bringing design to life with lean ux & lean engineering

lessons learned @ paypal in blending teams to create new experiences

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

cody evol (@codyevol)
checkout design lead, ued, paypal

lean day west, portland or, sept 2013

the schedule

08:30–08:45	**introductions (15)**
08:45–09:30	**the problem (45)**
09:30–10:00	**adopting lean (30)**
10:00–10:15	break (15)
10:15–11:15	**the lessons (60)**
11:15–11:45	**the tools (30)**
11:45–12:00	**Q & A**

the problem

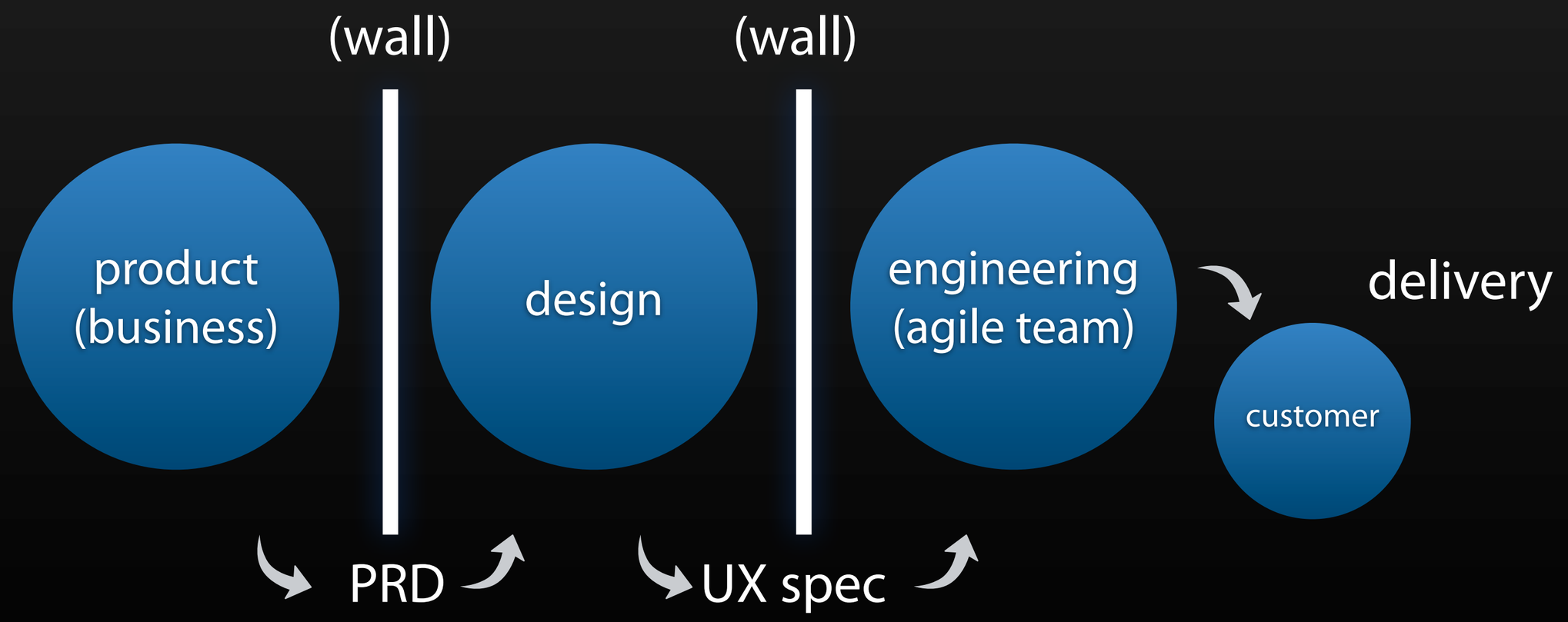
a look at where paypal has been. can you relate?

organizational model

standard process creates distinct work phases

boundaries are the hand-off points between roles

typical product life cycle



upon delivery, team disbands and forms into new teams

what was broken in design?

late 2011/early 2012

deep silos

iteration planning done by developers without designer's involved

designers hand off specs without prior involvement of developers

developer days ("dev days") valued over design time

frequent WFH days created low energy and less collaboration time

hyper-segmentation of products

broad team distribution

geographic distribution created wedges, duplications and blocked collaboration

lack of alignment with UED partners (not uncommon to have designers & engineers in same region to be working on different products)

lack of agile understanding

while UED interfaced with agile teams they did not participate directly in agile planning, retrospectives, etc.

agile machinery also did not account for experience design

no strong ownership

UED staff in a pooled/studio model instead of a dedicated model

once delivery happened the designers moved to another project

often engineers did not know who the designer was for a product to ask questions to

teams not empowered to make decisions as a gauntlet of other teams had to approve to go live

what was broken in product?

late 2011/early 2012

no measurement/learn culture

in several products there were no key performance indicators to measure & learn against

since a/b testing was hard to do, there was no concept of an MVP (minimal viable product)

feature-itus

since the organization rallied around projects instead of products, product tended to try to put as much on the train as possible

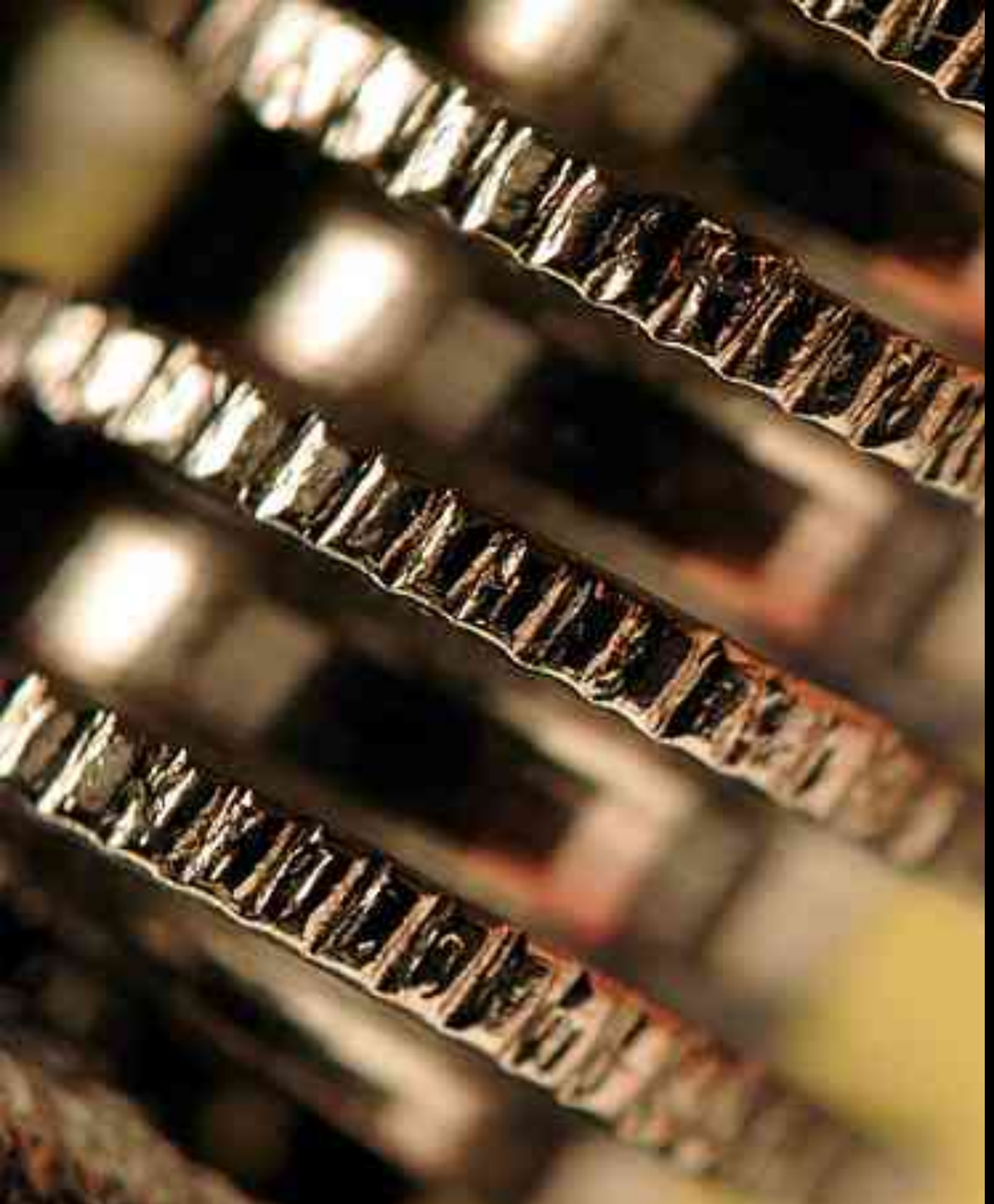
without kpis you guess more and more (F.O.G.)

without measurement you never get rid of features

too many silos

product was divided over 9 different organizations!

mobile was also a separate business, product and engineering silo



what was broken in engineering?

late 2011/early 2012

too many silos

just like our counterparts, we were broken into many different organizations

mobile was a separate organization

too hard to go live

37 tickets just to get a bug fixed and pushed to live

every organization was set up to say “no” to anything that might be innovative for fear of failure, risk, security issues, etc.

no devops, no CI/CD

technology broken

no modern services architecture

all solutions were built as silos

ui logic and business logic intertwined

technology platform assumed developers were not to be trusted

agile way too granular

one product had 20+ agile streams. 12 of these were experience streams. each stream was responsible for one small part of the experience

created nightmares of integration

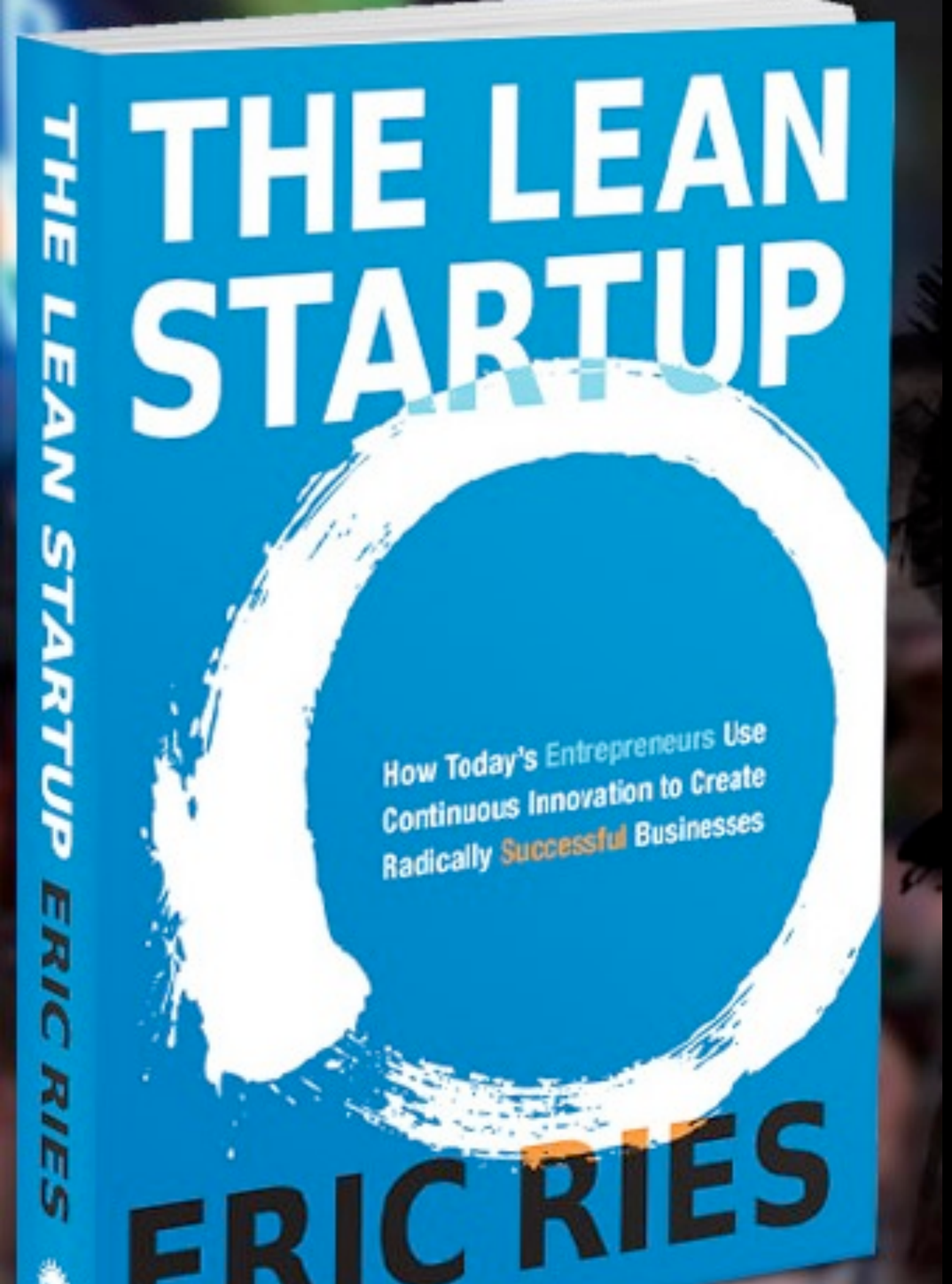
created a fractured experience for users

paypal circa 2011

roll your own. disconnected delivery experience. culture of long shelf life. inward focus. risk averse.

adopting lean

following a build/measure/learn mindset



lean startup

founded on build/measure/learn

get out of the building (GOOB)

invalidate your risky assumptions

go for the minimal viable product (MVP)

fail fast, learn fast

get to the pivot

lean ux

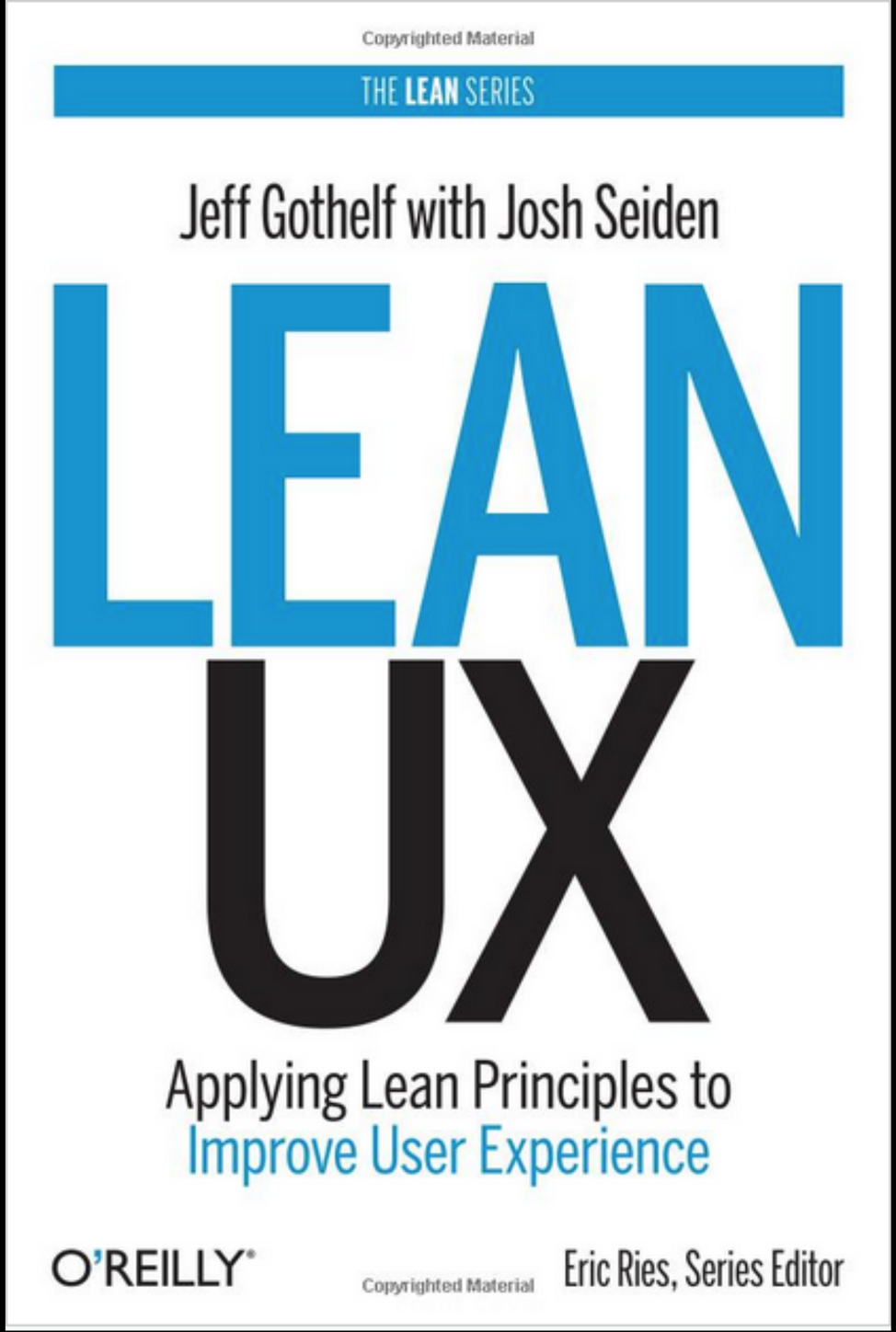
designing products for build/measure/learn **(lean startup)**

requires 3 rules to be followed at all times

get to & maintain a **shared understanding**

form **deep collaboration** across disciplines

keep **continuous customer feedback** flowing



LEAN ENGINEERING

Applying Lean Startup Principles
to
Bring Design to Life

engineering focused on **learning**

engineering the **build/measure/learn** cycle

shift the focus to minimal viable everything (MV*)

follows the key rules of lean ux:

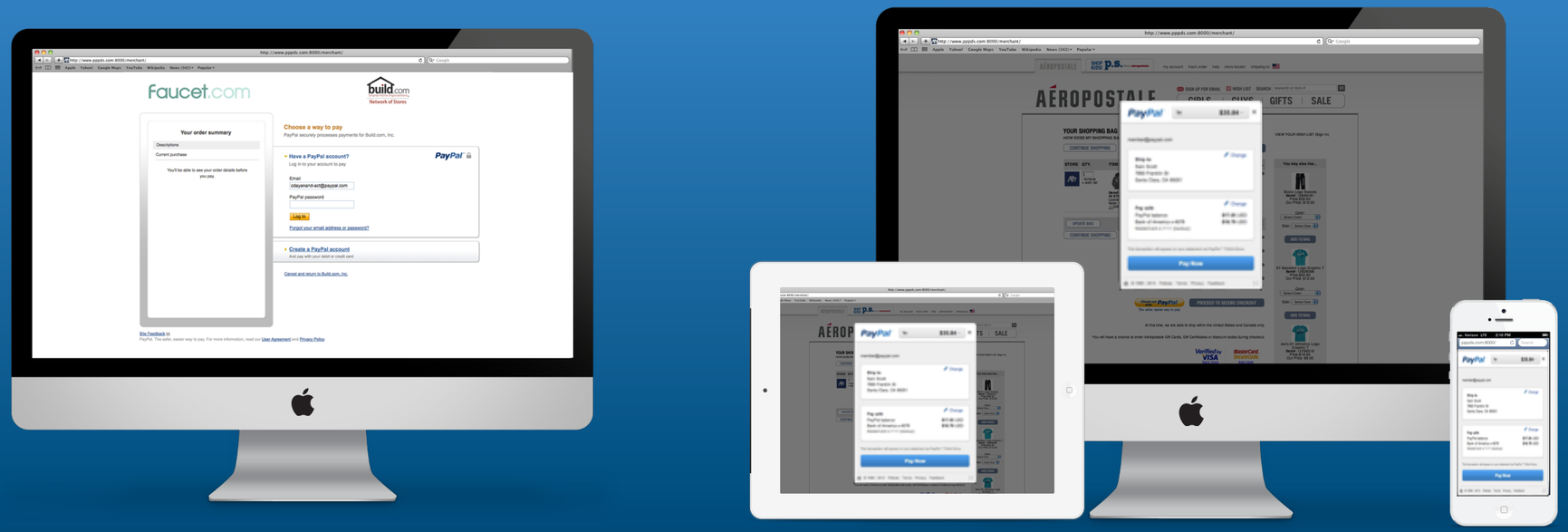
shared understanding

deep collaboration

continuous customer feedback

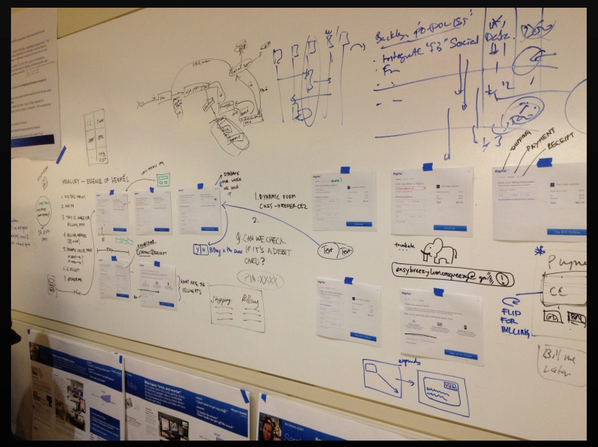
hermes project

re-inventing checkout with lean ux



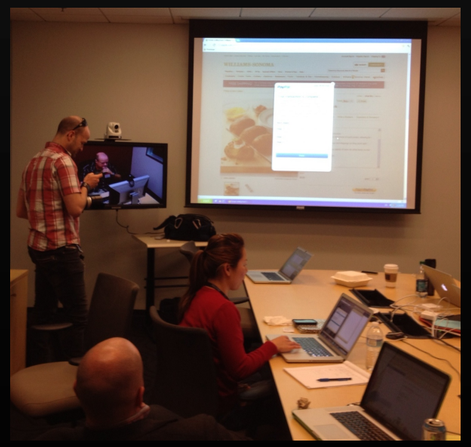
hermes project

lean ux in action



whiteboard
to code

code to usability

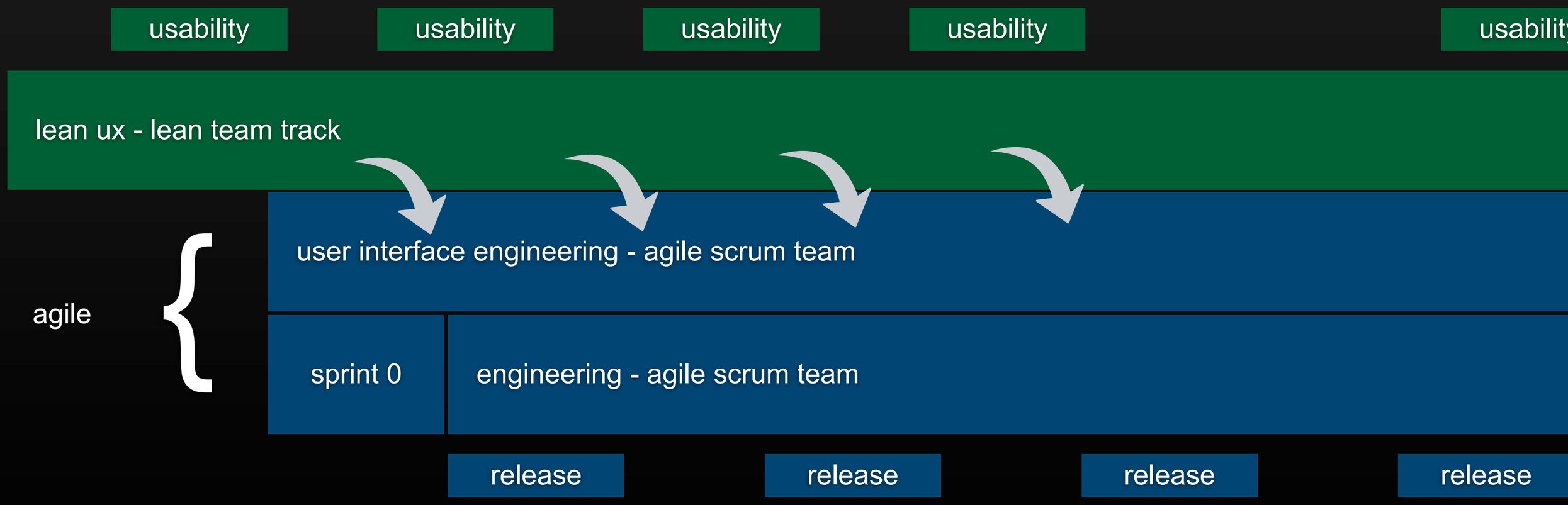


product/design/engineering teams

usability/customers

free to iterate independent of agile

lean ux can provide a brain for agile



three key principles that drive lean ux

remember these to keep your teams on track

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

lessons learned along the way

school of hard knocks

1

create a sandbox

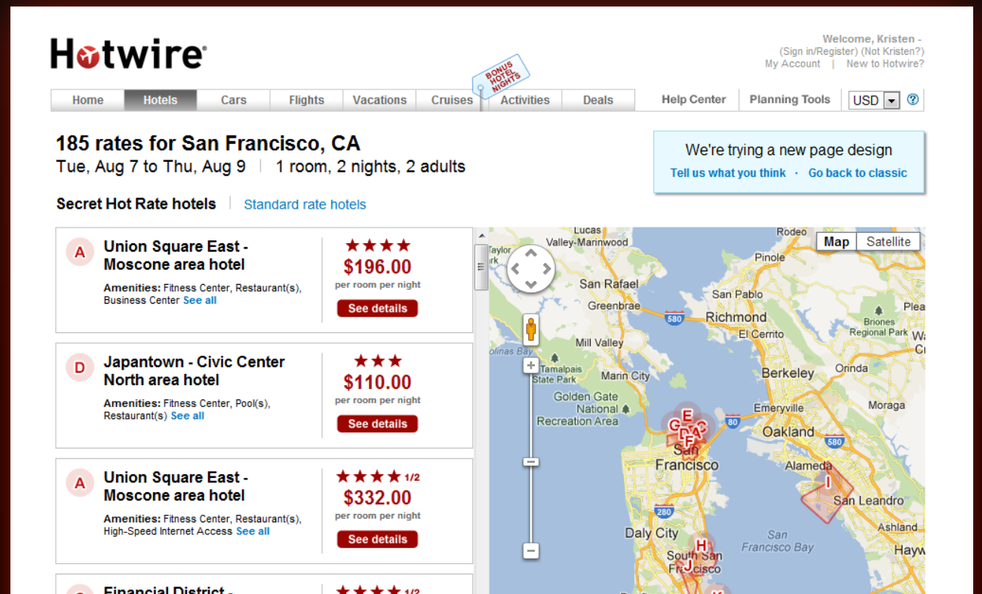
IMVU allows every engineer to put a test out to 1% of users

at netflix we often created additional tests that designers or engineers independently wanted to try as a solution to a hypothesis

hotwire case study

Source: “Lean Startup in the Hotwire Enterprise” by Kristen Mirenda & Karl Shultz

how do you protect the parent organization from the internal startup?
create a sandbox



hotwire case study: feedback

Source: “Lean Startup in the Hotwire Enterprise” by Kristen Mirenda & Karl Shultz

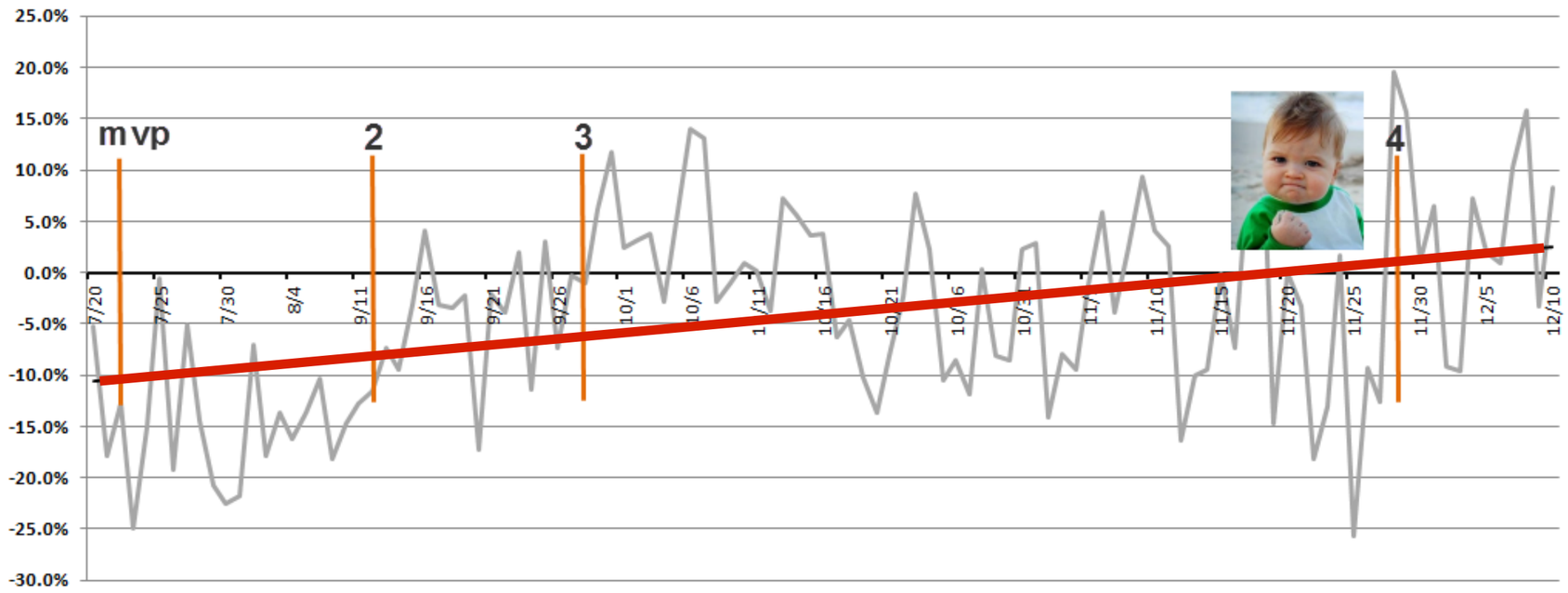
“I don't like it because you cannot filter the results or even sort them.. What were you thinking?”

“hate it - can't even sort anymore”

“absolutely blows...pure garbage. need to be able to sort asap. i'll come work for you and help you figure it out. wtf.”

hotwire case study: data

Source: “Lean Startup in the Hotwire Enterprise” by Kristen Mirenda & Karl Shultz



move to a “living spec”

break down barriers between prototyping and production

use developers for prototyping as forcing function

embrace RITE

avoid tools/process that get away from collaboration

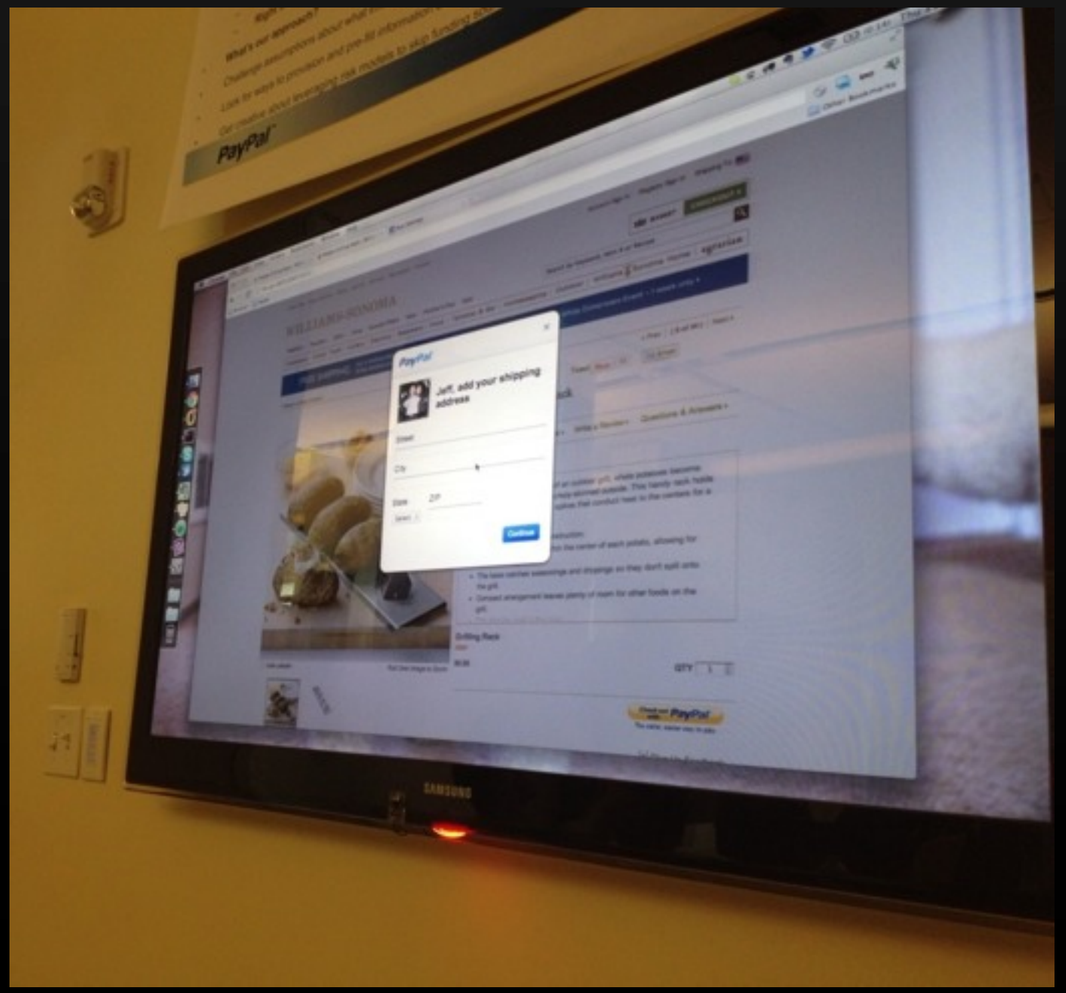
make the spec real

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?

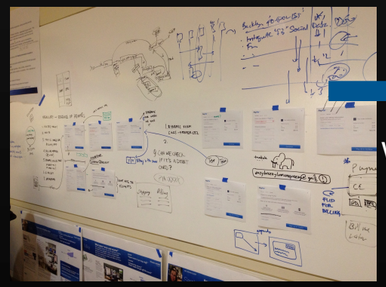
watch out for “predictive documentation”

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

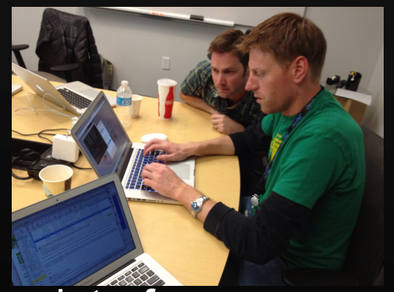
good documentation will enhance collaboration, shared understanding and disseminate learnings

use a prototype stack

to enable learning



whiteboard
to code



code to usability

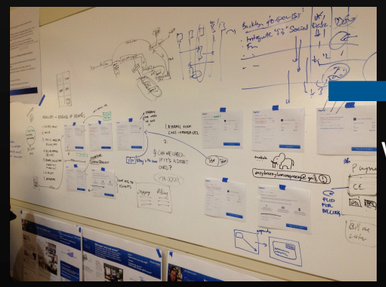
product/design
team

user interface
engineers

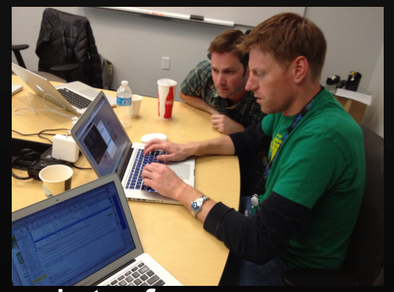
usability/customers

use a prototype stack

to enable learning



whiteboard
to code



code to usability

product/design
team

user interface
engineers

usability/customers

JS Templating
(dustjs)

JS libraries

less -> CSS

images

nodejs

enable sketch to code

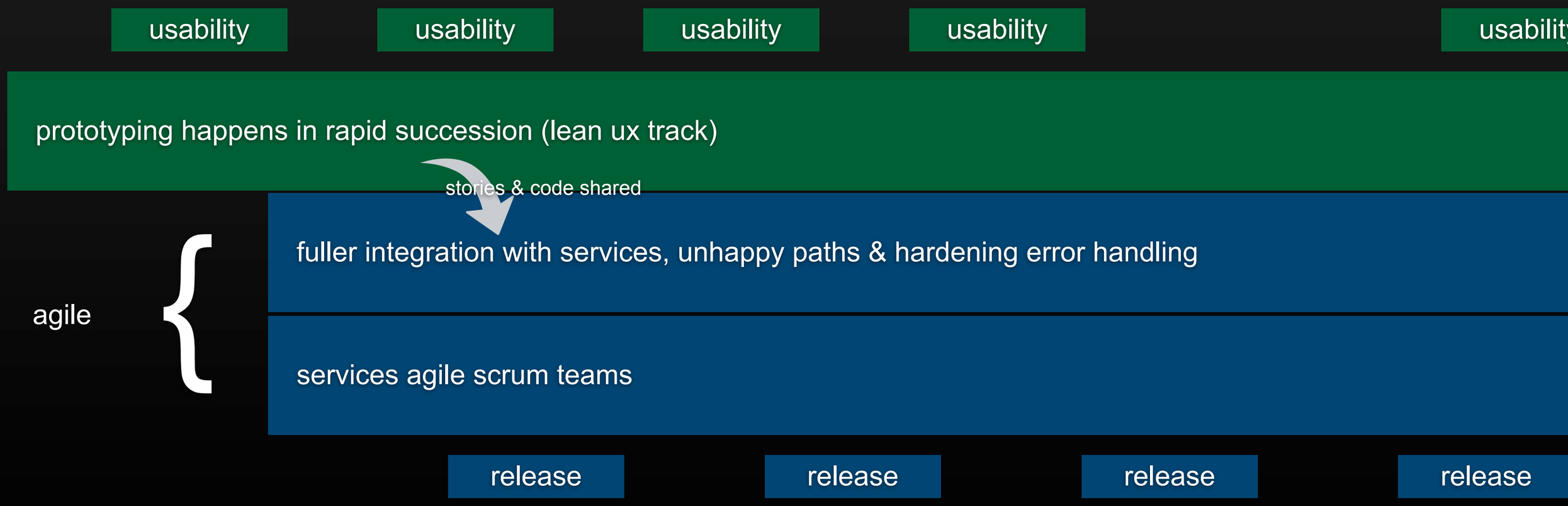
forcing function

- it brings about a close collaboration between engineering and design
- it creates a bridge for shared understanding

requires a lot of confidence and transparency

how lean & agile can play together

lean ux can provide a brain for agile



lean & agile teams should blend together

3

engineer for experimentation

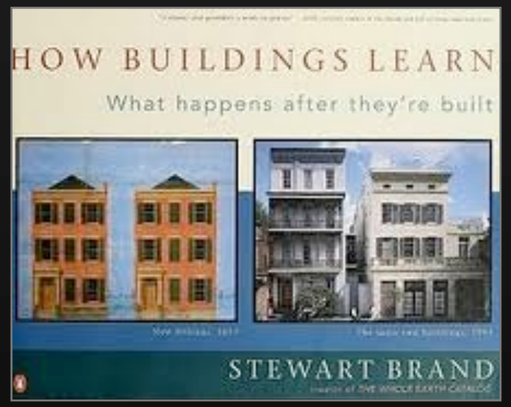
long shelf life to rapid experimentation

focus on learning not on delivery

design for volatility

refactor the tech stack with learning in mind

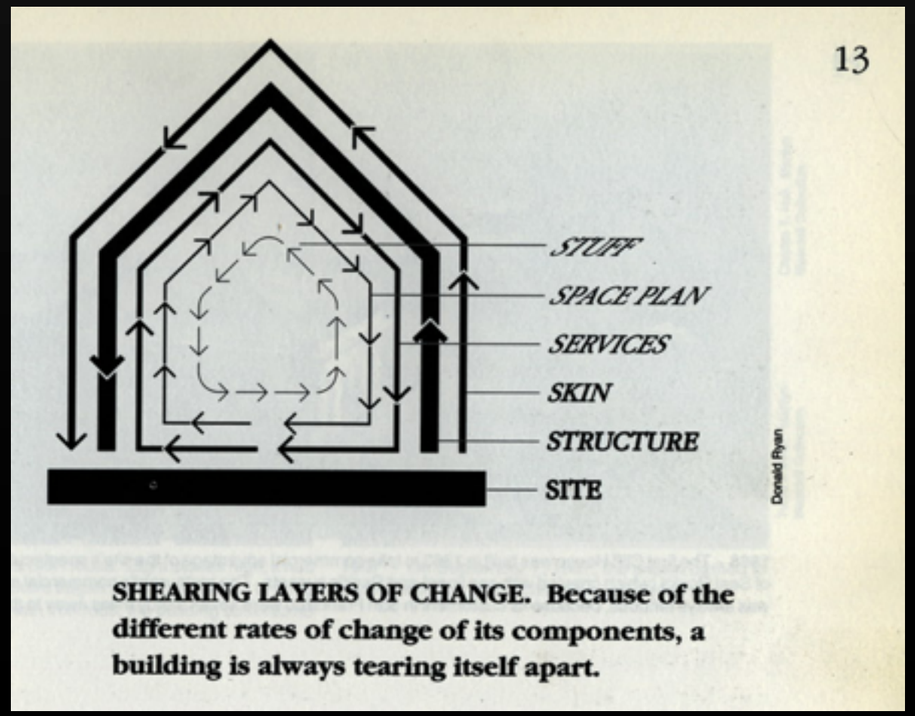
experiences must learn



All buildings are predictions.
All predictions are wrong.

There's no escape from this grim syllogism, but it can be softened.

Stewart Brand



SHEARING LAYERS OF CHANGE. Because of the different rates of change of its components, a building is always tearing itself apart.

Our software is always tearing itself apart (or should be)

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

why start with experience?

stay honest & pure by having experience be the driver

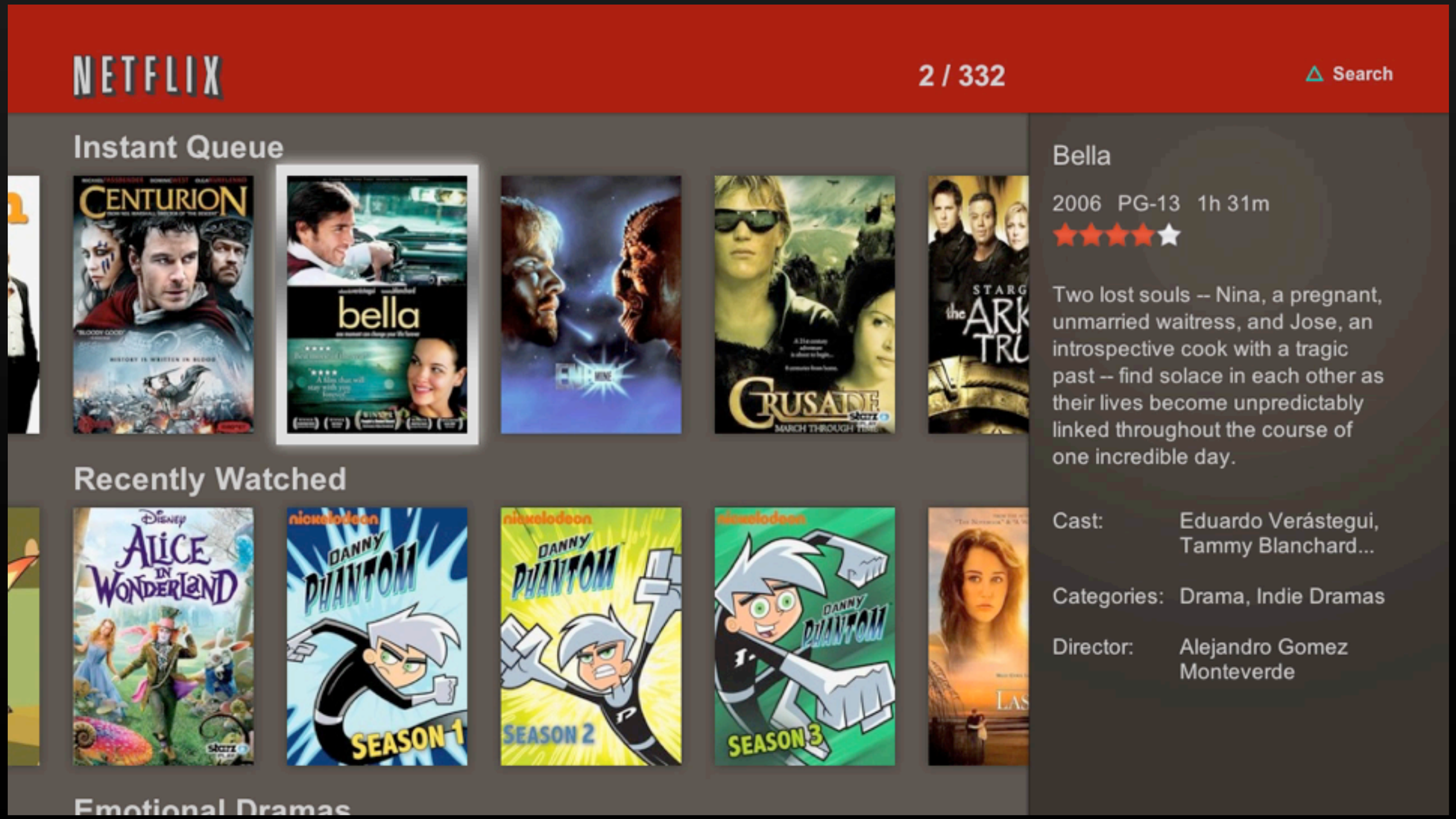
(not what your boss thinks or what looks good on your resume or what the loudest one in the room thinks)

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.

experience vs components

experience vs components



NETFLIX

Bill Scott | Your Account & Help

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	★★★★½	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

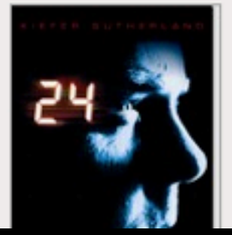
See all >

Your taste preferences created this row.

Suspenseful Action & Adventure

As well as your interest in...

24: Season 2



Chain Reaction

Westbound

Boxer's Adventure

tangled up technology

big problem. technology and processes not geared to build/test/learn.

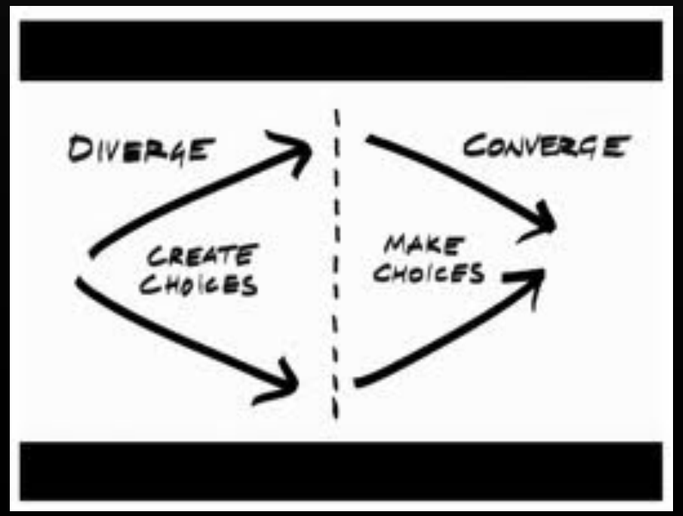
refactor your way out of technical and experience debt

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



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

1st step: fire up a prototype stack (nodejs)

ui bits

node.js

prototype stack

utilize opens source stack

express, connect, require.js

bring in javascript templating and other open source ui goodness

2nd step: bootstrap with bootstrap

ui bits

node.js

prototype stack

Bootstrap

able to create a new branded look in a few hours

enabled sketch to code

3rd step: use javascript templating

templates = JS
{dust}

text templates get compiled to javascript

Hello {name}

```
{dust}
```

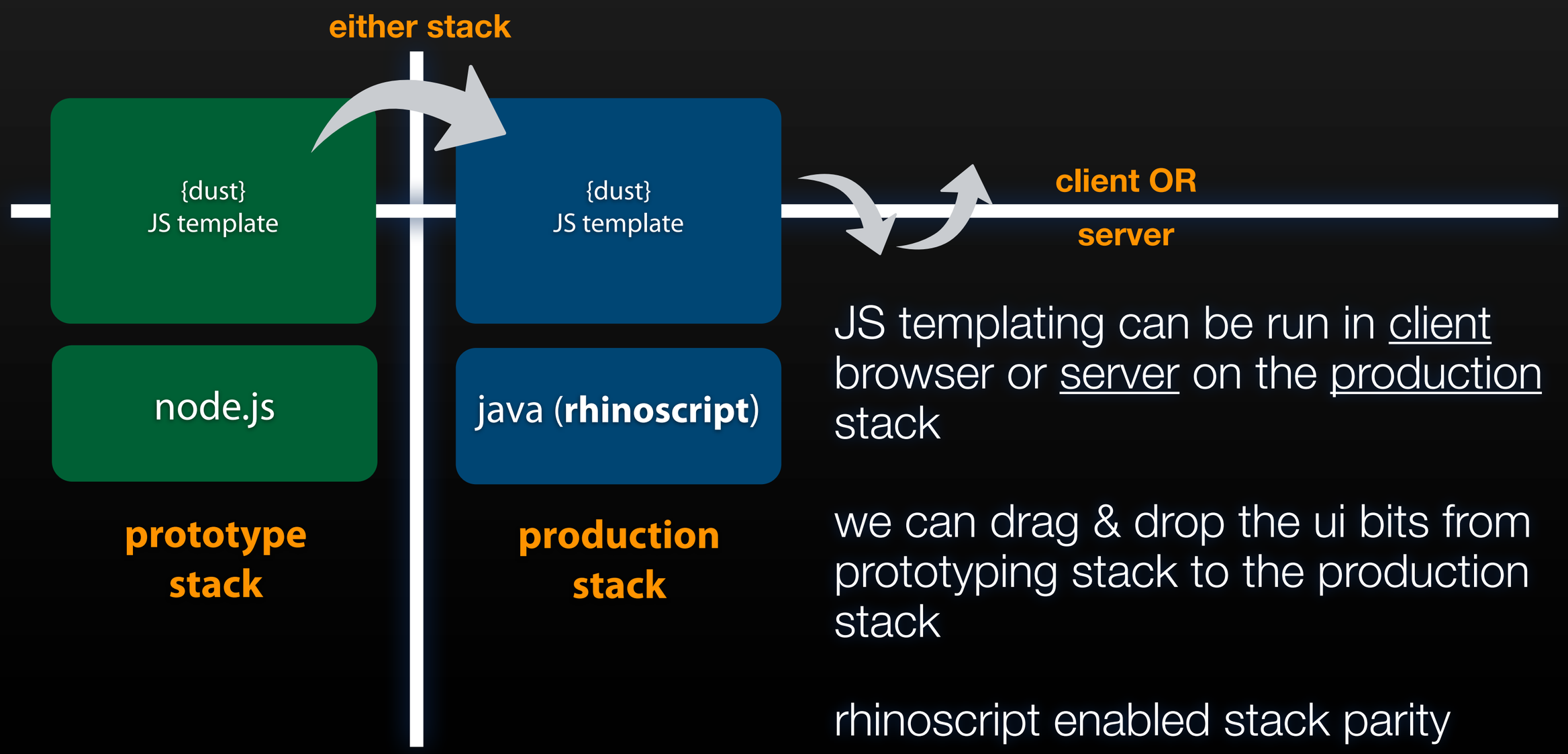
compiles to...

JavaScript

javascript is evaluated to render ui

dustjs templates execute wherever there is javascript

4th step: make ui bits portable to legacy



JS templating can be run in client browser or server on the production stack

we can drag & drop the ui bits from prototyping stack to the production stack

rhinoscript enabled stack parity

5th step: build on open source

Bootstrap

{less}

UNDERSCORE.JS

jQuery

BACKBONE.JS

{dust}

REQUIRE.JS

Passport

nconf

q

async

supertest

node.js

express

kraken

Chai Assertion Library

contains “webcore” for scaffolding and providing a lightweight framework for dev & production

mocha

prototype & production stack

github love

6th step: bring node to production

project kraken

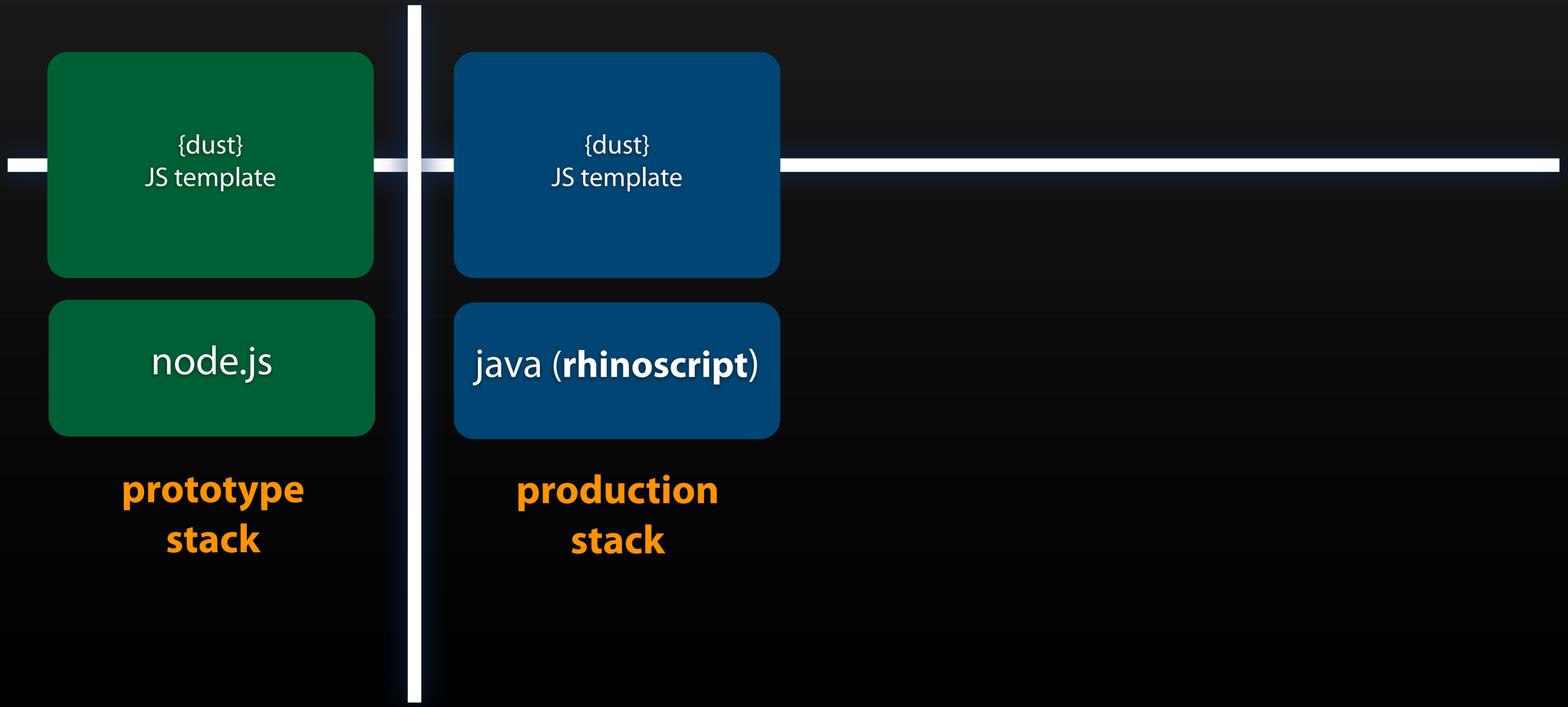
enable all of the standard paypal services WITHOUT looking like PayPal

but do it in a friendly npm way

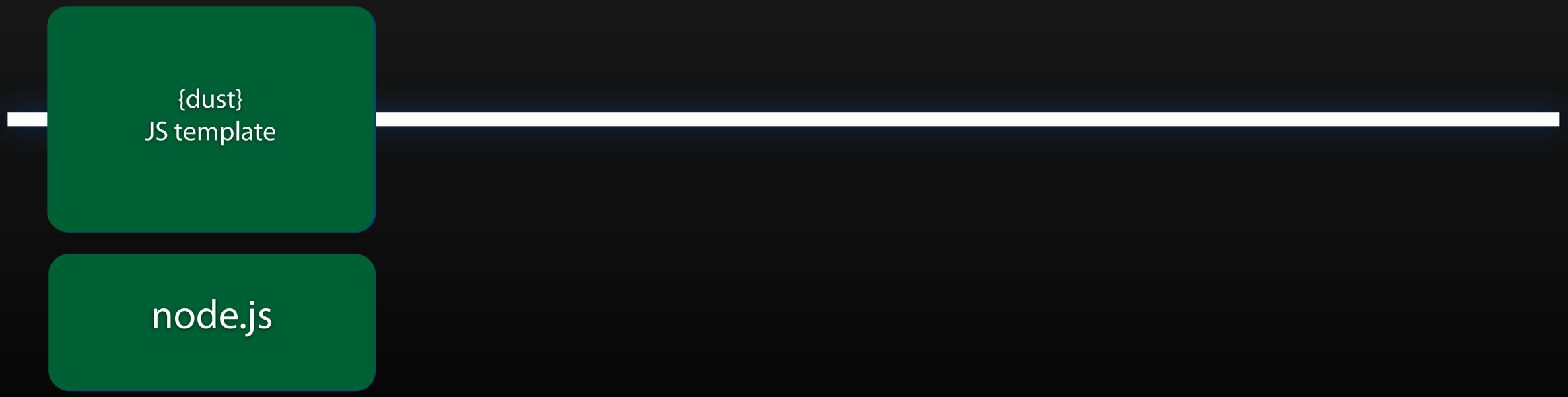
- monitoring, logging, security, content, locale resolution, analytics, authentication, template rendering, experimentation, packaging, application framework, deployment, session management, service access, etc.

simplifies creating an app in a few minutes with all paypal services

7th step: one stack to rule them all



7th step: one stack to rule them all



prototype stack

production stack

4

give agile a brain

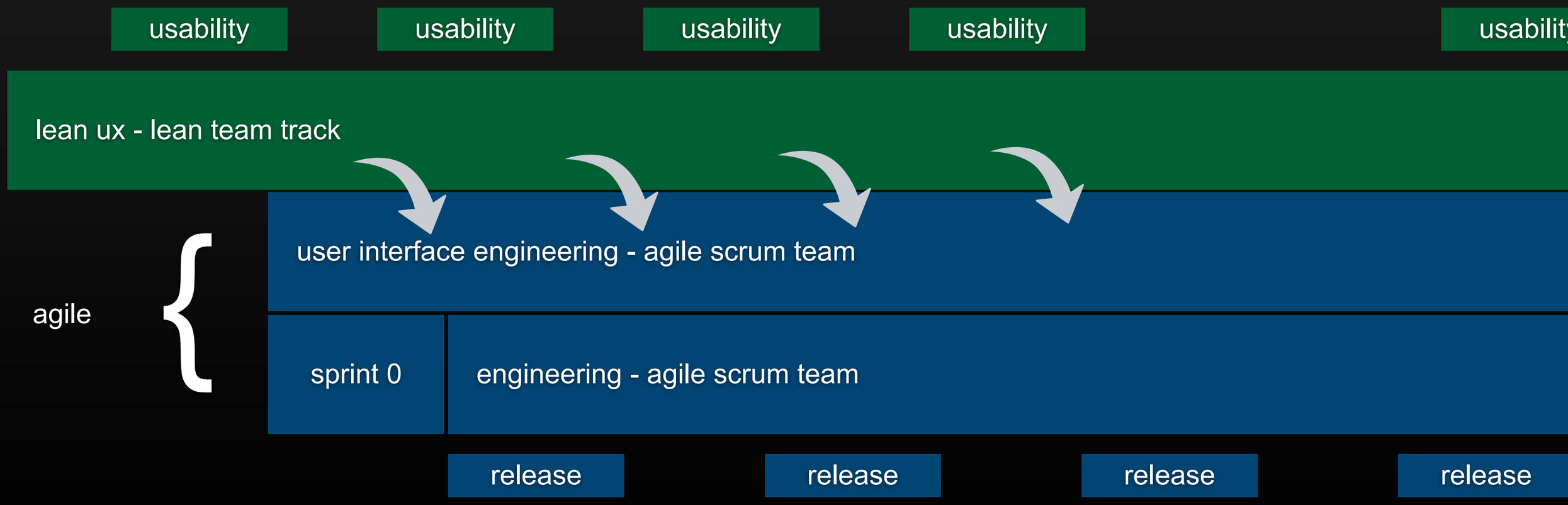
use lean ux as the brain for agile

develop a lean cadence

involve all members in lean ux (balanced teams)

free to iterate independent of agile

lean ux can provide a brain for agile



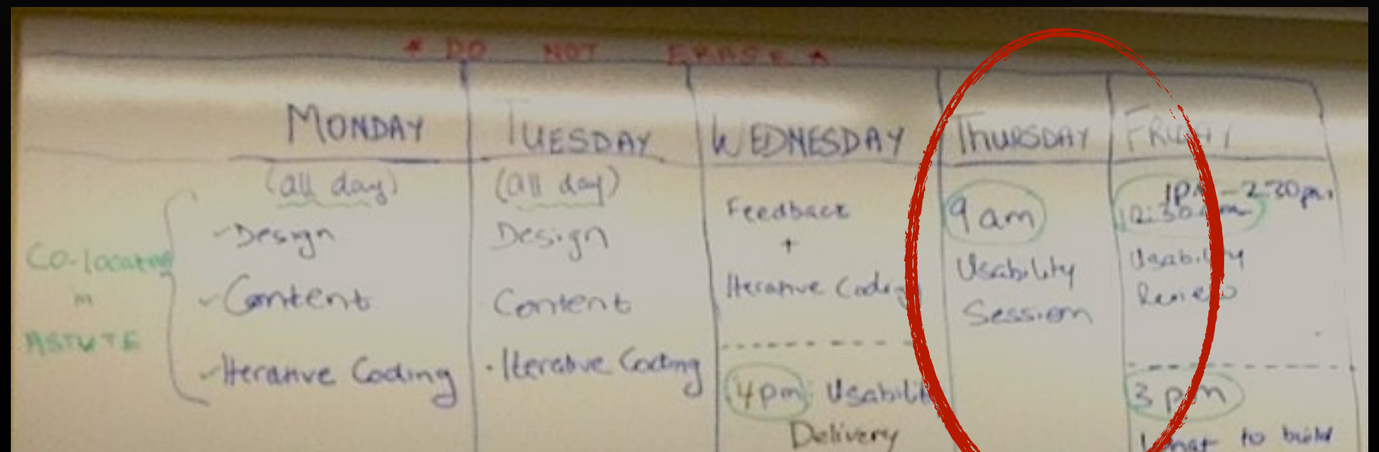
free to test frequently with users

sprint faster

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

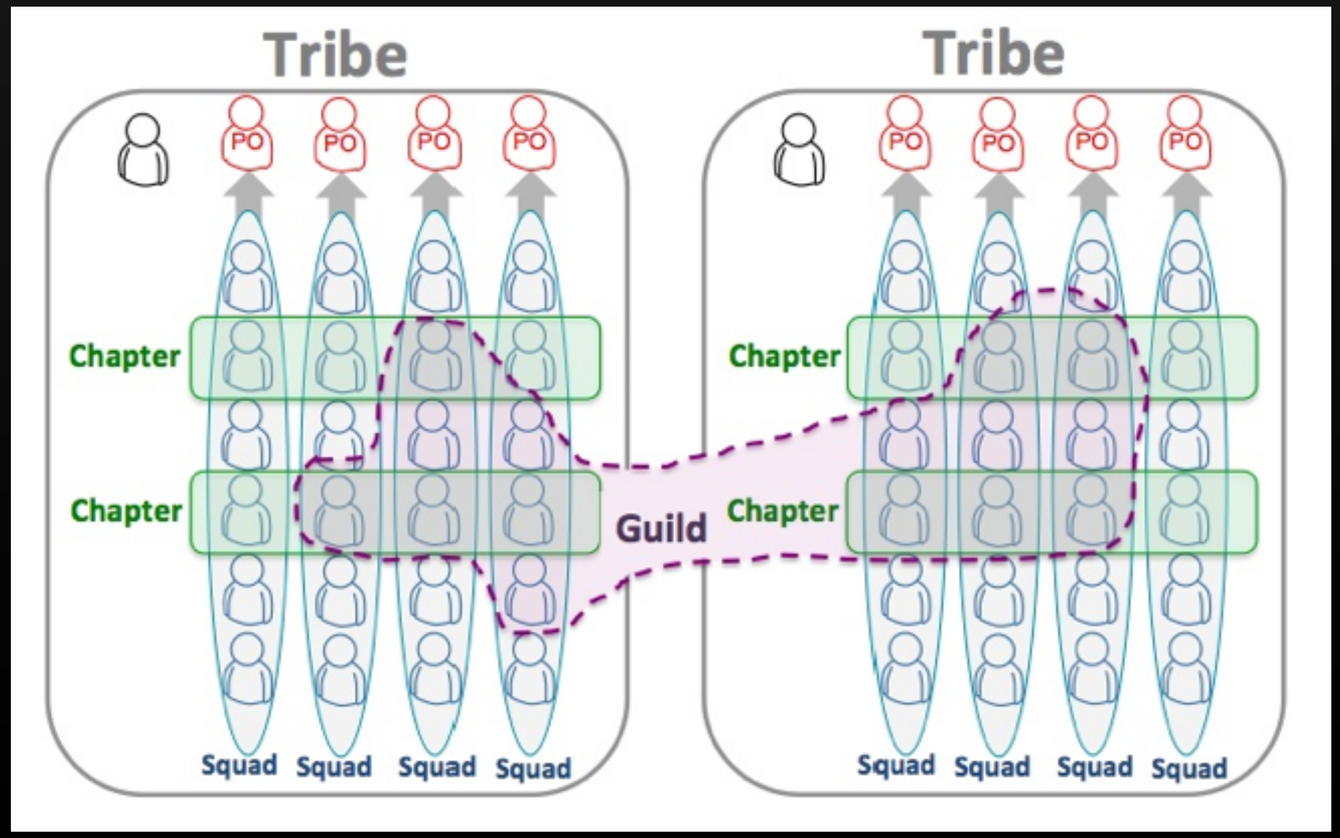
removes the politics in the team as this becomes the arbiter

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



example: spotify

squads run like lean startups



spotify: squad

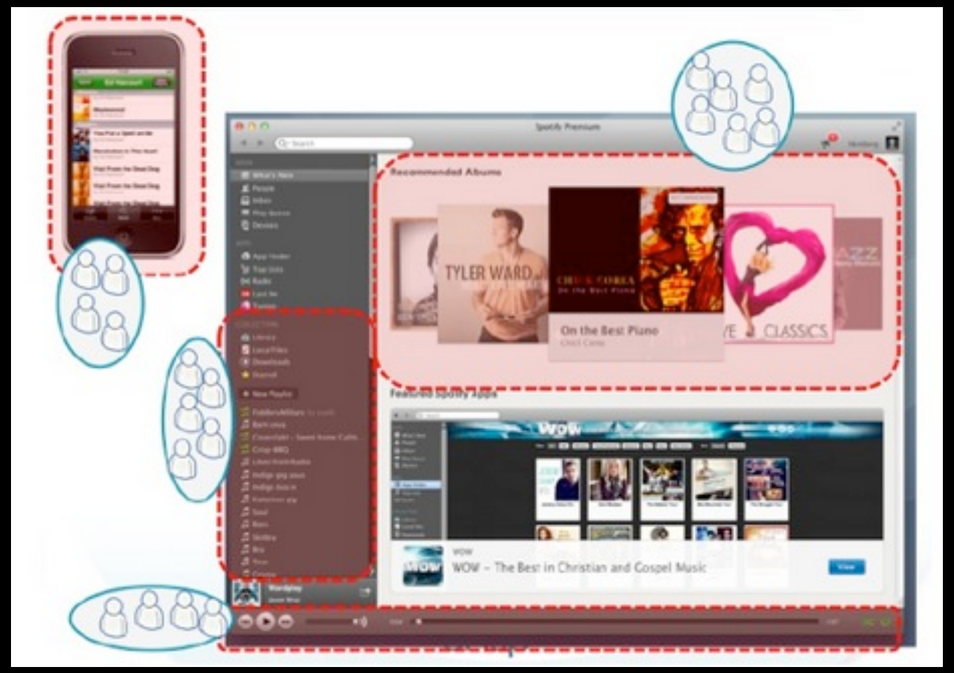
similar to scrum team. feels like startup

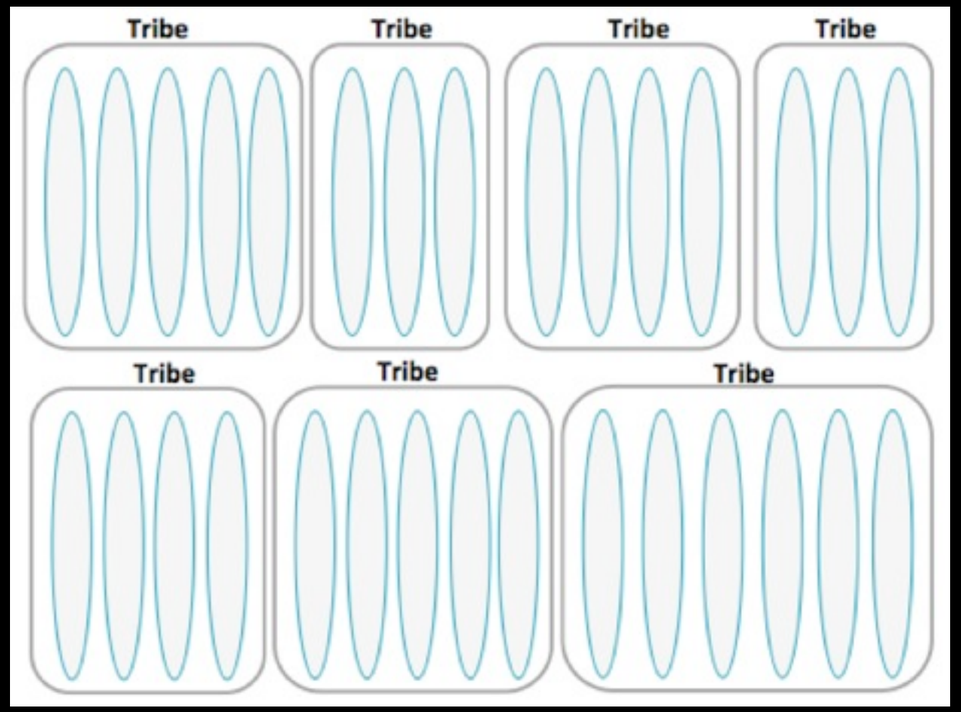
long term mission: build & improve the product. stay long term on the product.

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

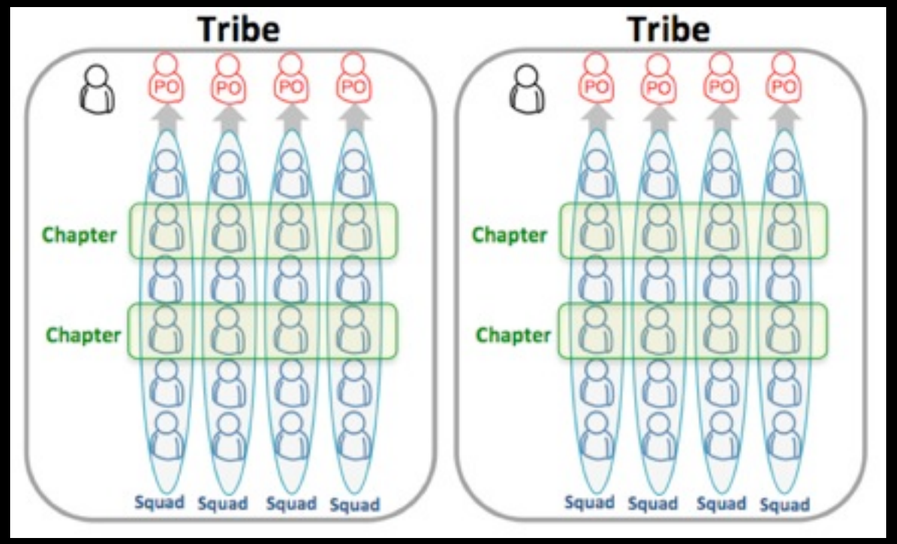
incubators for tribes

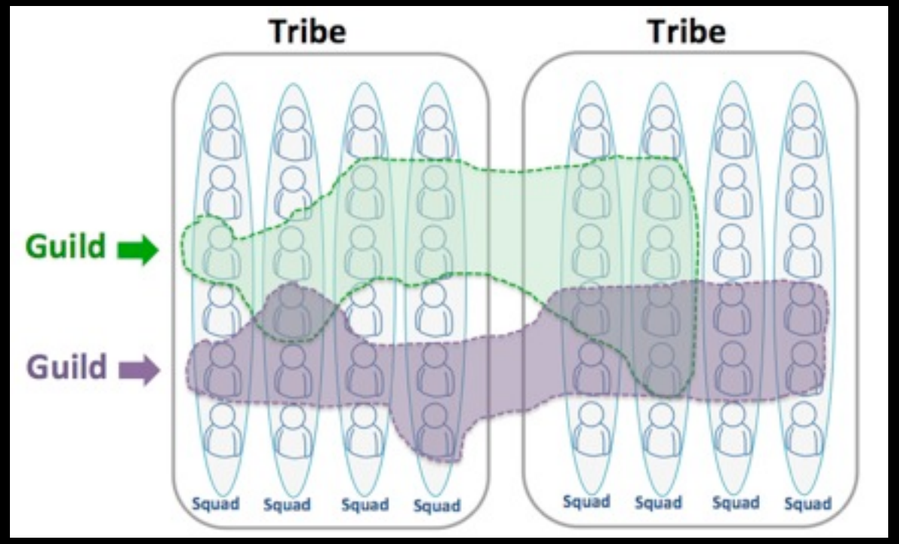
hold regular gatherings

spotify: chapters and guilds

chapters represent horizontal practices within a tribe

guilds represent horizontal practices across tribes





become hypothesis driven

learn to state design goals in terms of solving specific customer problems

don’t be afraid to invalidate hypotheses

look for the pivot

5

6

embrace the problem not the solution

engineering: don’t start with technology, start with experience

design: get your ideas out early

together: get in front of customers so problem is the focus, not our current solution

co-locate if at all possible

high bandwidth “meatspace” facilitates **shared understanding** and **deep collaboration**

also facilitates shared 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. single source truth.

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

tools that can help

tools

technologies (or lack thereof) that can help your team stay lean

tools

sketching/whiteboard

paper prototyping

patterns and visual design

ui software

prototyping

sketching and whiteboarding

stop talking, start drawing

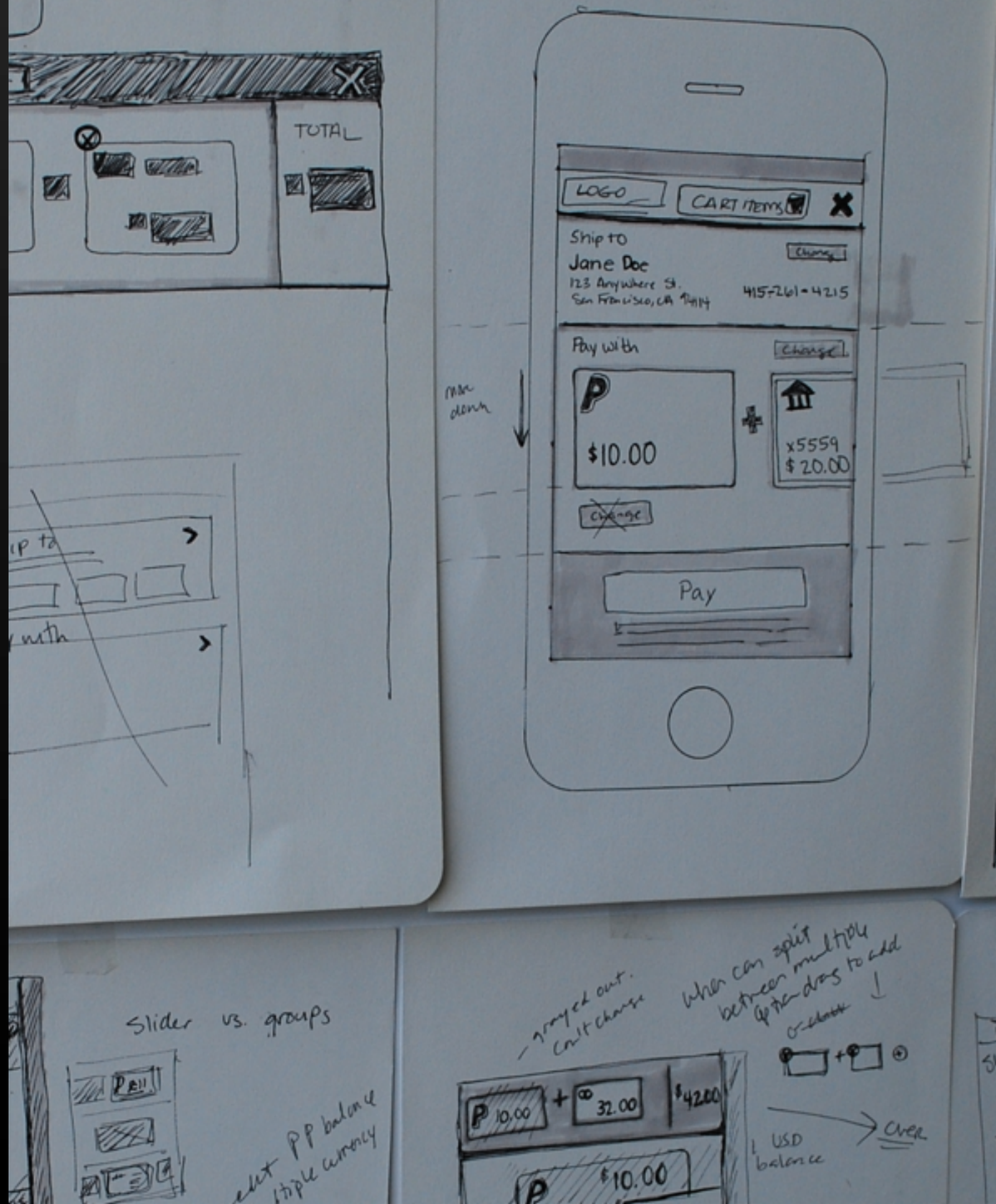
rapid ideation

throw away

validation

shared understanding

use as a part of deliverable



paper prototyping

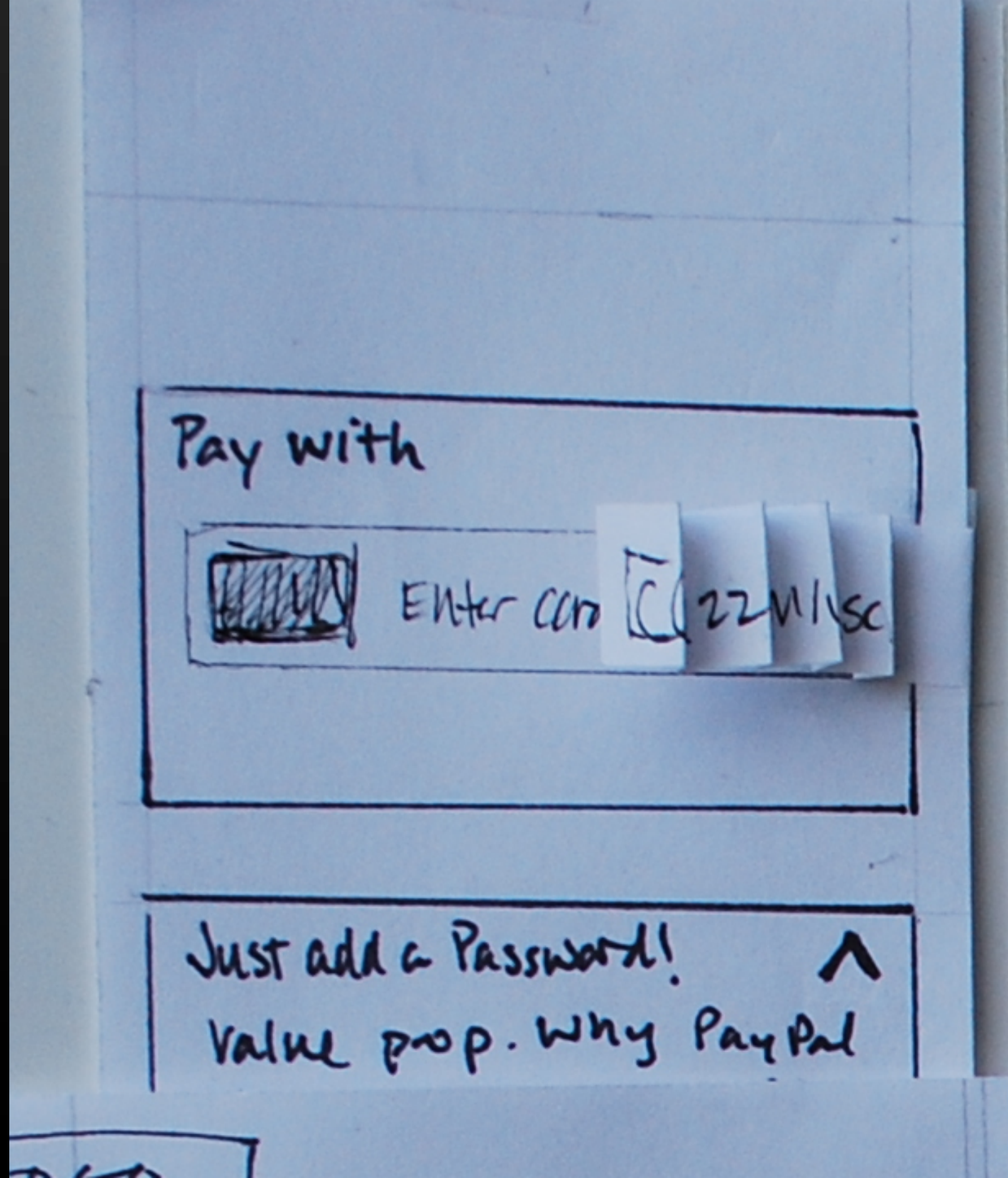
easy to go from paper to production

validate interactions

makes it easer and faster for developers to understand.

paints a clearer picture to business partners.

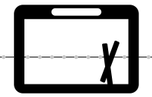
super fast



prototyping software

Fastest






Fast







Slower



code prototypes vs ui prototyping software?

use the right tool at the right time

- as you get closer to agile

axure, proto.io, POP and a host of other prototyping tools are amazing -- especially early in the learning cycle

code prototypes

- important once you get close into the actual agile sprints
- provide high fidelity to the user testing
- faster cycle from “learning to live”

suggestions for code prototyping

bootstrap is one of the quickest to get going with

we use it on our production stack as well

jetstrap allows you to drag and drop a bootstrap page to get a quick start

nodejs is really powerful for prototyping your full application (web, tablet, desktop)

node js™

Bootstrap, from Twitter

why templating is cool for prototyping

start with simple html mock for the page

convert to template: sprinkle in data placeholders, iterations

throw some mock json data in and you have a prototype

same template when generated with real json data from services is the live experience

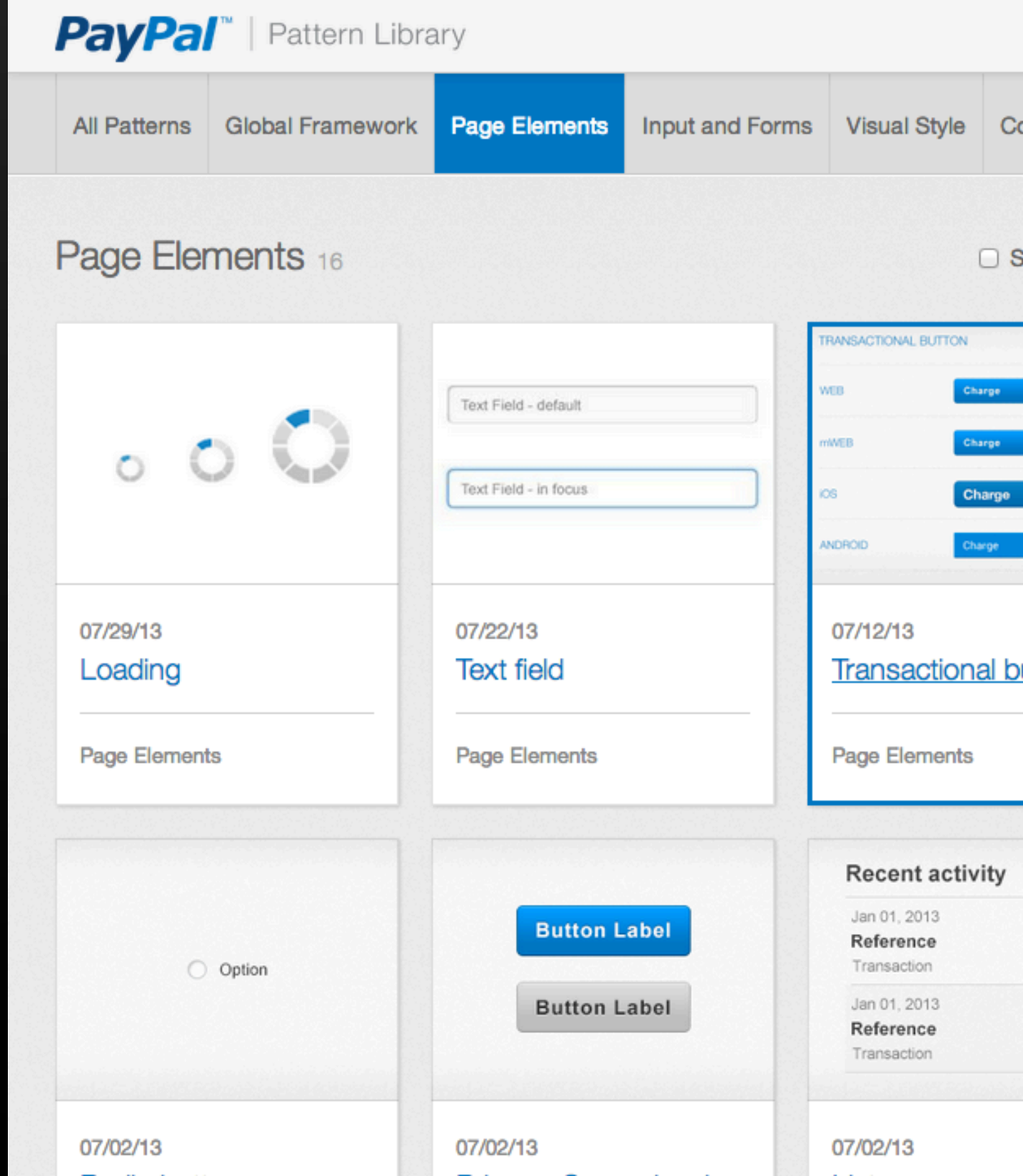
patterns & visual language

patterns enable rapid development

refine over time

ensure consistency

speed up design



three key principles that drive lean ux

remember these to keep your teams on track

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

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/fragglerawker/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/fragglerawker/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!