the uncertainty is to present the **estimates as a range**.



This has the advantage of visibly showing sponsors the uncertainty that comes with the project, and then letting sponsors decide how much risk they can afford to take on.

Fund incrementally

Another approach that makes a lot of sense, but unfortunately I don’t see applied often enough, is incremental funding.



With incremental funding you don't ask for the whole bag of money upfront. Only enough to spike through enough of the work, to report back a better number on how long it is going to take.

It's not foolproof. You can still run into trouble later on. But by giving teams $30-50K, letting them build something and seeing how long that takes, can go a long ways to reducing the variance in that upfront number.

You can read more about this way of planning and the advantages it brings in Beyond Budgeting (http://www.amazon.com/gp/product/1578518660/ref=as_li_tf_tl?ie=UTF8&camp=1789&creative=9325&creativeASIN=1578518660&linkCode=as2&20) by Jeremy Hope.

The Root Cause

We live in a world of annual budgets. And unfortunately this drives a lot of the tension and drama in our workplace. In some ways annual budgets are natural (we live and track time in yearly cycles) but in others they are terrible for planning as a year is a long time in the marketplace.

If you find yourself getting tripped up by the cone of uncertainty, just remember the whole point of software estimation is to determine whether the project is even possible.

Or as Steve McConnell says:

> The primary purpose of software estimation is not to predict a project's outcome; it is to determine whether a project's targets are realistic enough to allow the project to be controlled to meet them.

—Steve McConnell, Software Estimation: Demystifying the Black Art (http://www.amazon.com/gp/product/B00B9ZD20K/ref=as_li_tf_tl?ie=UTF8&camp=1789&creative=9325&creativeASIN=B00B9ZD20K&linkCode=as2&20)

Back → (http://www.agilenutshell.com/user_stories)

Unit Testing (/unit_testing)

Refactoring (/refactoring)

Continuous Integration (/continuous_integration)

Test Driven Development (/test_driven_development)

CONCEPTS

Burndown Charts (/burndown)

Cone of Uncertainty (/cone_of_uncertainty)

Management by Miracle (/management_by_miracle)

Three Simple Truths (/three_simple_truths)

XPISMS

Bill of Rights (/bill_of_rights)

YAGNI (/yagni)

Yesterday's Weather (/yesterdays_weather)

The Simplest Thing (/simplest_thing)

Testing (/could_possibly_break)

Production Readiness (/production_readiness)