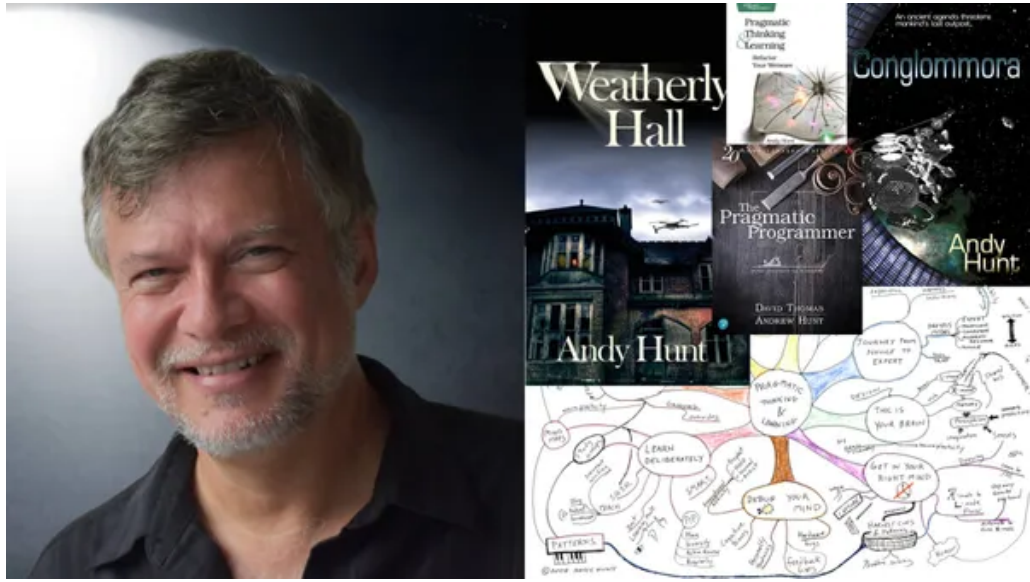
Ending The Age of Misery

Andy Hunt

- *The Pragmatic Programmer*
- 12 tech and fiction books
- *Agile Manifesto* author (1/17)
- Co-founder, The GROWS Method® Institute



What Are The Most Important Skills For A Software Developer?

Learning

Communication

Learning from the...

- Business domain
- Emergent properties of the system under construction
- Dynamics of the team
- New tech, updates, deprecations

Communicating with...

- Programming language/environment
- Subject matter/domain experts
- Users
- Team members
- Stakeholders, business owners
- Vendors/third parties

How Does Your Method Support Learning & Communication?

- Are these ideas a first-class part of your method?
- Does your method support these ideas *at all?*
- Ticket-driven development does **not**

"Scatter-Gather" / Ticket Driven Method

Scatter Phase:

1. Divide the project into pieces
2. Assign pieces of the project to "teams"
3. In the "teams", a lead divides the pieces up and creates tickets
4. Tickets are assigned to individuals who work for the lead

Adapted From Tim Ottinger, http://agileotter.blogspot.com/2018/07/the-scatter-gather-method-of-software.html

"Scatter-Gather" Method

1. Individuals complete their tickets
2. The small jobs are integrated/combined/summed by the lead
3. When small jobs are done, the lead delivers the finished work of the "team"
4. The integrated, completed project is delivered as soon as pigs take flight

Adapted From Tim Ottinger, http://agileotter.blogspot.com/2018/07/the-scatter-gather-method-of-software.html

Problems with Ticket-Driven Development:

- Small tasks disconnected with context
- There's no actual team
- Learning is siloed, happens individually, doesn't spread
- Communication gets harder

Communication And Learning Pathways

[n * (n-1)]/2

- Team size of 5 == 10 pathways
- Team size of 12 == 66 pathways
- Team size of 100 == 4,950 pathways

(Dr. Fred Brooks, The Mythical Man Month)

Communication And Learning Pathways

[n * (n-1)]/2

- How does learning get shared in these environments?
- What would be different if the **team** learned together?
- What would be different if the **team** developed code together? Like an actual team?
- What would be different if you didn't need tickets?

Why do we develop this way when we *know* better?

Agrarian Age → Industrial Age

- "Safety regulations are written in blood"
- 1900's London Smog, lethal Great Smog of 1952
- Triangle Shirtwaist Factory fire, 1911
- Child "spraggers" in mining...

Hardie, D. W. F., A History of the Chemical Industry in Widnes, Imperial Chemical Industries Limited, 1950.

Kids had worked alongside parents at home and farm.

Didn't translate well to factory work.

We had to learn how to live with the new technology.

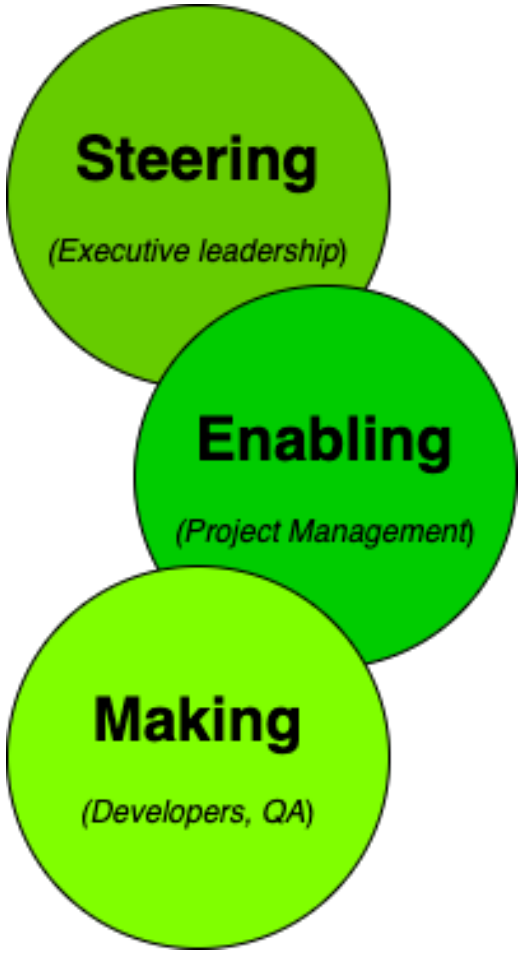
Industrial Age → Information Age

- Bogus "snake oil" tools & methods
- Endless ticketing systems
- Ponderous practices
- Disconnected leadership, stakeholders, product owners, etc.
- Angry users

Industrial Age → Information Age

- Destructive Social Media platforms
- Disinformation and manipulation
- Safety issues with self-driving cars, drones, appliances, etc
- Privacy issues with cameras, sensors, facial recognition, etc.
- Naive adoption of Large Language Models (chatGPT, et al. It ain't AI.)

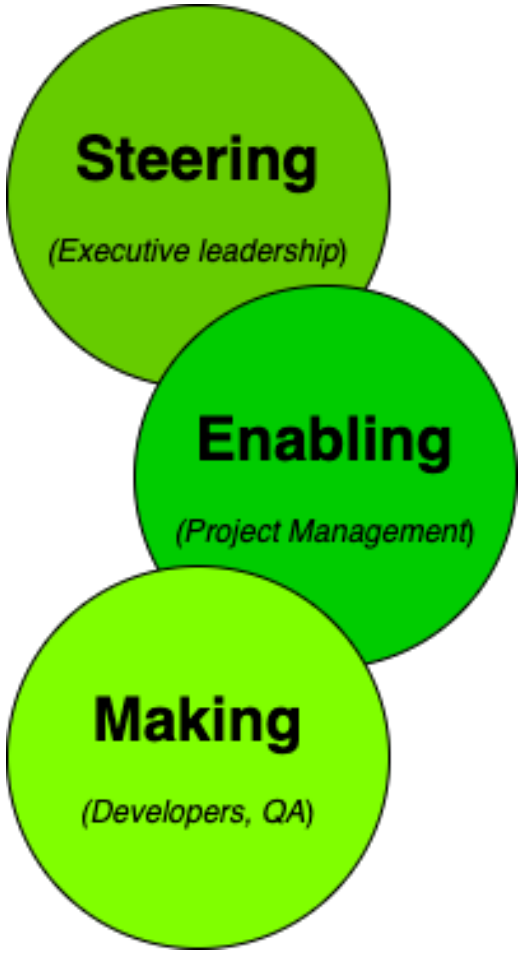
Our Roles



Via Esther Derby

What Do we Want?

- **Steering**: Speed, value. Ability to turn on a dime.
- **Enabling**: Manage uncertainty, seek reliability and predictability.
- **Making**: Autonomy. "Get out of my way and let me work."



Discovering Better Ways of Working

- Flow-based, always releasable
- Not auto mechanic or building construction
- Not greenfield development, but stuck-in-the-middle
- More like a surgeon: keep the body going
- **Dynamic**, not static

Eating Soup With A Fork

Companies embrace industrial age optimizations and slavery-era accounting practices

Knowledge Work != Factory Work

But it "kinda" works well enough...

Theory X

- Dislike their work
- Avoid responsibility
- Individual-goal oriented only
- Have to be micro-managed, threatened, to deliver
- Need to be told what to do
- Have no intrinsic incentive to work, need $$ rewards
- Viewed as **resources**

Douglas McGregor, MIT Sloan School of Management

Theory Y

- Work from their own initiative
- Involved in decision making
- Self-motivated to complete their tasks and deliver
- Seek and accept responsibility
- Work to better themselves
- Solve problems creatively and imaginatively
- Viewed as critical **assets**

Douglas McGregor, MIT Sloan School of Management

Which do you think is most appropriate for knowledge workers?

Are you a resource or an asset?

What Do We Need?

- Flow-based, CI/CD, Technical Excellence, Ensemble Programming →
- FINE Experiments, Fast Feedback Loops, Systems Thinking →
  - **F**ast Feedback, **I**nexpensive, **N**o permissions required, **E**asy
- Psychological Safety & Support, Self-Contained Teams →

What Do We Need?

- Engaged Leadership/Steering →
  - Dynamic/Adaptive Accounting ("Beyond Budgeting")
  - Move away from batch-oriented funding
  - Reward systems aligned to Theory Y behavior, not Theory X

What Do We Need?

- Escape the short-term Shareholder Trap

> *Unrestrained growth is cancer*

"Once a firm embraced maximizing shareholder value and the current stock price as its goal, and lavishly compensated top management to that end, the C-suite would have little choice but to deploy command-and-control management ***(aka Theory X)***, because making money for shareholders and the C-suite is inherently uninspiring to employees. The C-suite would have to compel employees to obey, even if this meant that employees would become dispirited. The authors didn't worry that **dispirited employees might become a critical constraint in an economy that would depend on innovation from engaged knowledge workers.** How firms were managed was not a matter of much interest to academic economists."

—Notes on *Theory of the Firm* by Michael Jensen and William Meckling, via Stephen Denning

I am of the generation of the microprocessor revolution.

We thought we could change the world.

agilemanifesto.org

- eXtreme Programming was *revolutionary*
- Ultimately co-opted and mangled to conform to status quo
- "agile" now == Half-assed Scrum plus Jira tickets

The Age Of Misery

- Resignations are up; Return-to-office mandates and burnout
- Of the 57,000 workers across 1,600 companies employee sentiment dropped **10x** faster since January 2023 than in the previous three years. *(BambooHR)*
- Tech industry happiness down **-145%** *(BambooHR)*
- **Half** of young women will leave their tech job by age 35 *(Accenture/Girls Who Code)*
    - *(Largely due to unsafe environments/culture)*

Enshittification of Business Models

"Here is how platforms die: first, they are good to their users; then they abuse their users to make things better for their business customers; finally, they abuse those business customers to claw back all the value for themselves. Then, they die. I call this *enshittification*."—*Cory Doctorow*

Complex

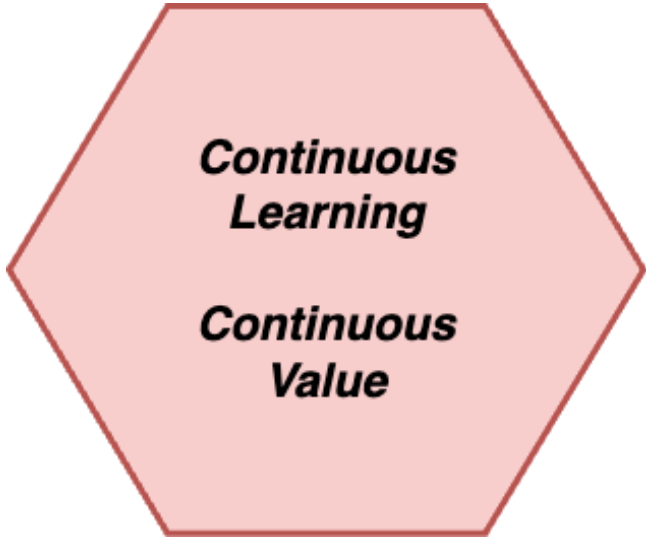
Not easily observable

No Identifiable Cause and Effect

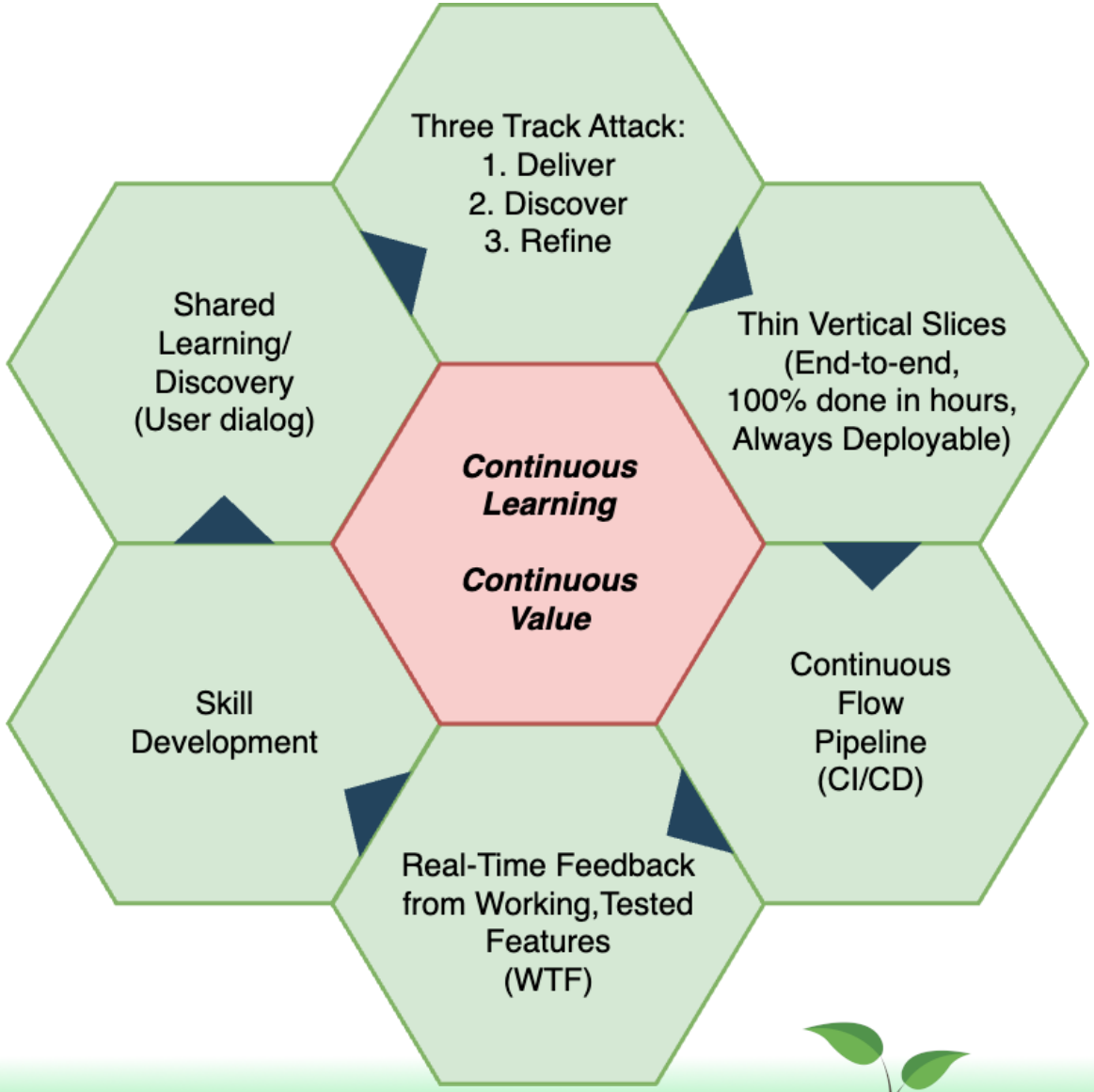
Can't Change "culture" Directly

No Single, Simple Solutions

The GROWS Model:



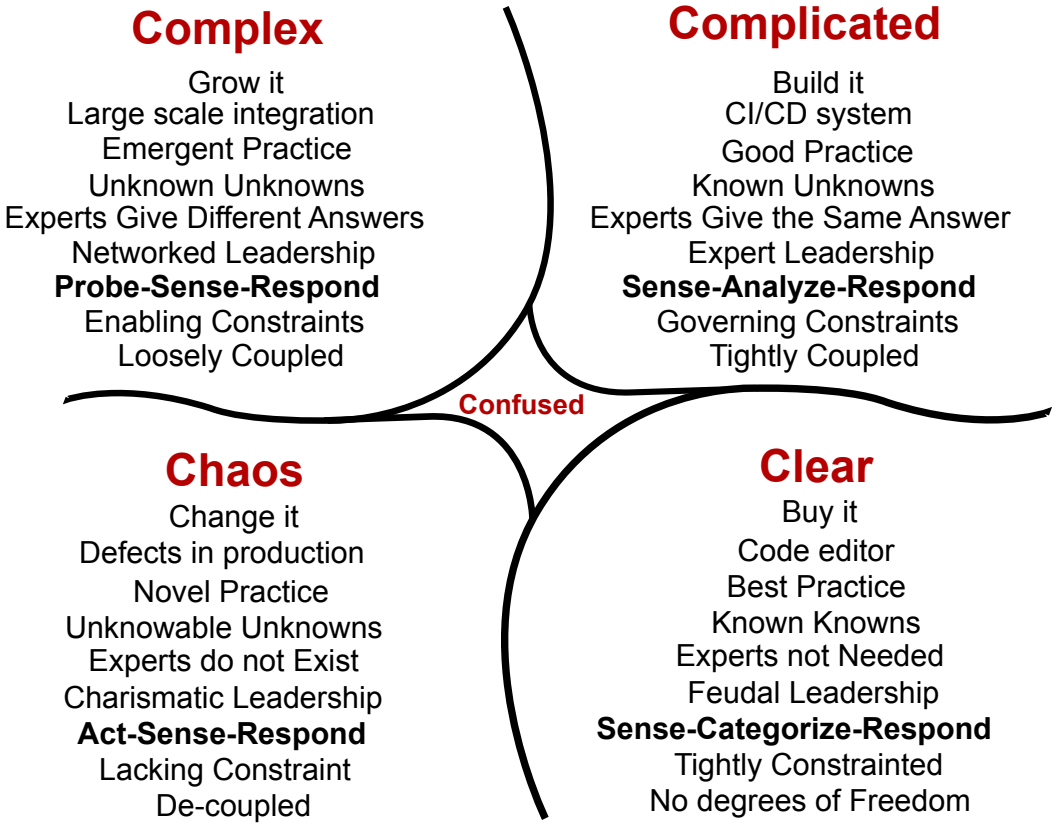
The GROWS Model:



GROWS Method®

The GROWS Model:

GROWS Method®



Practices are for Beginners

- Cynefin sense-making framework
- No "Best Practice"

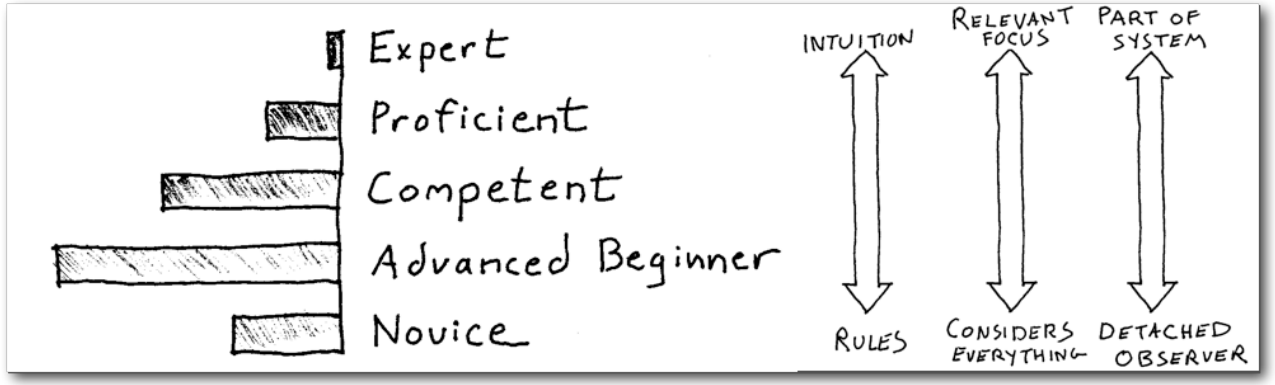
Dynamic Practices, not Static

> *"Practices can never be completely objectified or formalized because they must ever be worked out anew in particular relationships and in real time."*

—Dr Patricia Benner, *From Novice to Expert*

Rules, Intuition and Expertise

- Beginners need rules
- Experts rely on intuition from experience
- "Work to rule" limits you to a beginner's performance



Intuition from Experience

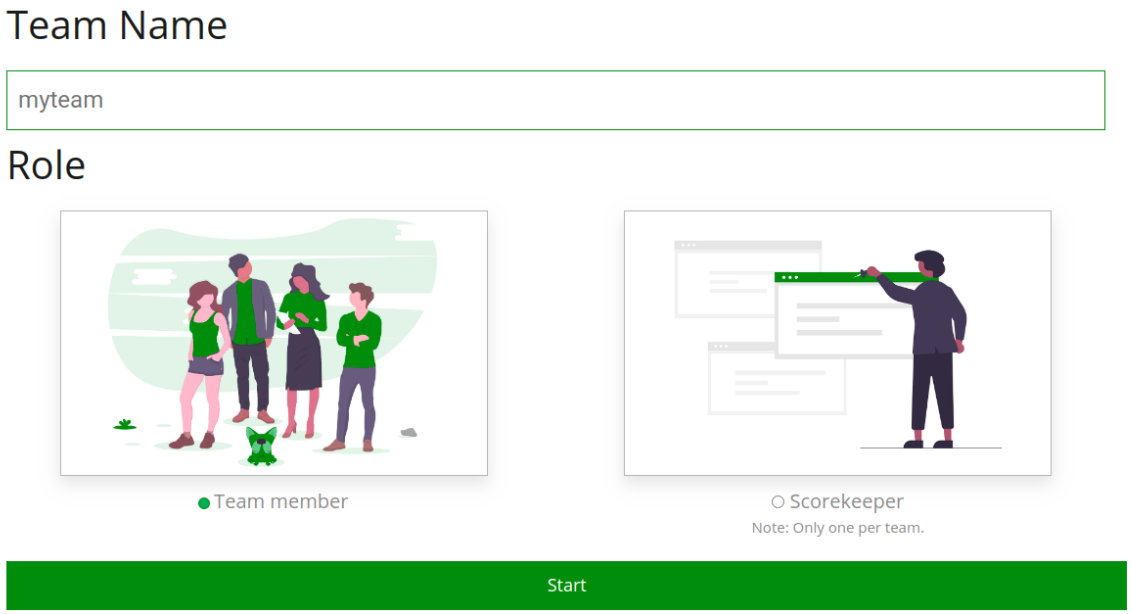
- But what if you don't have enough (good) experience?
- Many projects not "successful"
- Many work environments unsafe

How can you get experience?

- At psychological safety?
- At making good development/team decisions?
- At working with a team?

Simulated experience

- Experience: benefit of a simulation
- Psych safety: free to invent and evaluate options

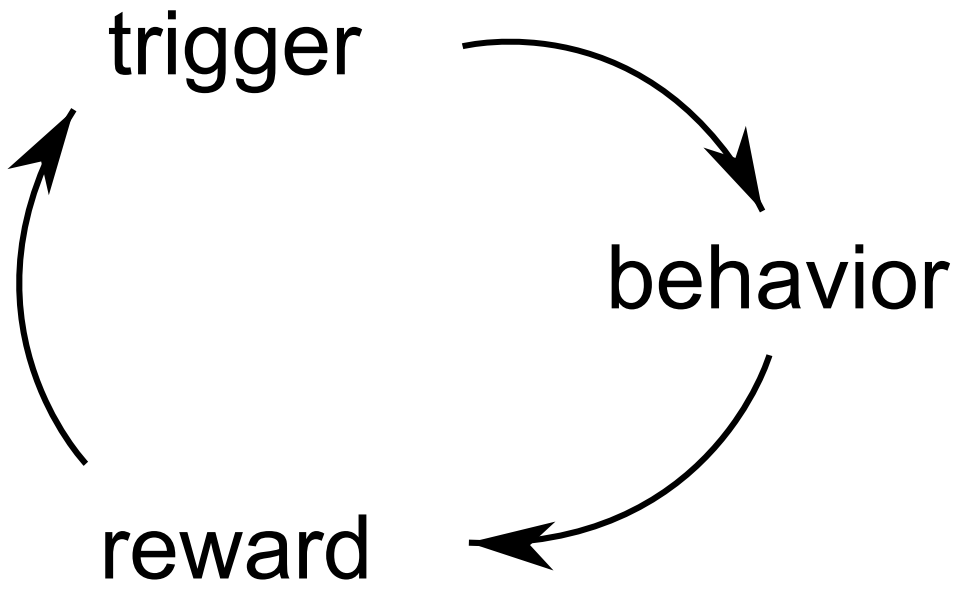


Habits from Experience

- Habits are often "subterranean," emergent from repeated behavior
- Continuous battle for *cortical real estate*

Can't Break an Existing Habit

But you can make new ones...



Habit Cycle:

- Can replace the "behavior" part

Key habits for "Theory Y" software development teams

What to fix first?

- Psychological Safety
- Reward Systems
- Small, frequent steps overall
- Direct User Involvement (no proxies)
- Short success horizon, End-to-end code (Tracer Bullet Development)

Psychological Safety

- Freedom to share thoughts
- Freedom to express concerns
- A sense of ease and support
- Ok to take calculated risks
- All options considered

Does *not* Mean:

- Saying anything you want
- Letting emotions fly
- Hands-off leadership
- Taking blind risks
- Tolerating everything, including bad behavior
    - "You get what you tolerate"

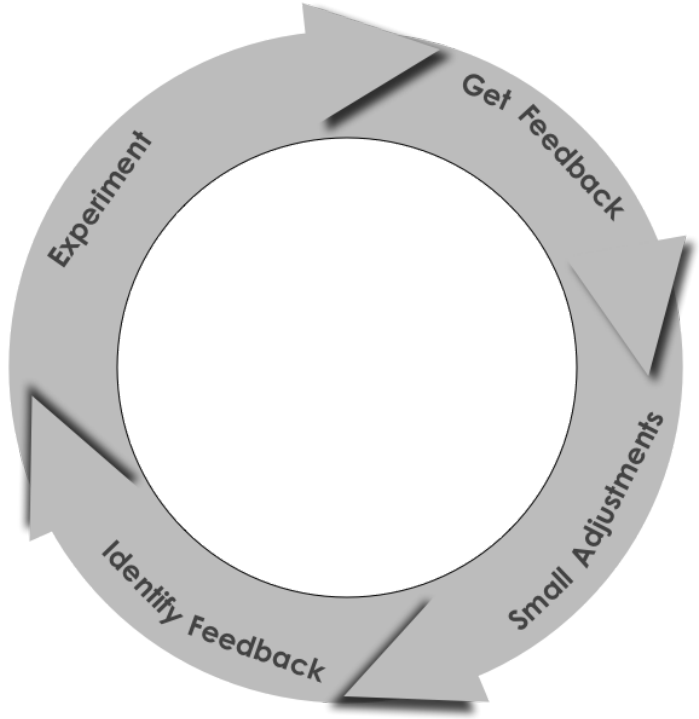
Activities that HARM Psychological safety

- Collecting metrics at the individual level
- Inappropriate comparisons between team-level metrics
- Unaligned reward structures (raises, bonuses, etc)

Always Look for Feedback

> *Get itchy, uncomfortable if there's no immediate feedback*

- TDD, Pair/ensemble programming, CI/CD, ...
- Meetings
- Technical decisions

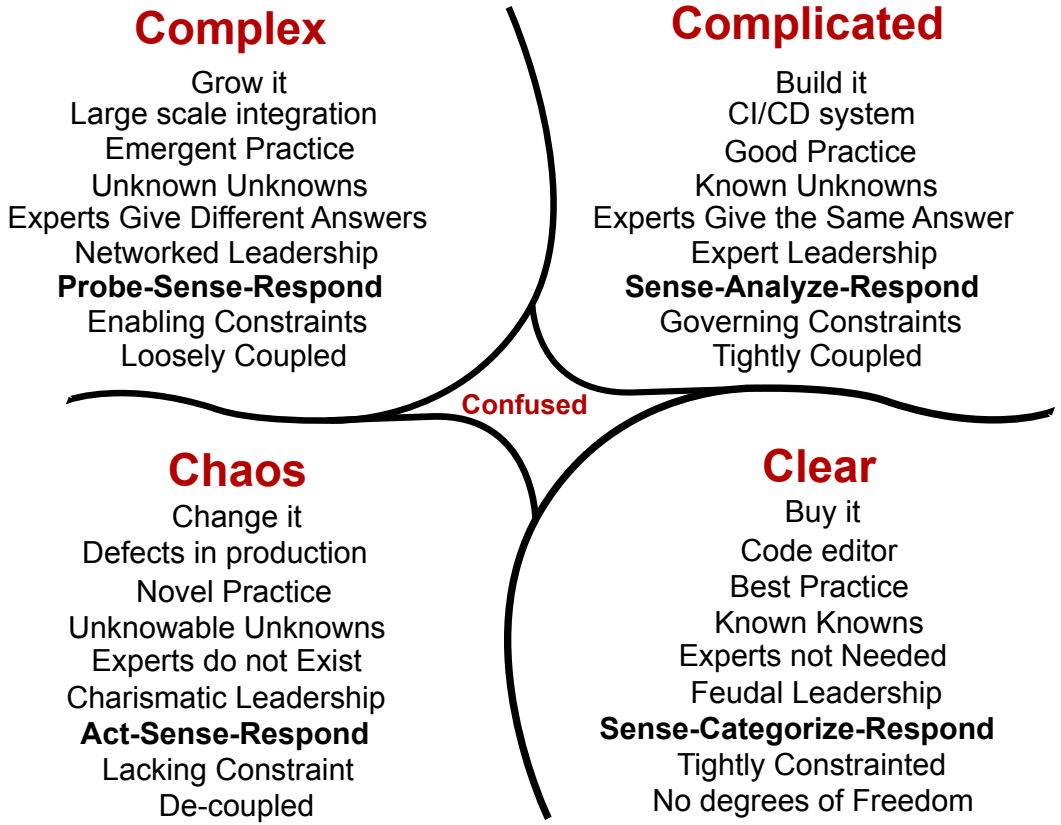


Experiment to Find Answers

> *Probe-Sense-Respond*

(Cynefin "Complex" Domain)

- User's options/issues
- Project's Technical questions
- Team's Process/Practices/Habits



Take Small Bites Always

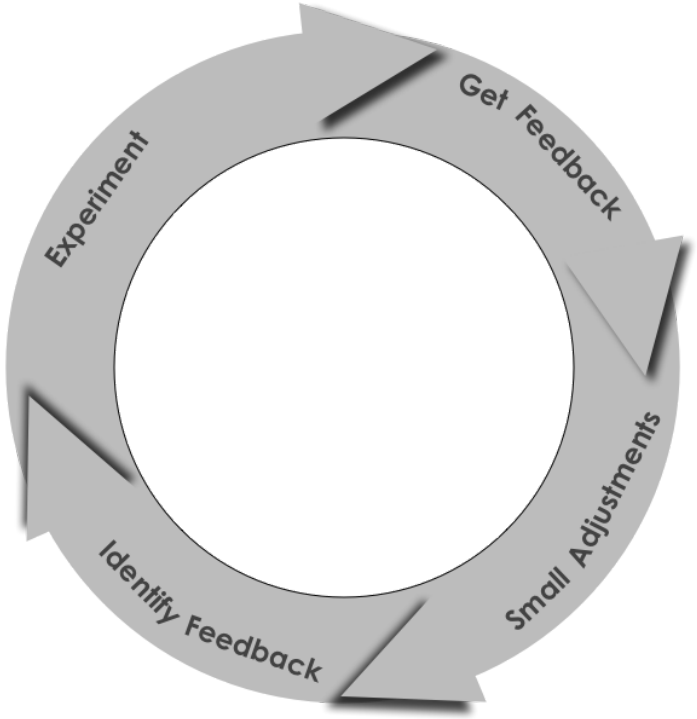
> *However small a step, make it smaller*

- Small achievable likely to succeed
- Faster feedback
- Small mistakes easier and faster to correct
- Not bound by large, long-term liabilities/commitments
- Reduces risk

Look for Loops, not Lines

> *The world is a circle*

- *Systems Thinking*; not naive linear, predictive thinking
- No simple Cause→Effect; multiple influences and outcomes
- Never "one and done;" every project has a before and an after



Look for Actuals, not Proxies

> *You want the real stuff*

- Actual capability, not a piece of paper
- Actual shipping software that generates value, not numbers on a spreadsheet
- Actual users in their native habitat, not a watered-down version in a memo or on a story card

Keep it Safe

> *Make sure your own behavior contributes to the psychological support and safety of the team*

- Free information flow is critical to success
- Westrum Continuum

Westrum Continuum

Pathological (Power)	Bureaucratic (Rules)	Generative (Performance)
Low cooperation	Modest cooperation	High cooperation
Messengers “shot”	Messengers neglected	Messengers trained
Responsibilities shirked	Narrow responsibilities	Risks are shared
Failure leads to scapegoating	Failure leads to justice	Failure leads to inquiry
Novelty crushed	Novelty leads to problems	Novelty implemented


What are you going to do tomorrow?

GROWS Method®

The GROWS Method®
Habits, Workshops

growsmethod.com

GROWS Method®

Andy Hunt

- andy@growsmethod.com
- @PragmaticAndy@mastodon.social
- weatherlyhall.com - *Psychological Thriller*
- conglommora.com - *Science Fiction*