Crash course into agile thinking

Ben Brandwood

Winchester, Jan 2020


jira

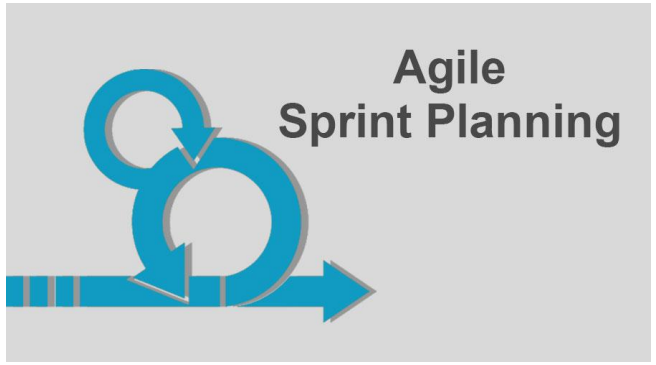

Basecamp


Scrum.org


Trello


看板
KANBAN


Agile
Alliance


Agile
Sprint Planning


Jira
Basecamp
Trello
Scrum.org
Agile Alliance
Agile
Sprint Planning
看板
KANBAN

None of these tools will make you Agile. Just the same as having a cement mixer makes you a builder, or having a ruler makes you an architect.

So what is Agile?

The Agile Manifesto

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

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

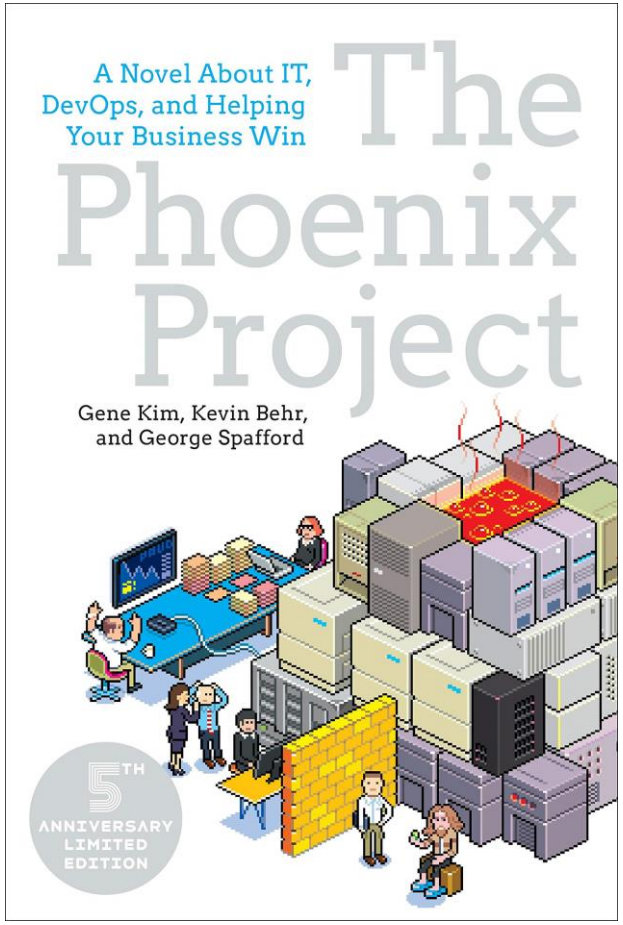
https://agilemanifesto.org - 2001

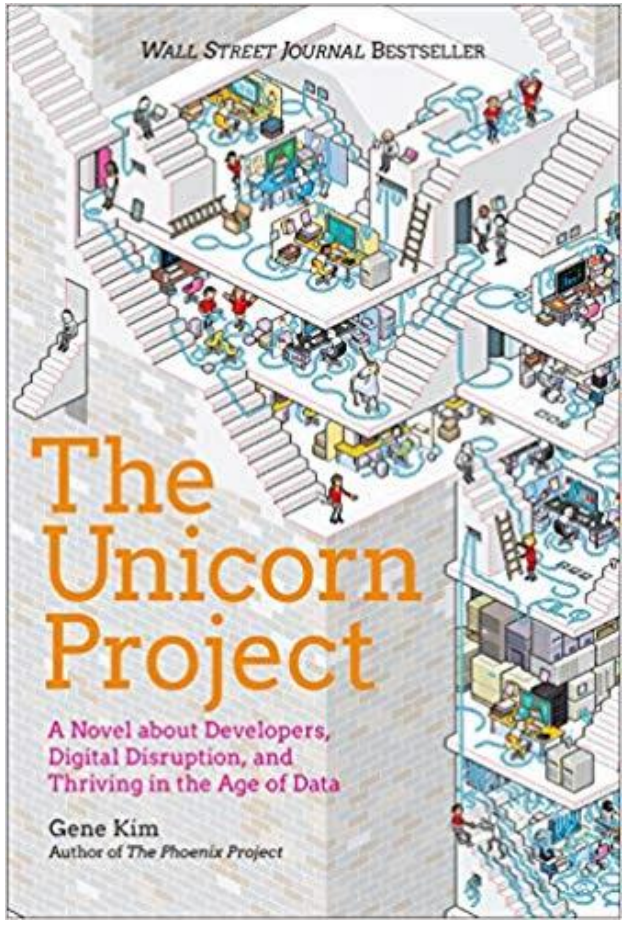
The Principles

- *We follow these principles:* Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- ***Working software is the primary measure of progress.***
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity--the art of maximizing the amount of work not done--is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

Agility in Practice

Read these two books by Gene Kim. It codifies agile thinking and approaches to real-world IT organisations and challenges. They are in the form of novels.





The “Five Ideals” - From *The Unicorn Project*

The First Ideal: *Locality and Simplicity*

- Reduce dependency links to increase autonomy and flexibility.

The Second Ideal: *Focus, Flow and Joy*

- Is our work marked by boredom and waiting for other people to get things done on our behalf?

The Third Ideal: *Improvement of Daily Work*

- From the Toyota Andon Cord principle, also used by Microsoft “we must raise improvement of daily work over daily work itself”

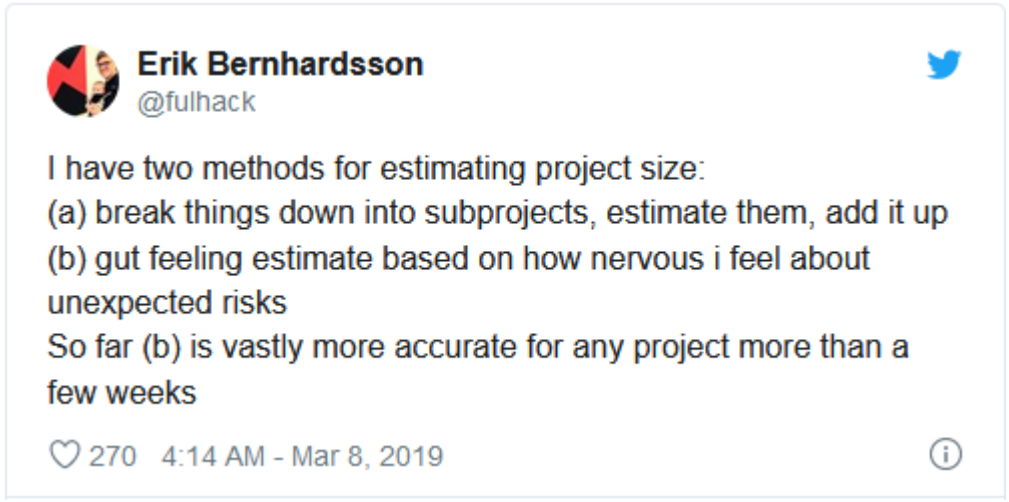
The Fourth Ideal: *Psychological Safety*

- Safety to talk about problems, foster acceptable risk taking. Honesty requires the absence of fear.

The Fifth Ideal: *Customer Focus*

- Where we ruthlessly question whether something actually matters to our customers, as in, are they willing to pay us for it or is it only of value to our functional silo.

The dead hand of planning



Summary

If my model is right (a big *if*) then here's what we can learn:

- People estimate the *median* completion time well, but not the mean.
- The mean turns out to be substantially worse than the median, due to the distribution being skewed (log-normally).
- *When you add up the estimates for n tasks, things get even worse.*
- Tasks with the most uncertainty (rather the biggest size) can often dominate the mean time it takes to complete all tasks.
- The mean time to complete a task we know nothing about is actually *infinite*.

https://erikbern.com/2019/04/15/why-software-projects-take-longer-than-you-think-a-statistical-model.html

The dead hand of planning

"When you add up the estimates for n tasks, things get even worse."

Wait up! Isn't that what we've been doing in agile? Writing user stories and tasks, estimating it and then adding all those up to say when things are going to be delivered?

Yes, and that's why estimates are rarely within 200% of actuals.

So what do we do without planning?

Coming back to the Agile Manifesto:

- ***Customer collaboration*** over contract negotiation
- Business people and developers must work together daily throughout the project
- Working software is the primary measure of success.

A delivery plan is ultimately a crude form of contract negotiation between the customer and IT. Instead of that, can we move the needle so that we have a conversation rather than a negotiation?

We aren’t building a house

In the data space we sit in a privileged position. We aren’t trying to build a bridge, or a house, or a rocket.

The tools at our disposal are small, discrete and modular. It’s Lego rather than a pile of cement. It’s quick to put together, quick to pull apart and easy to modify.

Working closely with our customer, we can put the first bricks down (“working software is the primary measure of success”) in days rather than weeks. We then ask the very simple question

“Where do you want to go next”

But what about estimation?

We all estimate all day every day, automatically.

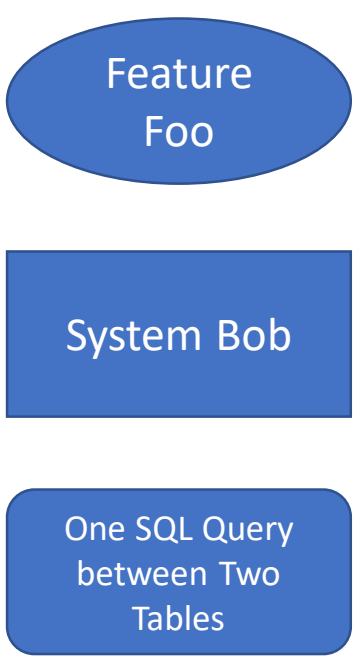
How often have we been asked?

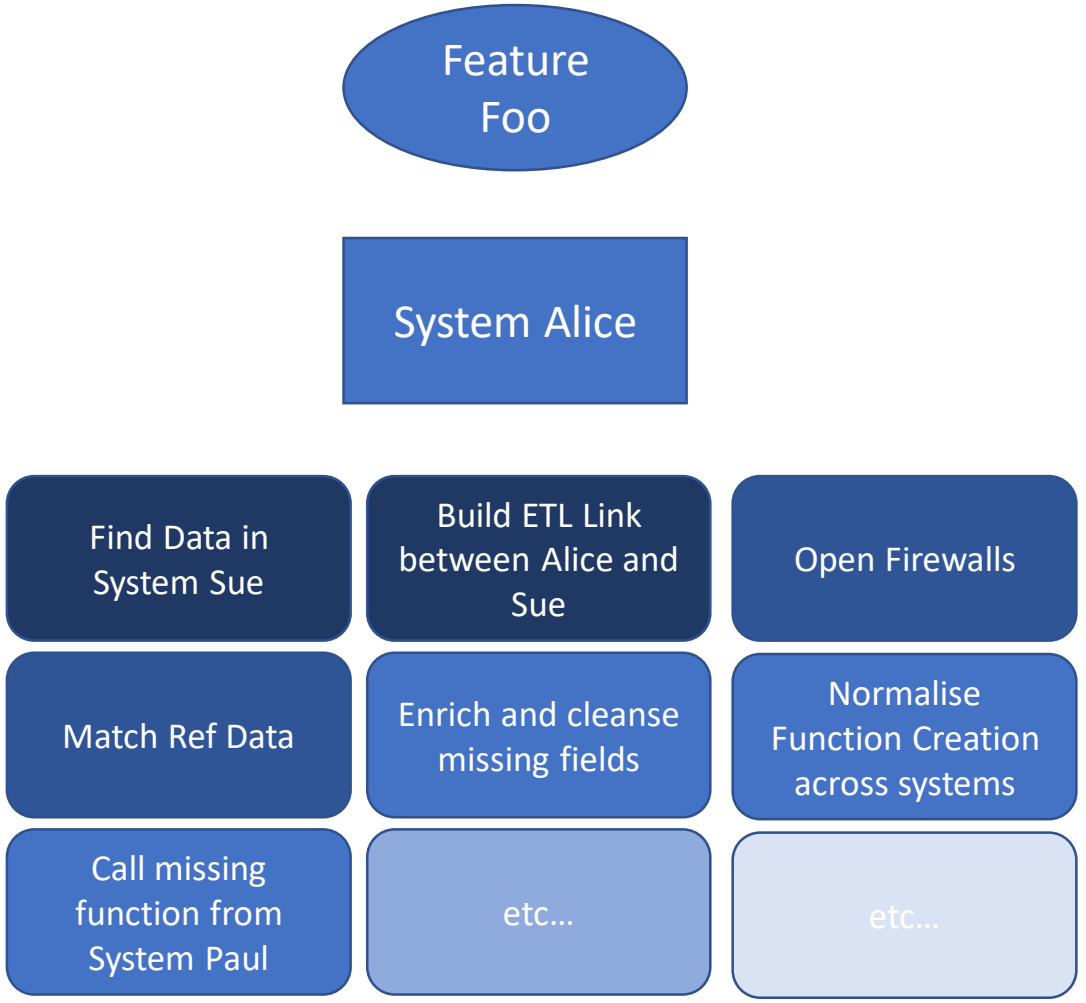
“Can you implement feature Foo in System Bob and System Alice?”

And while it’s the same feature, you know that in Bob it will take a day to do, but in Alice it would take months (if not longer!)

How do we know this?

Task Clouds





Assuming there's some knowledge of the systems we automatically construct task clouds with underlying tasks.

Without "formal" estimating we can see that for a functionally similar feature, one implementation is going to be "harder"

How does this help us?

People ***hate*** surprises! (which is why people want plans).

It’s a category error to assume the only way of expressing complexity is by placing dates on them.

Stakeholders can grasp complexity when presented with simple task clouds and that then factors in to their value judgements.