Intentional Experimentation

Why Everything We Learn Keeps Messing Us Up

GROWS™
Because one size does not fit all.
Agile Artisans
Who Am I?
Jared Richardson

Cofound of the GROWS Methodology

Author
Coder
Speaker

Started AgileRTP in 2007
Coach and Trainer

AgileArtisans.com
GrowsMethod.com
Learning to Ride A Bike

Remember?

Parents bought you a book

Took a 2 day course

Got a bike masters certification

Entered your first race

Sat on the bike for the first time!
Of Course Not

Study can’t teach you to ride a bike

Only a bike can teach you that
Learning to Code

Study or try?

Both!

Read this bit…

Then type “Hello world”
Experiments

↓

Experience
The only source of knowledge is experience

-Albert Einstein
Expertise Doesn’t Transfer

Remember Dreyfus Model?

Expert brain surgeon

Horrible cook

Threading vs UX

JS or Clojure

Can be an expert here…

… and clueless there
Our Expertise

Became an expert...

On the ~last~ project
Today is a New Project

New OS (Windows 10? El Capitan? Sierra? Pangolin?)

New language (Clojure? F#? Phoenix?)

New JavaScript library (What time is it??)

It’s all new!

Slightly different
Expertise Graph

Skill

Project One
Expertise Graph (Reality)
Really?

Waterfall

- Successful: 57%
- Challenged: 42%
- Failed: 14%
- 29%

Agile

- Successful: 49%
- Challenged: 42%
- Failed: 9%
- 9%

We’re Not Experts Anymore!

Dunning-Kruger Effect

If you think you’re an expert, you’re at risk!
Let’s Assume...

Past experience is of less value than we’d hoped

The ability to learn is invaluable

We can minimize risk by testing and proving assumptions

Every step of a project has experimental opportunities
How?

Let’s focus on a few areas…

Requirements

Architecture

Languages & libraries
Requirements

The Expert vs The Learner
Requirements: The Expert

I know what the customer needs
I’ll go write all the requirements for the next release
Leave me alone
Feedback slows me down
Requirements: The Learner

I’m not sure

I have ideas

I’ll write down one

Discuss with a customer

Review with developers and testers

Implement one

Show a customer

Adjust and try again
Include the Golden Triad

Before significant time and $$$ is invested

Dev

QA

Customer
Architecture

What do you do today?
Architecture: The Expert

I’m an Architect!

The last 5 projects? Development’s fault.

The Cloud is just like the mainframe

Tweak a few slides…

Here’s your slide deck architecture

Go implement it

BAM! (The Disappearing Architect)
Architect: The Learner

Tracer Bullet Development

Thin, vertical slices

Validate that technologies interact cleanly

Executable architecture

Engaged partner, working alongside
Technology Choices

- JavaScript
- Database
- Language
- Web stack (LAMP, WAMP, MEAN, ?)
Technology Choices: The Expert

This worked last time

Shut up and code

I know what I’m doing

It’s what I already know, so it’s what we’ll use again
Technology Choices: The Learner

Let’s try it

Let’s code a POC using it

Spend a few days… then regroup to discuss

That didn’t work as expected

This is great!
Why Doesn’t Everyone Do This?
I’M LATE! I’M LATE!
Don’t sharpen the axe
I don’t have time to learn
SHUT UP AND CODE!
Learn from history?

Re-implement the same bad patterns…

And we’re too busy to break out of the rut

flickr.com/photos/lauritopias/15594799343
THE KEY TO WRITING SOFTWARE: IS TO WRITE SOFTWARE
Developers disagree on which Javascript toolkit is best.

Developers disagree on which architecture is best.

Frequent demos discover communication failures early.

Continue until you can demo.

Demo.

Rewrite

Rewrite Adjust

Pick one

node.js
io.js
Vert.x

SOA
Monolithic
Peer to peer
Guidelines for Habitual Experimenters (from GROWS)

Intentional (setup loop before starting any activity)

Short (approaching real-time)

Real-world (e.g., “as deployed”)

Iterative (steady, cyclical rhythm)

Incremental (growing small piece at a time)

Anti-fragile (make it stronger when it breaks)
Source of Truth

Books often mislead
Friends overstate
Internet lies
Experiments strip away biases, prejudices, and assumptions
Experiments reveal reality
Experiments

This IS how it works

Not what I thought

Not what I expected

It’s easy to set up

It’s difficult to get started

Too slow

Ugly

Beautiful

Truth

flickr.com/photos/sandwichgirl/5109314424
Remember the Suck Curve

Moving from expert to novice is painful
Remember the Suck Curve

Moving from expert to novice is painful.

Killer Curve

Unconscious Incompetence

Not aware you suck

Conscious Incompetence

Aware you suck

Conscious Competence

Working hard to suck less

Unconscious Competence

You no longer suck
No Pain, No Gain

If something’s not hurting, you’re probably doing it wrong

Get comfortable being uncomfortable

flickr.com/photos/aandaphotography/8703788718
Summary

Assume what you just learned still works and you’ll be a statistic

Question everything

Prove every assumption

Validate choices before they’re permanent

Short, time-boxed experiments
Your Turn

Write down 1 situation from your recent past…

Experiments would’ve saved you

Bad technology choice

Bad architecture decisions

Bad requirements

How would’ve experiments helped?

Write down 1 and pass it to the front
Experience the 2 day GROWS Method workshop with your team!

GrowsMethod.com

jared@GrowsMethod.com

AgileArtisans.com