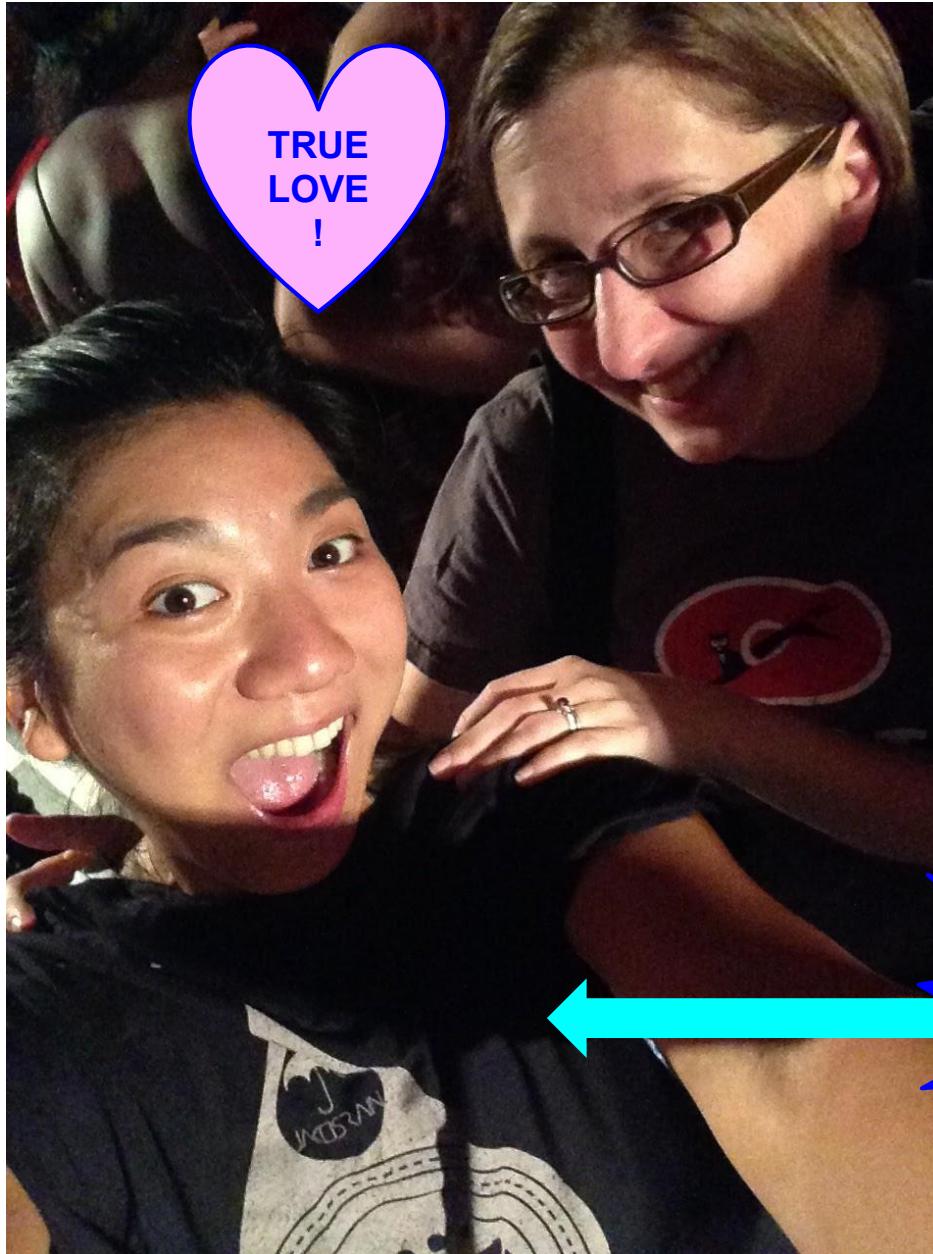


# I Don't Know What the F\*ck That Means!

Jenny Liu and Carol (Nichols || Goulding)

@liumonade and @carols10cents

# Carol and Jenny, sweaty-faced, at a concert





# The Agenda

- A Communication Problem
- How it happens (storytime!)
- Meta-Lessons
- What can YOU do about it
- Cat pictures (purr purr)

**So what's the  
problem?**

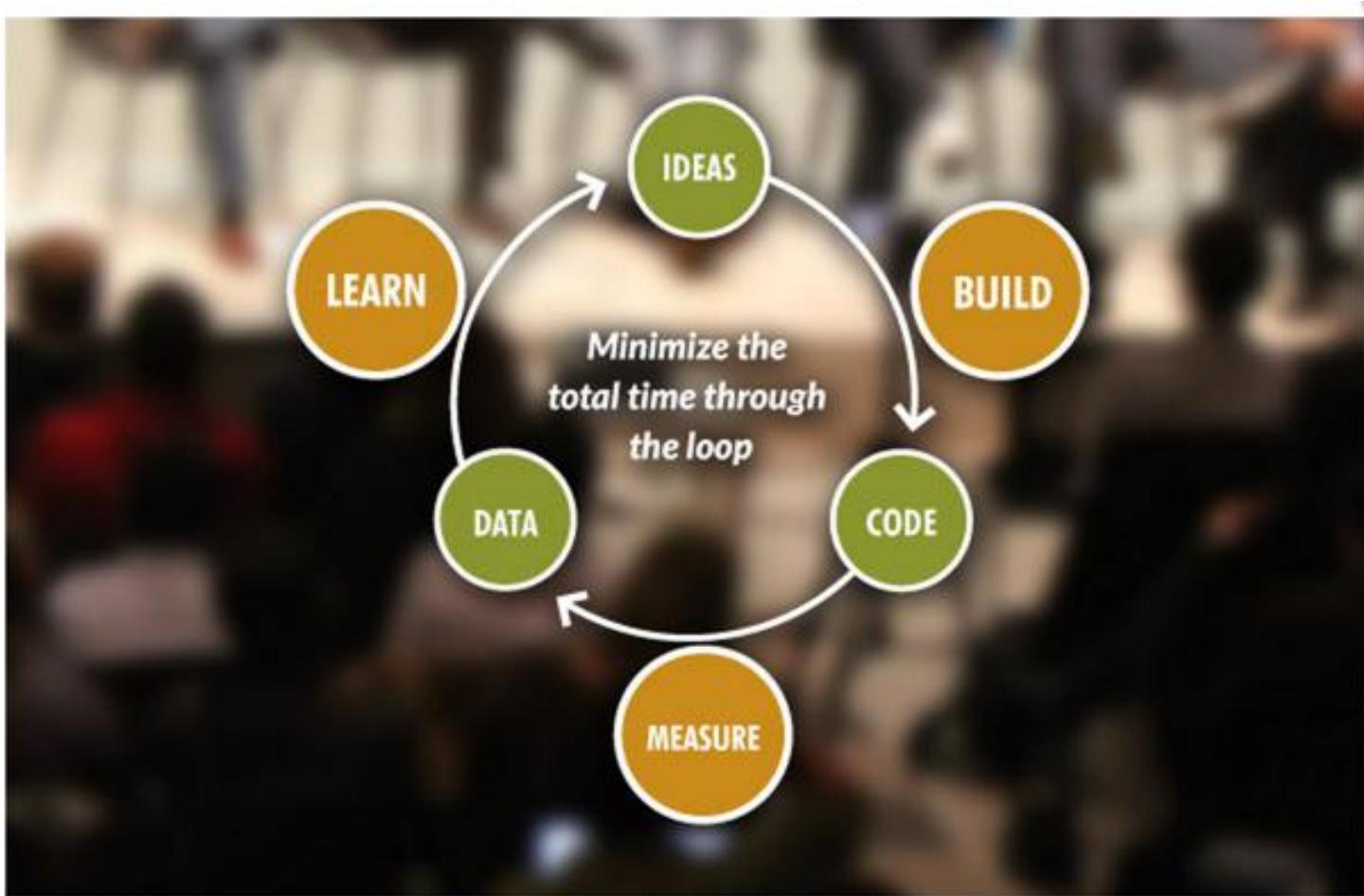
# **Exhibit A:**

# **MVP**

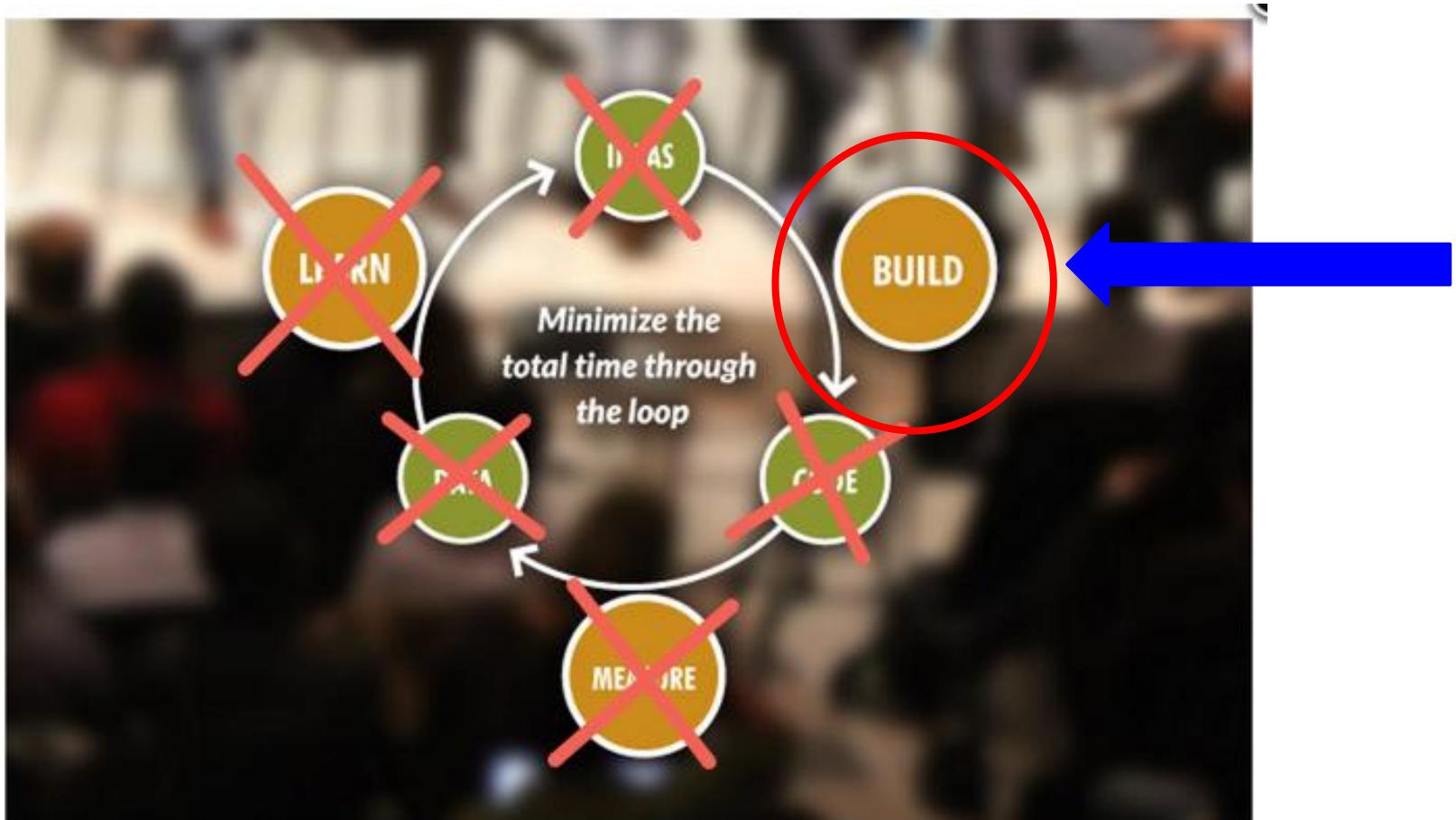
**aka ...**

**Minimum  
Viable  
Product**

# Eric Ries' Build-Measure-Learn Loop



# Not ideal, but at least it's honest





Unusable product  
(embarrassing)

Minimum Viable Product  
(loveable but limited)

Complete product  
(expensive)

# **Exhibit B:**

# **Technical Debt**

# Dat Technical Debt Backlog

Tech Debt Backlog Private

Show Menu

## Backlog

Upgrade Apangea to PG 9.4

Make Live Teaching Scalable

Merge apangea databases

Migrate to slack

Standardize Apangea Styles - Teacher/Admin

Convert PSP to Structured Math Content

Send down LP student + lesson data in one payload.

Convert ActionMailer templates to Mandrill

Add a card...

## Doing

Enforce LP refresh when a new version is available.

Upgrade Apangea to Rails 4

Upgrade DW to Rails 4

Remove Old LP

Move Live Teaching to Scalr

Add a card...

## Done!

Upgrade to ruby 2.1

Refactor the Attempter and Attempter subclasses

Configure deploy tracking in bugsnag  
<https://bugsnag.com/docs/deploy-tracking-api>

Use Rubocop in the build pipeline or in the PR with Hound

Student session should never timeout in dev mode

Using twice as many database connections as necessary

Declare migration bankruptcy and rely on schema load (which we do anyway)

Add a card...

## Ongoing Efforts

Deprecate erb in Apangea in favor of HAML templating.

Add a card...

# Is it Technical Debt?

**Ask yourself... If you said no to even one...**

*Is the code clean?*

*Is the code tested?*

*Is there a learning objective?*

*Is there a plan for payback?*

*Is the business truly informed?*

**Then you don't have Technical Debt**

**Or is it cruft?**

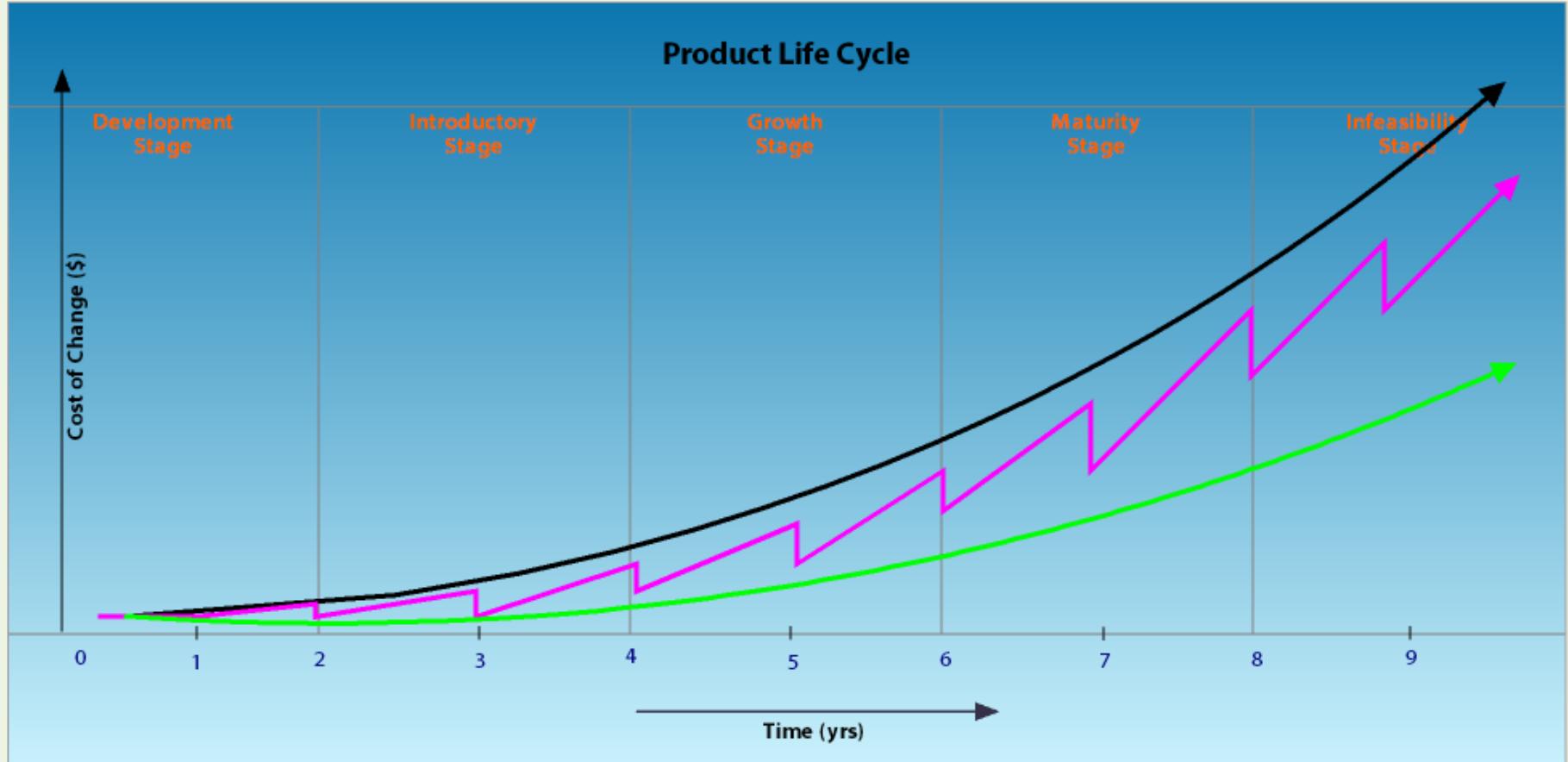
# **Intentional Cruft**

**(naughty naughty!)**

**vs**

# **Unintentional Cruft**

**(oops, i did it again)**



# **Exhibit C:**

**And for our last term ...**

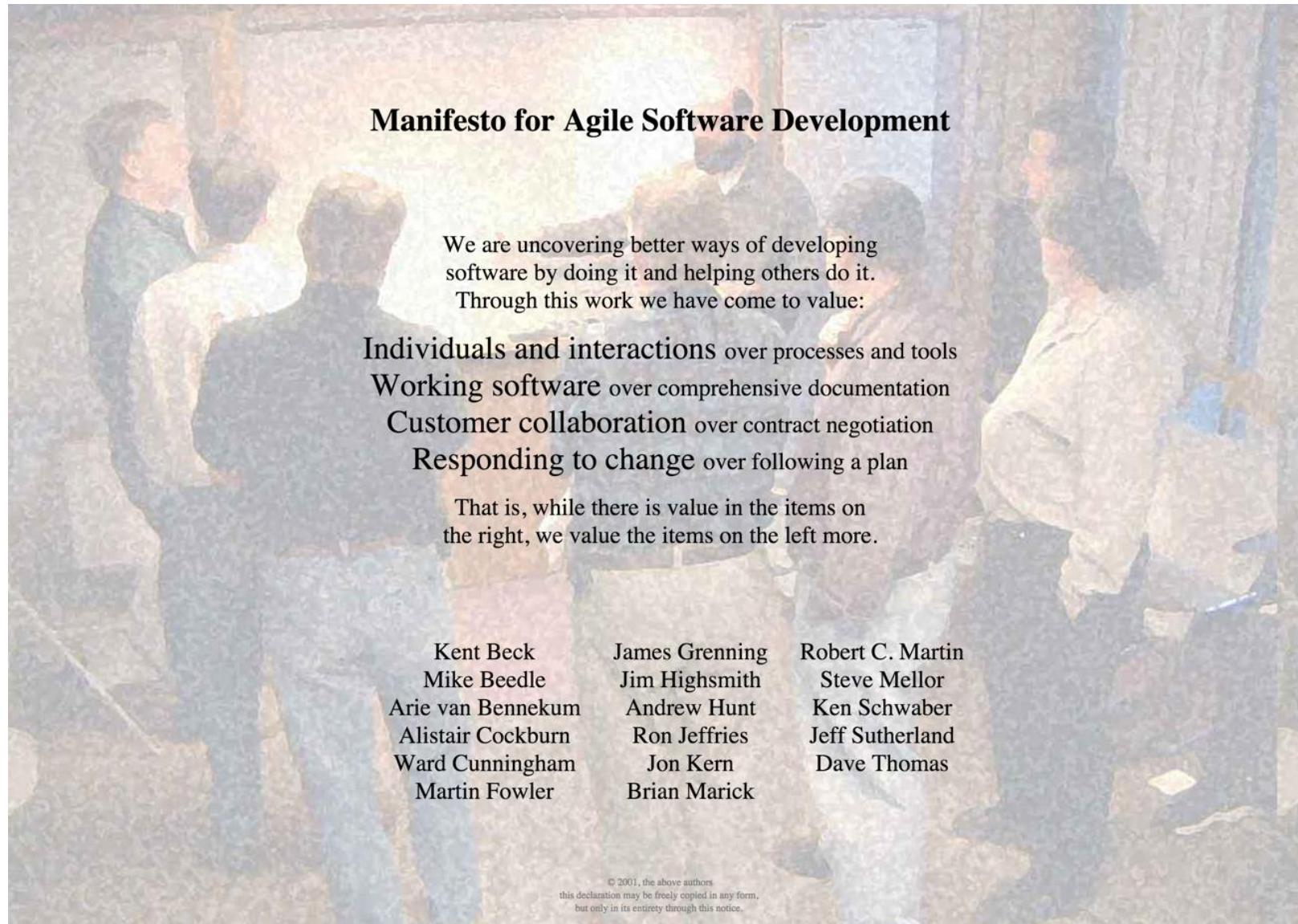
**Agile**

**How have YOU heard agile  
defined?**

# Here's what we've heard:

- Agile is when you do standups
- Agile is pair programming
- Agile is measuring progress with points and burndown charts
- Agile is the opposite of waterfall
- Agile is Scrum
- Agile means no planning
- Agile means no documentation

# Who doesn't love an origin story?



(this screenshot is probs too difficult to read)

# Ah, that's better!

## The Agile Manifesto

**Individuals and interactions**

over

Processes and Tools

**Working Product**

over

Comprehensive Documentation

**Customer Collaboration**

over

Contract Negotiation

**Responding to change**

over

Following a plan

*That is, while there is value in the items on the right, we value the items on the left more.*

# **Meta-Lessons; or, the Big Picture**

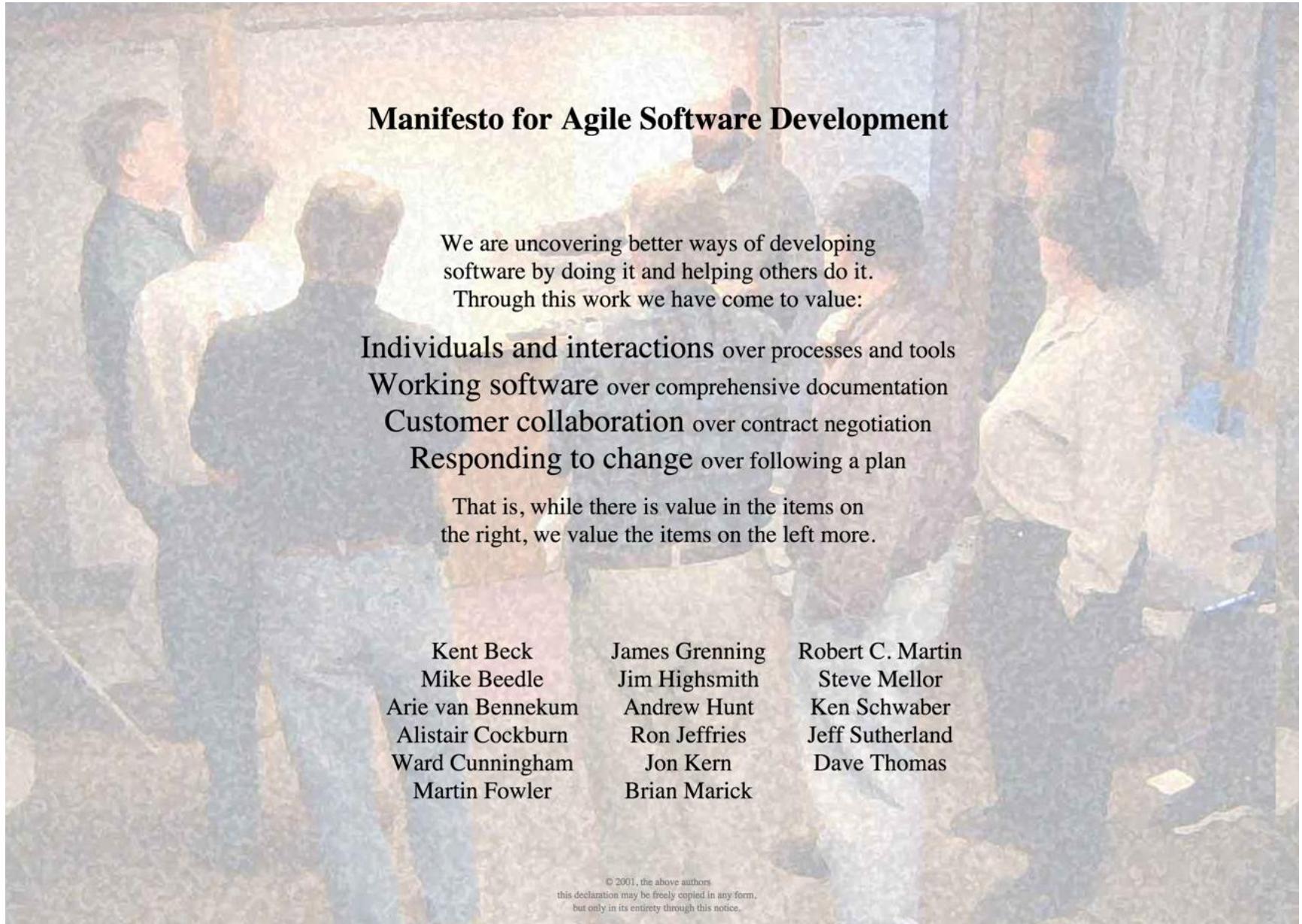
**So what are we *really* saying?**

# Brace yourselves



## FOR SOME DISCOURSE

# Meta-Lesson I: Consensus < Plurality



# Meta-Lesson I: Consensus < Plurality

## Manifesto for Half-Arsed Agile Software Development

We have heard about new ways of developing software by paying consultants and reading Gartner reports. Through this we have been told to value:

**Individuals and interactions** over processes and tools

*and we have mandatory processes and tools to control how those individuals (we prefer the term 'resources') interact*

**Working software** over comprehensive documentation

*as long as that software is comprehensively documented*

**Customer collaboration** over contract negotiation

*within the boundaries of strict contracts, of course, and subject to rigorous change control*

**Responding to change** over following a plan

*provided a detailed plan is in place to respond to the change, and it is followed precisely*

That is, while the items on the left sound nice in theory, we're an enterprise company, and there's no way we're letting go of the items on the right.

**Excerpts from the Half-Arsed Manifesto:**

**Customer collaboration  
over contract negotiation**

(within the boundaries of strict contracts, of course, and subject  
to rigorous change control)

**Responding to change  
over following a plan**

(provided a detailed plan is in place to respond to the change,  
and is followed precisely)

*That is while the items on the left sound nice in theory, we're an enterprise company and there's no way we're letting go of the items on the right*

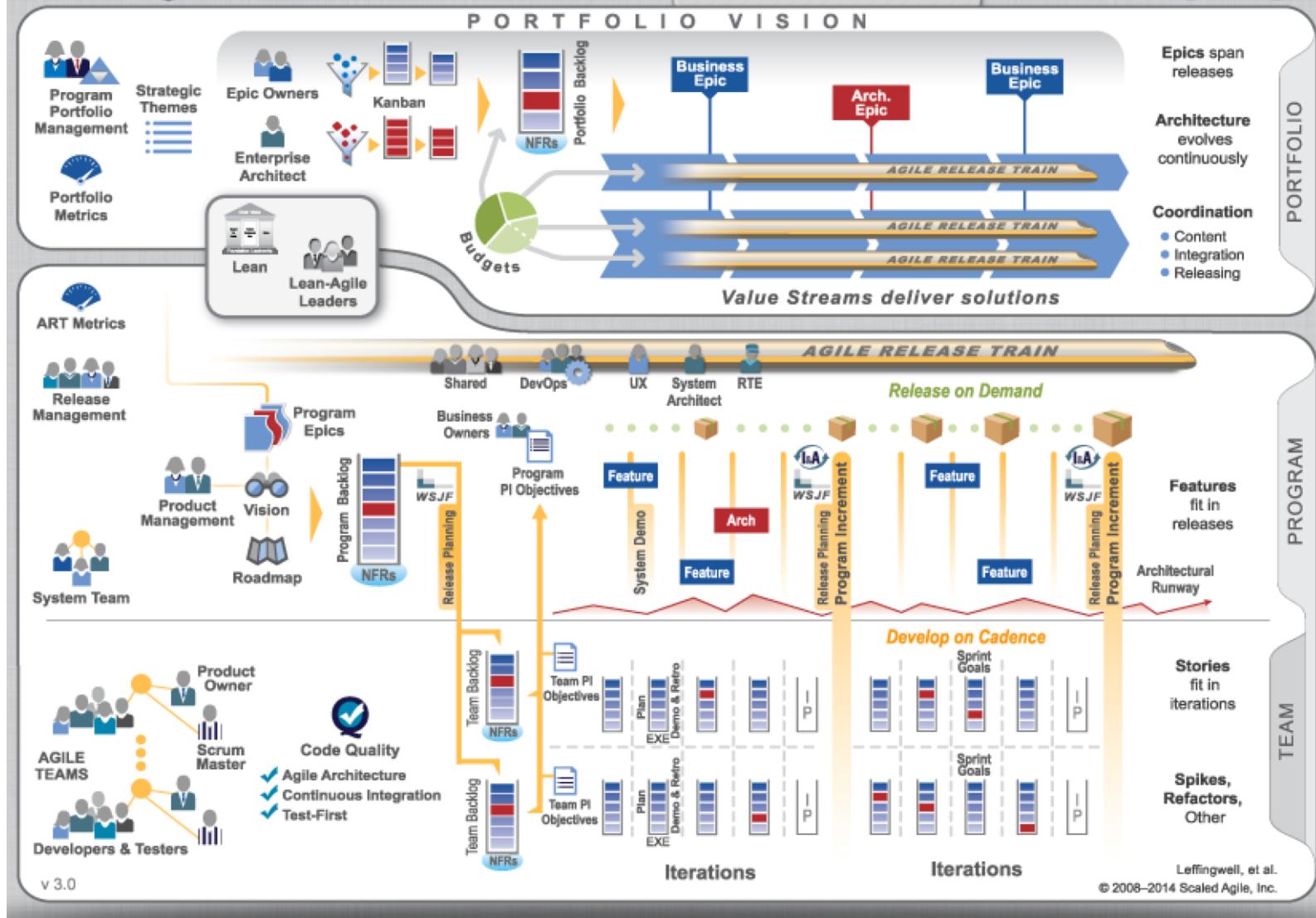
# “As Scrum is to the Agile team,

Scaled Agile Framework® 3.0



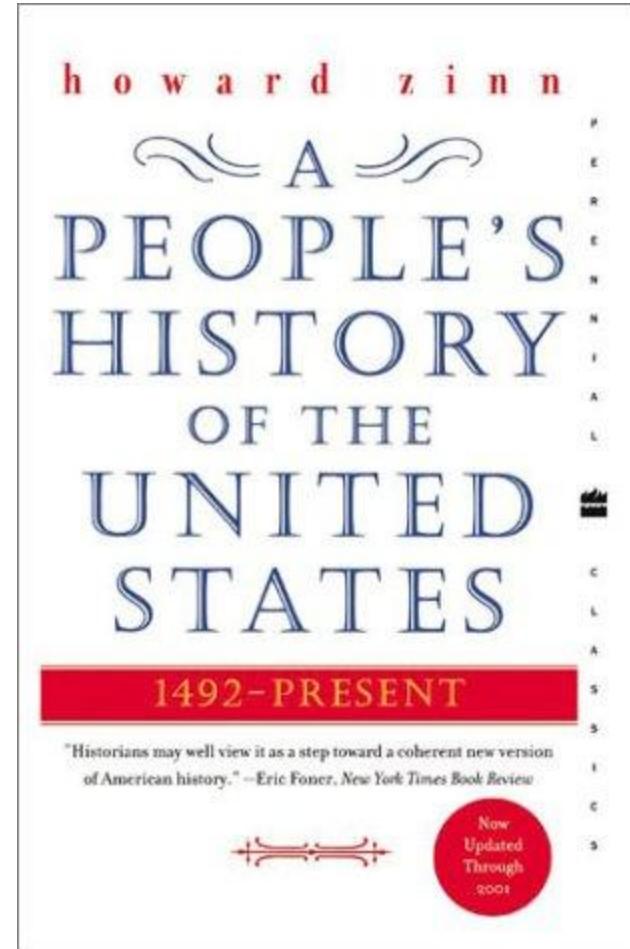
CLICK ANY ICON  
for detailed information

**SAFE**

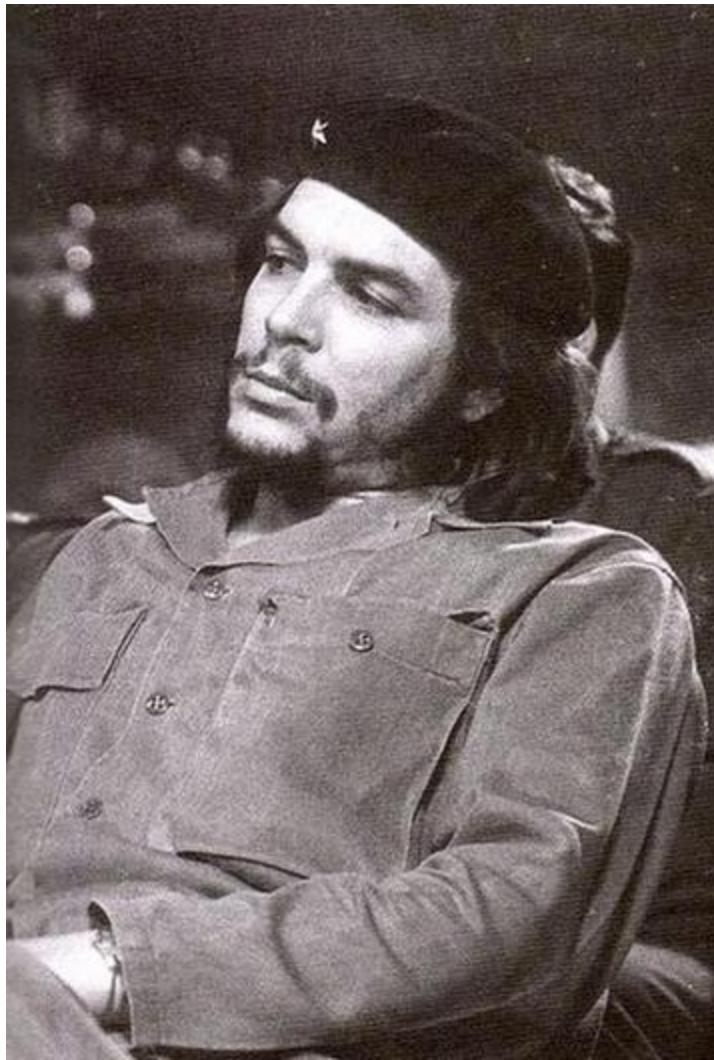


## SAFe is to the Agile enterprise”

# To be clear, plurality is GOOD



# Meta-Lesson II: Mainstream adoption is unavoidable



Only  
\$20.99 on  
Amazon!

# .... and the cycle continues



A Modern Software Development Suite. Because one size does not fit all.

[Home](#)   [FAQ](#)   [Contact](#)

## Evidence Based

GROWS™ is an empirical, evidence-based process. You don't blindly adopt any new practice: instead, you try an inexpensive, quick experiment. You begin with a *Directed Empirical Approach* that provides necessary support for experiments and demonstrates positive results early, helping you and your team see where the best value and best opportunities lie. Working with local evidence and tight feedback loops guides your growth to higher stages.

## The GROWS Method™

Grow  
Real-world  
Oriented  
Working  
Systems

An empirical, anti-fragile, pragmatic and evolutionary approach for the 21<sup>st</sup> century.

Follow [@growsmethod](#) on Twitter.

**What's with this ™ everywhere??** See [The FAQ](#) for the answers to this and other pressing questions.

## Skill Driven

The Dreyfus Model describes major differences in people as they acquire a skill. It defines five stages we all experience as we learn a skill, how we change along the way, and what we need to succeed. The GROWS Method™ uses these stages to implement skill-appropriate practices for the team members, team, and executives.

## Dynamic, not Static

Too many silver-bullet solutions promise a universal fix for your team; both agile and plan-based approaches only offer a static set of accepted practices. But the world is more complex than that, and your team isn't the same as mine, or anyone else's. And more importantly, your team as of yesterday is **not the same** as your team of tomorrow. You need a more personalized, dynamic approach to meet changing demands

## Local Fit

GROWS™ takes the best ideas from modern software engineering and the agile software development movement and helps you determine when and how can you use them. Just because a practice or an idea worked for someone else doesn't guarantee it will work for you. Don't guess: find out with actual data.

## Find Out More...

email address

**Subscribe**

Sign up for more information on how you can participate and use

# A noble, but ultimately futile effort

from the GROWS website:

What's with the ™ all over the place?

We've seen first-hand how words like "agile" and "scrum" have been used and abused, and we'd like to avoid the same fate. Claiming GROWS as a trademark give us some measure of protection from companies misusing this word too.

***We've seen first-hand how words like "agile" and "scrum" have been used and abused, and we'd like to avoid the same fate. Claiming GROWS as a trademark give us some measure of protection from companies misusing this word too.***

**So what can we do?  
(tactically, that is)**

**TELL  
EVERYONE HOW  
WRONG THEY  
ARE!!!**

**TELL  
EVERYONE HOW  
WRONG THEY  
ARE!!!**

# Communicate More Effectively!

- **Ask all the questions!**
- **Check for understanding, and use active listening**
- **Switch mediums**
- **Don't use buzzwords; just say what you mean**
  - **If you *do* use them, use responsibly**
- **Use CRITICAL THINKING**

**... and now, some cat pictures!**

**(purr purr)**



**No one puts Baby in the corner!**









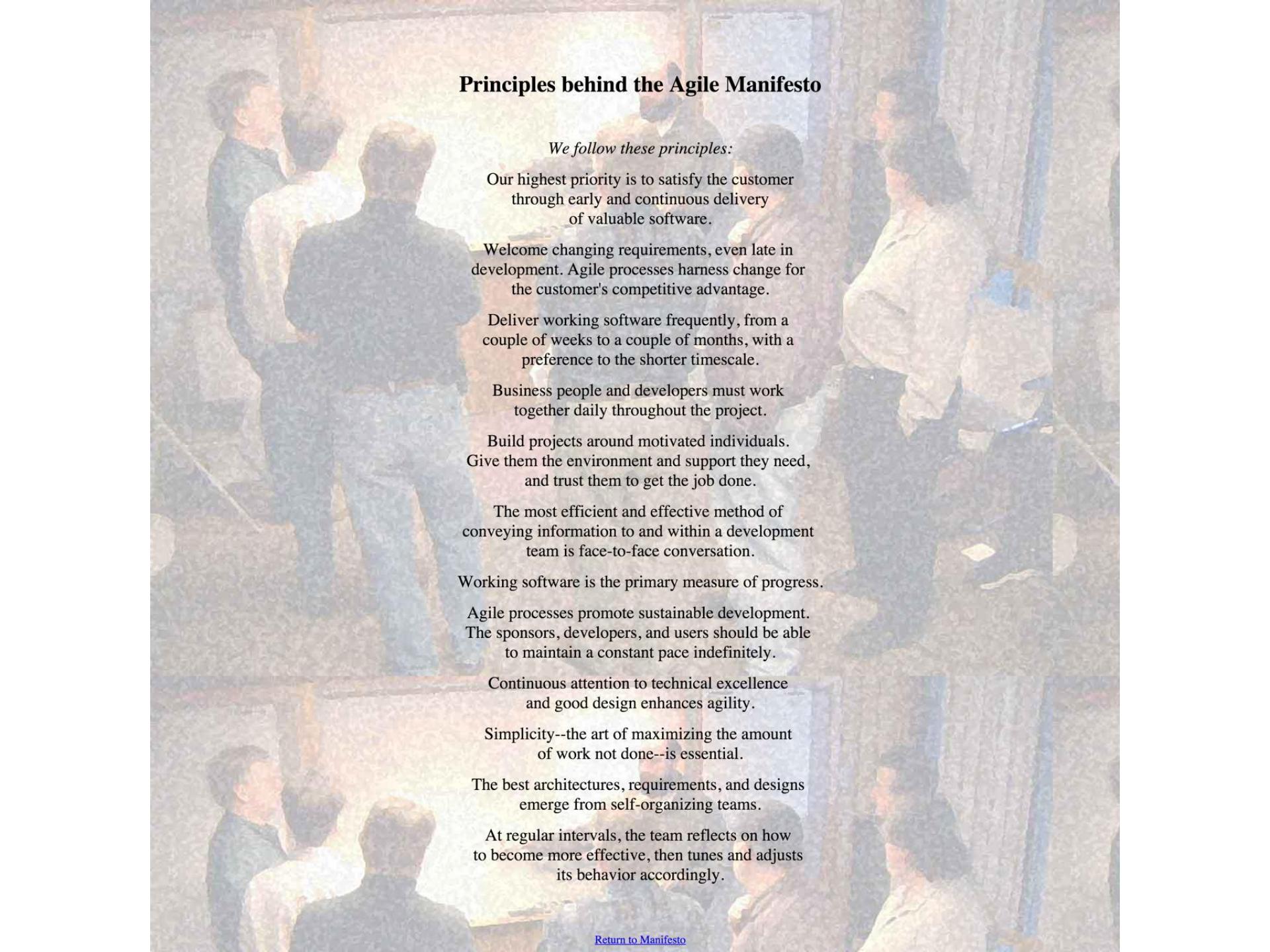
# References: Happy Reading!

- [How Spotify Builds Products](#) by Henrik Kniberg
- [The Lean Startup](#) by Eric Ries
- [Ward Cunningham on Technical Debt](#)
- [Doc Norton on Technical Debt](#)
- [Levent Gurses on Refactoring & ROI](#)
- [Manifesto for Agile Software Development](#)
- [Scaled Agile Framework](#)
- [Manifesto for Half-Arsed Agile Software Development](#)
- [Beyond Agile: New Principles?](#) by Ron Jeffries
- [The GROWS Method™](#) by Andy Hunt

# **Thanks for all the fish!**

Carol (Nichols || Goulding) and Jenny Liu

@carols10cents and @liumonade

A photograph of a group of people in a workshop or office environment. Some individuals are standing and engaged in conversation, while others are seated at a table, possibly reviewing documents or working on a computer. The scene is somewhat blurred, suggesting a candid, active workspace.

## Principles behind the Agile Manifesto

*We follow these principles:*

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

Simplicity--the art of maximizing the amount of work not done--is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.