Decoding estimates

The what, why and how of estimation

simply brilliant thinkers
making software brilliantly simple

The importance of estimates

Some reasons to estimate

- Budget setting
- Planning releases
- Stakeholder management
- Go/No-go decisions

What’s the best way to find out how long something will take?

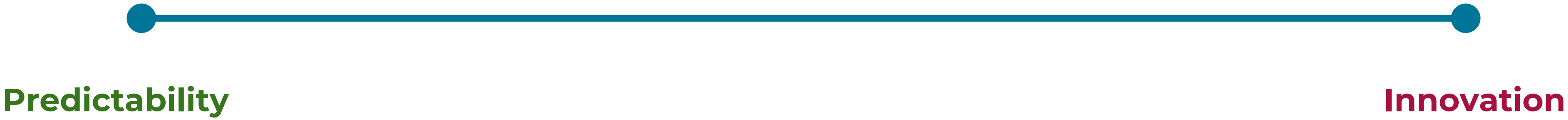
The only way to know for sure how long something will take, is to do it, and measure how long it took.

Everything else involves compromise.

What’s the *next* best way?

Do the same thing again and again.

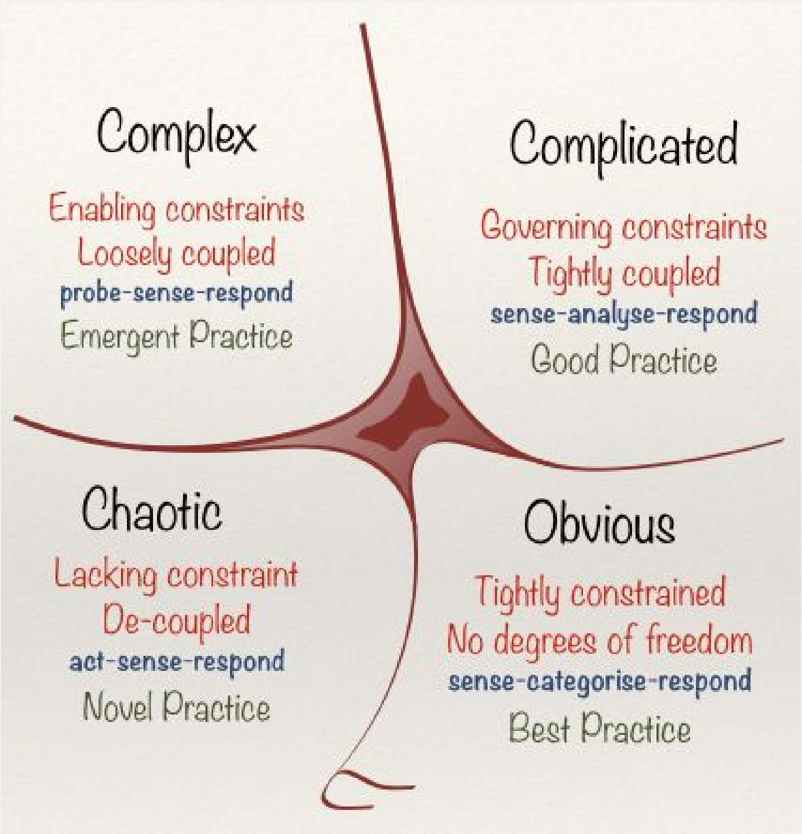
Predictability vs Innovation



Predictability

Innovation

Konstantin Kudryashov
http://stakeholderwhisperer.com/posts/2016/1/innovation-slider



Dave Snowden
https://commons.wikimedia.org/w/index.php?curid=33783436

What actually *is* an estimate?

> ***“A good estimate is an estimate that provides a clear enough view of the project reality to allow the project owner to make good decisions about how to control the project to hit its targets.”***

Steve McConnell
“Software Estimation - Demystifying the Black Art”

What *we* mean by an “estimate”

✓ A characterisation of the work to be done

✓ An estimate *range*

✓ *Your confidence in that estimate*

✓ A list of factors that could affect either the estimate or your confidence level

***Not* a raw number!**

Estimate examples

“This is a minor update to an existing project, mostly small features and tweaks.

We estimate 12 to 15 days for a single dev who was involved in the initial build, with a high degree of confidence.

If we have to give the work to someone new to the project the estimate could double as there is a lot of context to absorb.”

“This is a major new feature using technology we've got very little experience with.

It's between 4 and 6 sprints for a team that are already familiar with the client's previous projects.

Most of the work is actually fairly well understood, but a critical part involves the new tech.

We have a high degree of confidence that most of the work could be done in up to 3 sprints. We have a low degree of confidence that the novel parts could be done in 1 to 3 sprints. We can't be more accurate with our current understanding of the tech.

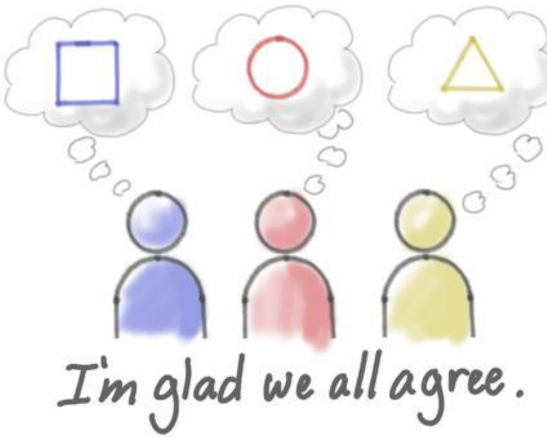
We could improve our confidence by spending a targeted 3 days doing a spike with the new tech.”

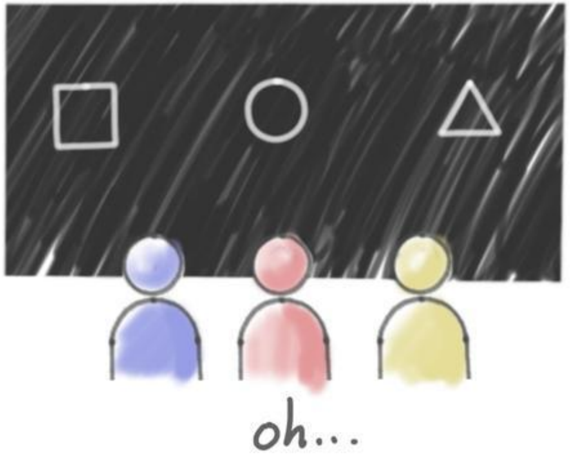
The challenge?

Predicting the future is hard!

The planning fallacy

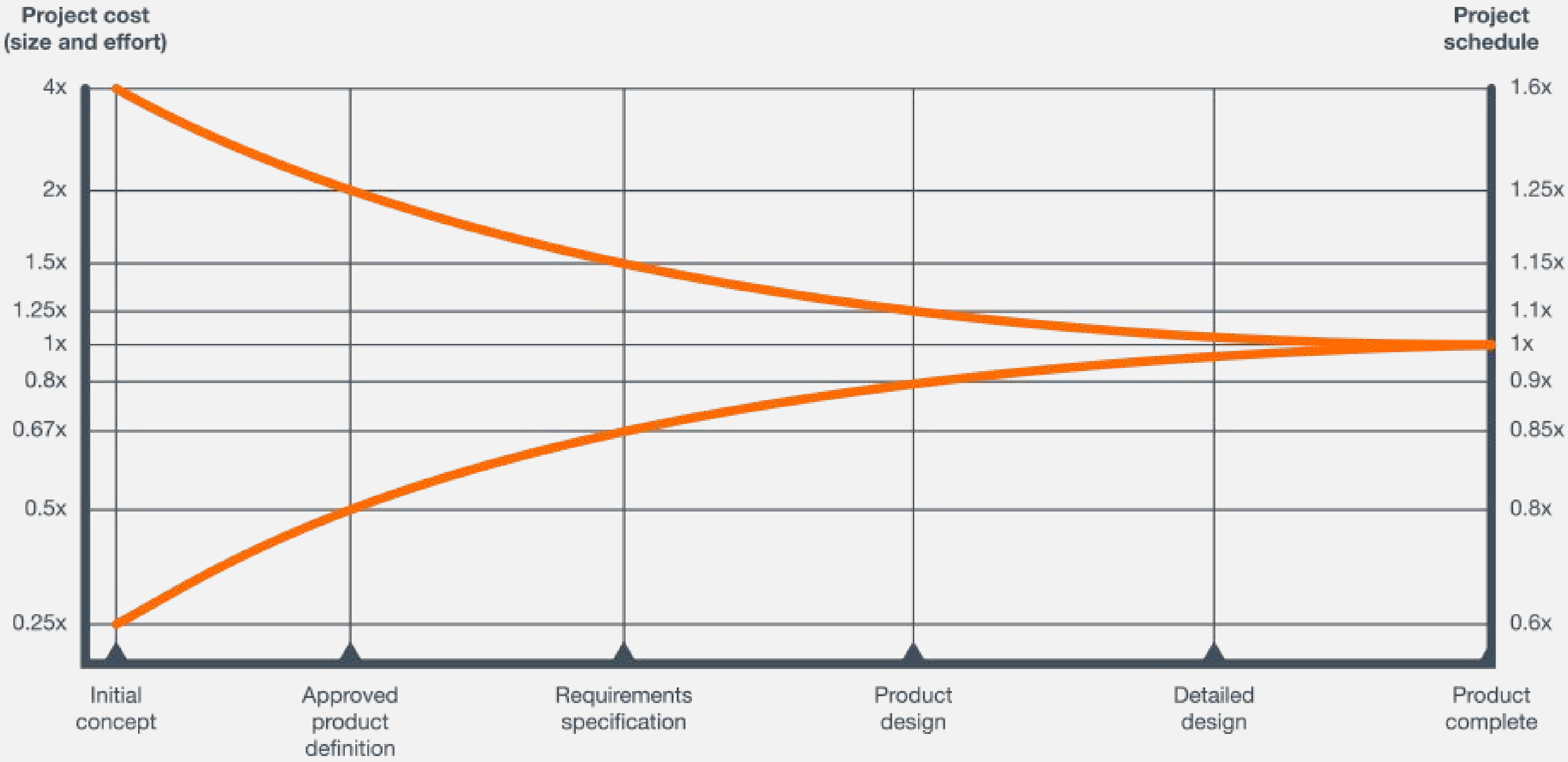
Specification problem/Heisenberg requirements





Jeff Patton
“User Story Mapping: Discover the Whole Story, Build the Right Product”

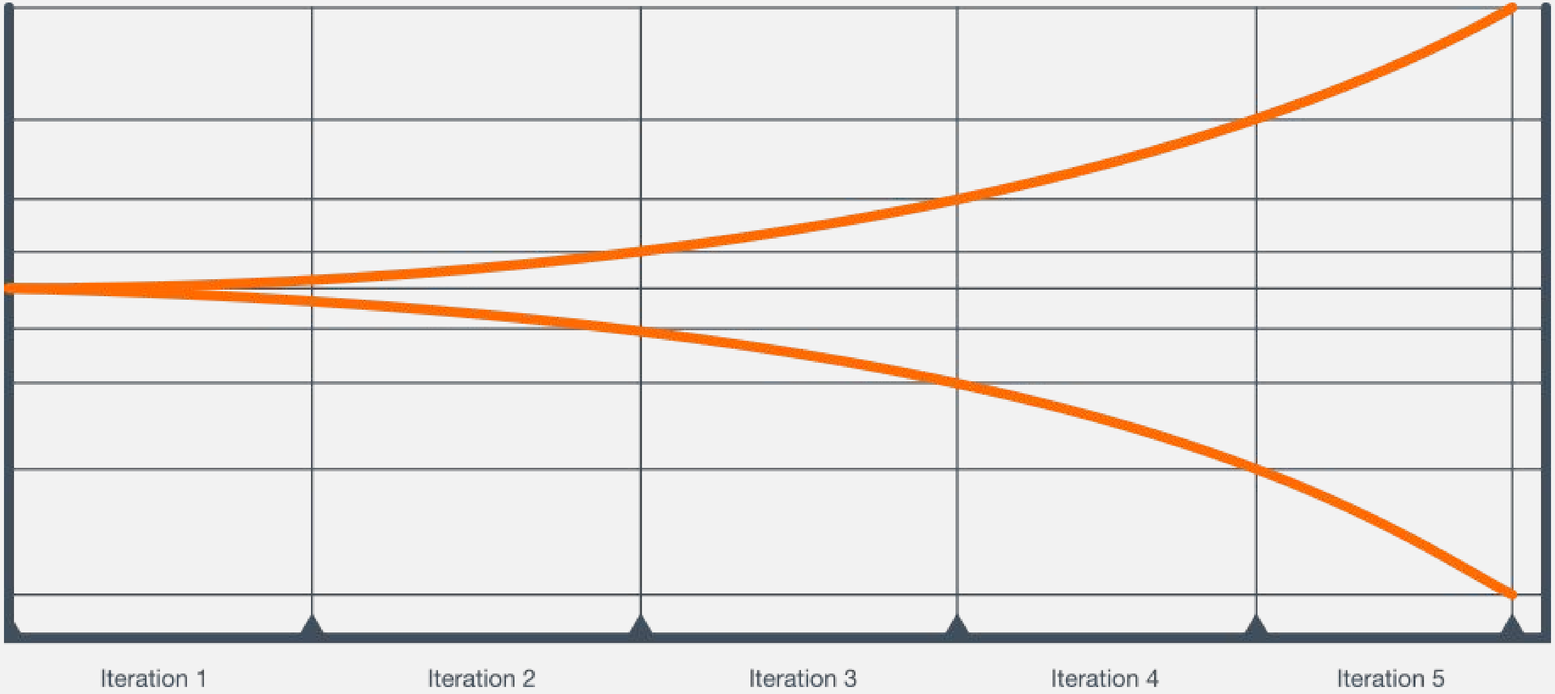
The cone of uncertainty



Steve McConnell
“Software Estimation - Demystifying the Black Art”

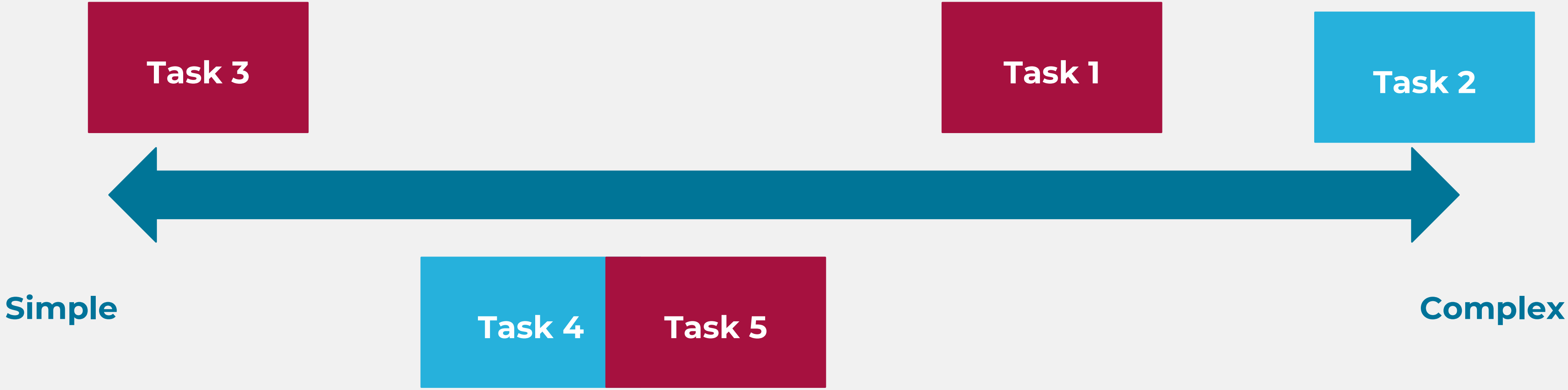
Mitigating uncertainty

Flip that cone around!



Compare with historical data

Buckets/Affinity estimation



Task 3

Task 1

Task 2

Task 4

Task 5

Simple

Complex

Bracketing to estimate

- One day?
- One year?
- One week?
- Six months?
- One month?
- Three months?

Key takeaways

- ✓ Some things are more predictable than others.
- ✓ You can’t get rid of uncertainty completely.
- ✓ But… you can work with it, rather than ignoring it.
- ✓ A broad estimate isn’t necessarily a bad estimate.
- ✓ And you *can* learn to produce better estimates.

Thank you!

simply brilliant thinkers

making software brilliantly simple