Deliberate Discovery

Dan North – DRW Trading

Imagine you got to redo your project...

...completely from scratch

- with the same objective
- with the same people
- with the same constraints

...except

- you knew then what you know now
- you had 20/20 *fore*sight

Ignorance is the constraint

Theory of Constraints

There is currently a constraint

Any inventory behind the constraint is waste

Any time not addressing the constraint is waste

But you don't know where the constraint is!

Ignorance is multivariate

What kinds of ignorance are slowing you down?

You are ignorant (to some degree) about...

- your domain
- the nature of the problem
- your incumbent technologies
- other possible technologies
- organisational constraints
- relationships with stakeholders

...and you might not even know!

Ignorance reduces in steps

oh.

Oh!

Crap.

So can we influence the reduction of ignorance?

Can you prepare for the *oh crap* moments?

This time we'll know better

This time it'll be different

This time we'll come in on time

This time will be exactly like the other times!

Deliberate discovery

Assume you are *always* operating in ignorance

Assume *some specific axis* of ignorance is your current constraint

Improve throughput by actively addressing ignorance

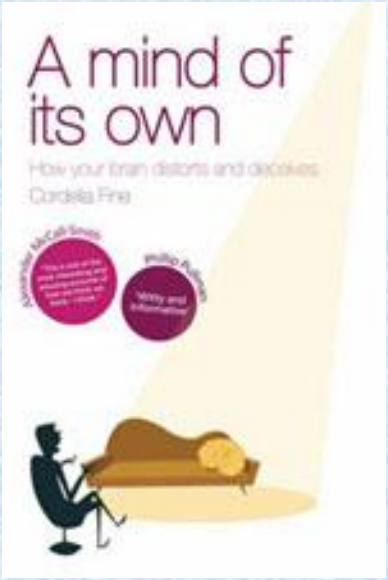
Second-order ignorance is a given

We are wired to resist this

We suffer with attribution bias

We suffer with confirmation bias

...but not as badly as everyone else!



Remember how the project started?

You got everyone in a room

You decomposed the problem into stories
- and more stories
- and more stories

You estimated the stories
- and estimated
- and estimated

Was that really the best use of your time?

So how can we apply this?

Software has a half-life

Shorter half-life means less 2OI

- less opportunity to not know what I don't know

So why not rewrite rather than redevelop?

- with multiple overlapping *implementations*

Deliberate discovery in planning

Plan for at least *some* unexpected bad things

Try natural planning (GTD)

1. Purpose
2. Mission/vision/goals
3. Brainstorm
4. Organise
5. Next actions

Figure out your axes of ignorance. Then do it again.

Beware the perils of fractal estimation

Deliberate discovery in analysis

Don't fear "analysis paralysis"

- as long as it's reducing *relevant* ignorance

Ethnography

- What are your stakeholders caring about?
- What *should* they be caring about?

Play it forward

- Who are you trying to reach?
- What do you want their experience to be?

Understanding the domain is a whole team activity

Deliberate discovery in programming

Spike-and-stabilise

- learn through evolving the “spike”
- choose to stabilise later – deferred, test-driven testing

Design for the second case

- you *might* gonna need it!

Indirect discovery

Travel in pairs

- optimise for learning: delivery will take care of itself

Deliberate discovery in testing

Exploratory testing

Randomised testing

Not just running the same old automated tests!

Good testers already know this

Deliberate discovery in (dev)ops

Get into production early

- get *anything* into production!

Design for monitorability

- push diagnostics rather than pulling an autopsy
- make it easy to listen

Design for discoverability

- what went wrong?
- what's *going* wrong – right *now*?

Summary

You don’t know what you don’t know

That ignorance is killing your throughput

Sometimes you can’t know what you don’t know

For everything else, there’s Deliberate Discovery

Thank you

dnorth@drw.com

http://dannorth.net

@tastapod