

A photograph of a multi-story brick building with several bay windows. The building is made of reddish-brown bricks and has dark shutters on the windows. In the foreground, several cars are parked in a row. The image is slightly tilted to the right. A semi-transparent dark band across the middle of the image contains white text.

lean front end engineering

bringing great design to life
by applying lean principles to front-end engineering

bill scott (@billwscott)
sr. director, user interface engineering, paypal

front-end masters series
nov 30, 2012

schedule

- 9:15 am. building products (60 mins)
- 10:15 am. break (15 mins)
- 10:30 am. lean ux (75 mins)
- 11:45 am. q & a (15 mins)
- 12:00 pm. lunch (60 mins)
- 1:00 pm. lean tech stack (80 mins)
- 2:20 pm. break (20 mins)
- 2:40 pm. anti-patterns (75 mins)
- 3:55 pm. q & a and wrapup (20 mins)
- 4:15 pm. dismiss.



building products

what the problem is and why we are here today

quick poll

purpose

equip you with how to work with your partners to bring great experience to life

apply lean UX & lean technology approaches to front-end engineering

end of the day: bring awesome experiences to life

paypal?

what can we learn from paypal?

“culture of a long shelf life”

team roles



standard process creates distinct work phases

boundaries are the hand-off points between roles

product

product manager, business analyst

owns the features. doesn't do design. drives by hypothesis.

typically produces PRD





design

UED, UX, ID, IA, VizDe,
Content, designer

not responsible for
engineering the experience,
but designing the
experience.

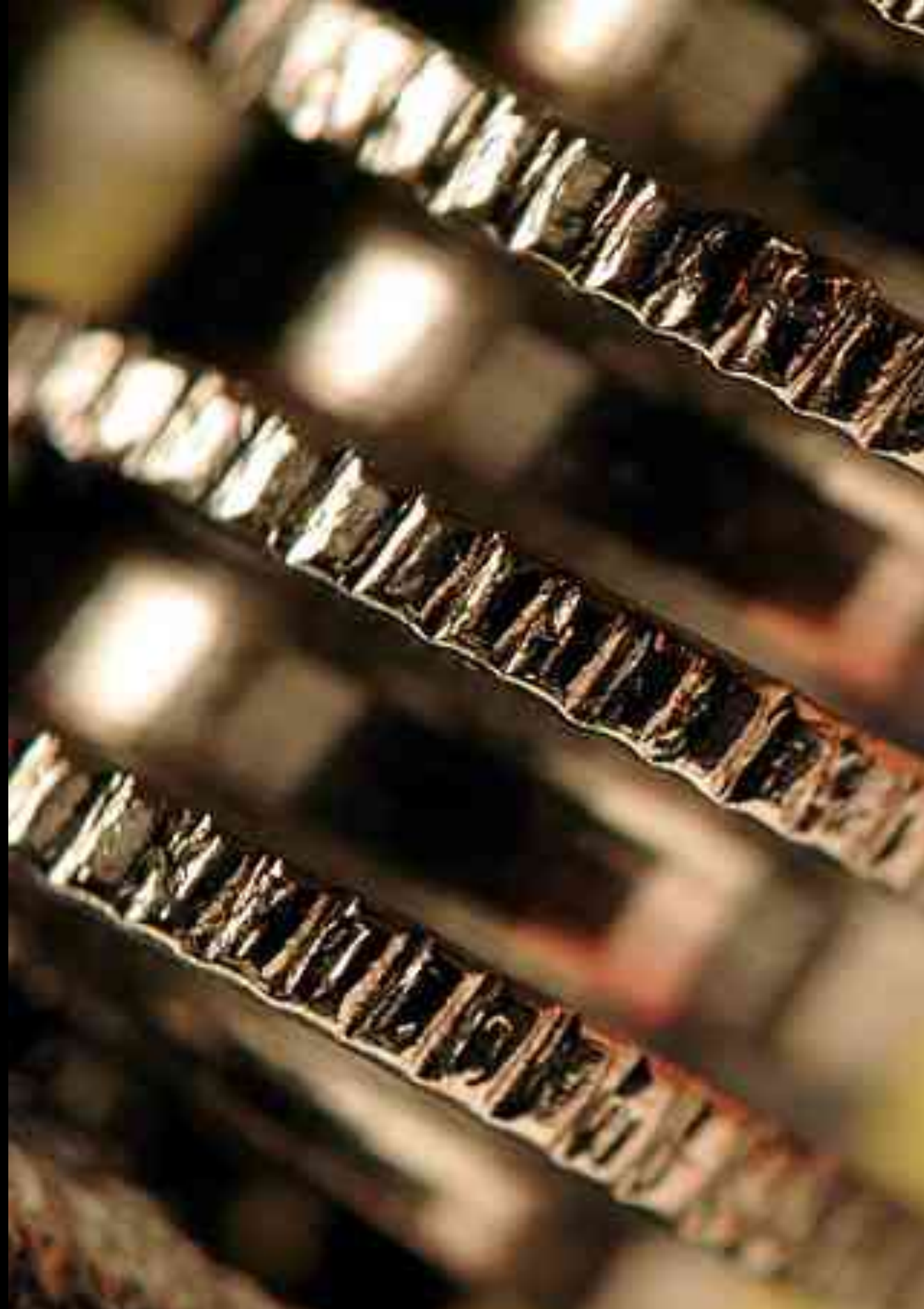
typically consumes PRD and
produces design specs.

engineering

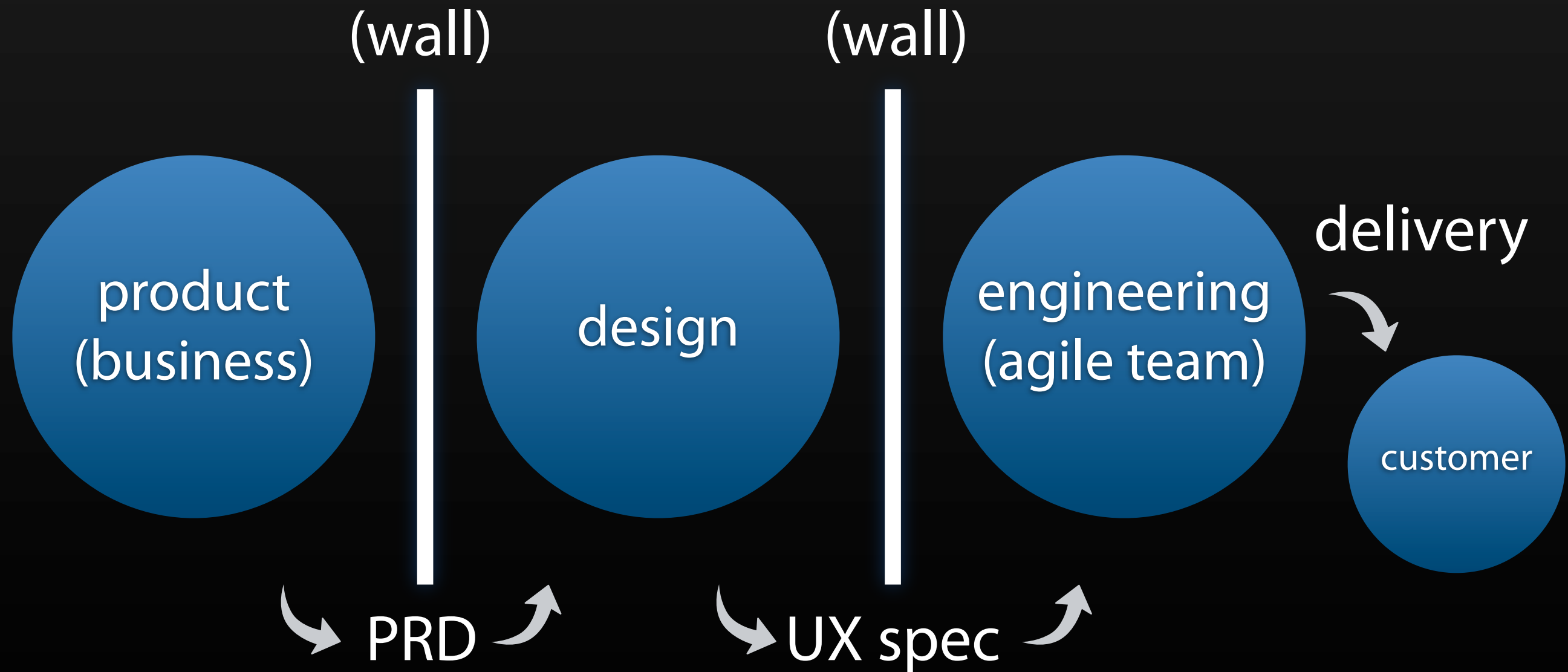
front-end engineer, user
interface engineer, web
developer.

not the designer, but the
engineer that creates the
experience.

typically consumes UI specs
and PRDs (for context).



typical product life cycle



upon delivery, team disbands and forms into new teams

what's wrong with this?

teams work in isolated phases

process & documentation stand in to patch this together

focus is on delivery instead of continuously improving the experience

lean ux

a good portion of the workshop today will be focused on lean ux as a way to

- break down the walls between teams

- continuously learn from users

- work in a highly collaborative manner

rapid experimentation

@netflix

different way of working

only customer is the member (no internal customers)

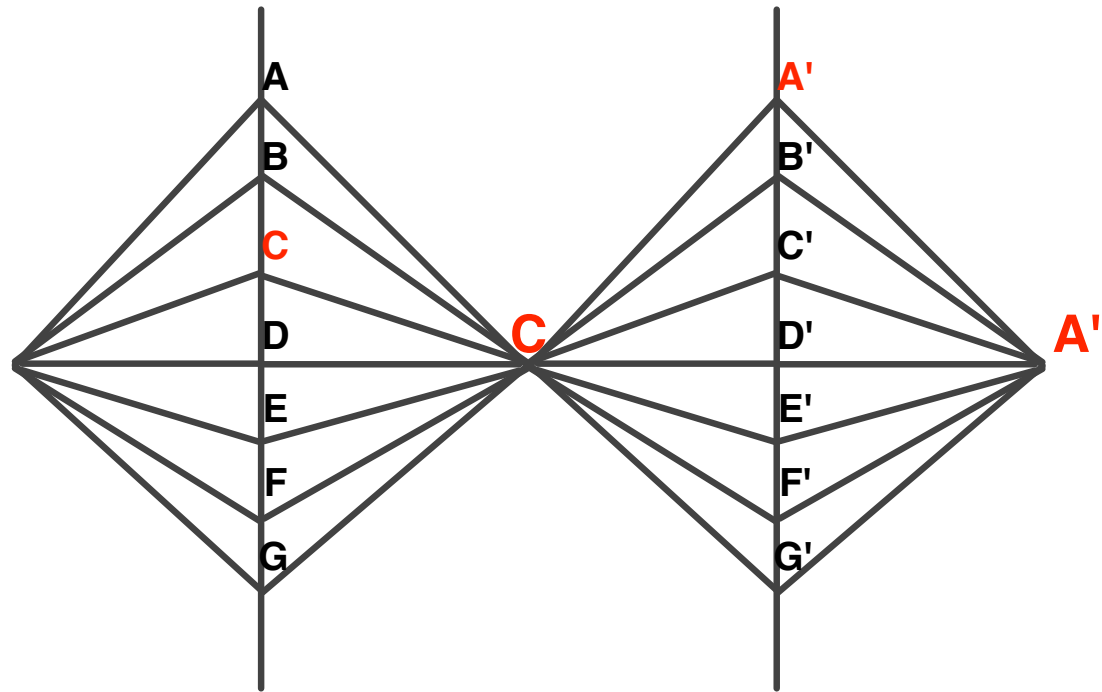
rarely talked about the “machine”. Talked about members.

culture - rapid experimentation vs. long shelf life

get it out live as fast as possible

fail fast/learn fast

don't over think it



At Netflix 90% or more of the UI code was thrown away every year.

Doesn't take too many tests to result in lots of throw away code.

netflix

Follow Build-Test-Learn

Design for volatility

Design for throwaway-ability

intuit before

most sales of TurboTax
happened at tax season. this
led to conservative culture

one major initiative a year.

intuit after

test over 500 different
changes in a 2 1/2
month tax season.

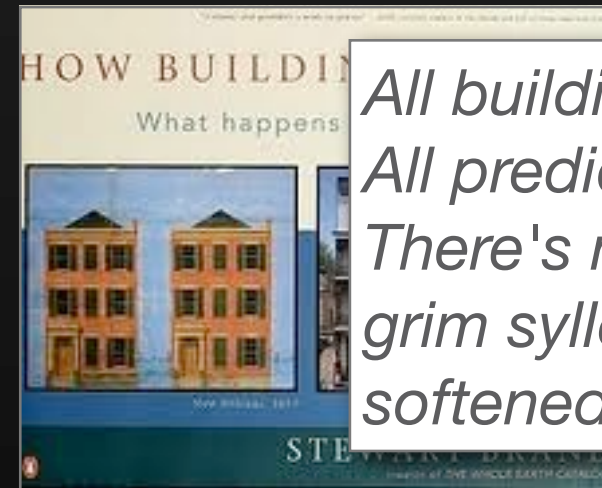
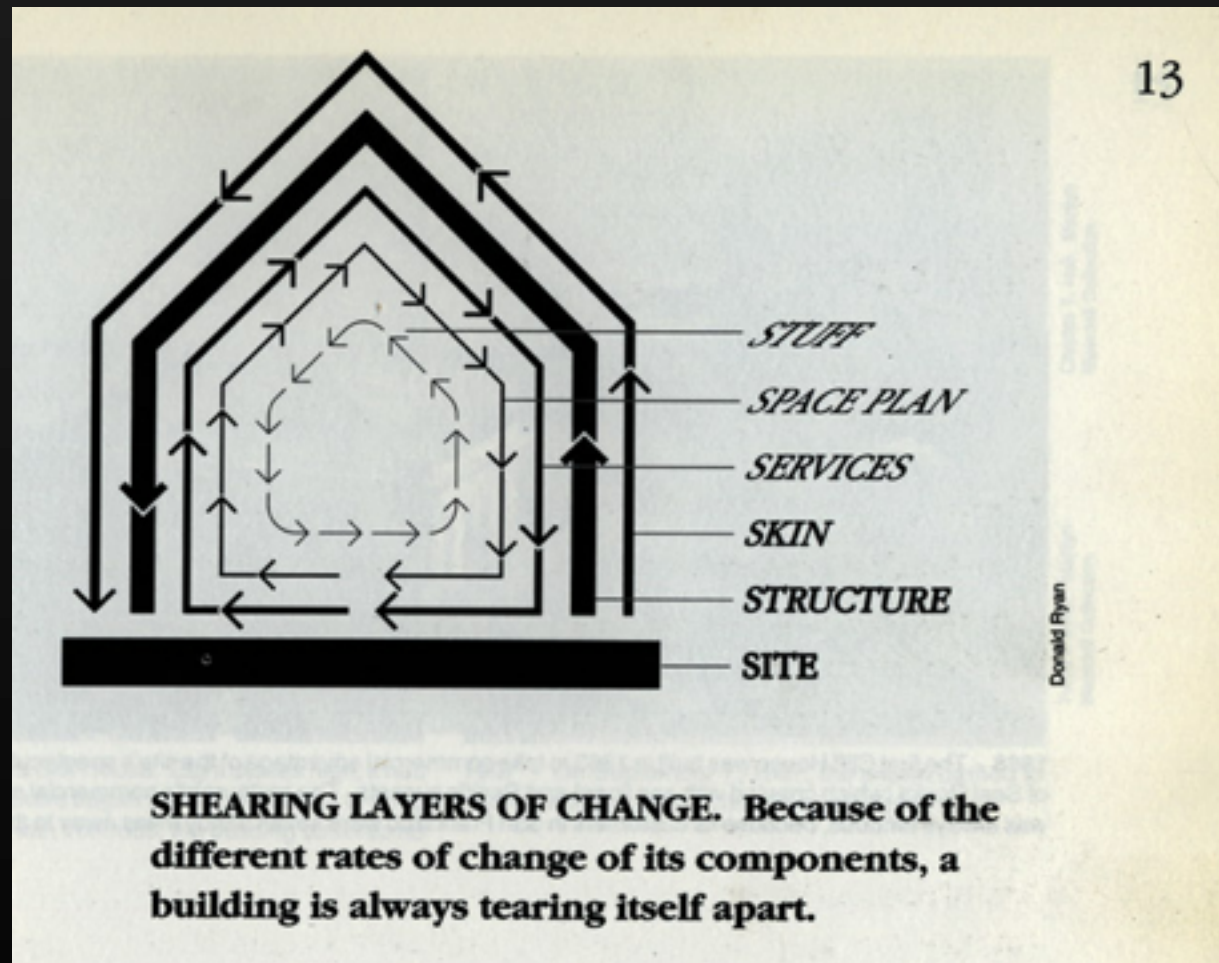
running up to 70 tests
per week.

lessons learned

The background is a composite of abstract light streaks in warm tones (yellow, orange, white) against a dark, almost black, background. These streaks suggest motion and energy. A solid dark horizontal band runs across the middle of the image, serving as a backdrop for the text.

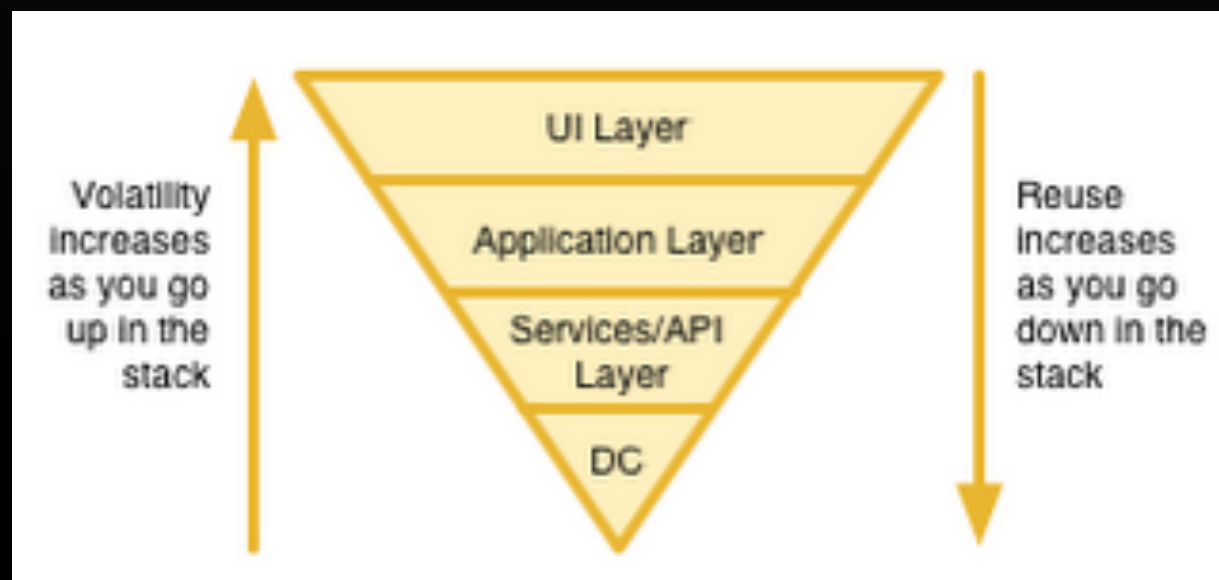
design for volatility

how buildings learn



*All buildings are predictions.
All predictions are wrong.
There's no escape from this
grim syllogism, but it can be
softened. - Stewart Brand*

our software is always
tearing itself apart (or at
least it should)



recognize that different
layers change at different
velocities

velocity changes by layer

recognize that different parts of tech stack change at different velocities

“any building is actually a hierarchy of pieces, each of which inherently changes at different rates” - Stewart Brand. How Buildings Learn.

design for throwaway-ability (volatility)!

“use before you reuse” (optimize for change)

utilize packaging or paths to capture experiments



start with experience

experience vs components

experience vs components

NETFLIX

2 / 332

Search

Instant Queue



Bella

2006 PG-13 1h 31m



Two lost souls -- Nina, a pregnant, unmarried waitress, and Jose, an introspective cook with a tragic past -- find solace in each other as their lives become unpredictably linked throughout the course of one incredible day.

Cast: Eduardo Verástegui, Tammy Blanchard...

Categories: Drama, Indie Dramas

Director: Alejandro Gomez Monteverde

Recently Watched



Emotional Dramas

Watch Instantly

Browse DVDs

Your Queue

★ Suggestions For You

Movies, TV shows, actors, directors, genres 🔍

Genres ▾

New Arrivals

Starz Play

Instantly to your TV

You recently watched:

[See all](#)

Danny Phantom: Ssn 2: Reality Tr ...



Play Next

Heroes: Ssn 1: Homecoming



Resume

2m 43m

Alice in Wonderland



Resume

0m 108m

Bill, rate what you've seen to reveal suggestions just for you

Rate *Heroes:*
Season 1

Haven't Seen It

Suggestion

1

Suggestion

2

Suggestion

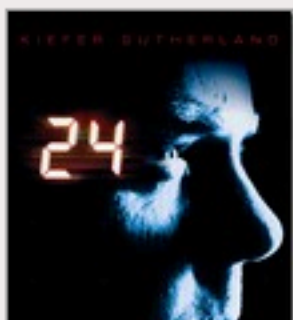
3

Suspenseful Conspiracy Action & Adventure

[See all >](#)Your taste preferences
created this row.Suspenseful
Action & Adventure

As well as your interest in...

24: Season 2



Chain Reaction



Westbound



Boxer's Adventure



why start with experience?

stay honest & pure by having experience be the driver

(not what your boss thinks or what looks good on your resume or what the loudest one in the room thinks)

remember

- use before you reuse

- let the experience drive the engineering

- reuse is an engineering optimization. use is what users do. reuse is what engineers do.

A man with curly hair and glasses, wearing a white lab coat, is smiling and pouring a dark liquid from a flask into a beaker. He is standing behind a wooden table cluttered with various bottles, some labeled 'GeoSalmakki' with numbers 11 and 13, and a small blue penguin toy. The background is a brick wall.

build in rapid experimentation

build in rapid experimentation

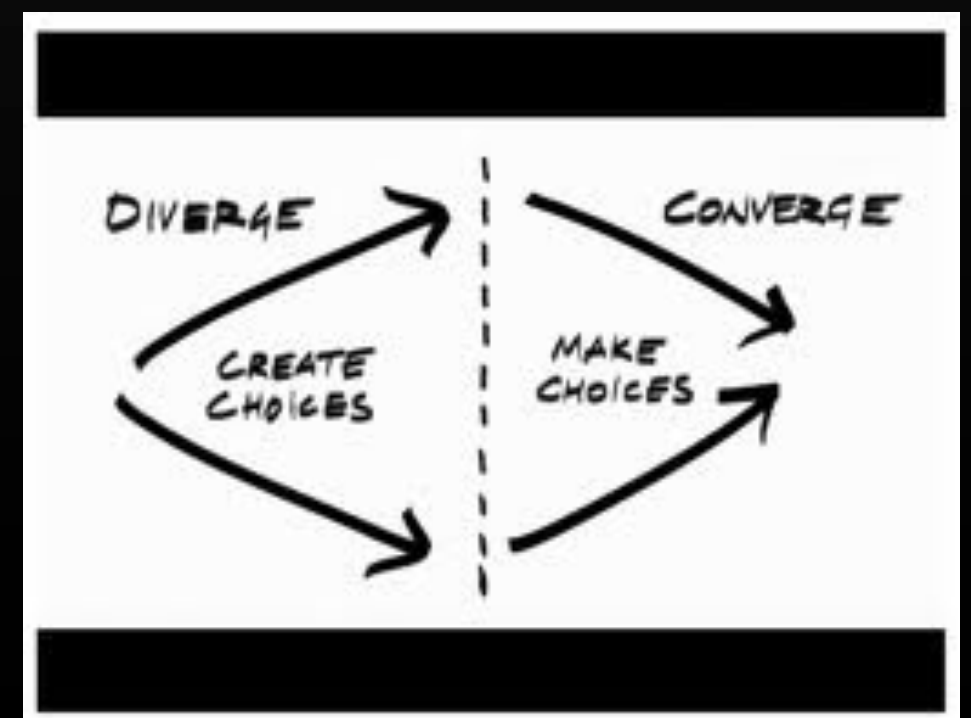
think of the UI layer as “the experimentation layer”

early rapid prototyping leads to learnings to get into the right ballpark

follow with live A/B Testing. Lots of it.

creates a healthy environment with constant customer feedback loops

contrast this with “long shelf life” culture





q & a





break

20 minutes





lean ux



lean manufacturing

comes from the manufacturing revolution

- draws upon the knowledge of individuals

- shrinking of batch sizes

- just in time production and inventory control

- acceleration of cycle times

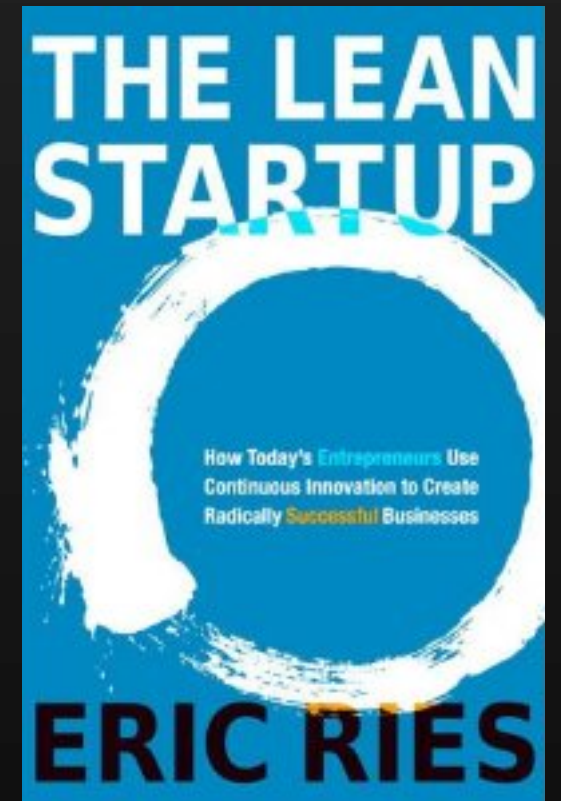
methodologies like kanban which say stories aren't complete until learnings happen

lean startup

built on the principles of lean manufacturing

a startup is a human institution designed to create a new product or service under conditions of uncertainty

developing experimentation systems that allow teams to move at the speed of these systems instead of the speed of “caesar giving a thumbs up or down”



lean startup

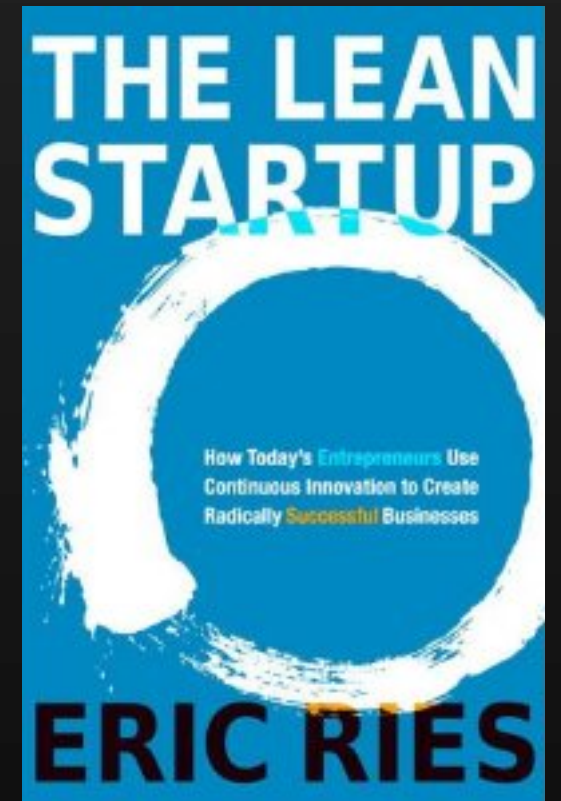
minimum viable product (MVP)

build/test/learn

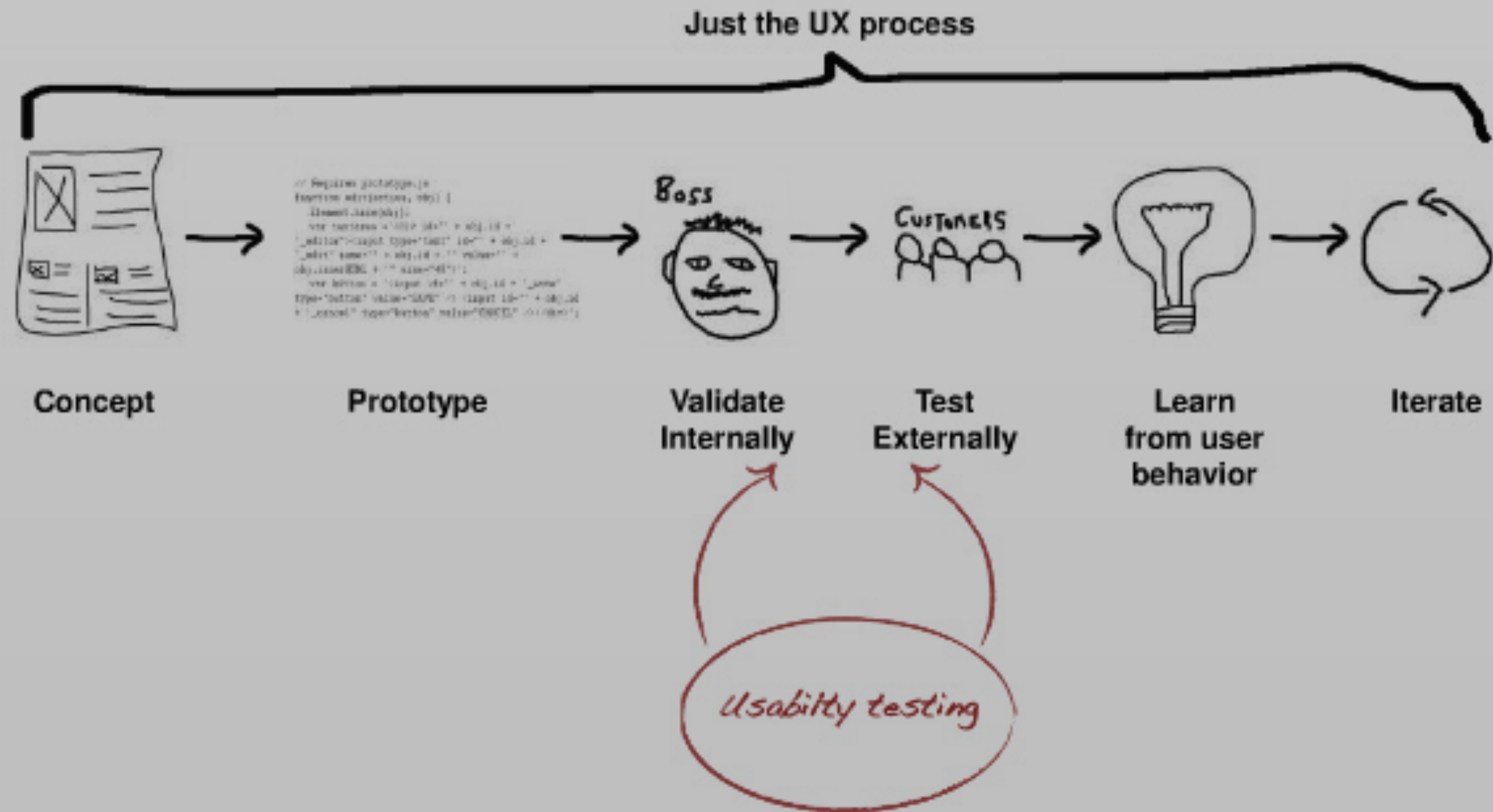
when to pivot (or persevere)

kanban - simple tracking of stories; end state is validated or not

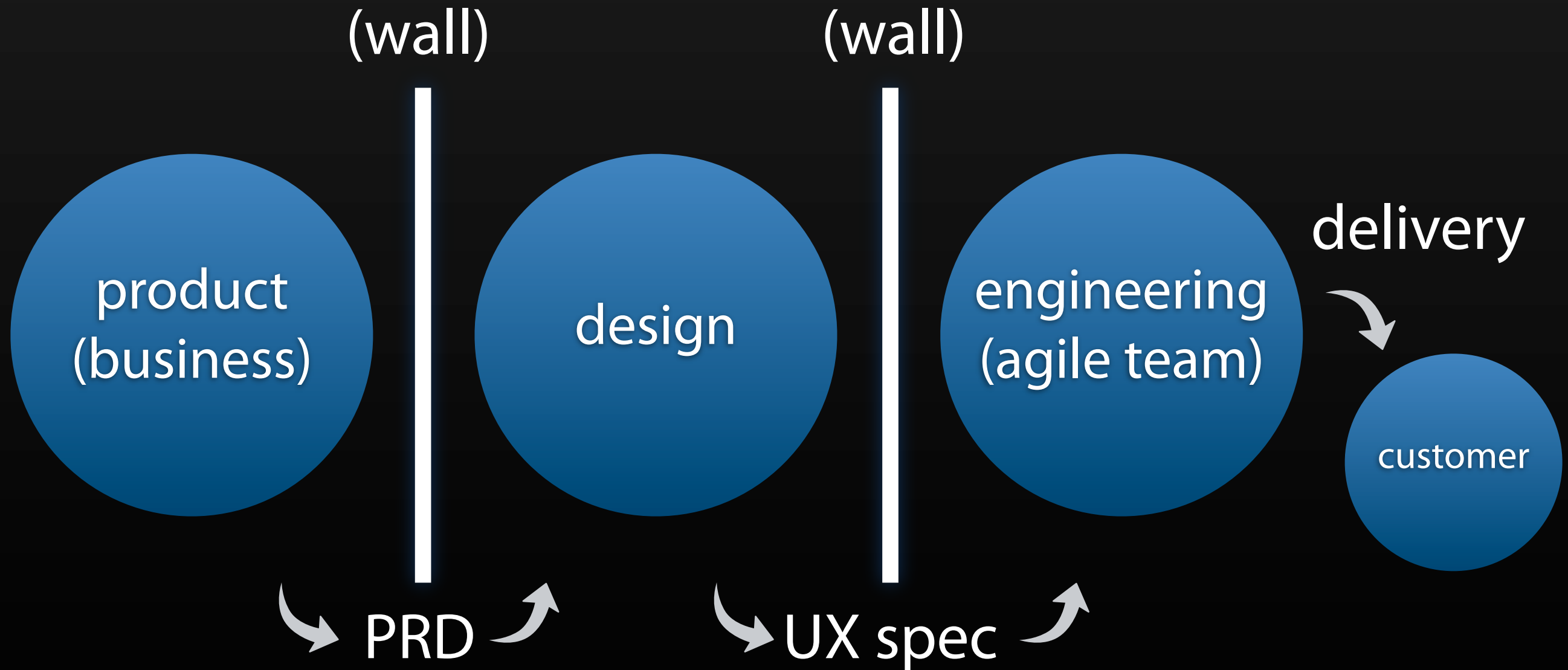
small batches



lean ux?



typical product life cycle



what is wrong with current UX?

became a deliverables-based practice instead of experience-based practice

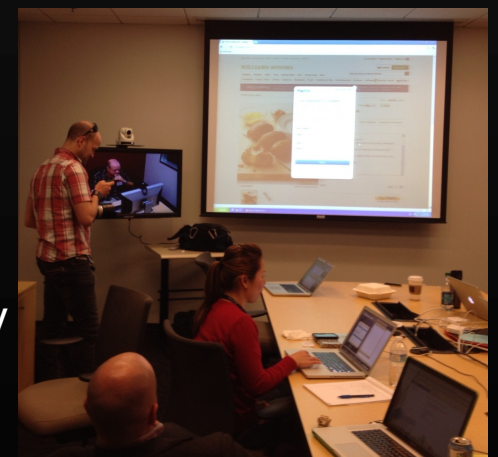
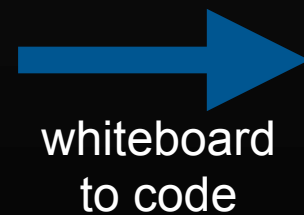
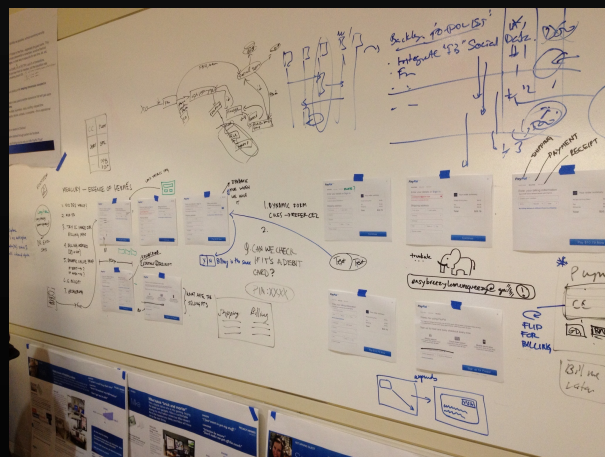
results in tons of waste when coupled with waterfall methodologies

even with agile development, the design process is still waterfall

depends on predictive documentation

lean ux @paypal

PayPal co-located project



product/Design
team

user interface
engineers

usability/
customers

lean ux

minimum viable product (MVP)

build/test/learn

when to pivot (or persevere)

kanban - simple tracking of stories; end state is validated or not

small batches



three key principles

for lean ux

shared understanding

the more understanding the
less documentation

but this doesn't mean NO
documentation

you need whatever is
needed to gain a shared
understanding





deep collaboration

strong belief that ideas come
from many different voices

trust is essential

all efforts never stray far from
collaborative efforts

continuous customer feedback

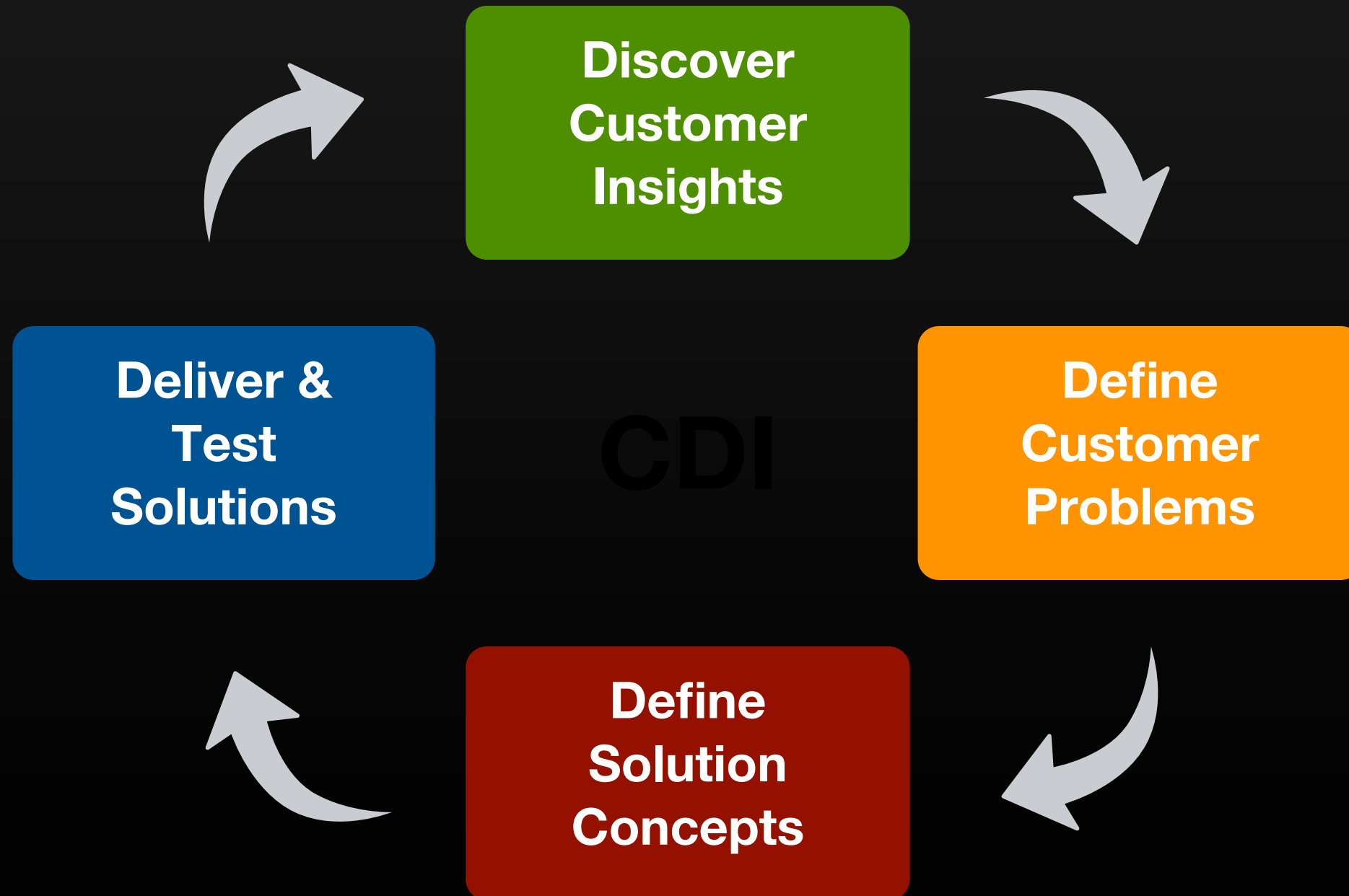
this is the lifeblood of the
team

gets rid of politics

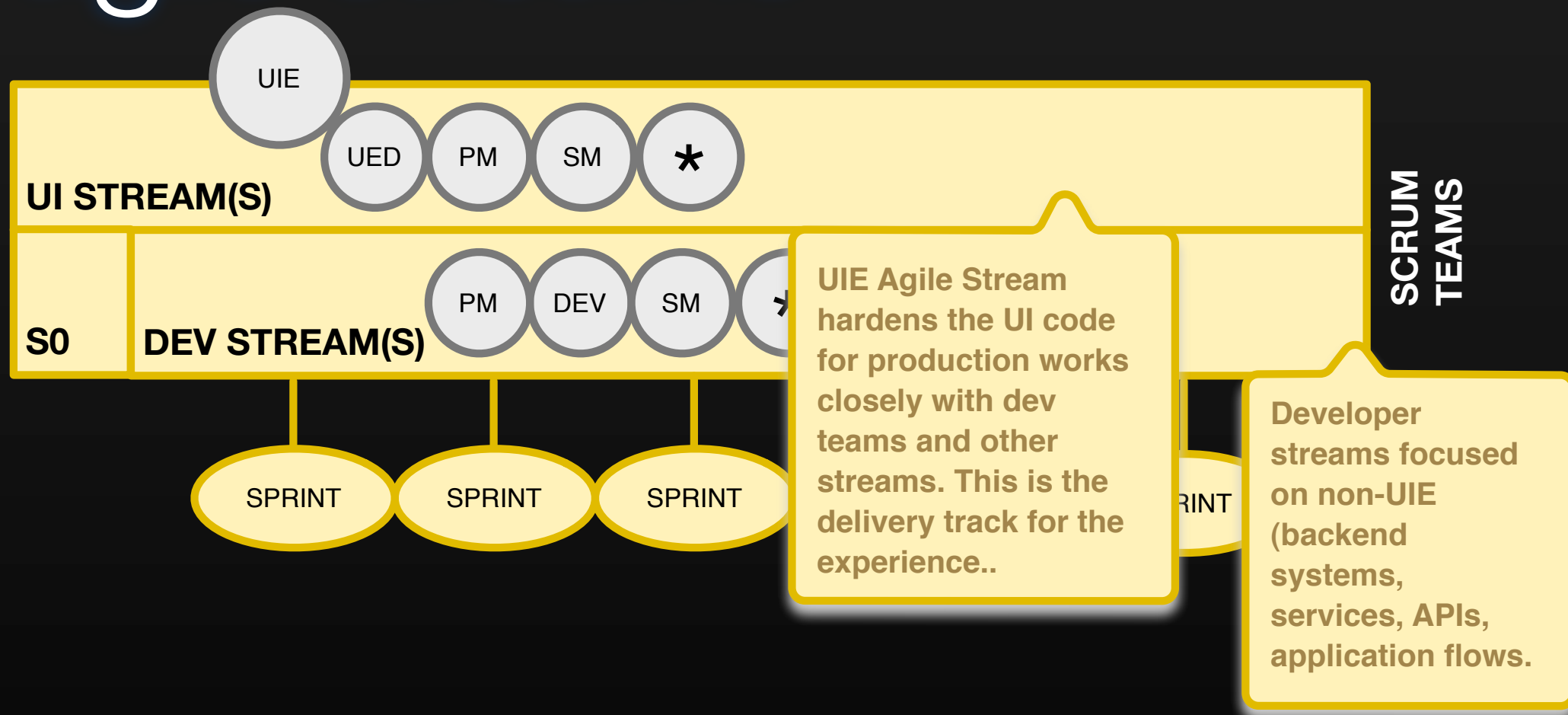
turns a team outside-in



healthy product life cycle



agile streams



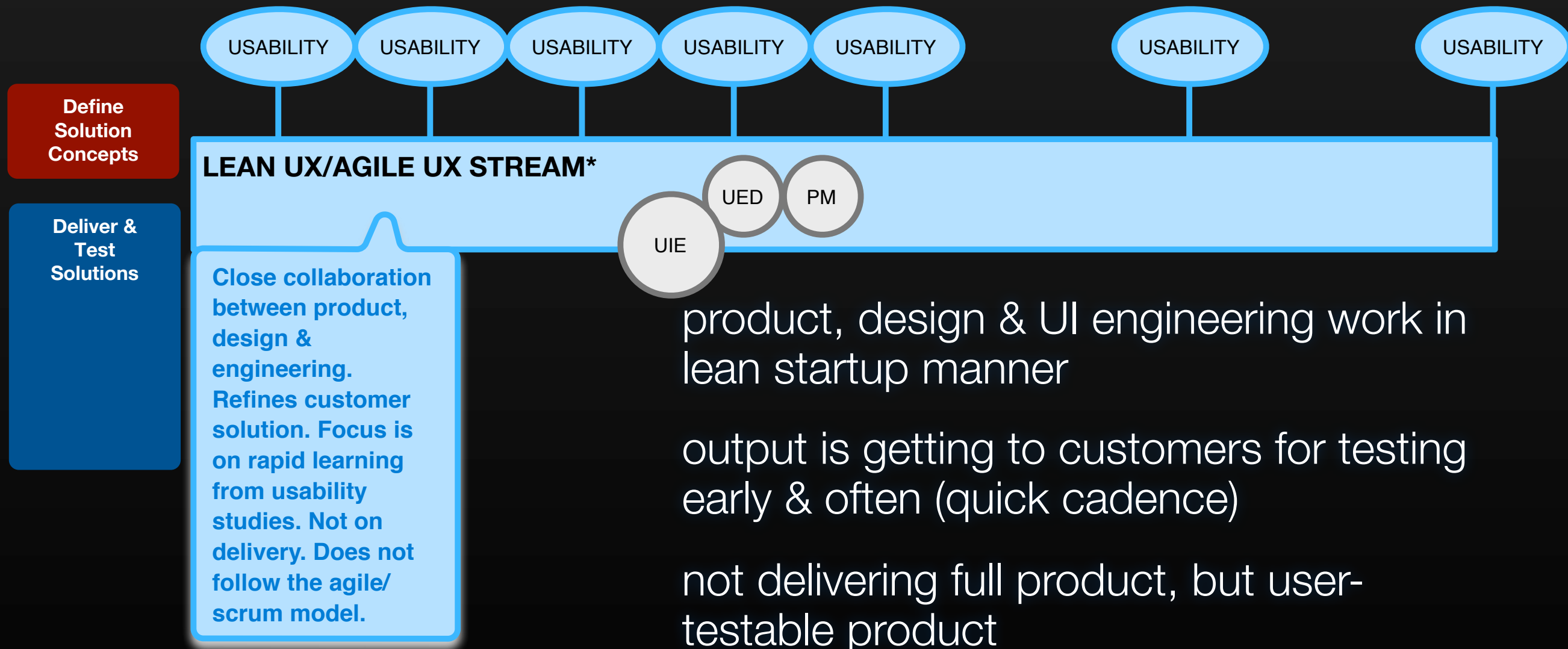
this is the agile process we all know and love

will have UI focused streams

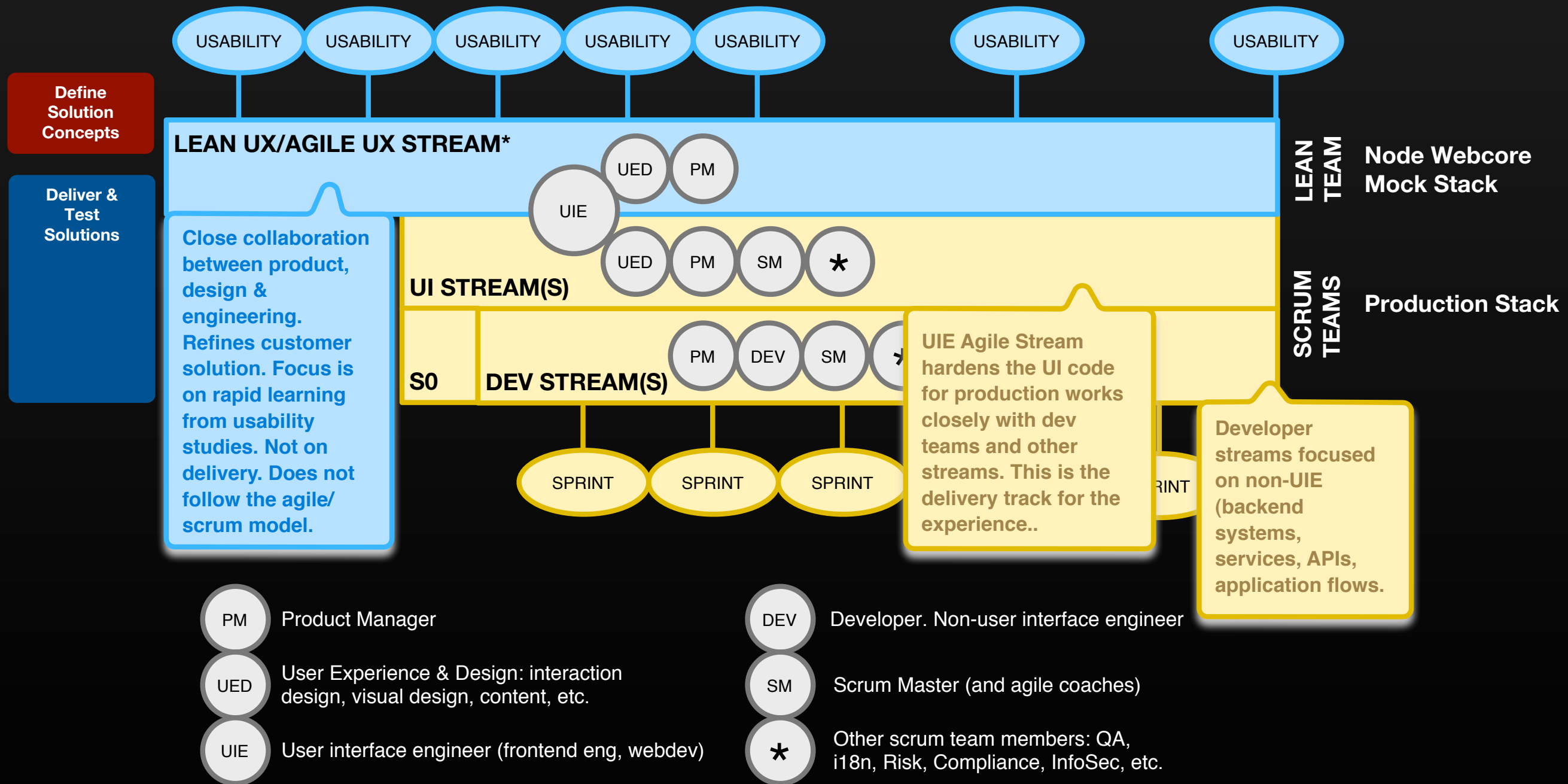
will have some form of sprint 0 (S0) to resolve architecture and starting point stories

fed by the lean track as well as other exercises to create story backlog

lean ux/agile ux



lean & agile





key learnings

takeaways from our lean ux teams

lean is not agile

avoid adopting all of the agile ceremonies in the lean ux track

agile vs lean

agile focuses on engineering delivery

lean focuses on learning

agile contains many “ceremonies”

lean contains few “ceremonies”

agile vs lean

	Agile	lean ux
outcome?	working product at the end of each sprint	user testable experience at the end of each sprint. Refines UI, contracts & data models. Feeds
cadence?	2 or 3 week sprints	starts with 1 week sprints, later moves to less frequent sprints
definition of done?	pass acceptance criteria	learn something from customer
use of stories?	stories feed the agile stream. It is the unit of work	stories are much simpler. solution concepts replace stories
planning tools?	utilizes tools to be able to manage backlog and stories	no need for anything more complex than a simple list
working code?	yes	UI bits: yes dev bits: no. simulated
focus?	engineering delivery	experience learning
ceremonies?	full set of agile ceremonies: scrums, scrum of scrums, backlogs, grooming, planning, retrospectives, t-shirt sizing, etc.	light on ceremonies: emphasis on as little process as possible, but has its own form of backlogs, planning, retrospectives.
relationship to agile		feeds stories into the agile stream. UI agile stream is tightly coupled to lean ux stream. loosely coupled to the dev streams.

co-locate if at all possible

high bandwidth “meatspace” facilitates shared understanding and deep collaboration and time with the customer

suggestions

at a minimum teams should come together for the first few weeks to build shared understanding, deep collaboration and getting feedback from customers

for distributed members use high bandwidth communication where possible (skype, tele-presence)

high bandwidth communication necessary.



github counterpoint

electronic: discussion, planning and operations process should be in high fidelity electronics.

available: work should be visible and expose process. work should have a URL.

asynchronous: almost nothing should require direct interruption.

lock-free: avoid synchronization points.

cooperation without coordination

<http://tomayko.com/writings/adopt-an-open-source-process-constraints>

create a team
working agreement

team working agreement

decide who is the decision maker

define your cadence

define how you will work together

define your hypotheses

* DO NOT ERASE *				
	MONDAY	TUESDAY	WEDNESDAY	THURSDAY
	(all day)	(all day)		
Co-located in ASTUTE	- Design	Design	Feedback +	9am
	- Content	Content	Iterative Coding	Usability Session
	- Iterative Coding	Iterative Coding	-----	12:30pm - 2:30pm Usability Review
		Hand-off	4pm Usability Delivery	3pm What to build next week
* DO NOT ERASE *				
INVOLVED	UGD UG PO			

team working agreement

* DO NOT ERASE *				
MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY
<p>(all day)</p> <ul style="list-style-type: none">- Design- Content- Iterative Coding	<p>(all day)</p> <ul style="list-style-type: none">- Design- Content- Iterative Coding <p>Hand-off</p>	<p>Feedback + Iterative Coding</p> <p>-----</p> <p>4pm Usability Delivery</p>	<p>9am Usability Session</p>	<p>12:30pm - 2:30pm Usability Review</p> <p>-----</p> <p>3pm What to build next week</p>
UED UIG PO	* DO NOT ERASE *	ERASE *		

sprint faster than agile

deliver to customers as often as possible

sprint faster

focus on getting to customer as early and as often as possible

removes the politics in the team as this becomes the arbiter

you can slow down this cadence after you converge on key hypotheses and potential solutions

	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY
	(all day)	(all day)		9am	1pm - 2:30pm
Co-located in ASISTUTE	- Design - Content - Iterative Coding	- Design - Content - Iterative Coding	Feedback + Iterative Coding	Usability Session	Usability Review
			4pm Usability Delivery		3pm What to build

sketch to code

sketch to code

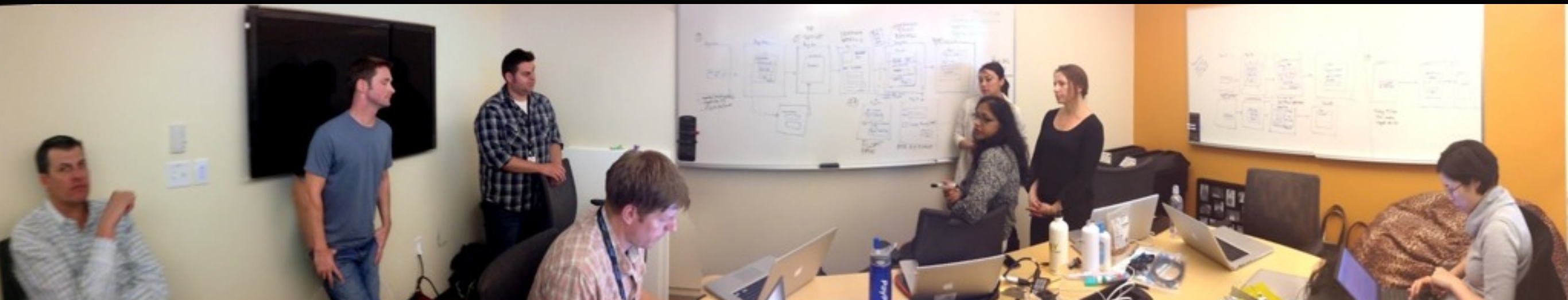
this is a forcing function.

it brings about a close collaboration between engineering and design

it creates a bridge for shared understanding

requires a lot of confidence and transparency

we will discuss rapid prototyping later



make the spec real

the prototype becomes the spec

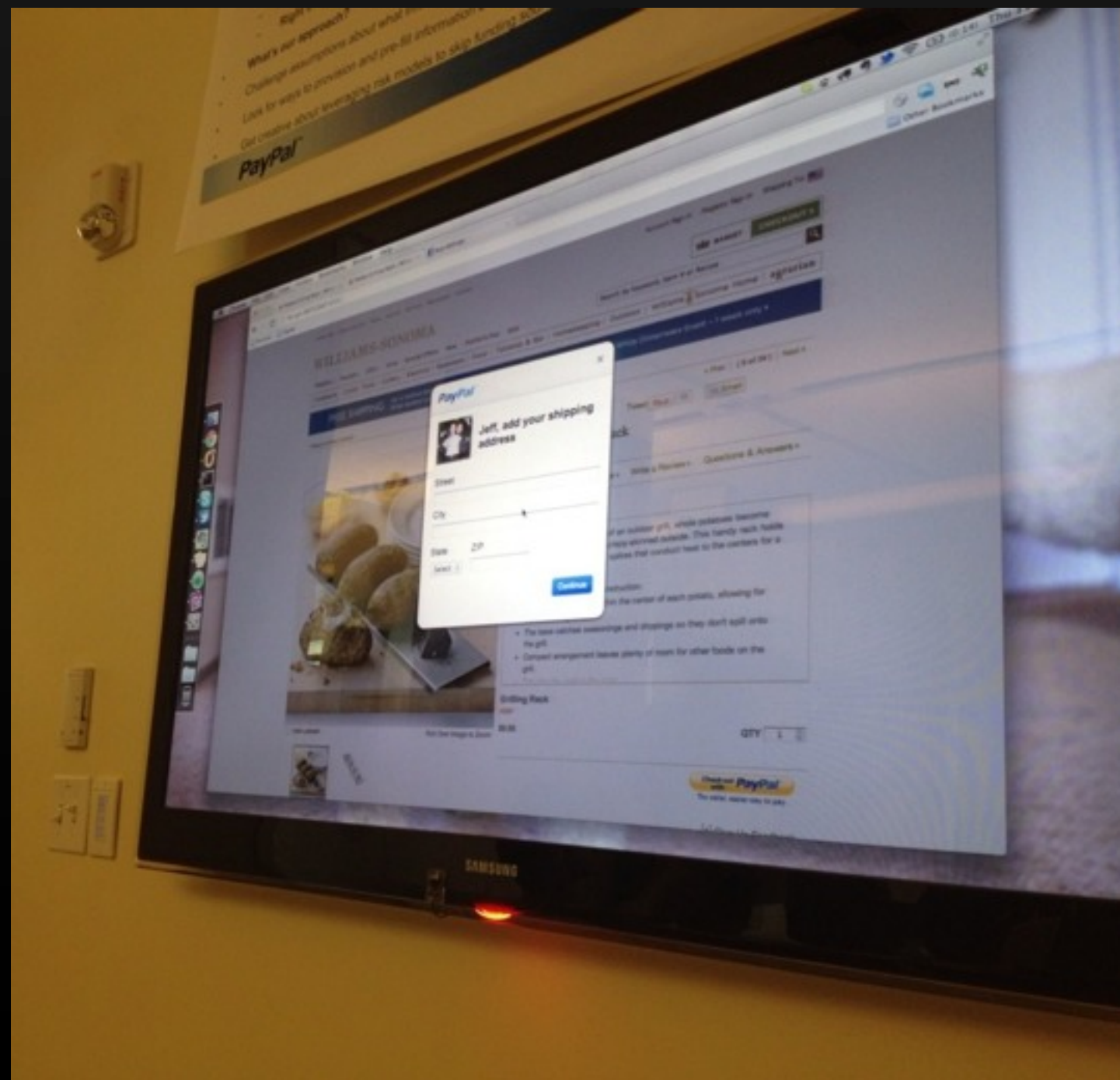
make the spec real

there are many, many prototyping tools available now

you can create a living spec with these

however the fidelity is never the same as real code

recommend HTML prototyping
(more on this later)



but what about docs?

watch out for “predictive documentation”

watch out for documentation that replaces collaboration or is a band-aid for bad process

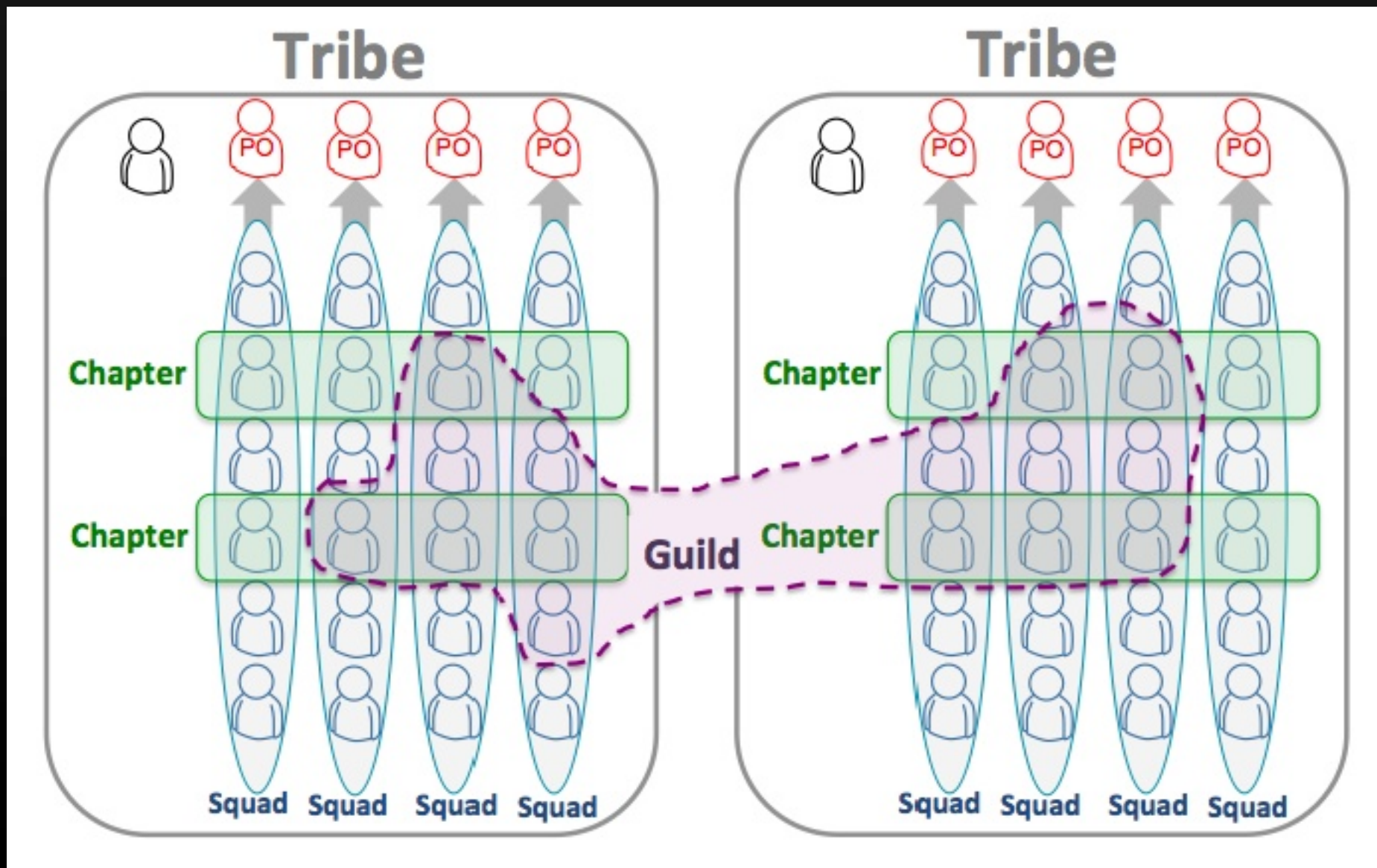
good documentation will enhance collaboration, shared understanding and disseminate learnings



example: spotify

example: spotify

squads run like lean startups



spotify: squad

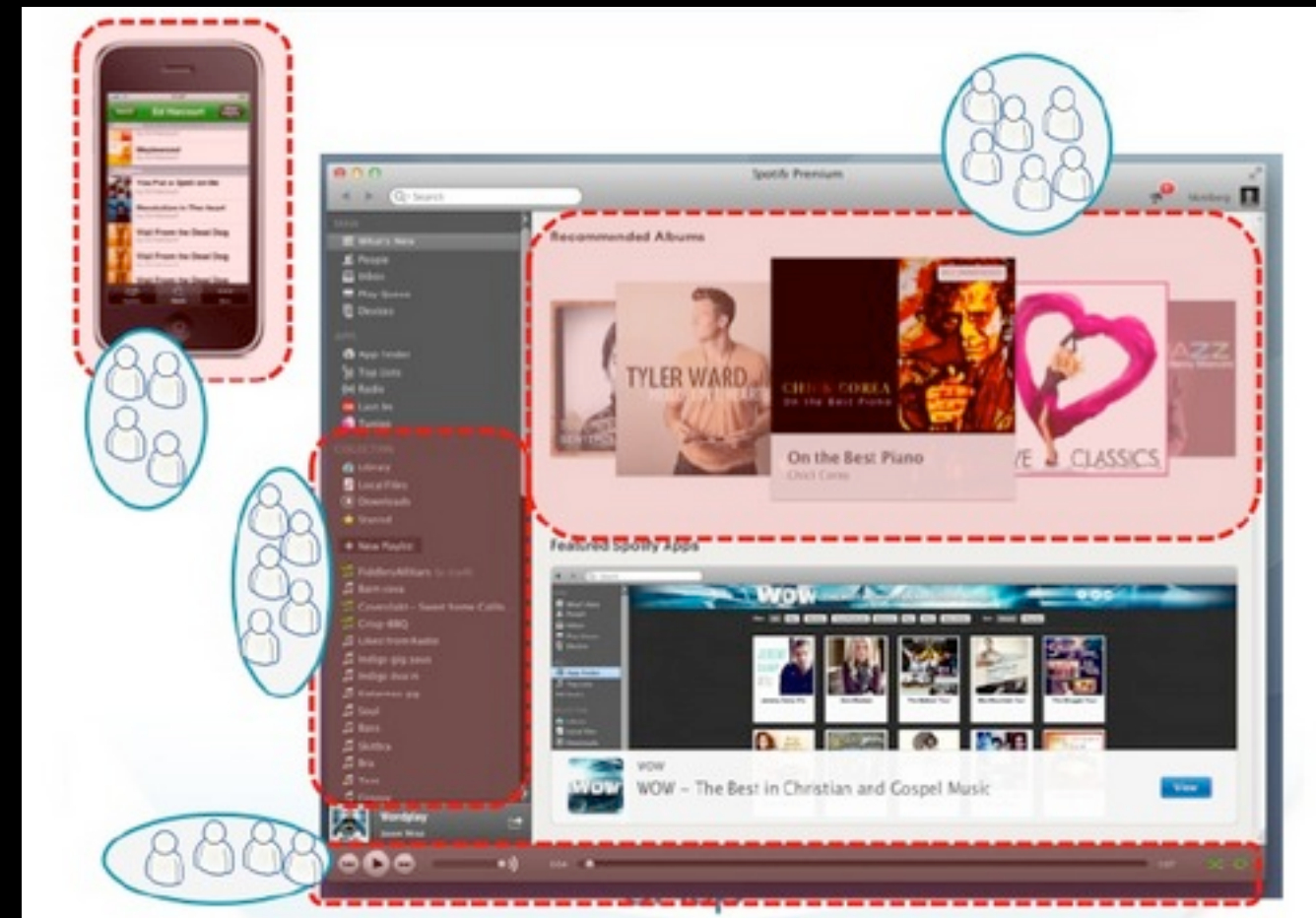
similar to scrum team. feels like startup

long term mission: build & improve the product. stay long term on the product.

apply lean startup principles like MVP

“think it, build it, ship it, tweak it”

emphasis on great workspace

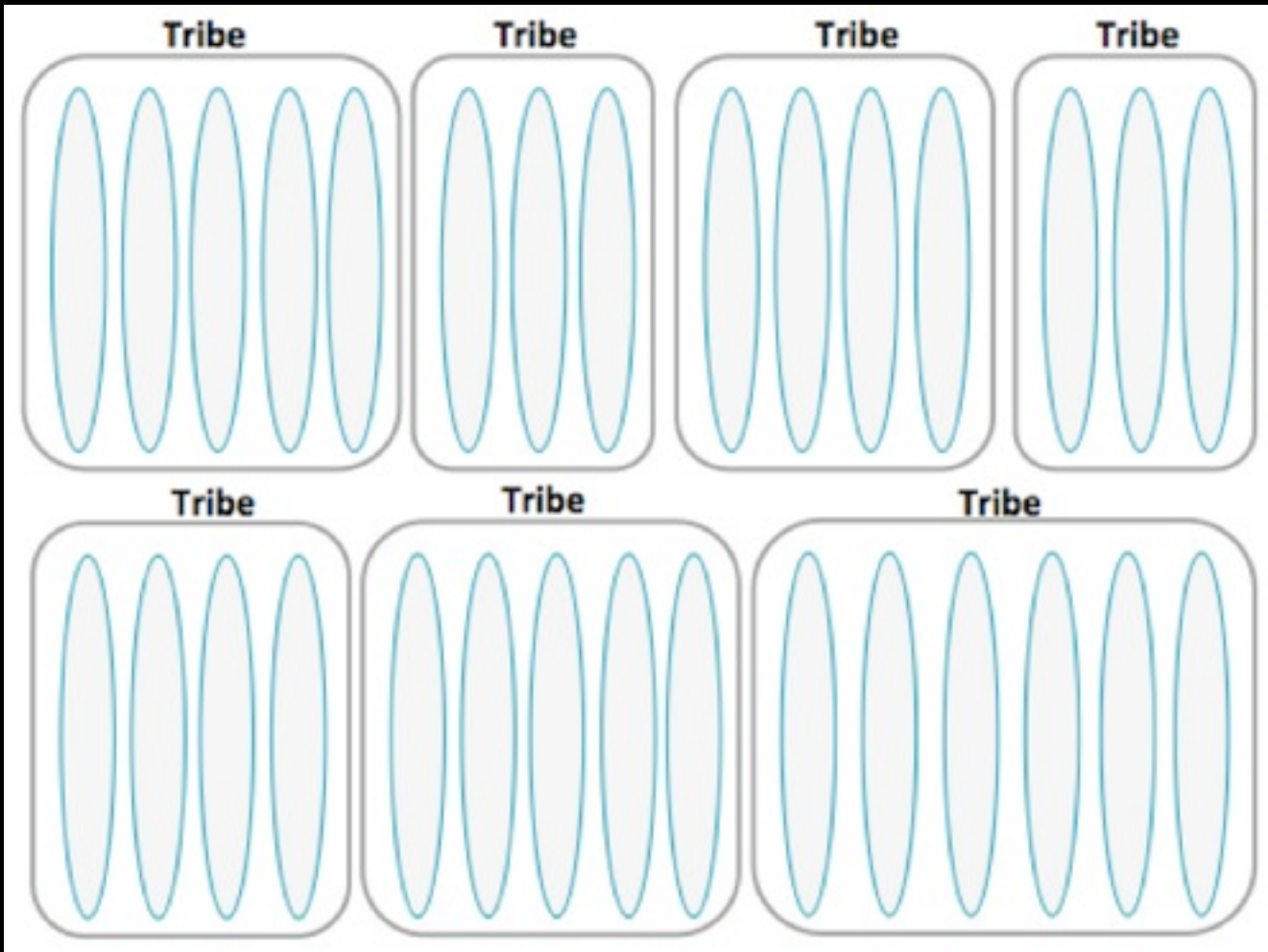


spotify: tribes

collection of squads that
work in a related area

incubators for tribes

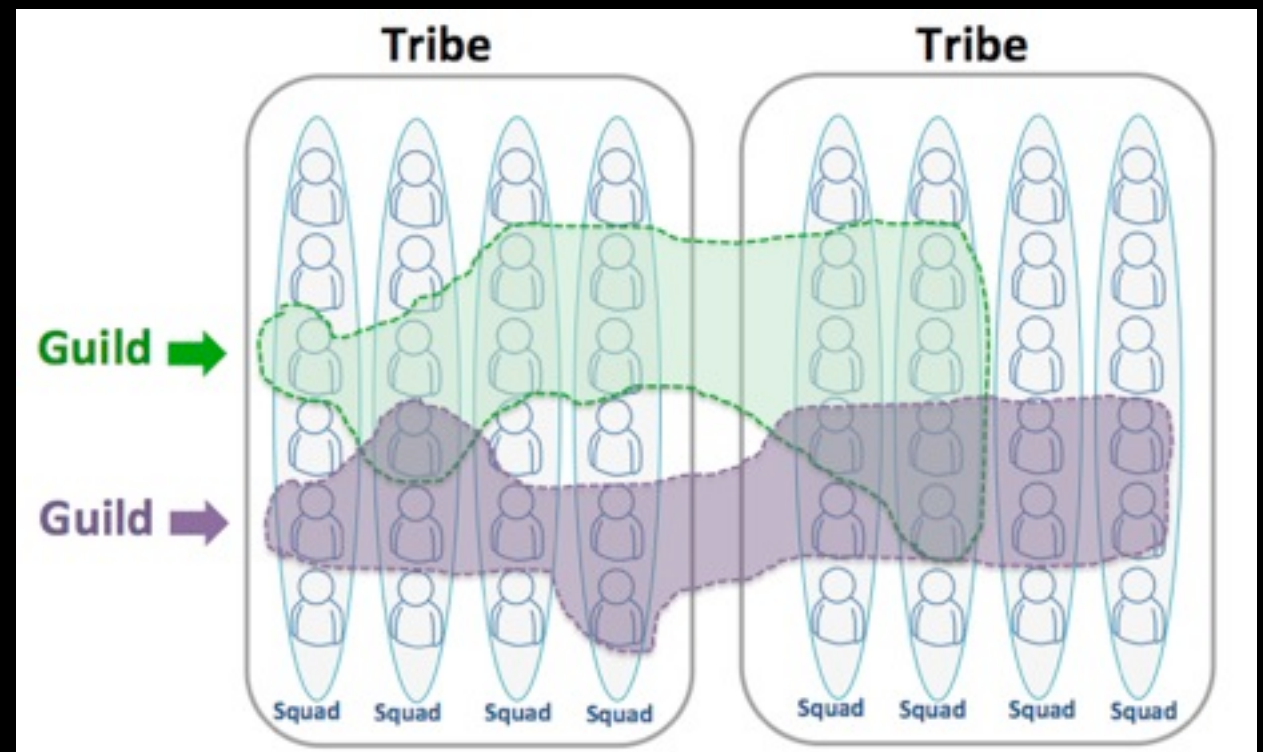
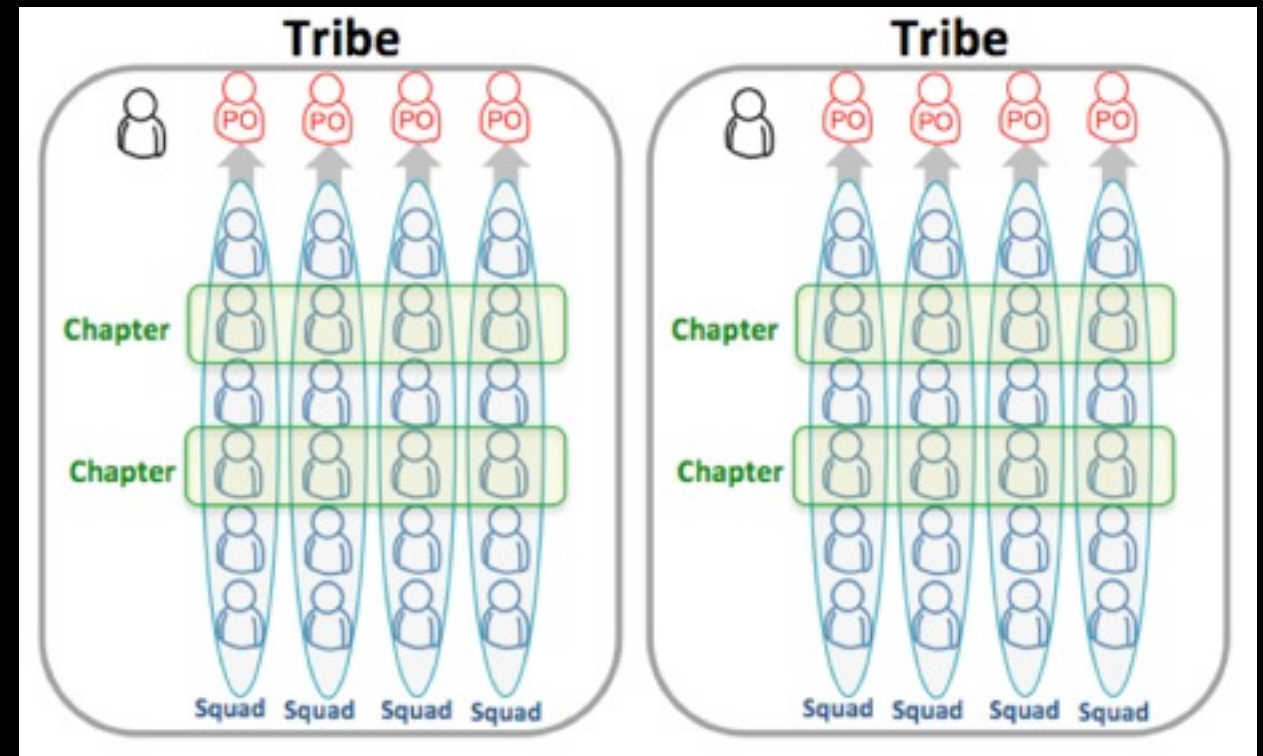
hold regular gatherings



spotify: chapters and guilds

chapters represent horizontal practices within a tribe

guilds represent horizontal practices across tribes



more Info

- Jeff Gothelf - the lean ux advocate
<http://www.jeffgothelf.com/blog/>
- lean ux article
<http://uxdesign.smashingmagazine.com/2011/03/07/lean-ux-getting-out-of-the-deliverables-business/>
- article I wrote back in 2010 on principle of shared understanding
<http://52weeksofux.com/post/2403607066/building-a-shared-understanding>



lunch break





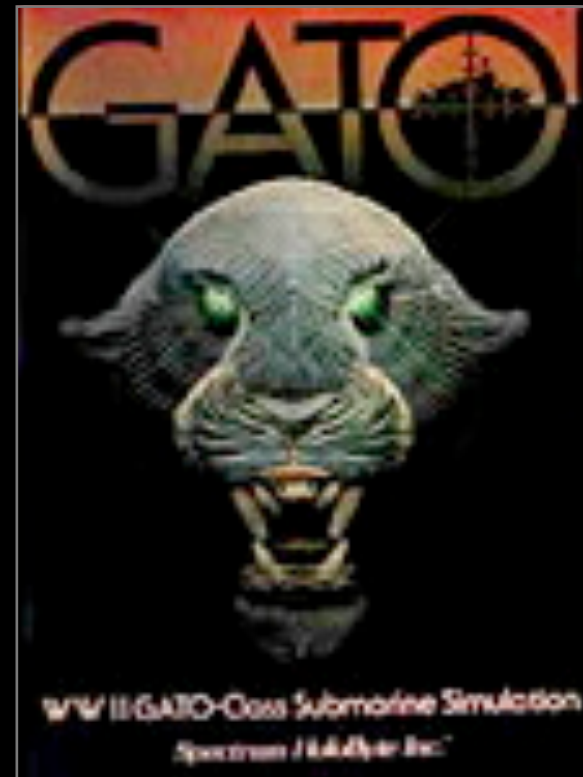
lean engineering

enabling lean ux through the technology stack



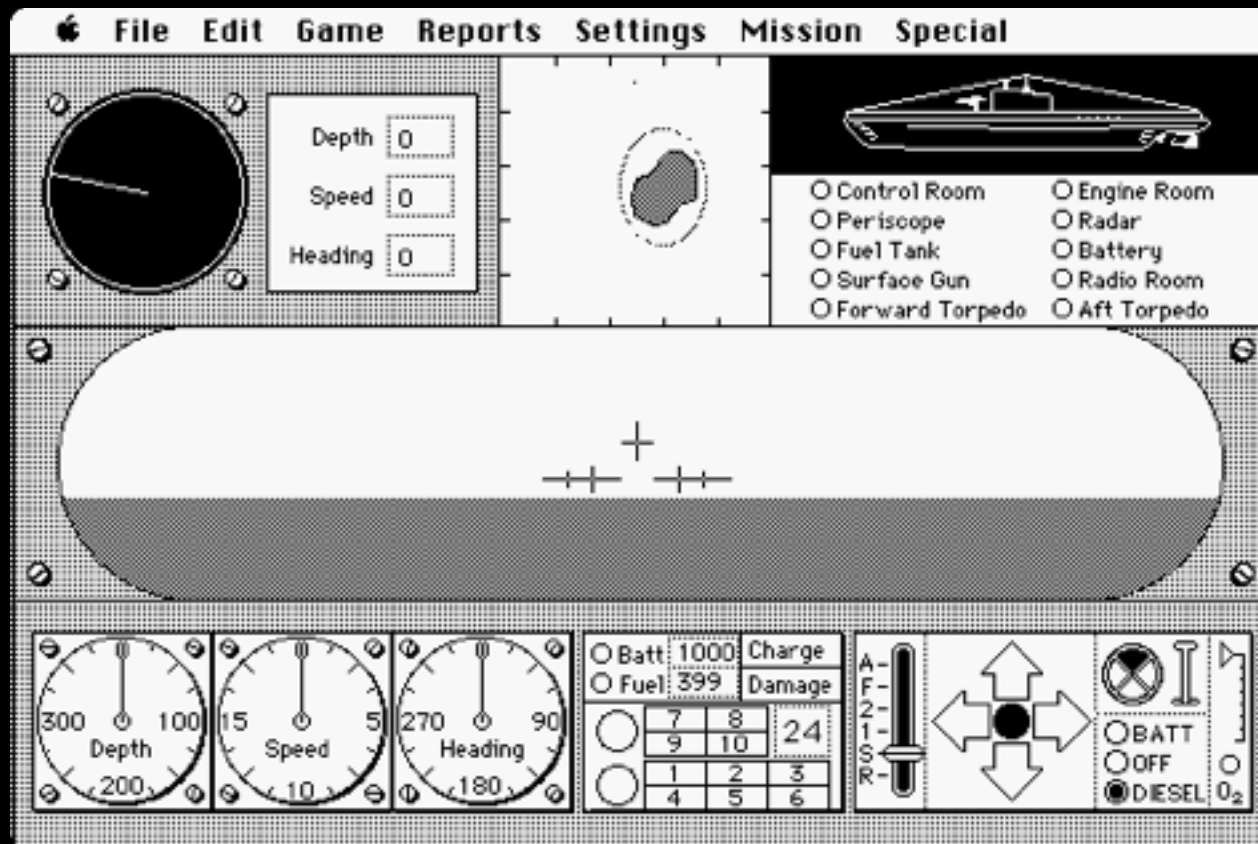
the way it was...

my journey through software development



building a game circa 1985

original Mac Quickdraw Toolkit provided some GUI framework pieces (like the Open File Box)



this bit of “path of least resistance” was a powerful boost to consistency and creating nice looking Macintosh UIs

but there was still a lot left to create on your own

developing a UI was hard



no internet. open source was practically non-existent

hard to build (all native + assembly code)

long shelf life (long release cycles with 3.5" disk deployment)

developing a UI was hard

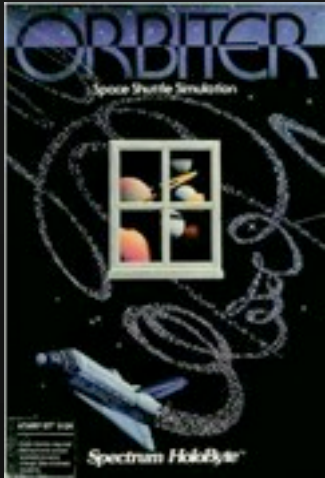


no internet. open source was practically non-existent

hard to build (all native + assembly code)

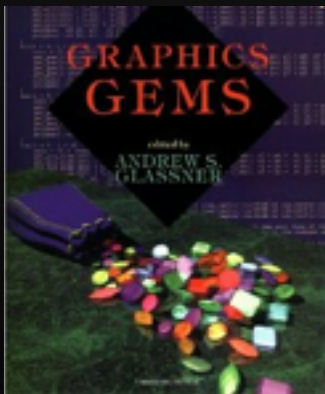
long shelf life (long release cycles with 3.5" disk deployment)

'85 - '05. proprietary



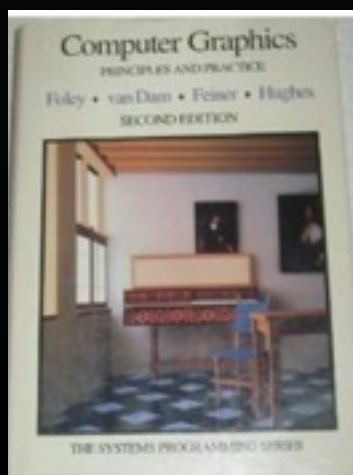
orbiter 3D graphics library

ESYView. Wargame simulator & briefing tool. Wrote everything from the ground up



C++ frameworks

tcl/tk frameworks



multiple JSP frameworks

open source



rico. one of the early ajax/js frameworks (2005)



launched yahoo! design pattern library (2006)



worked closely with yui team (and built first carousel)

moving to a lean tech stack

using open source at paypal

Bootstrap, from Twitter

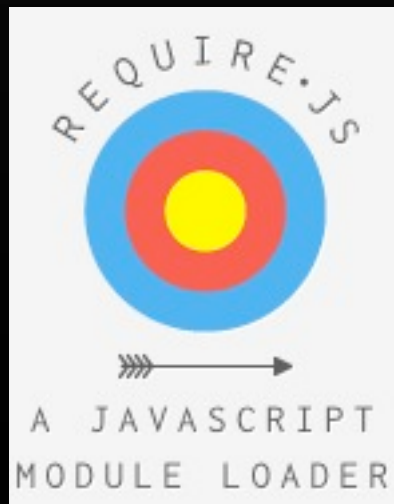


BACKBONE.JS

UNDERSCORE.JS



{dust}



jQuery
mobile framework

jQuery
user interface

working in open source model

internal github revolutionized our internal development

rapidly replaced centralized component/platform teams with de-centralized, distributed, social coding style of development. innovation democratized.

every developer encouraged to experiment and generate repos to share as well as to fork/pull request

giving back to open source

we have a string of projects that will be coming to external github

- node bootstrap (similar to yeoman)

- contributions to bootstrap (for accessibility)

- contributions to bootstrap (for internationalization)

- component repository framework for github (similar to bower)

- and more...

we made our ui bits portable

the ui bits can be delivered continuously

the ui bits can be run in client or server

the ui bits can be deployed on cdn

Lean UI Stack

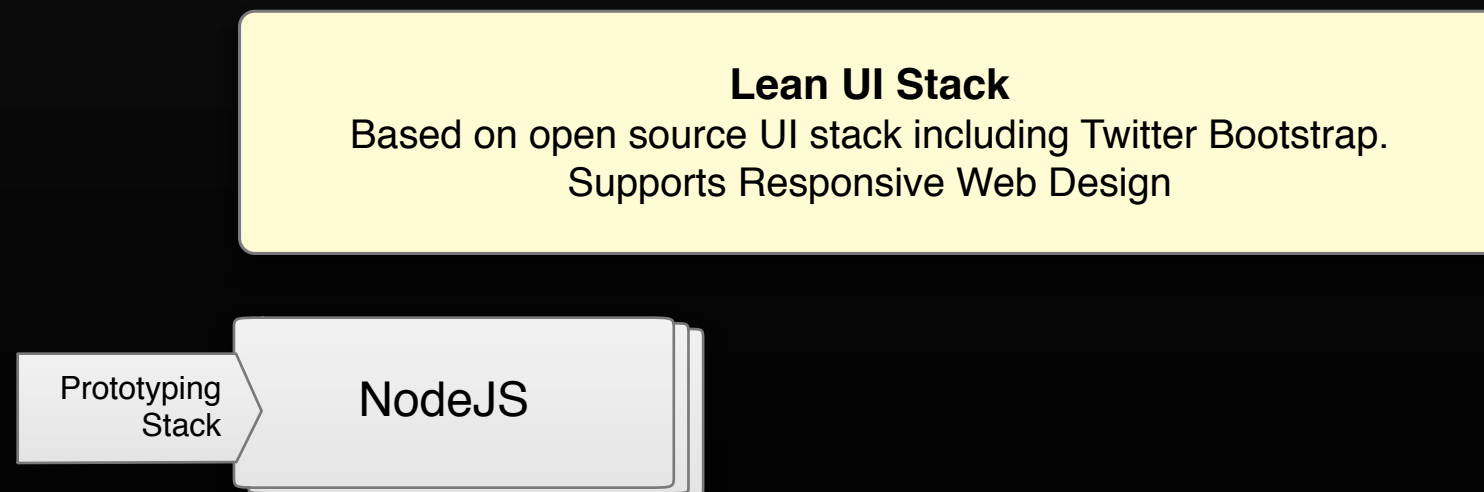
Based on open source UI stack including Twitter Bootstrap.
Supports Responsive Web Design

we made our ui bits portable

the ui bits can be delivered continuously

the ui bits can be run in client or server

the ui bits can be deployed on cdn

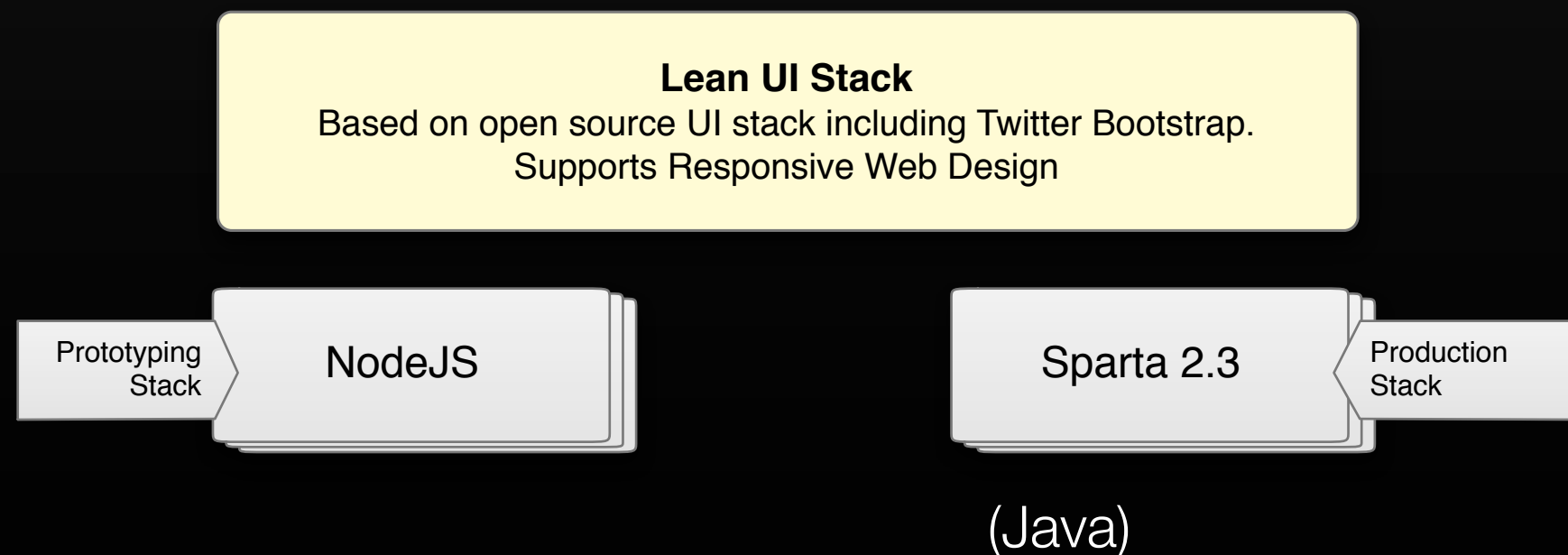


we made our ui bits portable

the ui bits can be delivered continuously

the ui bits can be run in client or server

the ui bits can be deployed on cdn

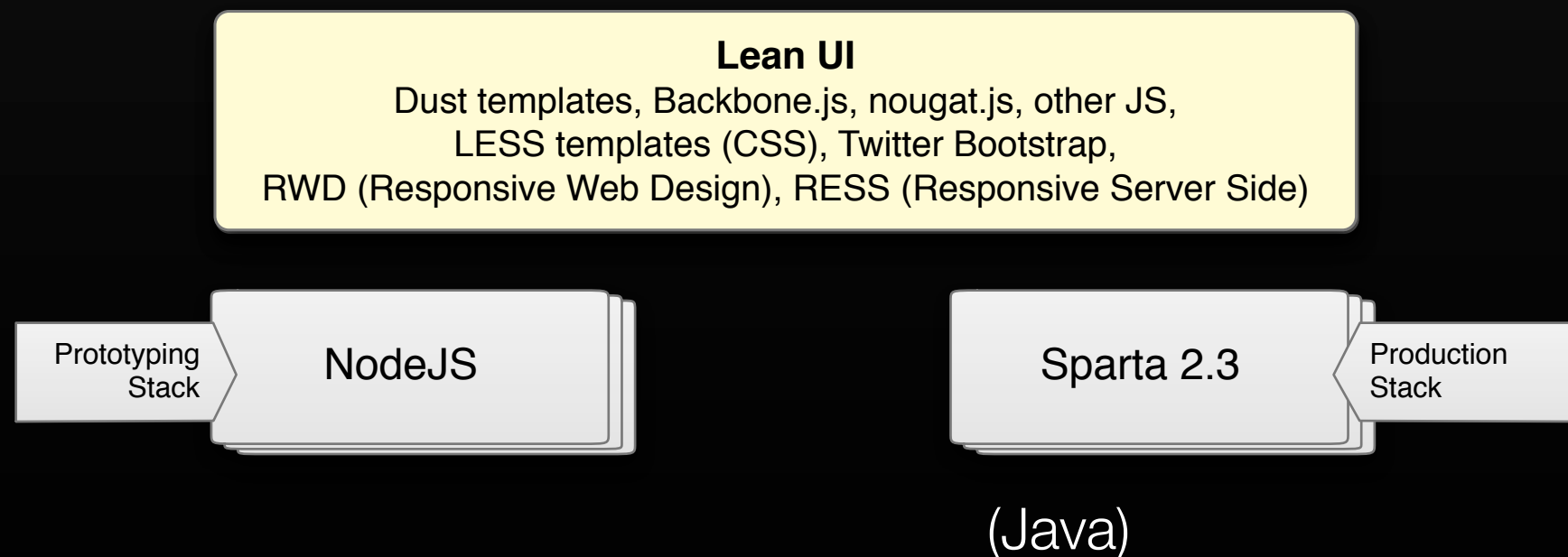


we made our ui bits portable

the ui bits can be delivered continuously

the ui bits can be run in client or server

the ui bits can be deployed on cdn



requirements for lean Stack

independent of the backend language

flexible enough to run in either the server or in the client

equally good at building web sites as it is building web applications

pushable outside of the application stack (publish model)

cleanly separated from the backend/app code (ideally by JSON)

utilize what is common to developers

quick & easy to build & tear down components



rapid prototyping

what to keep in mind for prototyping

use html prototyping

html vs prototype tools

why html5/css3/js?

with kits like twitter bootstrap or jetstrap it is really easy to create a nice looking UI really fast

right fidelity for user testing

can keep the UI bits for production

is a forcing function to get front end engineers and designer to collaborate

suggestion:

twitter bootstrap (and jetstrap)

prototyping tools

see:

list of prototyping tools on my blog: <http://bit.ly/SfWygk>

two that we also use:

Axure RP

InVision

use chrome

chrome is your os

don't initially worry about cross-browser

though don't be stupid

chrome is most consistent, compliant browser across
windows and mac

great developer eco-system

use node.js for app prototyping

rich eco-system

multitudes of npm modules available

recommend express for routing, asynch for simplifying event'ed programming

something like require.js for modularization of code

node webcore

discuss our use of node.js, nodewebcore and how our UI bits stay the same on the proto stack as well as the production stack

even better if you can have node as your runtime

yeoman?

use js templating

not just any templating, JS templating

why?

makes all of your rendering bits be javascript

can run the bits on the client or the server (node.js or rhinoscript, etc.)

can treat UI bits as CDN assets

templating in general makes life much simpler for rendering UI bits

forces clean separation of view & model

use the power of css

suggestions

don't hack your prototype (keep your css clean)

start with something like LESS

if you are using bootstrap that fits nicely and you can change stylesheet for your company's visual L&F

gives you the power of variables and mixins

use CSS animations and just let the other browsers degrade to lesser experience

there are also gradient generators for cross browser

(<http://www.colorzilla.com/gradient-editor/>)

apis are your friend

apis

public APIs can bootstrap you

node is a great way to mock APIs as well

can prototype the apis as spec for the dev team

json is the bridge

duh. what else would you use?

use the cloud

use the cloud

get your code live on a URL as fast as possible

pick a solution that uses hosted node.js

component libraries

components

twitter bootstrap

twitter bower is a good example of managing components via github

(we will be releasing an simpler open source library called garnish that is similar to bower)

other suggestions

other suggestions

resist the desire to “beautify” the design. Focus on learning

use a tool like Asana for collaboration

test on real devices early & often

use github

use continous deployment



UI Architecture Concerns

client vs server

which is better?

which is better?

sage wisdom: **“It depends”**

for page-to-page, server side rendering is usually simplest

for app (like in mobile), client rendering often works best

but often it is a blend of the two (utilizing ajax on client side to fetch server-side rendered html)

best solution is to be able to support either via rendering portability

twitter #fail

recall twitter had to back away from their “all client side rendering approach”

in reality this was a very naive and bad idea

- stopped the double rendering of !#

- removed javascript execution on client and switched to server side

- loading only what is needed (using commonjs modules)

<http://engineering.twitter.com/2012/05/improving-performance-on-twittercom.html>

application vs page

application vs page

one size doesn't fit all

but watch out for mobile, it is often a mistake to do
page to page experience

exception is sometimes managing a flow

native vs web

native vs web

native is great in low latency and for really high fidelity touch experiences. especially if there is a lot to scroll.

however you lose: rapid release cycle, simple a/b testing and the ability to use standard web developers to build the experience.

hybrid is another approach

- avoid trying to mimic native controls

- use something like phonegap or use cordova views for a bridge

RWD vs RESS

responsive web design (RWD)

Let's remove the religion and remember the purpose:
we want to have an omni-channel experience

For content, RWD is a no-brainer. Use CSS media queries and adapt

And BTW, always right your code so that is adaptive

Forces the conversation with design/product

Downsides: payload maybe too large on small device;
fights the ability to freely experiment in a channel

responsive server side (RESS)

RESS. coined by Luke Wroblewski.

use server-side detection (like WURFL) to choose the code to “responsively” deliver

advantages: smaller, customized payload;
experimentation in different channels encourage

mobile first

mobile first myths

myth: mobile first means designing for mobile devices first and in isolation. only when the entire mobile experience is designed / built / launched should you consider the other form factors.

fact: no! absolutely not! mobile first means designing for an ecosystem of form factors simultaneously – mobile, tablet, desktop – but through the prism of mobile. (omni-channel)

the limited real estate of a mobile device forces simplification that is then applied also to the other form factors.

mobile first myths

myth: Mobile First means we will be building for lowest common denominator (like feature phones).

fact: No! Mobile First is about mobile capabilities as much as the constraints.

mobile first myths

myth: We always build one experience that adapts across screens (mobile, tablet, desktop, etc.)

fact: In some cases this is true. But in others we may have distinct experiences for each channel.

mobile first myths

myth: Mobile First means we are creating web (non-native) since we want the experience to adapt to other channels.

fact: Mobile First is not tied to the technical solution. There are strategies that different technologies may allow and others not. But Mobile First is about the product and design approach that can be applied to any tech stack.

mobile first myths

myth: Mobile First means that we always build and launch a mobile experience first.

fact: No, but we should do a rationalization of minimum viable product (MVP), and mobile first is a technique to arrive at this.

mobile first is really a mindset to always be rationalizing product and features in light of all the ways users interact with our products.

mobile first is thinking about a user's goal for a page and what the screen should provide to accomplish it. It's goals and needs, not just mobile.

mobile first myths

myth: only the mobile team needs to deal with gesture and touch interactions

fact: it is the responsibility of all teams to understand and account for the way a customer will interact with an interface (touch/gesture/keyboard/mouse)

mobile first myths

myth: mobile first limits creativity when building experiences.

fact: mobile first affords the ability to focus on what's needed and then explore appropriate innovative solutions for full browser/desktop capabilities.



break


20 minutes





lean ux anti-patterns

what should you be on the look-out for?

A hand-drawn sketch on a piece of paper, possibly a napkin or a scrap of paper, is shown on the left side of the image. The sketch features a central point from which several lines radiate outwards. Some of these lines are straight, while others are curved. There are also some small circles and a wavy line. The paper is white and is placed on a red surface. The sketch appears to be a rough idea or a conceptual drawing.

genius designer

all design emanates from an huber designer. Team doesn't collaboratively participate in design/ideation.

solution: Keep the inspiration of genius designer but bring in others to brainstorm. focus on MVP (minimal viable product) to test with customers immediately. critical to build team success early.

tribal group

when a team is very small
members are forced to work
across disciplines. As soon as
team gets bigger, tribes reform
around disciplines.
collaboration stops.

solution: You have to keep
team reasonably small. And
the leaders in each discipline
must form a tribe that works
across disciplines.





newcomer

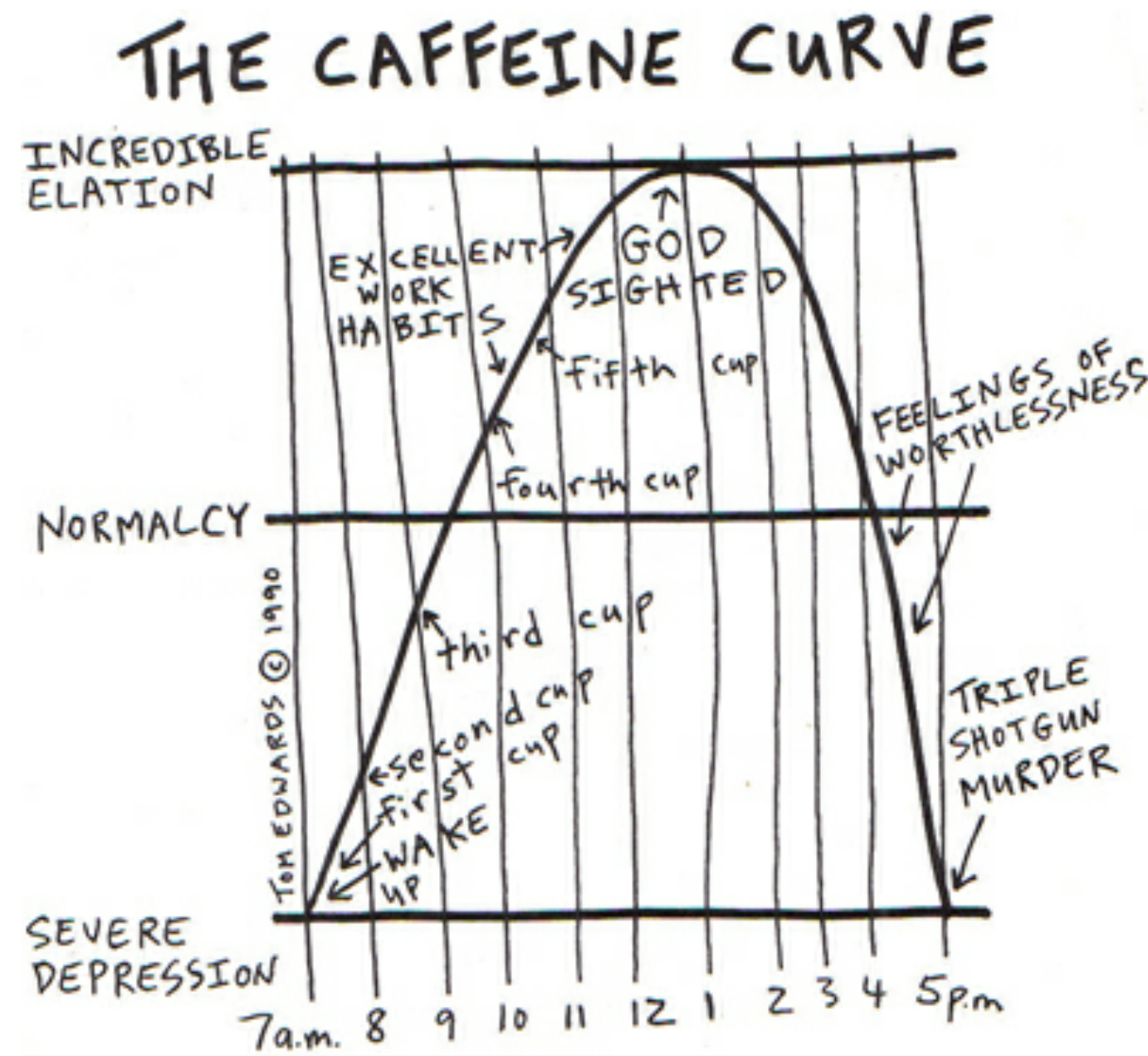
lean teams will form shared understanding. however, when new team member joins we assume this hard earned understanding will just happen.

solution: the team must immediately stop and initiate the newcomer. be patient, answer questions, reset vocabulary and enjoy the new voice in the team.

addicted

teams will often make a good start by trying out new behaviors and seemingly leave old behaviors behind. beware! old habits will creep back in.

solution: you must do it long enough and be successful with lean ux to ensure team members internalize the new habits.





naysayer

with collaboration so important it is key to believe in the process to create great products. a single naysayer can bring the team down in an instant.

solution: the naysayer must either learn new techniques or leave the team.

visitor

input from outside the team is essential. however, watch out. people cycling in & out of the team can cause the same disruption that the **newcomer** anti-pattern causes.

solution: customer trumps visitor. take input. test early and often with customers. that is the only “visitor” that ultimately matters.





magic tool

design & prototyping tools can accelerate ideation and design. however, be careful, tools that empower prototyping can enable designers to work in isolation.

solution: use tools as means to collaborate. never revert to “delivery” model of design.

going dark

when a developer, product manager, or designer goes dark for more than a day (or two) the team is losing valuable collaboration.

solution: working in isolation is necessary from time to time. however, limit to short periods of time. make work continuously visible.



change of cadence

change of cadence is actually a good and normal happening. however, whenever the rhythm changes it can bring productivity down.

solution: prepare the team for the change and quickly get focus and re-establish cadence.



too many cooks

the work needs to be divided up among different types of cooks (Chef de cuisine, Sous-chef, Chef de partie)

solution: have clear decision makers in each discipline and have specific roles (you can also rotate these functions).





not enough pizza

when a team suddenly scales up in size the team is in danger of losing cadence, shared understanding and focus

solution: keep teams to 2-pizza size. clear lines of responsibilities and laser focus for the team must be maintained

tower of babel

shared understanding is key to lean ux. however, it is easy to assume too quickly that team members are speaking the same language

solution: always ask, “what do you mean by x?”. always ensure other disciplines understand your jargon.





you got mail

teams can revert to email over collaboration. also, geo-graphically distributed teams can fall into delivery by email vs collaboration.

solution. Utilize high bandwidth communication (face to face, skype, telepresence, magic whiteboards, phone, etc.)

inmates run the asylum

this is from Alan Cooper's classic book of the same title. when engineers drive design the inmates are running the asylum.

solution, front end engineers must partner with product/design and get out ahead of backend engineers.





perfectionist

not embracing the challenge of the unknown, the perfectionist will not share their work till it is perfect. easy for designers to fall into this trap.

solution: engineers must not judge rough designs, instead they should use as springboard for collaboration. designers must realize iterative will yield better designs.

weakest link

team members who aren't up to the challenge of close proximity & transparency can cause a team to stumble

solution: talent acquisition must match this style of product delivery. must have freedom to replace talent.





the wall

walls between teams can happen when

- we allow tribes to form
- we see the other teams as separate delivery factories
- geo-distributed teams

solution: always work in small teams, collaborating not delivering and build shared understanding.

tangled up technology

unless the technology stack is built to have a clear separation from experience & services the lean team cannot make rapid progress. watch out when dev teams care too much about the specific version of the UI.

solution: key patterns include building services, APIs and CLIs. Keep the services & UI separate.





q & a

final q & a and wrapup

follow me on twitter @billwscott

picture credits

<http://www.flickr.com/photos/wuschl2202/531914709/sizes/o/in/photostream/>
http://www.flickr.com/photos/a_ninjamonkey/3565672226/sizes/z/in/photostream/
<http://www.flickr.com/photos/funky64/4367871917/sizes/z/in/photostream/>
<http://www.flickr.com/photos/emdot/9938521/sizes/o/in/photostream/>
http://www.flickr.com/photos/gregory_bastien/2565132371/sizes/z/in/photostream/
<http://www.flickr.com/photos/trvr3307/3703648270/sizes/z/in/photostream/>
<http://www.flickr.com/photos/legofenris/5426012042/sizes/l/in/photostream/>
<http://www.flickr.com/photos/cleaneugene/6866436746/sizes/c/in/photostream/>
<http://www.flickr.com/photos/66309414@N04/6172219058/sizes/l/in/photostream/>
<http://www.flickr.com/photos/nicmcphee/2954167050/sizes/l/in/photostream/>
<http://www.flickr.com/photos/pasukaru76/6151366656/sizes/l/in/photostream/>
<http://www.flickr.com/photos/brianmitchell/2113553867/sizes/o/in/photostream/>
<http://www.flickr.com/photos/ciscel/422253425/sizes/z/in/photostream/>
<http://www.flickr.com/photos/zebble/6817861/sizes/l/in/photostream/>
<http://www.flickr.com/photos/nicasaurusrex/3069602246/sizes/l/in/photostream/>
<http://www.flickr.com/photos/nathangibbs/98592171/sizes/z/in/photostream/>
<http://www.flickr.com/photos/neilsingapore/4047105116/sizes/l/>
http://www.flickr.com/photos/smb_flickr/439040132/
<http://www.flickr.com/photos/therevsteve/3104267109/sizes/o/>
<http://www.flickr.com/photos/st3f4n/4193370268/sizes/l/>
<http://www.flickr.com/photos/eole/380316678/sizes/z/>
<http://www.flickr.com/photos/cobalt/3035453914/sizes/z/>
<http://www.flickr.com/photos/mbiskoping/6075387388/>
<http://www.flickr.com/photos/fragglerrawker/2370316759/sizes/z/>
<http://www.flickr.com/photos/soldiersmediacenter/4685688778/sizes/z/>

picture credits (continued)

<http://www.flickr.com/photos/dahlstroms/4083220012/sizes/l/>

<http://www.flickr.com/photos/don2/53874580/sizes/z/>

http://www.flickr.com/photos/hao_nguyen/3634552812/sizes/z/

<http://www.flickr.com/photos/42573918@N00/8194636033/>

<http://www.flickr.com/photos/pagedooley/2420194539/sizes/z/>

<http://www.flickr.com/photos/neilsingapore/4047105116/sizes/l/>

http://www.flickr.com/photos/smb_flickr/439040132/

<http://www.flickr.com/photos/therevsteve/3104267109/sizes/o/>

<http://www.flickr.com/photos/st3f4n/4193370268/sizes/l/>

<http://www.flickr.com/photos/eole/380316678/sizes/z/>

<http://www.flickr.com/photos/cobalt/3035453914/sizes/z/>

<http://www.flickr.com/photos/mbiskoping/6075387388/>

<http://www.flickr.com/photos/fragglawker/2370316759/sizes/z/>

<http://www.flickr.com/photos/soldiersmediacenter/4685688778/sizes/z/>

<http://www.flickr.com/photos/janed42/5033842895/sizes/z/>

<http://www.flickr.com/photos/9619972@N08/1350940605/>

<http://www.flickr.com/photos/alanenglish/483251259/sizes/z/>

thanks flickr!



APPENDIX

JavaScript Landscape

JS Libraries

YUI

Ext.js

jQuery

Dojo

and 100s more

Component Libraries

YUI

jQueryUI

Ext.js

Twitter Bootstrap

Foundation

Kendo UI

JS Templating Solutions

See LinkedIn article on “Leaving JSPs in the Dust” for a reference to template smack down

Many of these exist (maybe 50+). Few of note:

Mustache, Handlebar, Dust, Ember, Underscore, JsViews, Pure, Jade, Google closure templates

Client Frameworks

backbone.js

ember.js

angular.js

spine.js

knockout.js

Full Stack

tower.js

derby

wakanda

meteor

flatiron

mojito

Mobile/Touch

Sencha Touch

jQuery Mobile

Lighter versions of jQuery Mobile

Zepto.js

Snack.js

\$dom

140medley

xui

<http://tutorialzine.com/2012/04/5-lightweight-jquery-alternatives/>

Mobile native/web solutions

appcelerator

phonegap/cordova views

Packaging/Modules

require.js

labJS

script.js

curl.js

node-browserify

ender

Build solutions

lumber

code surgeon

requireJS

brunch

grunt