Thanks for coming everyone.

Customary to start with an introduction

Me: Cennydd *(pronounced Ken-iTH)*

User Experience Designer at Clearleft
http://icanhaz.com/agiledesign

So hello! I'm one of the Clearlefties. Live in Brighton. Me on the pier (not my yacht). I'm Welsh. Ken-ITH.

Been at CL a year, working on WWF, Gumtree, JustGiving, startups.

Before: in-house, government IA, Lead UX at uSwitch / UpMyStreet

Qualified PM too, albeit in Prince2.

Run Agile, waterfall for many years innie and outtie. Still do. At CL, UX = project lead, many PM responsibilities.

Wrote an ALA article

Running times.

Backchannel: please do use. Going to be interactive, so feel free to ask questions, comments etc in person or on backchannel if shy etc :)

And there's one more thing.

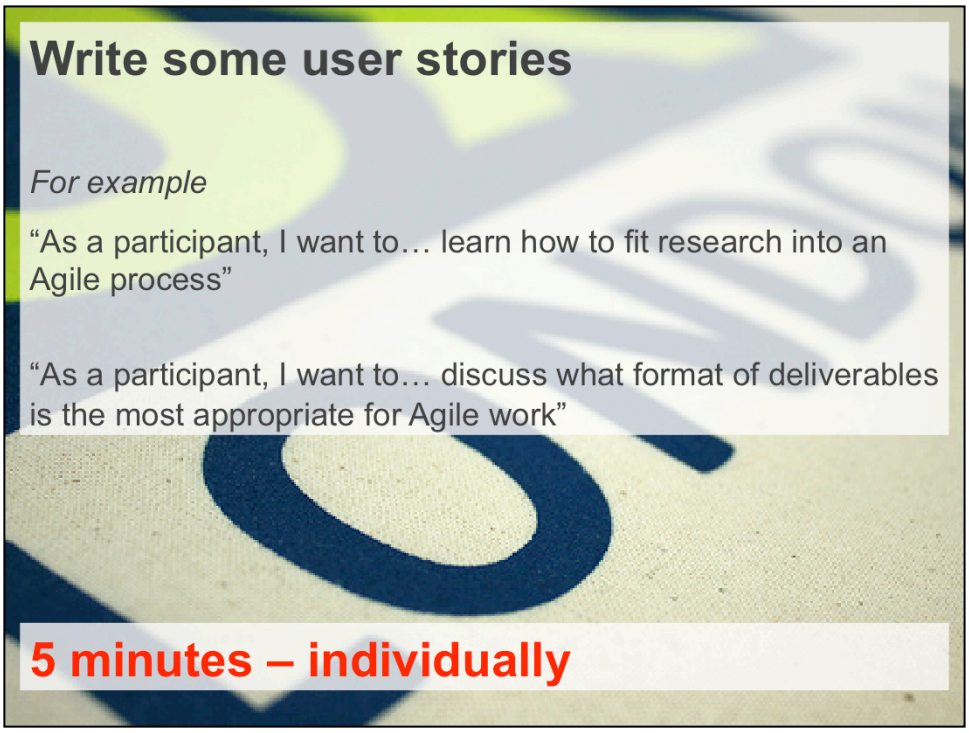
Today's my 30th birthday.
Cake will be served.
I'm down the pub afterwards.
Be nice. Particularly since…

This is very much an Agile session. Anyone running an Agile UX workshop must practice what they preach. So I've not done big design up front. Nor am I arrogantly assuming I know what you want. Although 30 mins = me talking, the rest is group discussions etc.

I don’t have all the answers. Maybe a few. Hopefully you have some too. No one’s really worked this out.

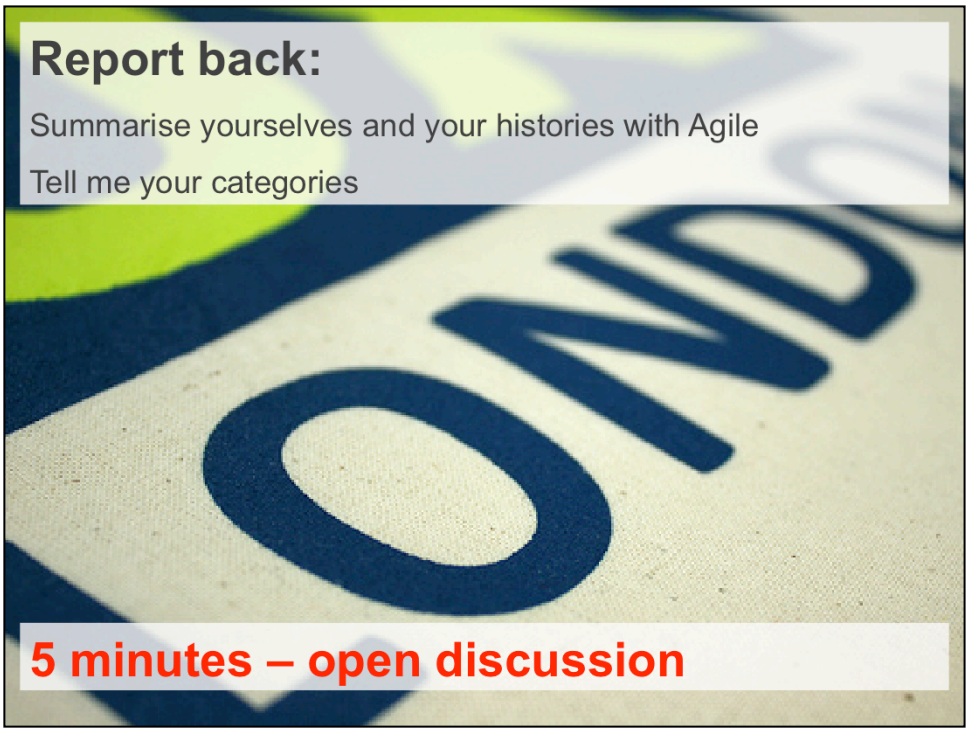
Also a bit more talking than doing. Hard to be that hands-on; it's an abstract concept. But we need to be collaborative. You will help make this session a success. In fact…

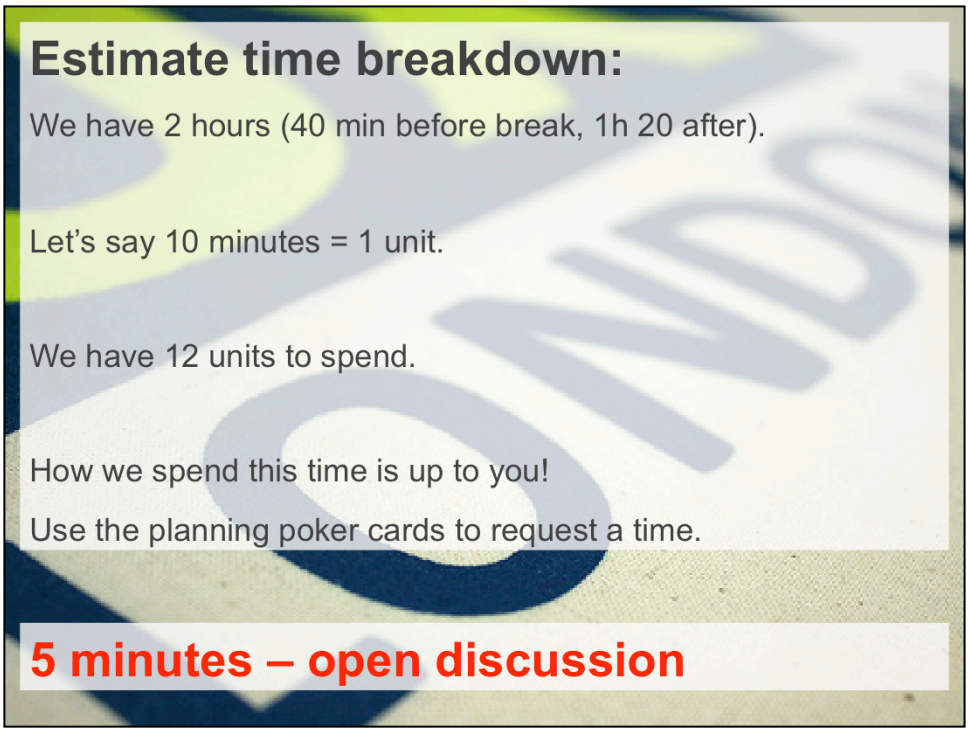
One Agile principle is that it's self-organising. I want you to set the agenda. Not just me winging it – I *have* prepared some bits and pieces; hopefully you'll ask similar stuff. If not, that's fine, we'll work it out together.
This is why no slides available: I don't know what we'll talk about. Will put together a PDF soon though


Write some user stories
For example
“As a participant, I want to… learn how to fit research into an Agile process”
“As a participant, I want to… discuss what format of deliverables is the most appropriate for Agile work”
5 minutes – individually

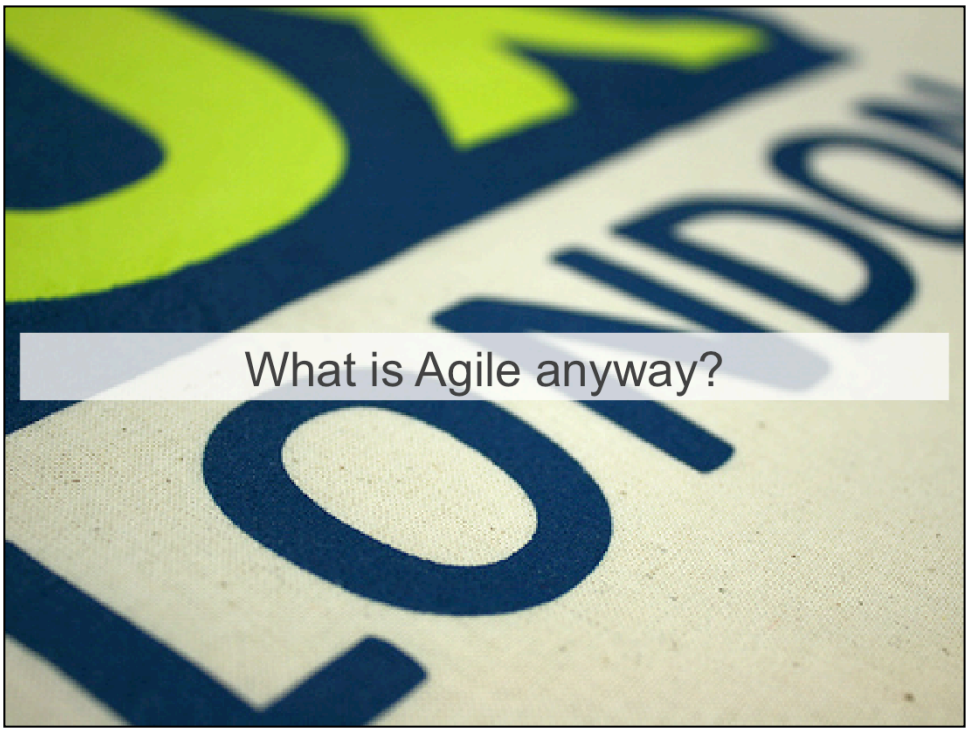
{exercise}


Group your user stories into themes
Introduce yourselves, your background and your history with Agile
Discuss your user stories
Card sort your user stories into themes or categories
5 minutes – group discussion


Report back:
Summarise yourselves and your histories with Agile
Tell me your categories
5 minutes – open discussion

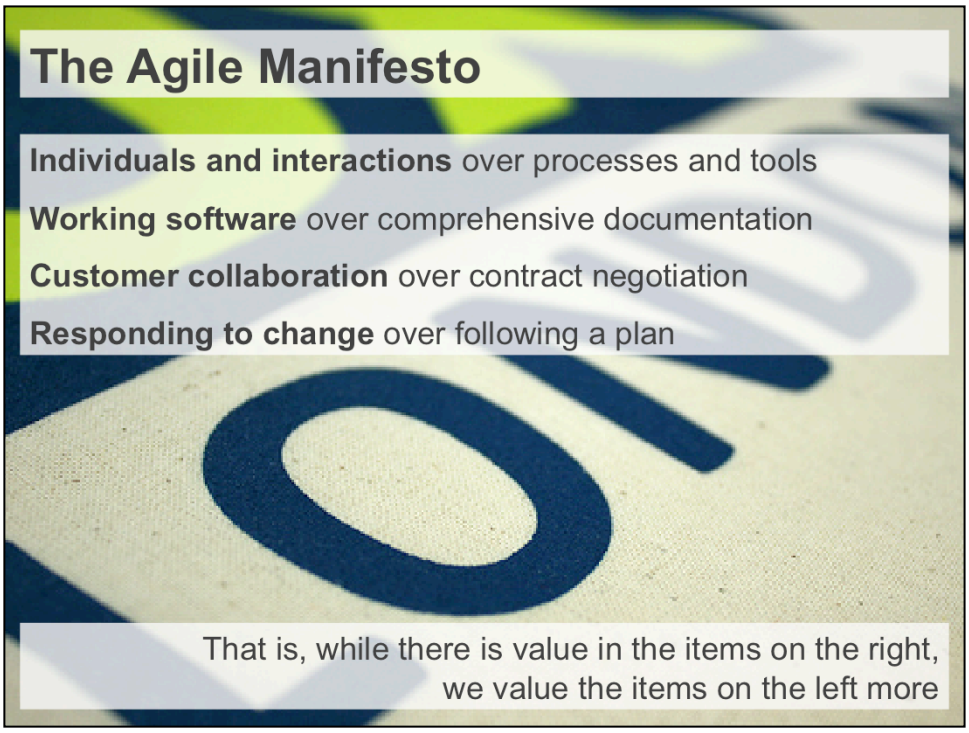


When do you want the break?



Need to start with some common understanding.

Normally hate Defining The Damn Thing, but 1% of time it’s necessary

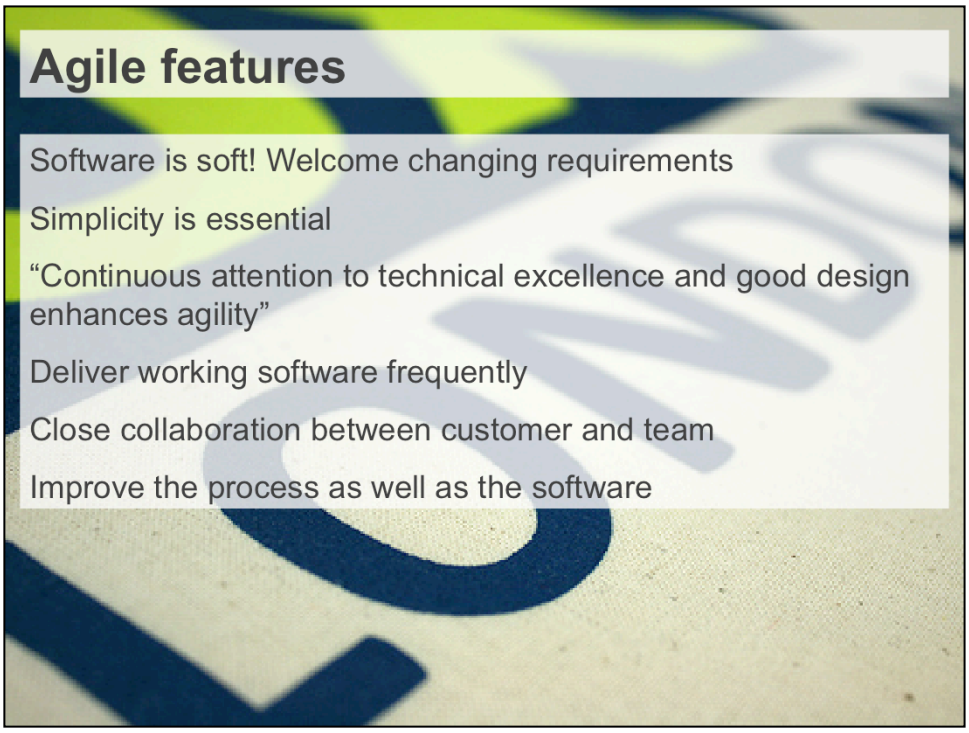


Manifesto. Before we go through, some background.

2001. Seventeen developers interested in ‘light’ methods met in a Utah ski resort

Few expected to find much common ground

Name ‘Agile’ and the Manifesto was the result. Only Martin Fowler had problems. Being a Brit, he knew Americans can’t pronounce Agile.



Also 12 principles, exerpted and paraphrased here.

Software is soft. Doesn’t have to be like architecture, industrial design etc. Changing requirements good: adjust scope etc instead. Minimising risk. Very reactive, not predictive (one reason why it’s hard for UX)

Simplicity: the art of maximising work not done. Aimed at developers, but twist it for users too!

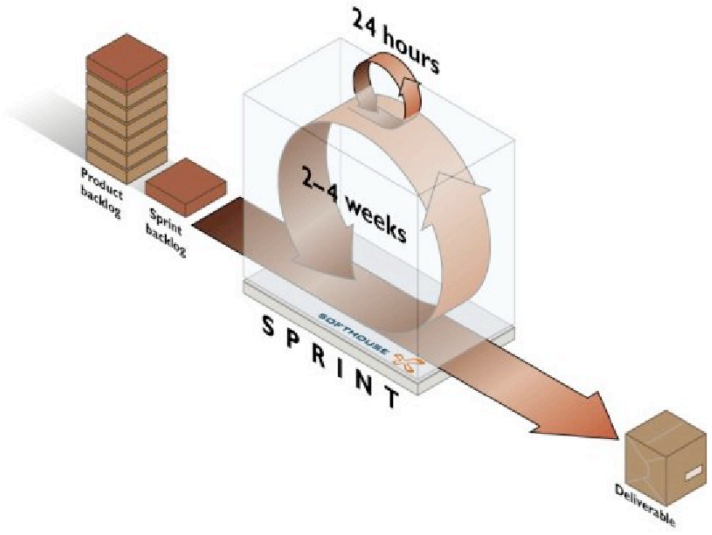
Attention to good design!

Deliver working software

Customer contact - Who is the customer? Product demos

Iterate on the process: retrospectives. Self-aware. Self-organising. Even scrums do this.

Agile features: Iteration



And, of course, Agile is iterative. So much so that our timeboxes for work are called iterations

Or sprints. I use the two interchangeably, although they're from different flavours of Agile

Snowman diagram from Scrum

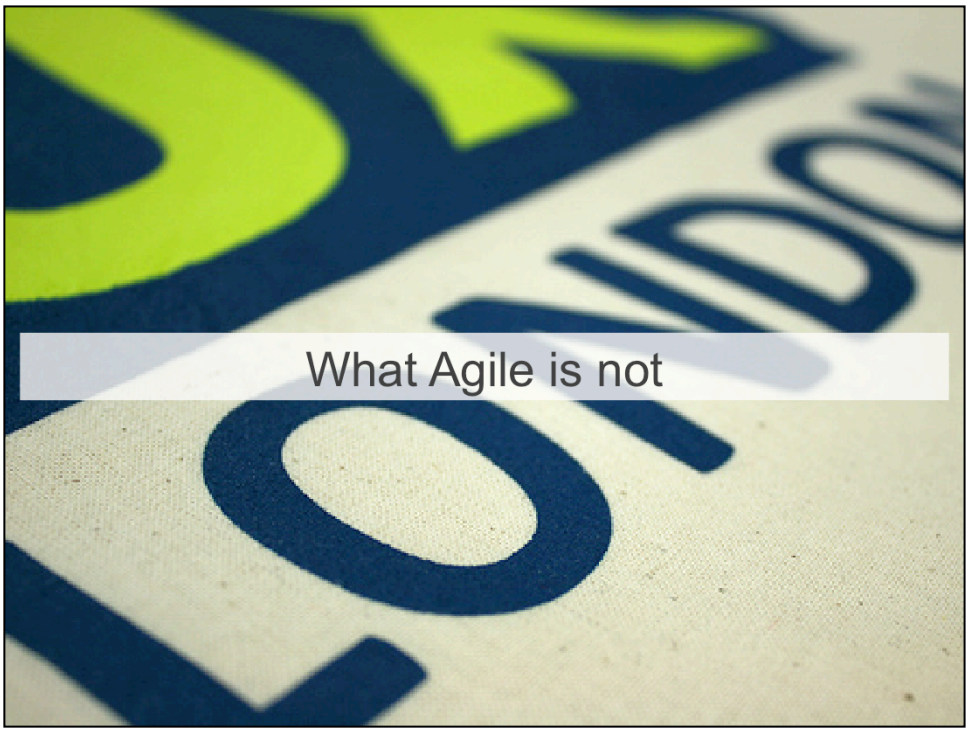
Product backlog created earlier. Pick some for this iteration, lasting 2-4 weeks (normally 2). Day-long sprint between e.g. scrums.

Other common feature: conjoins design and development. Another hair raiser for UX. But it doesn't merge them!

First rant of the day: doing Big Design Up Front, throwing it to developers who then do it in iterations isn't Agile!

Agile features: Estimation

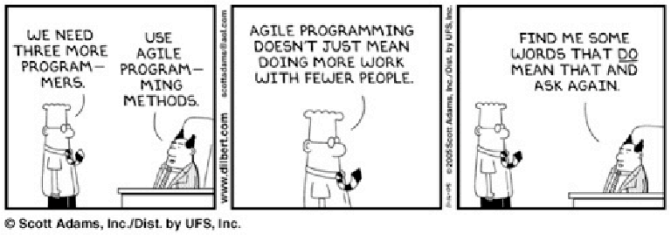
Beforehand, estimate work required. Typically developer-led. Pick what we can fit in an iteration. Planning poker based on Fibonacci (impress your mathematically-minded developers by pointing out that the ratio between the Fibonacci numbers tends to the Golden Ratio at its limit)



Helpful to explain

Agile is not:

Quicker or cheaper*



* (necessarily)

Sorry for a Dilbert slide. Terribly uncool.

People think it's cheaper & quicker. Particularly managers and Agile-reluctant designers

It's mostly about sanity in your process, putting the power where it should be: on the front line

It can be quicker and leaner, sure. But only with practice.

House of brands, the Agile methodologies

Extreme Programming (XP): Kent Beck. Pair programming. Anarchist method. Value-driven. Look ‘em up. You’ll like ‘em

Scrum: Snowman diagram. “Process skeleton”. ScrumMaster in lieu of PM. Accept requirements churn. Pig & chicken role. Daily scrums: only pigs can speak.

DSDM: Dynamic Systems Development Method. From RAD. Three phases (pre, lifecycle, post). Lifecycle: feasibility, business, functional model, design & build, implementation)

Rant 2: it’s not just about development in the coding sense.

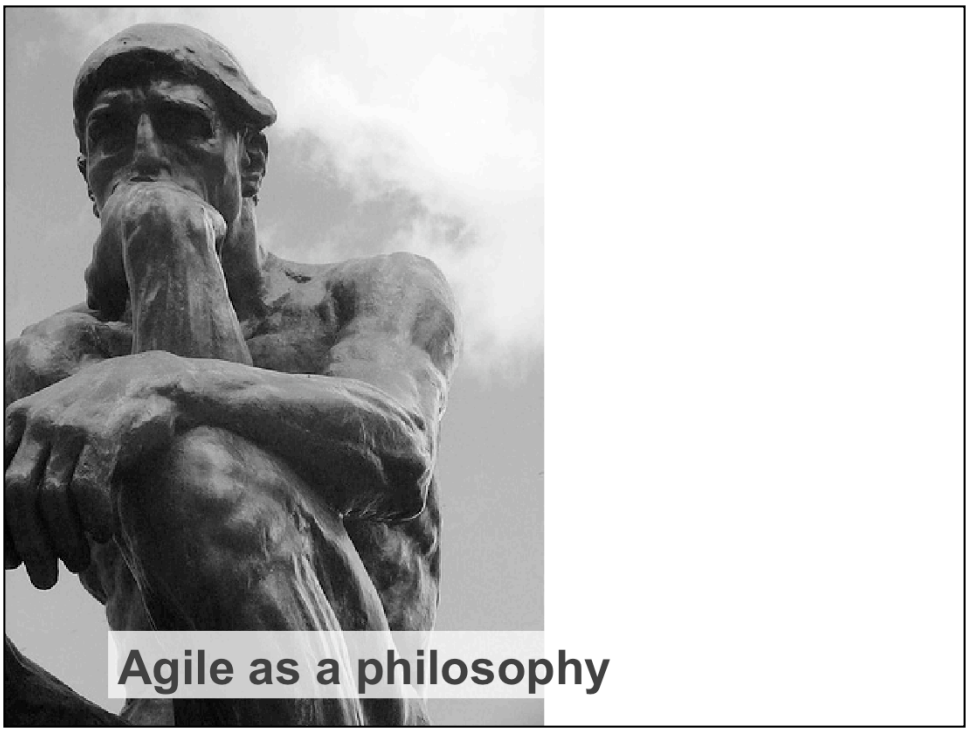
“Agile development” encompasses all, including design

“Agile development process” isn’t about how to code.

It's not a jigsaw. You don't pick a few stories, do them and forget about them

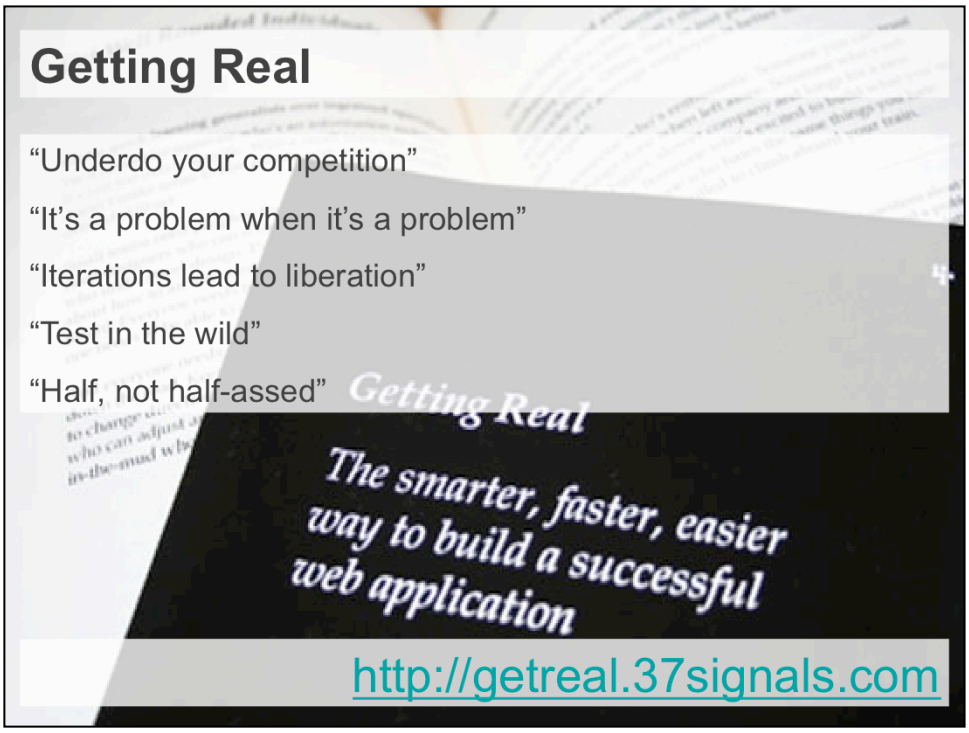
Must be able to revisit earlier stories

Need to manage customer expectations carefully; they may question "didn't we just do that bit?"



For me, it’s a philosophy. Mindset. Studies show most successful design is done with this approach.

Runs through whole software process, including business values



Who’s read it?

"Underdo your competition" – this runs through the entire software lifecycle

"It's a problem when it's a problem" - just-in-time thinking

"Half, not half assed" – focus on the really important stuff. What's going to make the big difference to your customers. Do you need log ins?

"Iterations lead to liberation” – you don’t have to be right first time. It’s not failing, it’s finding a way that doesn’t work

"Test in the wild" (although they rant inaccurately about usability testing. More on that later)

All compatible with UX

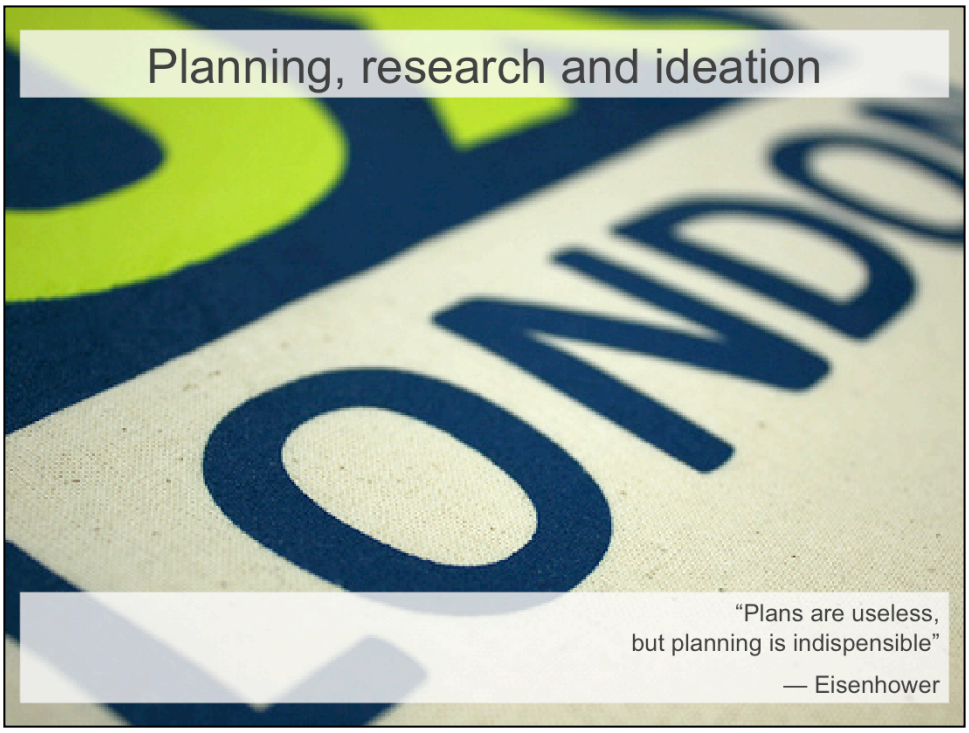
Shouldn’t be dogma. Lots of that around. UX has plenty too, which causes problems.

Things I espouse could be called post-Agilism.

Non-linear management. Anarchic self-organisation. Thinks Agile misapplied and overly controlling.

Perhaps. But I think benefits in looking at philosophy, and ignoring dogma.

If you have a dogmatist, good luck!



What do you want to know here?

Number one concern amongst people I speak to

And rightly so

Bad Agile leads to

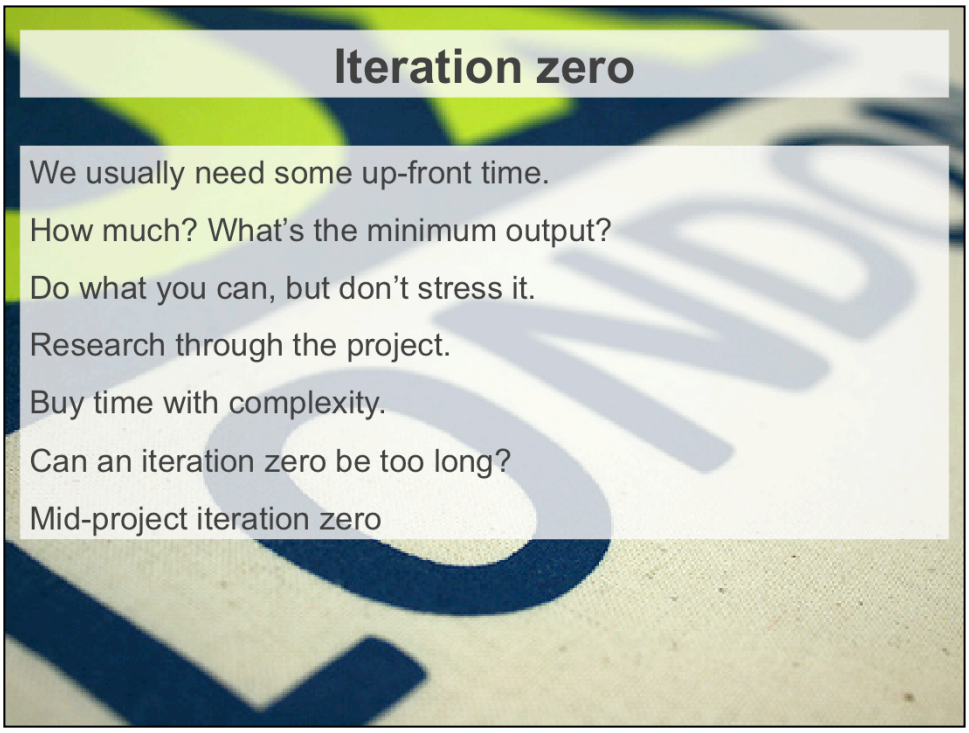
Most people’s concerns are that Agile doesn’t give us time for research and ideation. "Pure” Agile means coding on Day 1

Designers may not even be present. If we are, we're reduced to "User-scented design”. Please spread the love about this phrase ;)

No research, so designers use best guesses from previous experience. Does anyone identify with this? Skinning some database and developer-led system.

Can lead to risk averse design, or just copycat design of a competitor. Hollywood sequel syndrome.

If you’re in this situation, rely on your knowledge and design principles. You’re not doing UCD, but you can limit the damage. Universal Principles of Design.



Must fight for SOME up-front research for any decent sized project; Only way to combat user-scented design.

How long should it be? Enough to know you can solve things. This comes with experience (more later). Try to get a whole sprint (typically two weeks)

Decide what you want to get out of it. Are personas necessary? Do you have stand ins? (Often the customer is the user in Agile, eg 37signals)

Don't think you have to nail it all up front. Research can carry on through the project. *draw diagram*

Buy time with complexity! Coding *can* start on day 1 of iteration zero if it has to. Devs that they need to set up environments, get the backend structures etc set up. Or start on really obvious stuff, maybe logins. Or search. But don't let them start coding that irreversibly affects the design! That's designing from the inside out and is evil.

If your iteration zero HAS to be a month or more, you're probably not ready to consider executing it yet. Agile *may* not be right for you. Take on the fight, and good luck!

Mid-project iteration zero. Rare, but helpful. Developers like to tidy their code (refactoring). Meant to be part of Agile but often they'll have had to hack things together, just as you have. Run summative testing, check brand. Try scheduling around holidays! Might not need a whole iteration

{Draw minimax sketch}

Horizon effect: told not to worry about solutions to X. Tricky story may get pushed down the line, but you know it’s tricky.

Could break your whole system

Hard to check the end-to-end experience if modular

Some solutions:

Separate UX strategy project. What’s the experience vision? Tie into business goals.

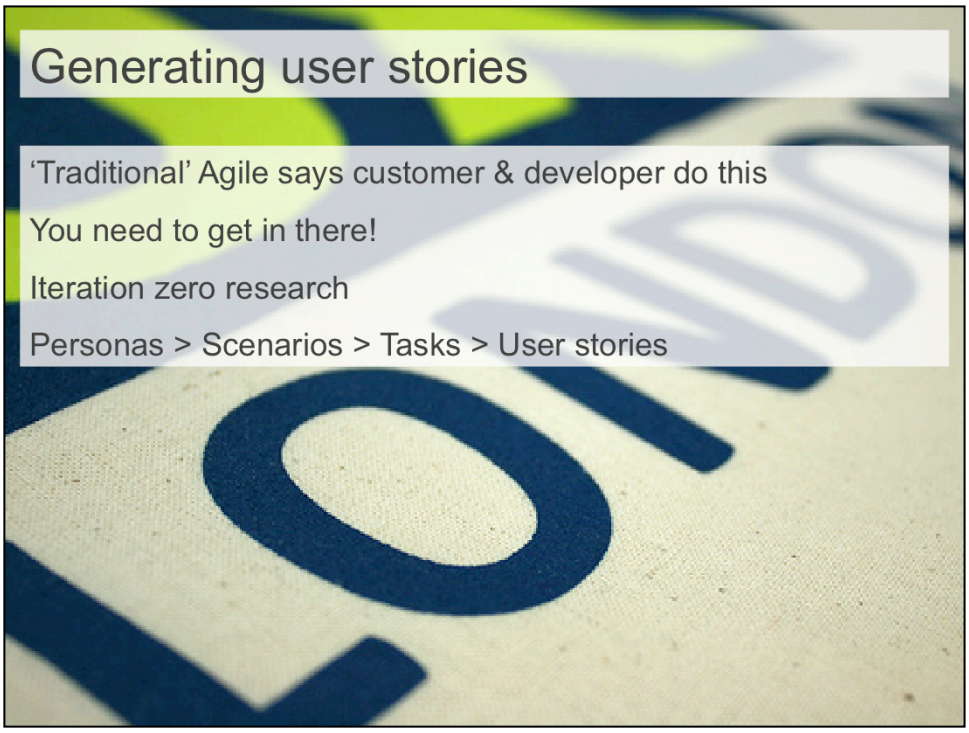
Design principles

What do you want to cover here?

Requirement specs are always wrong

Agile doesn’t do that (although I’ve seen some try)

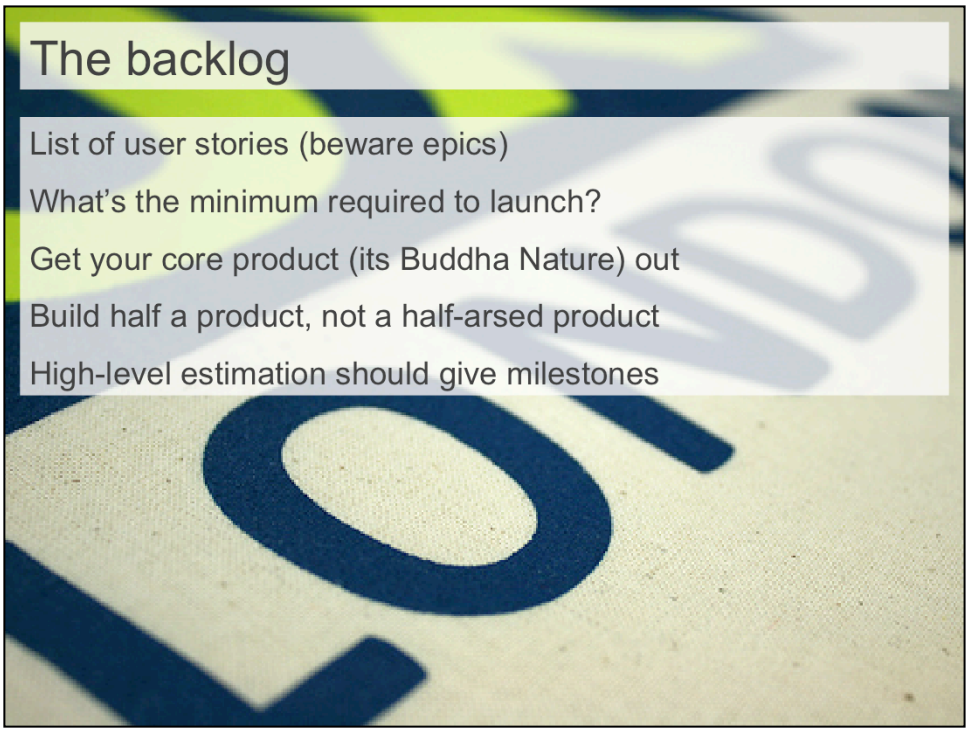
But you need to know what to build, right?

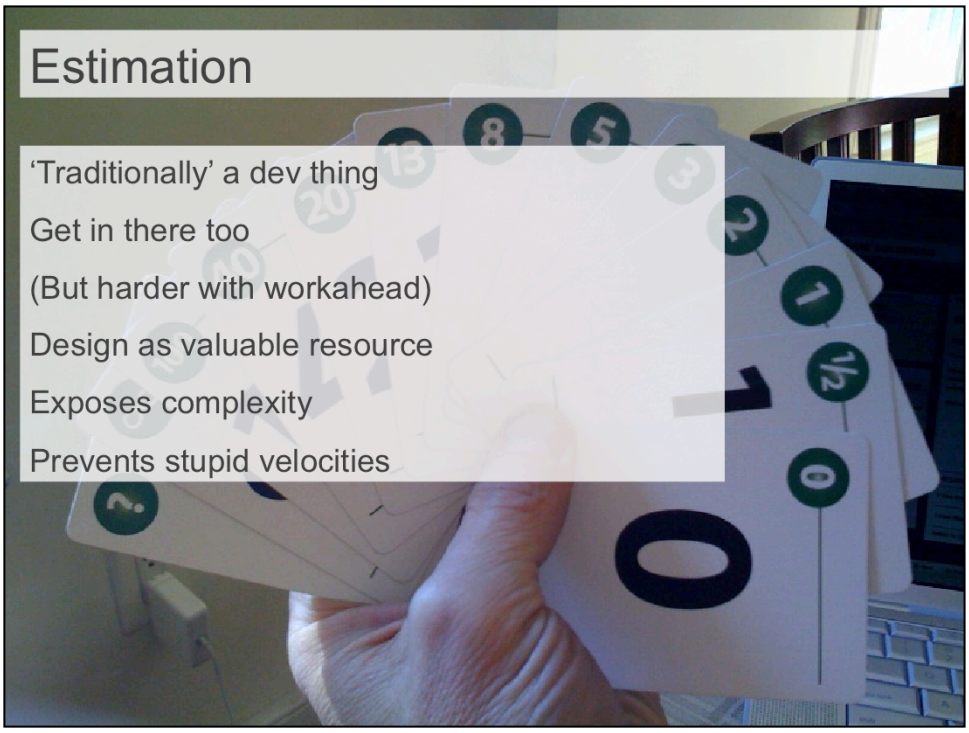


UX designers should get stuck in here

Introduce knowledge from your iteration zero

Personas create scenarios which create tasks. These map excellently on to user stories


The backlog
List of user stories (beware epics)
What's the minimum required to launch?
Get your core product (its Buddha Nature) out
Build half a product, not a half-arsed product
High-level estimation should give milestones



Important to sit in on iteration planning sessions (ask your PM / Scrum Master to put UI-related stories at the start so you needn’t sit in on back-end estimates)

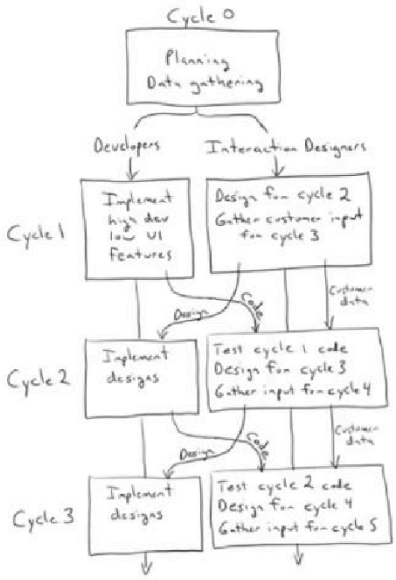
Exposes that design isn’t just a commodity; hard questions need time

Prevents stupid workloads

One concern is that Agile dictates your design process: Tail wagging the dog

Estimation important (see User stories, features and functionality)

Workahead (parallel track)



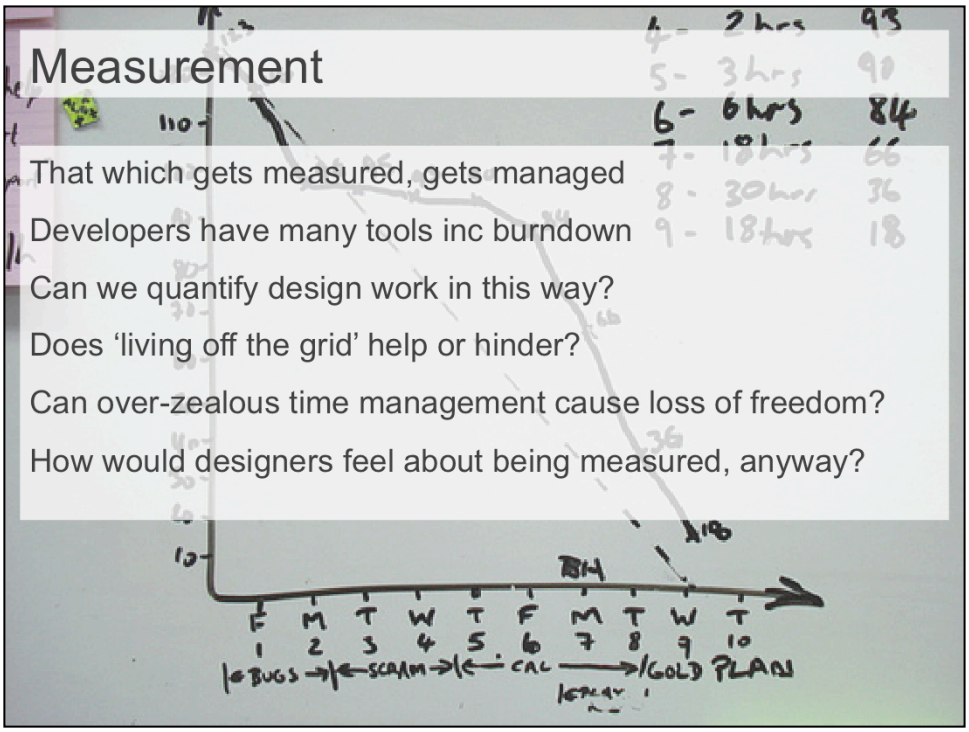
Best practice suggests research (n+2), design (n+1), support n, review (n-1)

But these are arbitrary timeboxes

Do your own high-level estimation of each story (hopefully you'll have done this in the big user story session)

Use your experience to identify "red flag" stories, tackle them early

Measurement



That which gets measured, gets managed

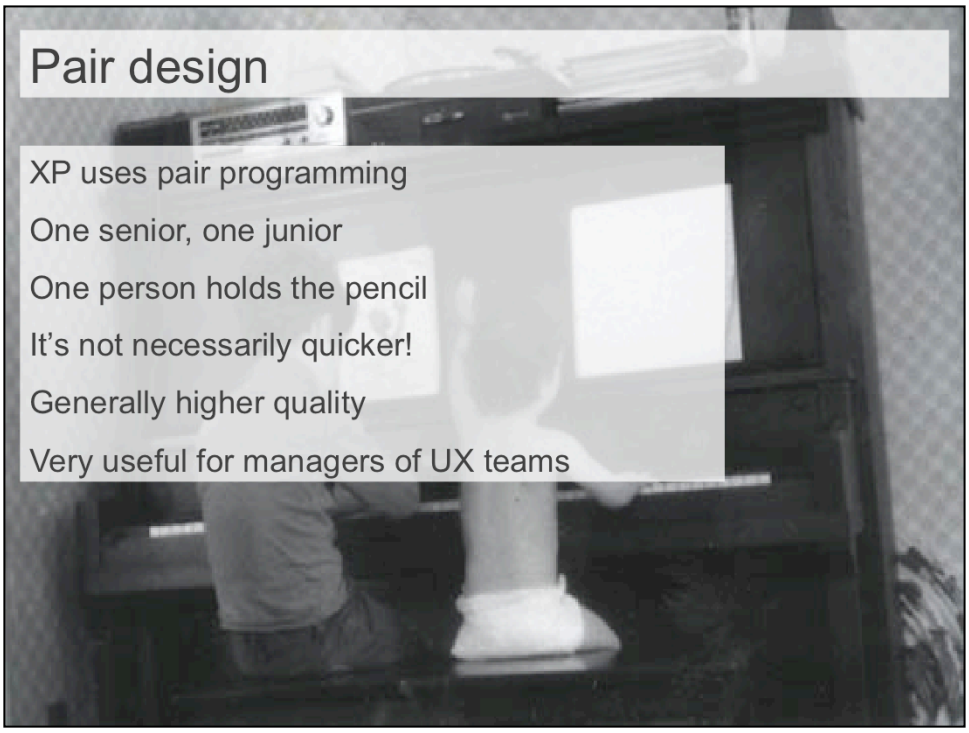
Developers have many tools inc burndown

Can we quantify design work in this way?

Does ‘living off the grid’ help or hinder?

Can over-zealous time management cause loss of freedom?

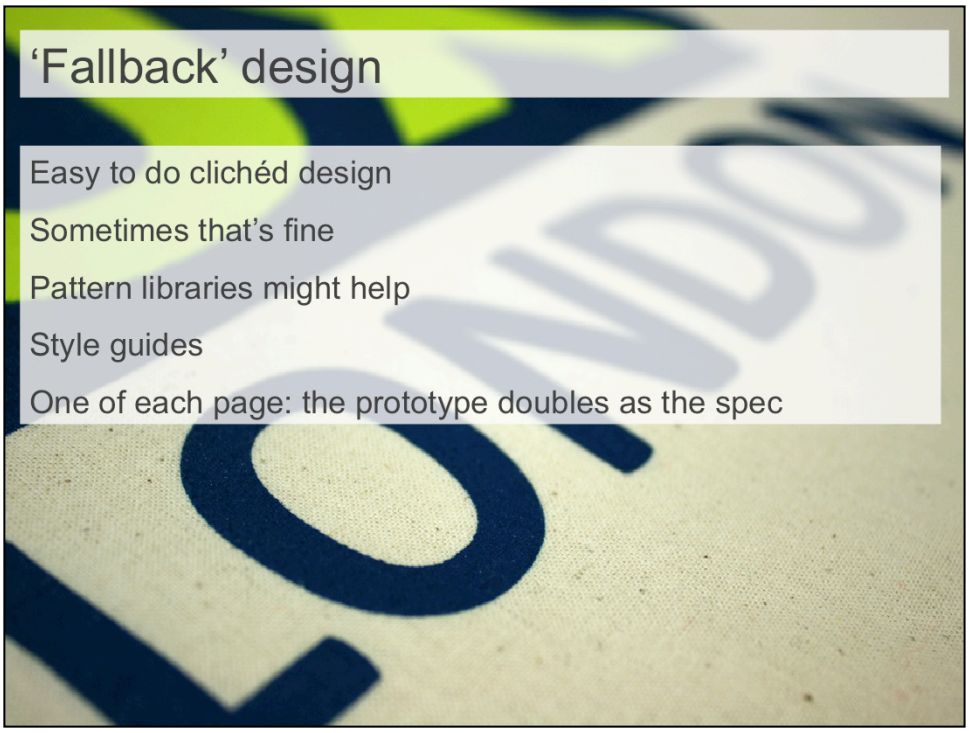
How would designers feel about being measured, anyway?

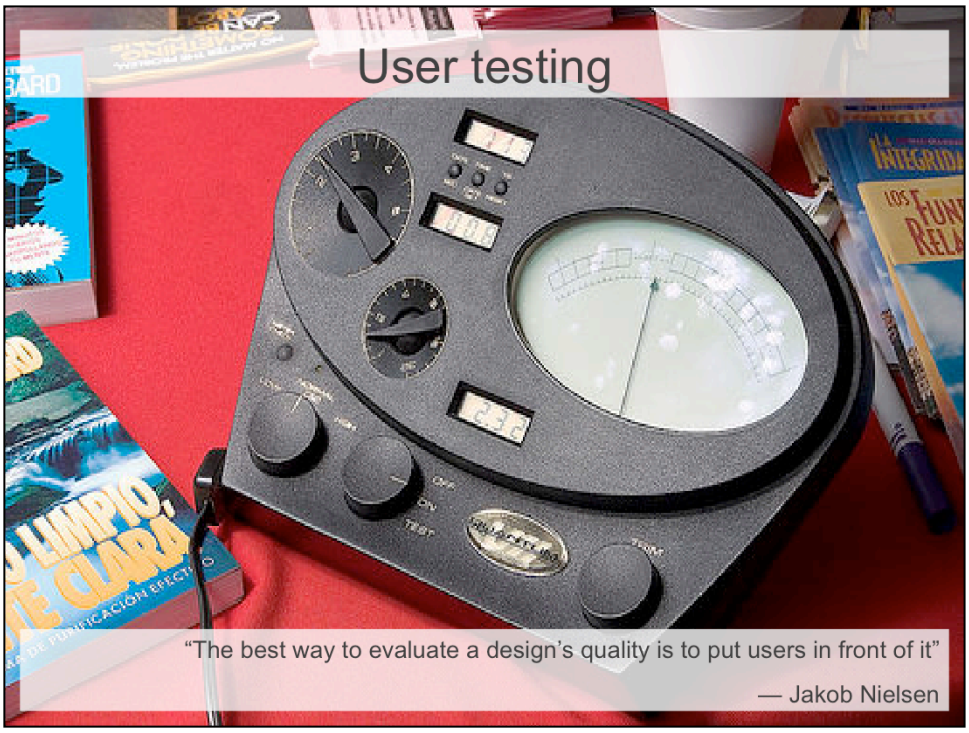


This might be a luxury

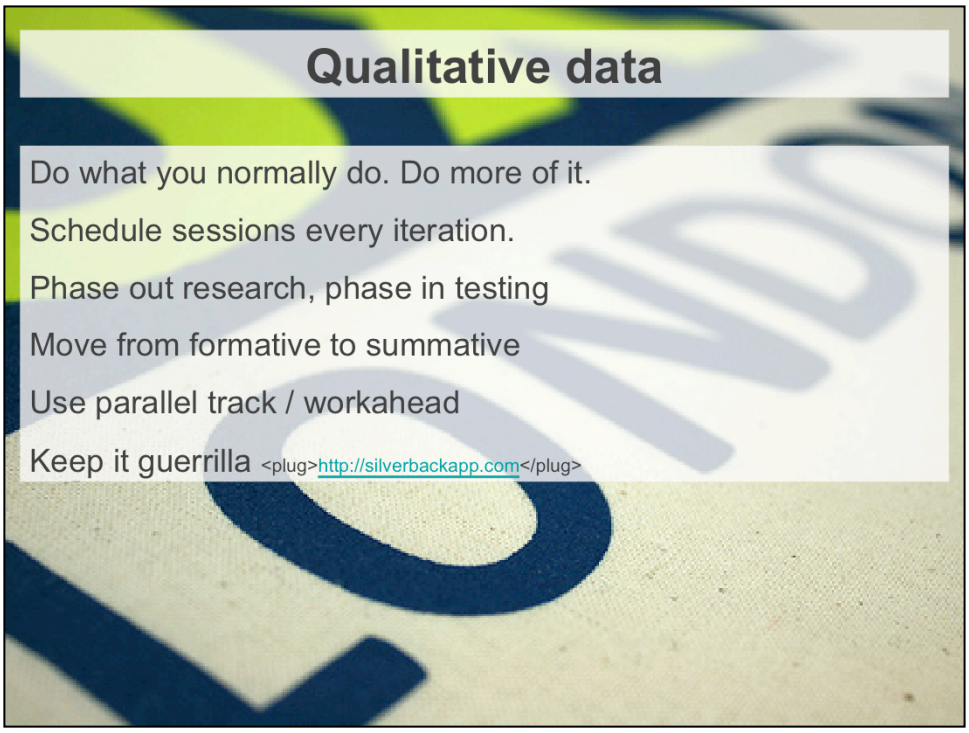
Different from collaborative design in that you’re both specialists

Can use design terminology etc


‘Fallback’ design
Easy to do clichéd design
Sometimes that’s fine
Pattern libraries might help
Style guides
One of each page: the prototype doubles as the spec



Agile is great in this regard. Gets stuff out there. What better way to test than on a real website? Can mix formative and summative, qual and quant.



Schedule a slot every sprint

Recruit weekly (not up front); requirements may change, you may want to cancel. Don't pay extra recruitment fees.

Move from formative to summative over time. Combine with research. Phase research out, and testing in

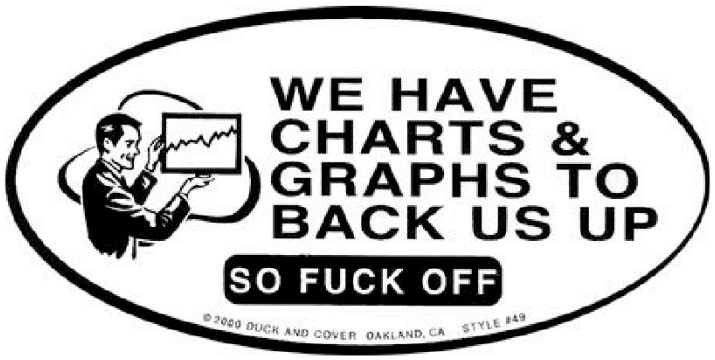
Workahead says research (n+2), formative test (n+1), evaluate (n-1).

"How do you currently X?" "Can you show me?" turns it into a mini usability test

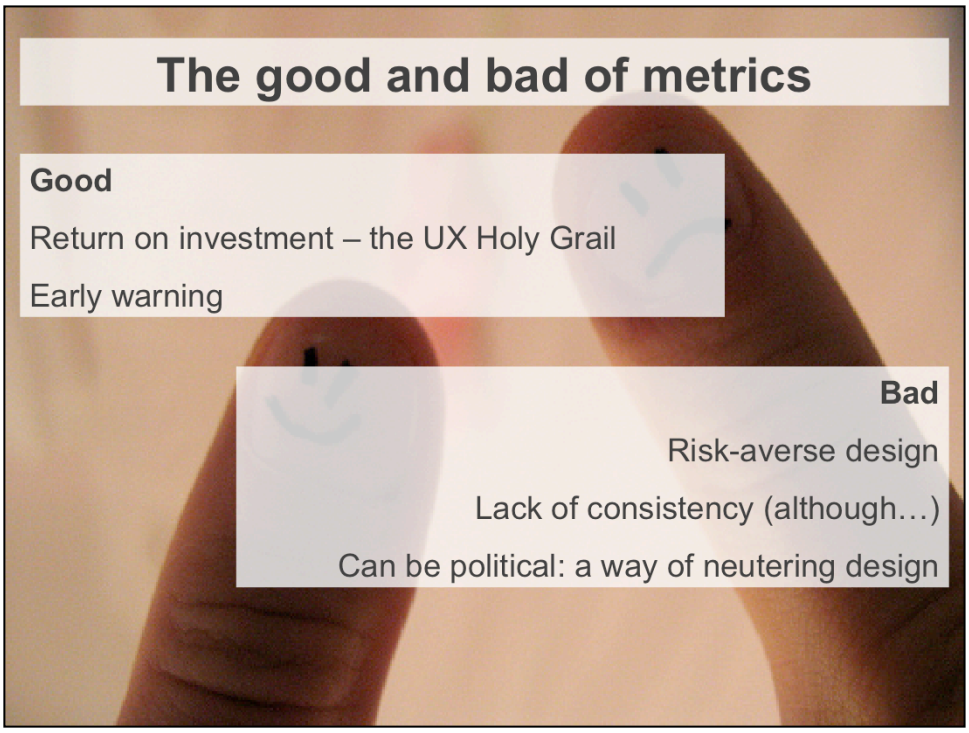
Keep it guerrilla. Formal lab settings aren't very Agile. We use Silverback, of course.

Share results with stakeholders in product demo. Even videos.

Quantitative data



Plenty of quant data available in Agile

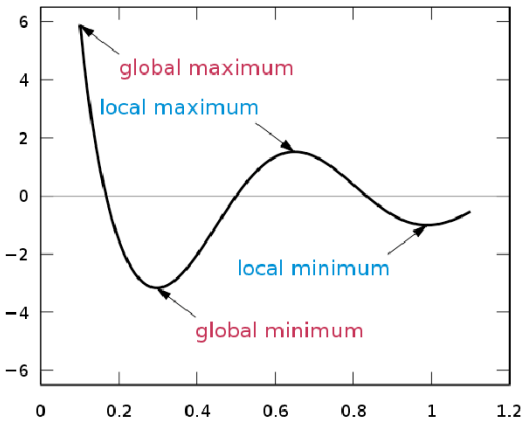


Types of metric: PVs, conversion rates, etc etc. A/B metrics

Benefits of metrics: RoI. High visibility / early warning of major successes and failures

Bad points too

Maxima and minima



Most sites are stuck in a local maximum

To get to a global takes large redesign effort (and courage)

To get from a fair to good solution may require getting worse first

Make just 20% of the changes required and you may find metrics go down

Needs courage to persist in the knowledge that the complete design makes sense

Deliverables

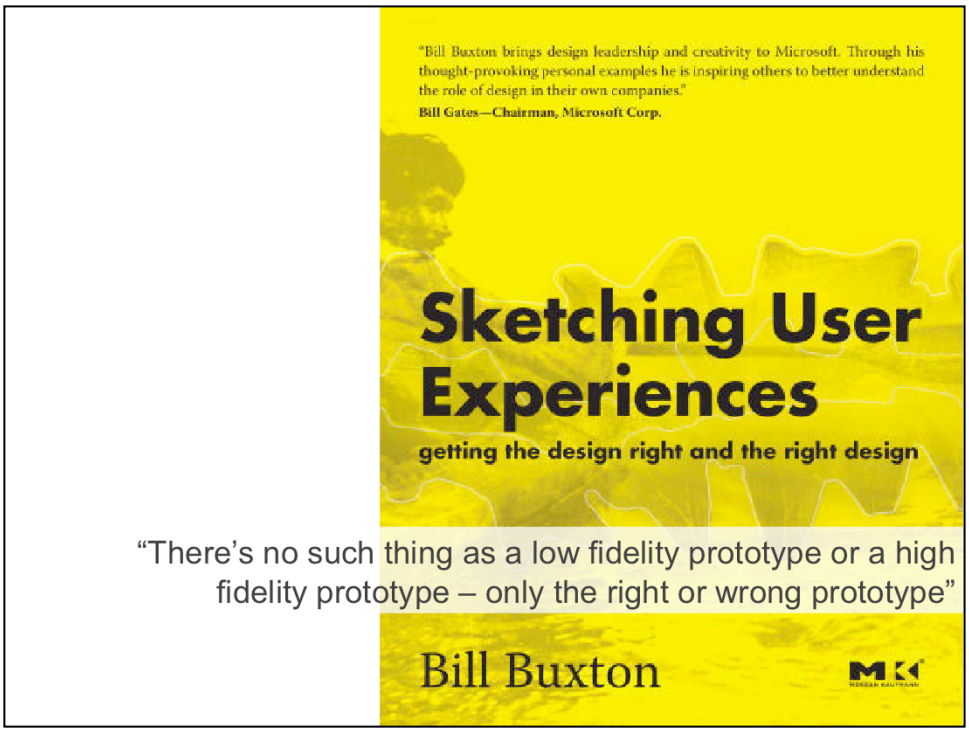
"Working software over comprehensive documentation"
— The Agile Manifesto

See this quote? Not going to be heavy on deliverables

Agile takes the view that no point creating glossy deliverables to give to the dev sitting opposite

A face-to-face discussion is better

Tends to mean low fidelity is the way to go, but:

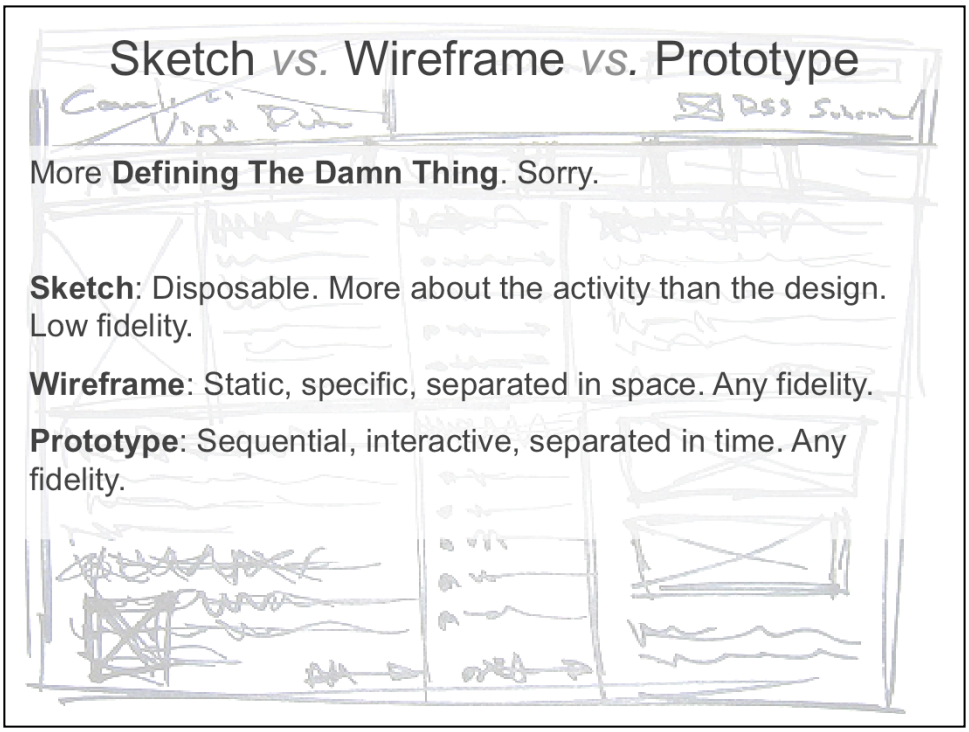


So be flexible. No one approach is right.

At Clearleft we vary considerably. Now I'm doing HTML and whiteboard sketches. Last project was all OmniGraffle.

Classic "it depends" moment. You need to have skills in sketching, wireframing, prototyping

What's the difference?



Sketch: Not a deliverable. A question, not an answer. If you lose it, that's ok. Shows function and general theme. The 'parti' perhaps?

Wireframing: static, paper based. More about specifying and handing over. Any fidelity. Separated in space. Comic strip. Something you read. Can be a deliverable.

Prototyping: sequential, interactive. Separated in time. Film. Something you experience. Can be a deliverable. Great for testing (only live software is better).

For me, sketching and prototyping *feel* more Agile.

Show of hands: who uses what?

Axure: suited to: prototyping. Pro: Very walkable, relatively easy to change Con: Expensive, learning curve

HTML: prototyping. Pro: developers love you. Quick, if you have good frameworks. Browser native -> great for testing Con: Requires skill. Harder to set up.

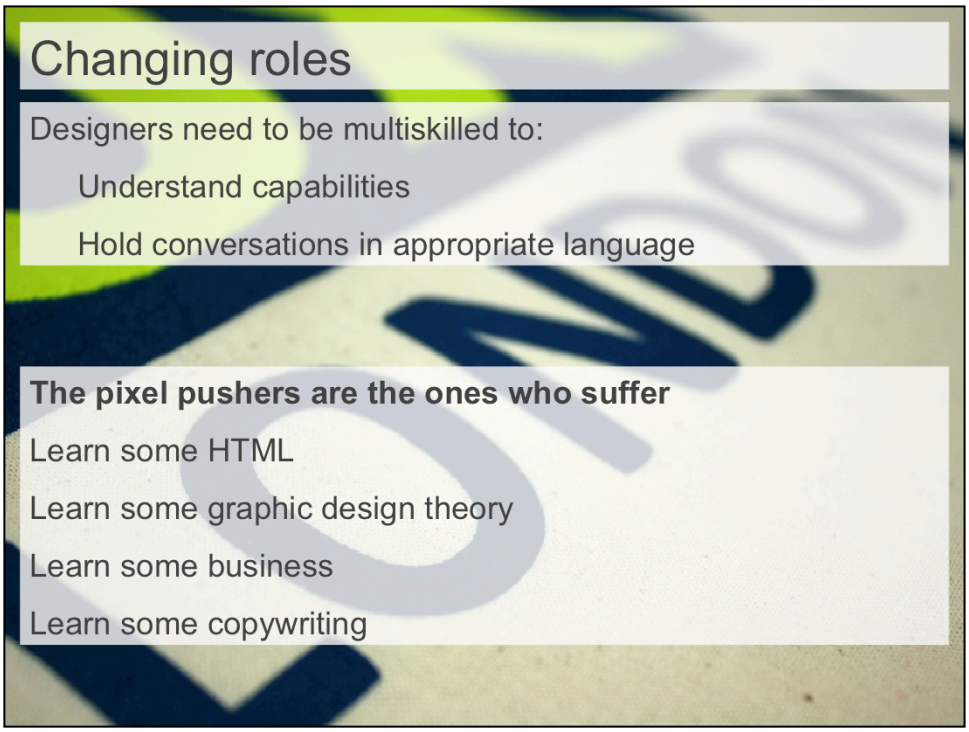
Balsamiq: wireframing, sketching. Pro: sketchy. Deprecates visual representation, focuses on content, hierarchy, functionality Con: Worst of both worlds? Comic Sans doesn't belong on a design tool. Sometimes you need visual rep.

Pen & paper: sketching, wireframing. Pro: quick, always on. Con: not great for deliverables unless you have great skill. Non digital; hard to edit.

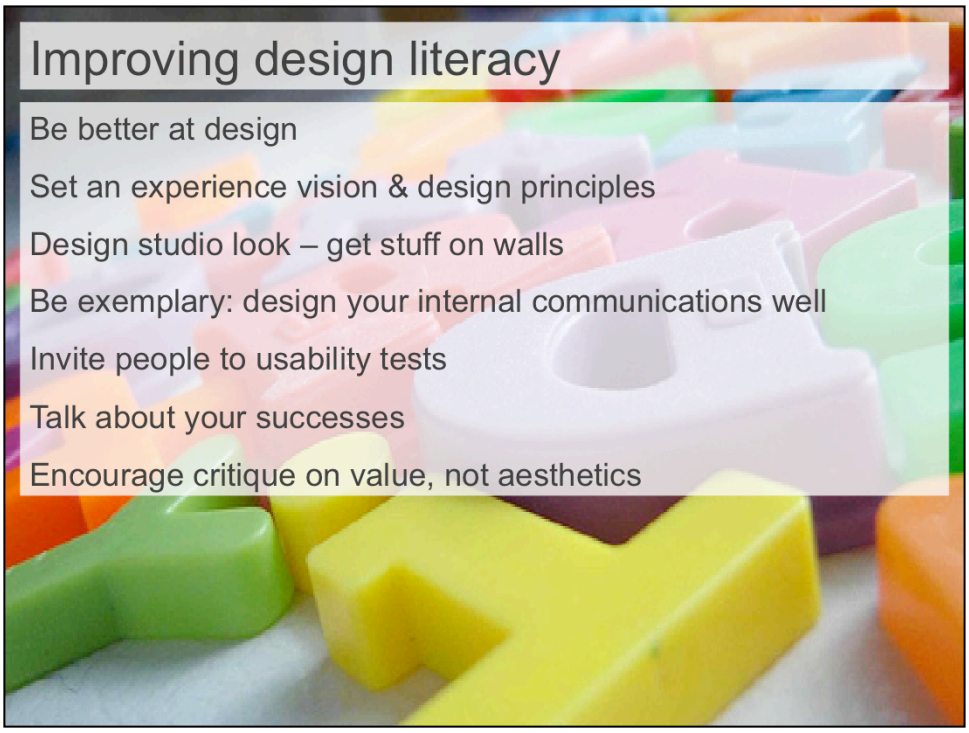
Visio etc: wireframing. Pro: Low learning curve. Con: static, boxy. Not designed for Uis, so hacky. Look more complete than they are. Wireframing isn't a faithful test of system behaviour. "Why are interactions like Canada?" "Because they're usually overshadowed by the States!" (Buxton). Visio wireframes favour states.

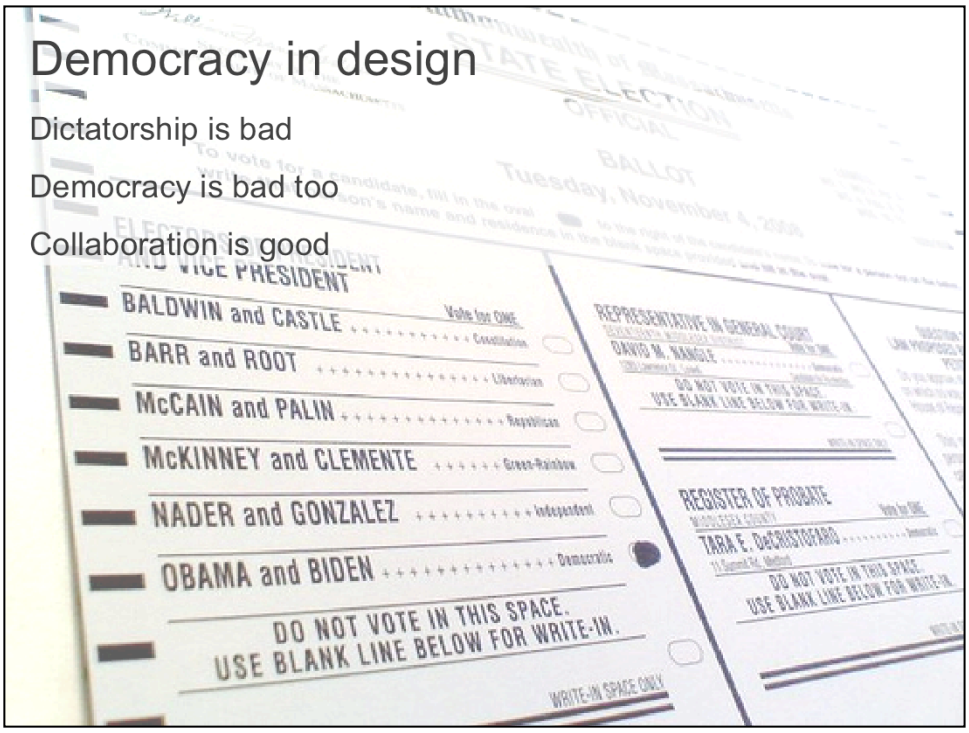
My preferences.

Need to change the way we handle internal structure



But surely you should know these things anyway?


Improving design literacy
Be better at design
Set an experience vision & design principles
Design studio look – get stuff on walls
Be exemplary: design your internal communications well
Invite people to usability tests
Talk about your successes
Encourage critique on value, not aesthetics



Dictators don't have buy-in

Democracy assumes all are created equal. And you're a better designer. Don't be afraid of that, be proud of it.

But a) you haven't time to do it all yourself b) you're not so arrogant to think you're the ONLY source of ideas c) you want to increase the chance of execution d) you want to improve the standing of design in your org.

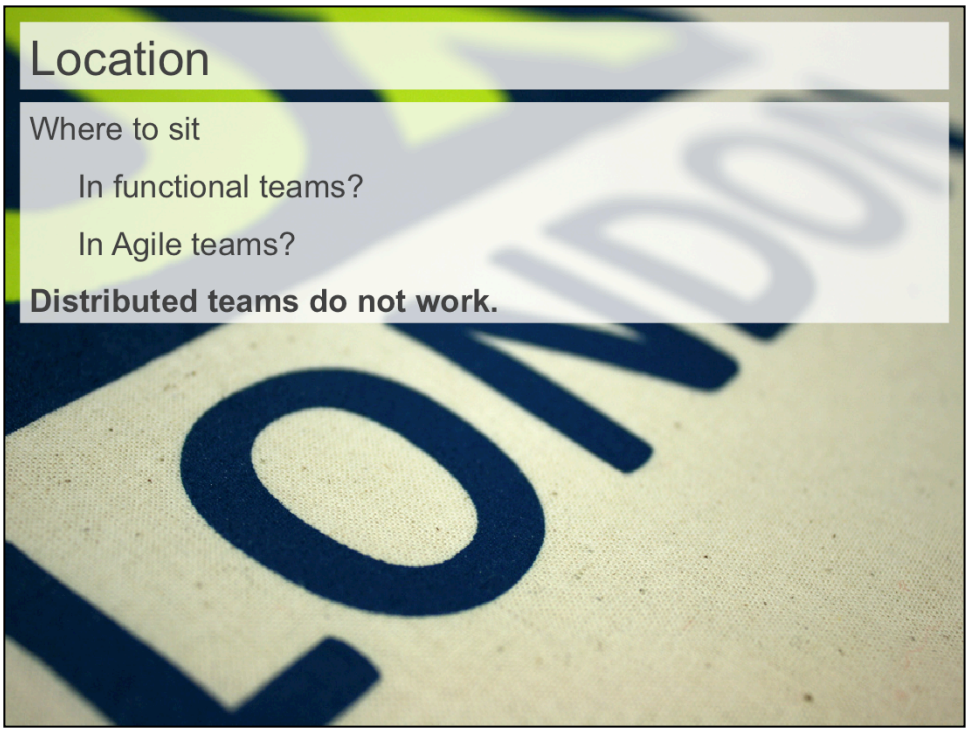
So engage people in collaborative design. Design is FUN. Everyone likes to do it. Everyone thinks they can do it. And they're right. But you can do it better.

If you don't, they'll do it anyway. Better to control and use that output than let it run riot.

Some useful for setting vision, some useful for helping with page design.

These take an extrovert!

Divide the dollar links in nicely with Page Description Diagrams

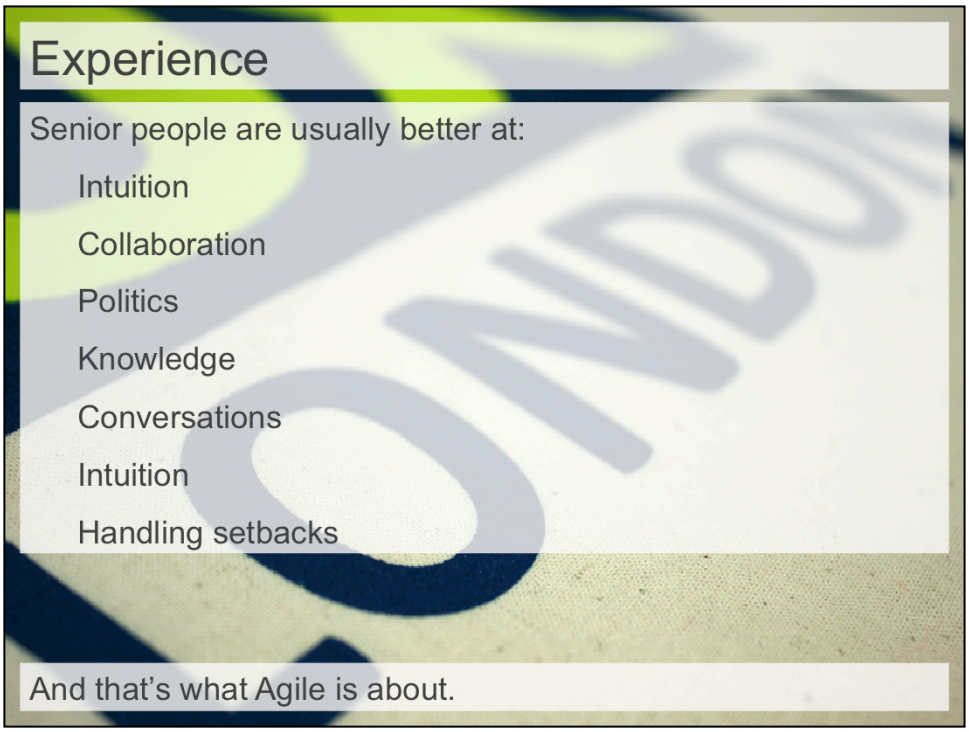


Tell story from previous employment

Distributed teams are nightmarish in Agile. Ref: Johanna Kollmann MSc dissertation. See also “Bridging the Distance”

We’ve done it on a couple of *really* small concepts that had Agile elements

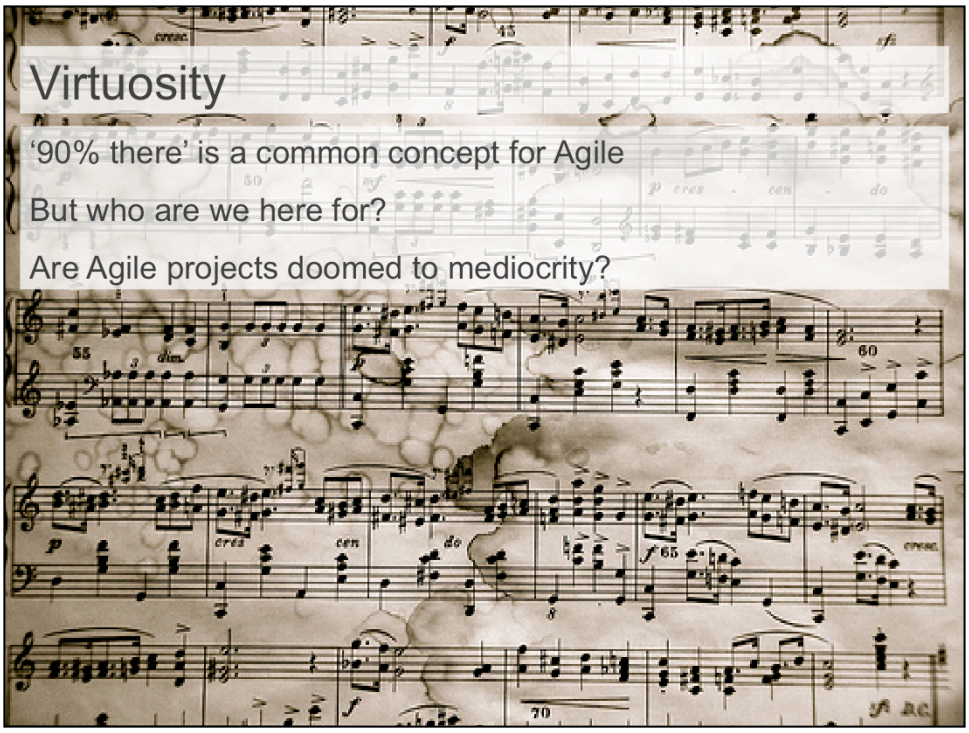
Tried to replicate the environment where possible: daily scrum over Skype etc



Junior UX staff can struggle with Agile
It can appear confrontational
Goes against their education
Often retreat into negativity or passivity
Takes real skill to coach them through this phase
Age and emotional maturity are important factors too
Developers are the same too – they recommend senior people

Quality

Perception that Agile encourages a lack of quality



Letting go: 90% right is part of life in Agile

To an extent, we're to blame. Is 90% that bad?

Remember who we're here for: users

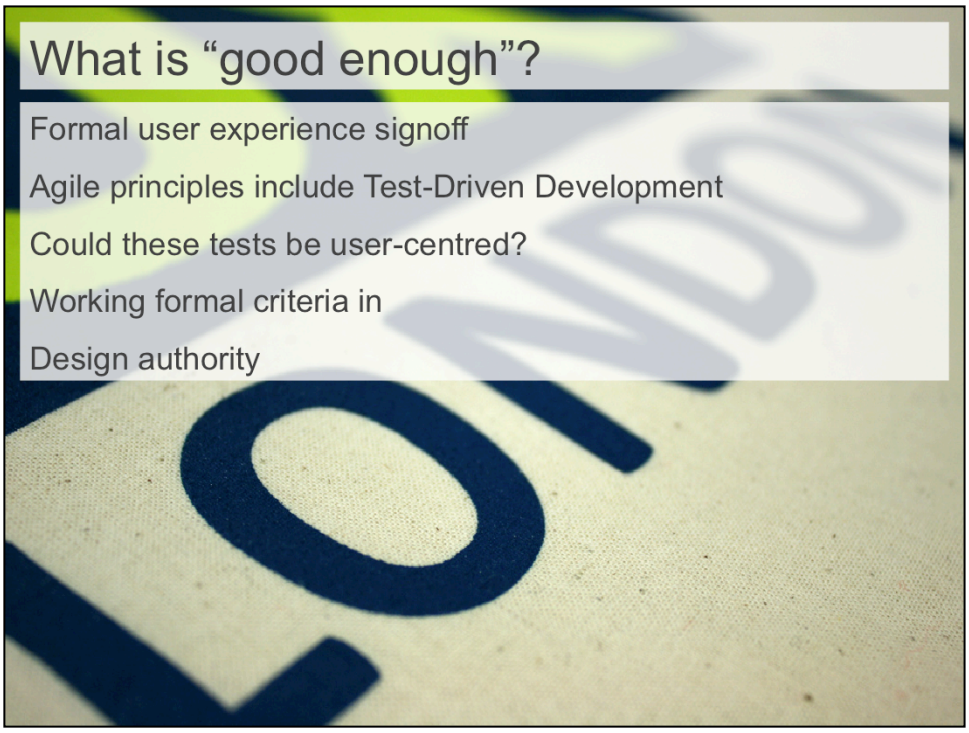
Realistically, do users notice *exquisite* interactions, typography etc?

Impressing other designers is good for the ego but otherwise harmful

Does this mean that Agile projects are never spectacular?

Perhaps. To create a true breakthrough product needs us to answer a new user need. This requires analysis and ideation.

Obvious that the more time you have for ideation, the more likely a novel solution

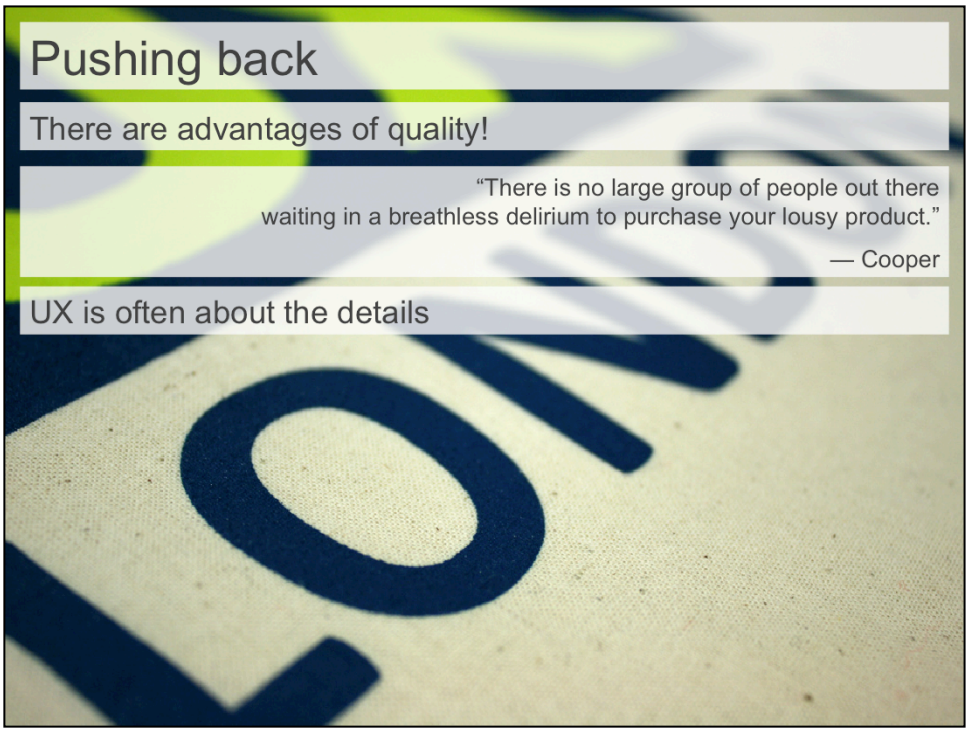


Do you have formal signoff?

You can decide, or users can? How should this work? Assess against the personas

Boolean 'yes it's fine' v 'no, it's not good enough' is crude. Can work formal criteria in. Either metrics from live data (error rate, dropout, time) or quant data from testing (completion rate, happy scores etc)

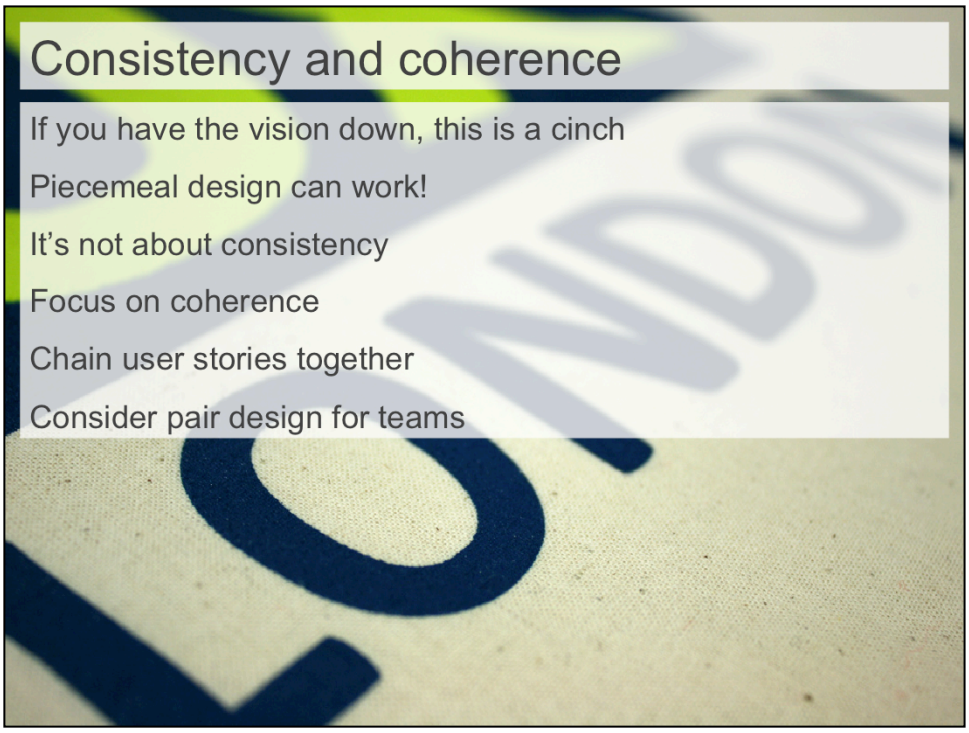
If no such design authority exists, that's a cultural problem. Work on your bosses.



Marketing advantages of quality?

Best to market v first to market

Sometimes UX is about the details, although not as often as we like to think



Vision and design principles

The 'parti' – Luke W

Don't create a big style guide at the start

Let emergent design patterns evolve and use them as the spec

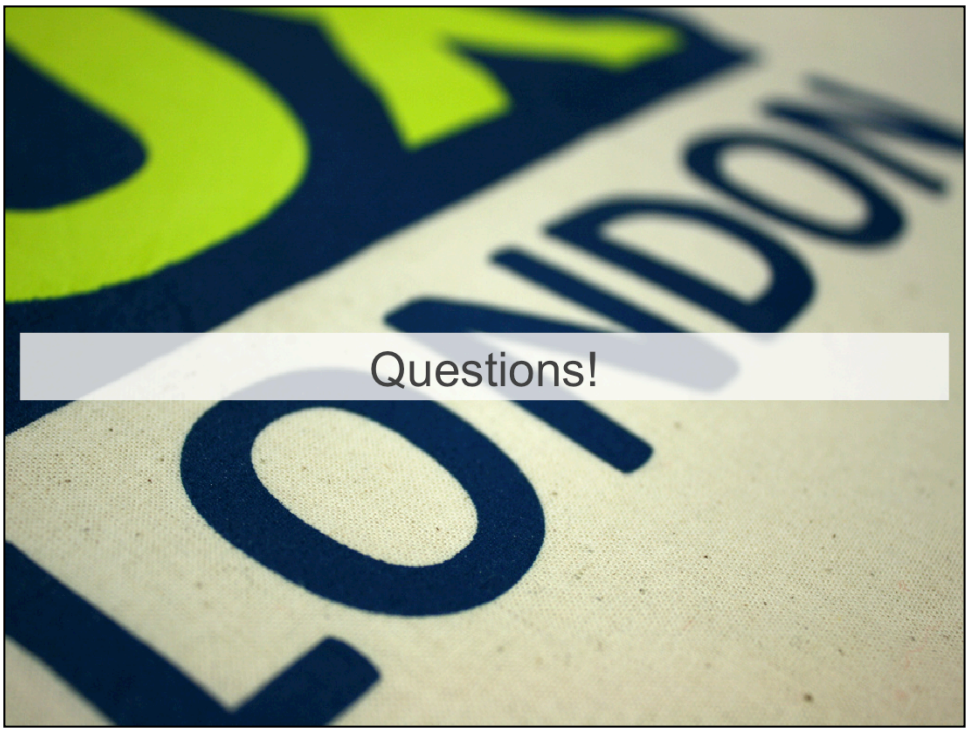
But don't slavishly follow them; flex where appropriate

Consistency doesn't matter. Coherence. Part of the same family. Make it appropriate to the user's task.

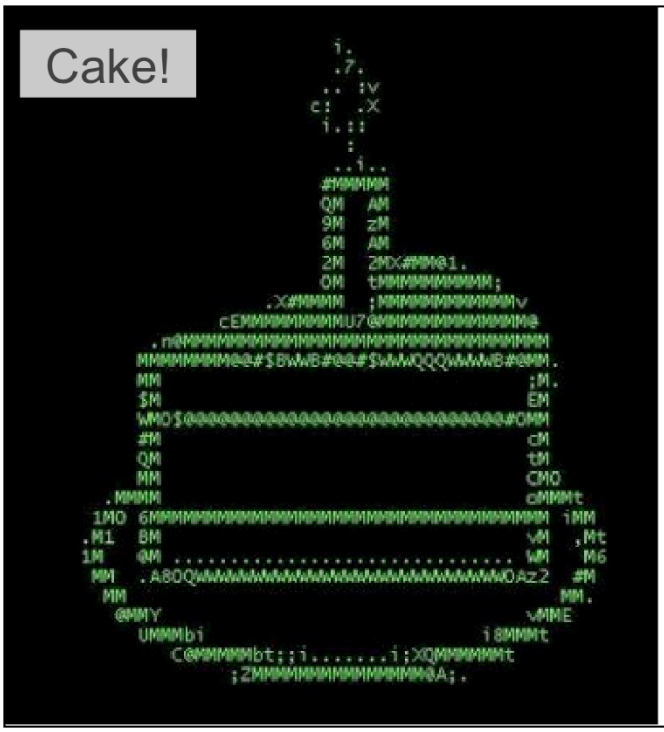
Chain users stories together. Design them all at once to ensure you at least have consistency along that one major chain. Don't have to be developed in same iteration.

(Look for epic stories; these often represent a long chain of user interactions)

If multiple designers, consider pair design to improve consistency


LONDON
Questions!

Cake!




The end! The pub!
www.clearleft.com
www.cennydd.co.uk
@cennydd
Birthday boy http://www.flickr.com/photos/mag3737/
Cennydd http://www.flickr.com/photos/anna_debenham/
Question mark http://www.flickr.com/photos/earlg/
Jigsaw http://www.flickr.com/photos/queen_of_subtle/
Thinker http://www.flickr.com/photos/resetreboot/
Jelly Beans http://www.flickr.com/photos/meltingmama/
Downstream http://www.flickr.com/photos/fredericksburg/
Scientology bullshit http://www.flickr.com/photos/giantginkgo/
Fingers http://www.flickr.com/photos/assbach/
Big red button http://www.flickr.com/photos/wlodi/
Clock http://www.flickr.com/photos/klarakristina/
Burndown http://www.flickr.com/photos/jnicho02/
Scribbly wireframe http://www.flickr.com/photos/maxpower/
Kitchen tools http://www.flickr.com/photos/xdjio/
Fridge magnets http://www.flickr.com/photos/taylorhood/
Ballot paper http://www.flickr.com/photos/uzi978/
Post its http://www.flickr.com/photos/missnita/