

WADU

PRODUCT

VS

CRAFT

Juan Delgado @wadus

May / June 2016

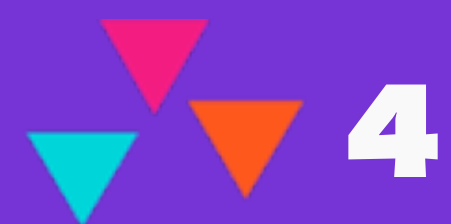
NOTE: This is a modified version of the slides for the talk, probably only make sense to the people that attended one of the conferences.

If you haven't, in the [Product vs Craft post in my blog](#) there're links to recordings you can watch.

Thanks :)

วิสต้า

**WE'RE A DIGITAL
PRODUCT STUDIO**



Studios



People

**OUR CONTEXT:
WE SOLVE
PROBLEMS
AND UNCOVER
NEW
OPPORTUNITIES**

These are some of the people we work with, from startups to huge multinationals. Their needs and software development practices are very different, giving us the opportunity to test our ideas about development in very different scenarios.

 **Network** Locum

 **streams**
by DeepMind Health



Wayfindr



DEFINITIONS

- **Product:** The output of the software development process: an app, a website, a digital poster... This also includes the business plan, the value exchange map, etc.
- **Craft:** Set of practices for software development.

THE PROBLEM

Either we execute to the best of our abilities at all times or we are doing it wrong. Moreover, certain practices are being adopted and spoken about in absolute terms without enough reflection (by some, not you of course!).

THE HYPOTHESIS

Effective software engineers adapt the way they develop software based on the product vision and the point in the lifecycle that the product is.

QUALITY IS A SPECTRUM



QUALITY?

What is “quality” when we are talking about software?

QUALITY IS NOT AN INTRINSIC PROPERTY OF SOFTWARE

“[...] quality often depends on the context in which a software component or feature operates. **The quality of a software component is not an intrinsic property** - the exact same component can be of excellent quality or highly dangerous depending on the environment in which it operates or the intent of the user.

[...] contextual nature of software quality is a fundamental challenge [...] **what is elegant in one situation might be downright unworkable in another”**

QUALITY?

“Quality” is a vague and hard to measure property. It’s also very polarising language.

Carlo Pescio is trying to come up with concrete and measurable software properties. If you have the time, watch his talk at DDD Europe. It’s a fascinating topic!

<https://www.youtube.com/watch?v=WPgYju3KnIY>

“properties”

Extendible

Readable

Fragile

Adaptable

Rigid

Maintainable

Clean

Abstract

Coupled

Viscous

Elegant

Modular

Reusable



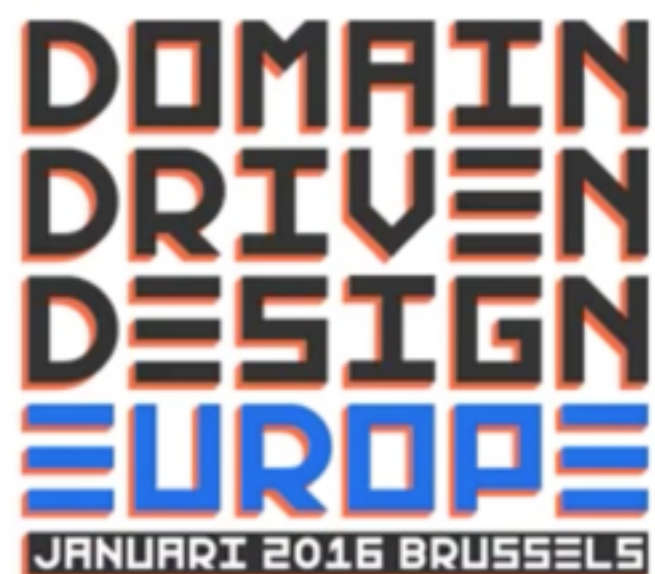
Carlo Pescio

*Software design and
the Physics of Software*

**DOMAIN
DRIVEN
DESIGN
EUROPE**
JANUARI 2016 BRUSSELS



Carlo Pescio
*Software design and
the Physics of Software*



“properties”

When the design preserving
methods are harder to employ
than the hacks, then the
viscosity of the design is high

Uncle Bob

Viscous

Coupled

Elegant

**HOW SHOULD
THE VISION OF
A PRODUCT
AFFECT YOUR
CODING
PRACTICES**



Say a client wants to build a new email client, why would users switch? What's the selling point?

**MOST
BEAUTIFUL
EMAIL CLIENT**

AS A DEV...

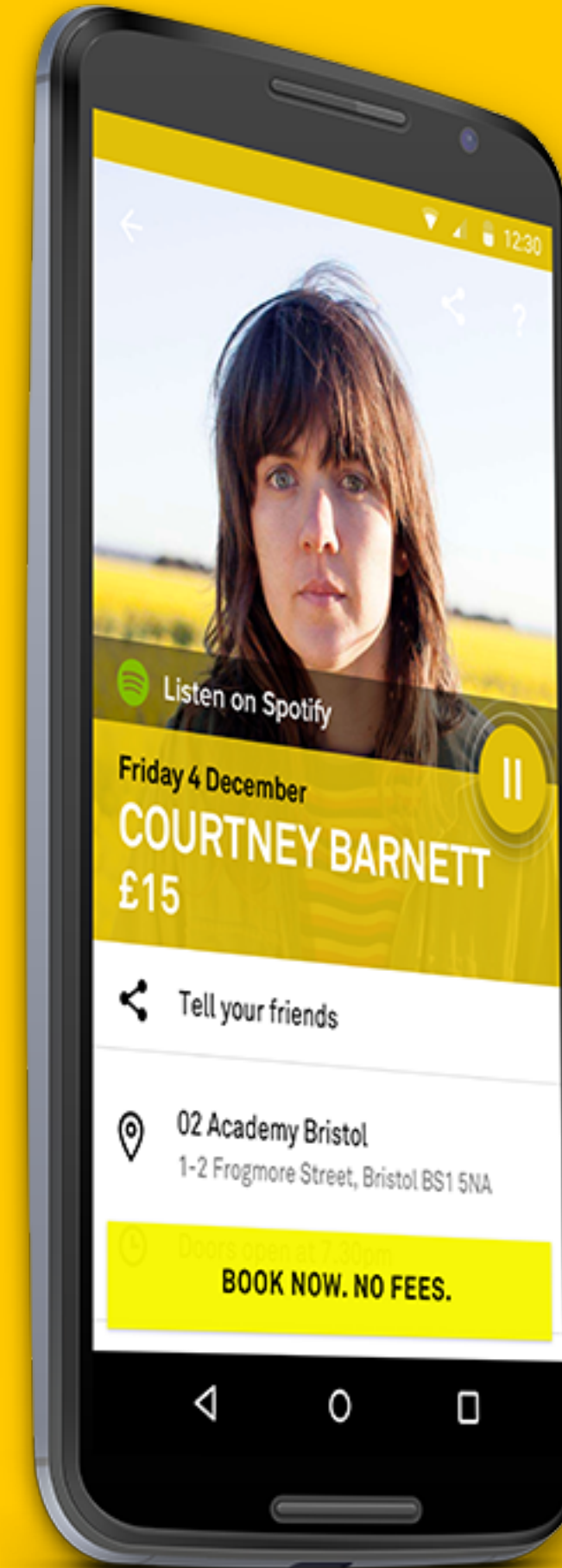
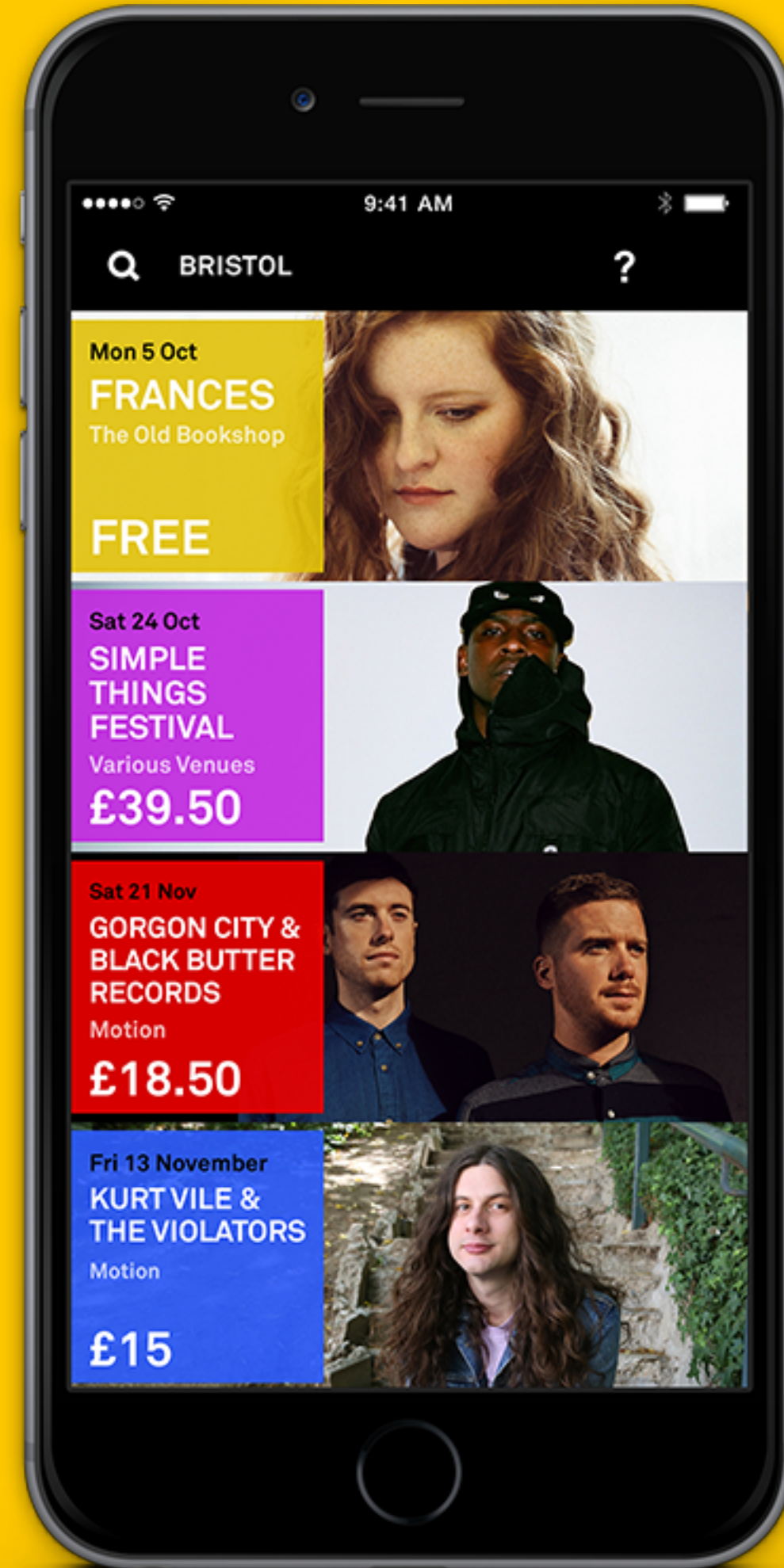
- **Would use 3rd party motion libs.** Bespoke effects, extremely rich UI.
- **TRADE OFF:** UI automation on rich UIs is very complex or sometimes simply just not possible.

**MOST
SECURE
EMAIL CLIENT**

AS A DEV...

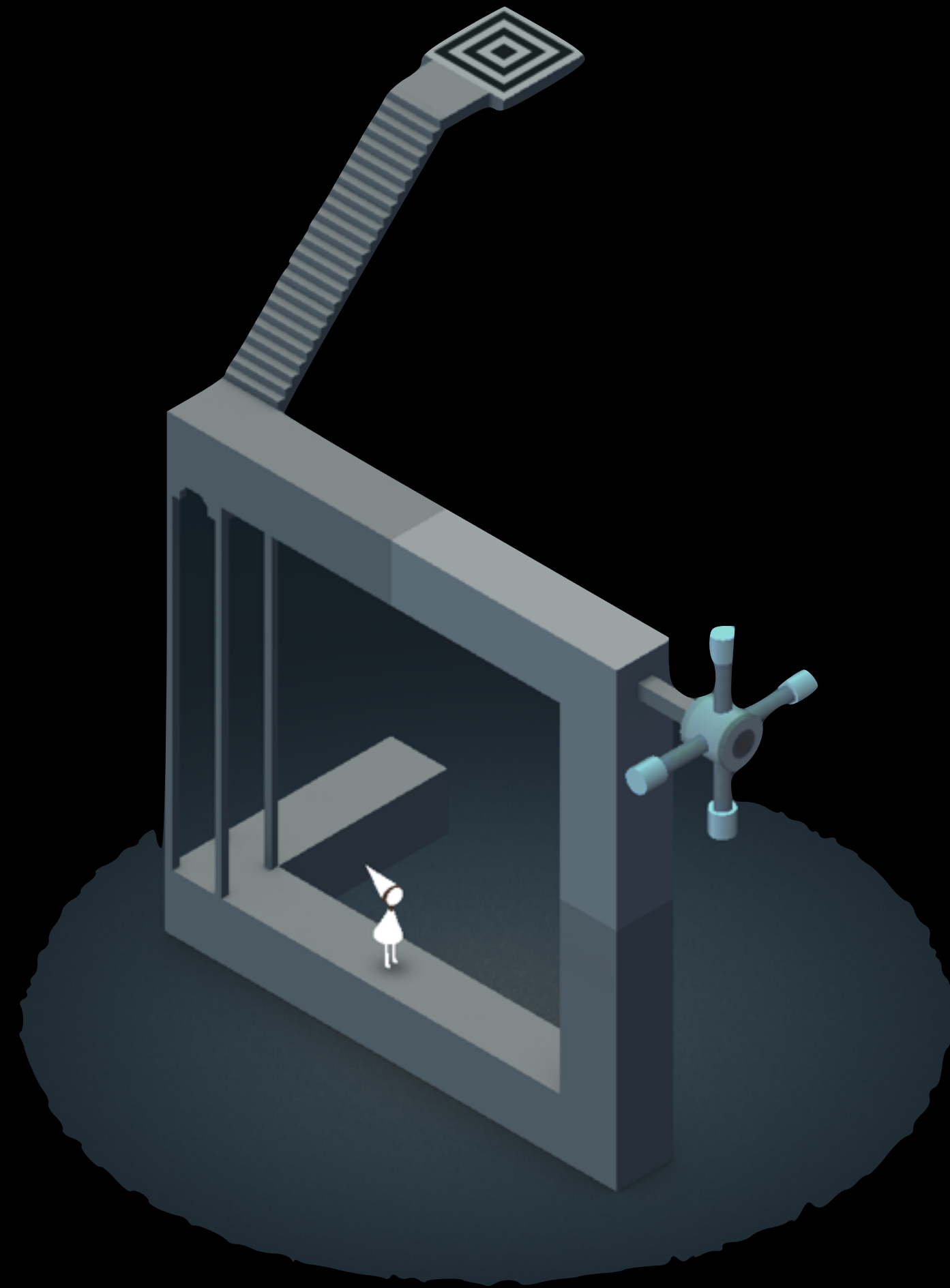
- **Would limit or avoid 3rd party software** since it's a source of security bugs through XBI or outdated dependencies (including dependencies of dependencies). Might even write or hire a sec expert to write own networking library.
- **Would strive to 100% test coverage**, no excuses.

**HOW SHOULD
THE MATURITY
OF A PRODUCT
AFFECT YOUR
CODING
PRACTICES**



Currently DICE.FM's backend is a modern web app: CI/CD, scalable, independent microservices...

Its 1st version was a Google Doc behind an API put together in half a day. Anything more would've been an unnecessary risk and potential waste.



Monument Valley is a stunning looking
ustwo game that sometimes anchor
new clients to beautiful UI execution.

This tends to be an issue early on
projects when we are focusing on
substance.

**HOW MUCH
QUALITY
IS ENOUGH
QUALITY**

THE HIGGIS PRINCIPLE



HIGGIS PRINCIPLE

**ENOUGH QUALITY IS THE QUALITY THAT
ENABLES:**

- **CORRECTNESS**
- **THE ABILITY TO ADD NEW FEATURES OR
ITERATE OVER CURRENT ONES IN THE
MID TERM**

EXAMPLES

JOCELYN GOLDFEIN

- **VP of Engineering** at VMWare
- **Engineering Director** at Facebook

The right way of building software.

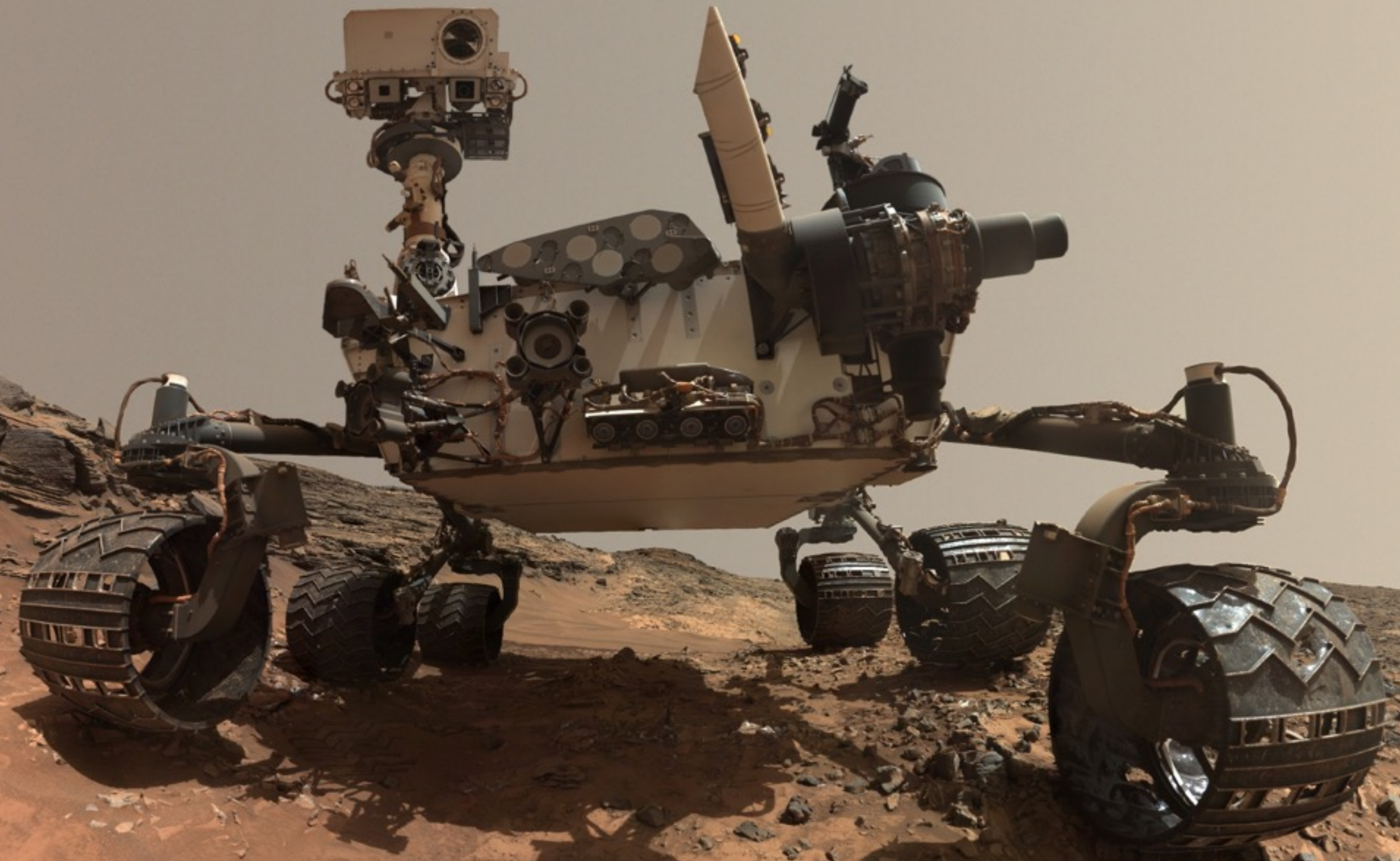
“When it was a startup, VMware needed to offer predictable dates and high reliability because they had to convince conservative enterprises to buy operating systems from an upstart new vendor.”

“In Facebook’s startup days, they needed to move quickly because first-mover advantage meant everything for a product based on network effects.”

Development practices at VMWare and FB were very different and yet both were **right** because they were aligned with product and business needs.

MARS SCIENCE LABORATORY AKA CURIOSITY

- 250 million miles away
- \$2.5 billion budget
- 40 people software team
- 5 years development
- OTA remote updates
- No access to PROD!
- 100% test coverage
- Logic verification
- ~80 lines of coding / day for the whole team
- Only 10 coding standard rules



TO SUM UP

QUALITY IS NOT AN INTRINSIC PROPERTY

“[...] quality often depends on the context in which a software component or feature operates. **The quality of a software component is not an intrinsic property** - the exact same component can be of excellent quality or highly dangerous depending on the environment in which it operates or the intent of the user.

[...] contextual nature of software quality is a fundamental challenge [...] **what is elegant in one situation might be downright unworkable in another”**

SO THERE'S NO WRONG WAY OF BUILDING SOFTWARE?

There are very wrong ways of building software, but they have nothing to do with isolated dev practices (TDD, BDD, CI, etc.). It has to do with applying those practices when and if they are appropriate and inline with product and business needs.

**“BUILD ME A
SOFTWARE”**

- No one, ever.

FRONT AND BACK

Why do we still split teams between server and client software? Users don't care about front and back, they care about working software.

Enough front and back, it's all about the product.

QUOTES

It's ok if you don't agree with them or make you slightly uncomfortable as a software developer. They are here to challenge your assumptions and make you reflect about your own coding practices. If you enter the debate, please be open minded and respectful!

Really good engineering

is finding adequate solutions
to problems that matter, in a
way that someone in the
future can understand and
improve on.




Jan Lehnardt

@janl

 **Follow**

Shipping means making trade-offs. We are putting our plugin feature on hold, to be able to ship @hoodiehq 1.0: [hood.ie/blog/hoodie-up ...](http://hood.ie/blog/hoodie-up)

<https://twitter.com/janl/status/669216118159069191>



Ideally, no one would write code that didn't make the app/site/product better. Engineering time spent writing code that isn't creating a better user experience or bringing in more revenue is a loss. Useless code creates technical debt and wastes everyone's time.

<https://medium.com/@iamchrstruman/product-responsibility-99c5bf2140d4>



Scott W. Ambler

@scottwambler

 **Follow**

Every practice has advantages and disadvantages, there is no such thing as a "best practice." dld.bz/euYza

<https://twitter.com/scottwambler/status/737593225603473408>



Steven Whitaker

@ThatSteveGuy

 **Follow**

YES!!! This -->

"It is OK to write ugly code if it helps your users."

<https://twitter.com/ThatSteveGuy/status/668851223475445760>



Gregory Brown

@practicingdev

 **Follow**

If software development wasn't a very high leverage way of making an impact on individuals and the world, I wouldn't be a programmer.

<https://twitter.com/practicingdev/status/734876082185371649>

THANKS

Juan Delgado @wadus