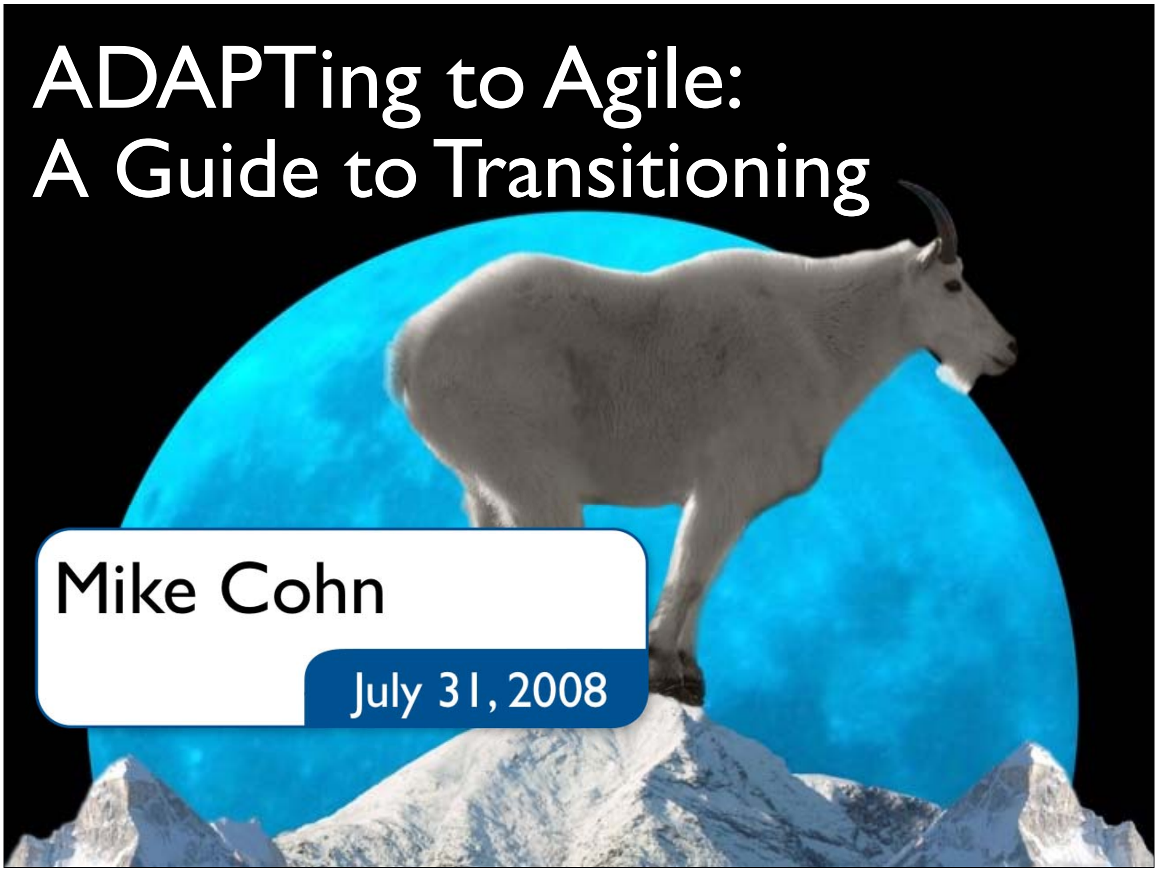
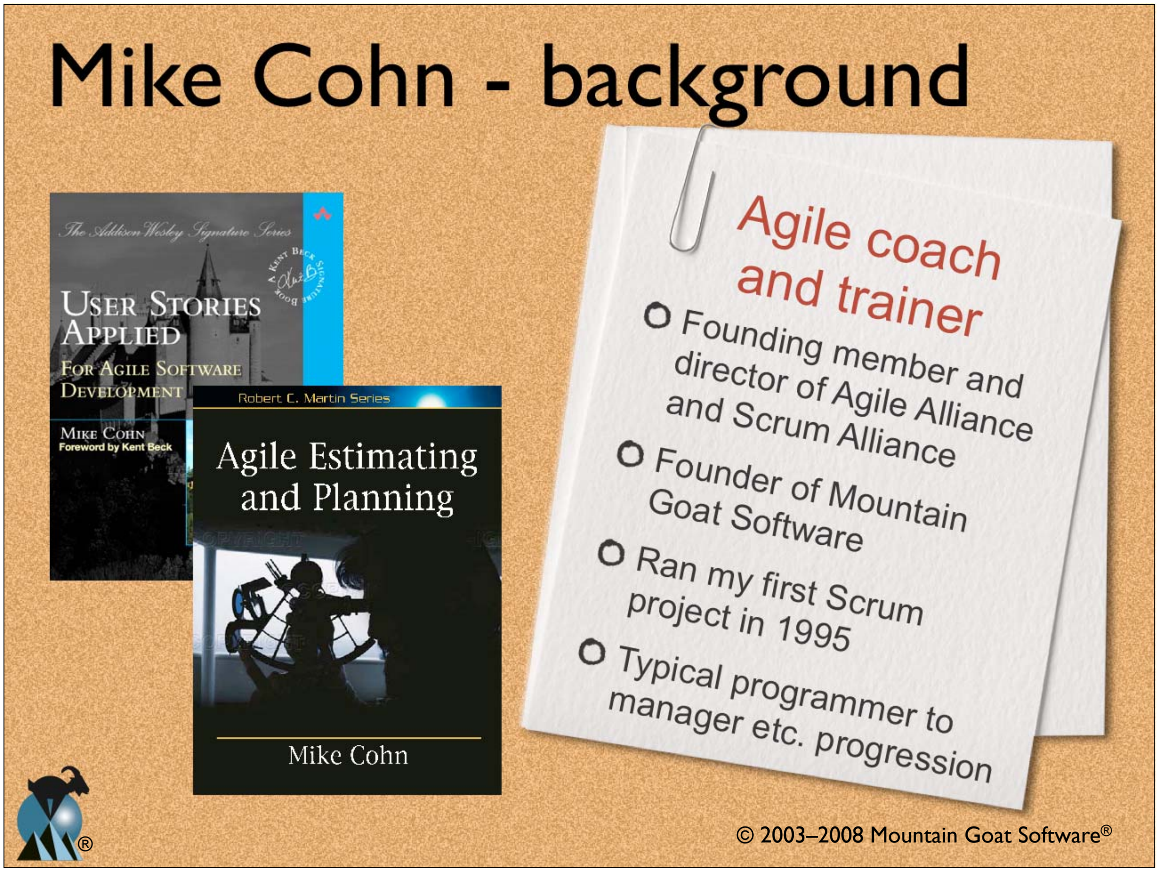


# ADAPTing to Agile: A Guide to Transitioning

Mike Cohn

July 31, 2008

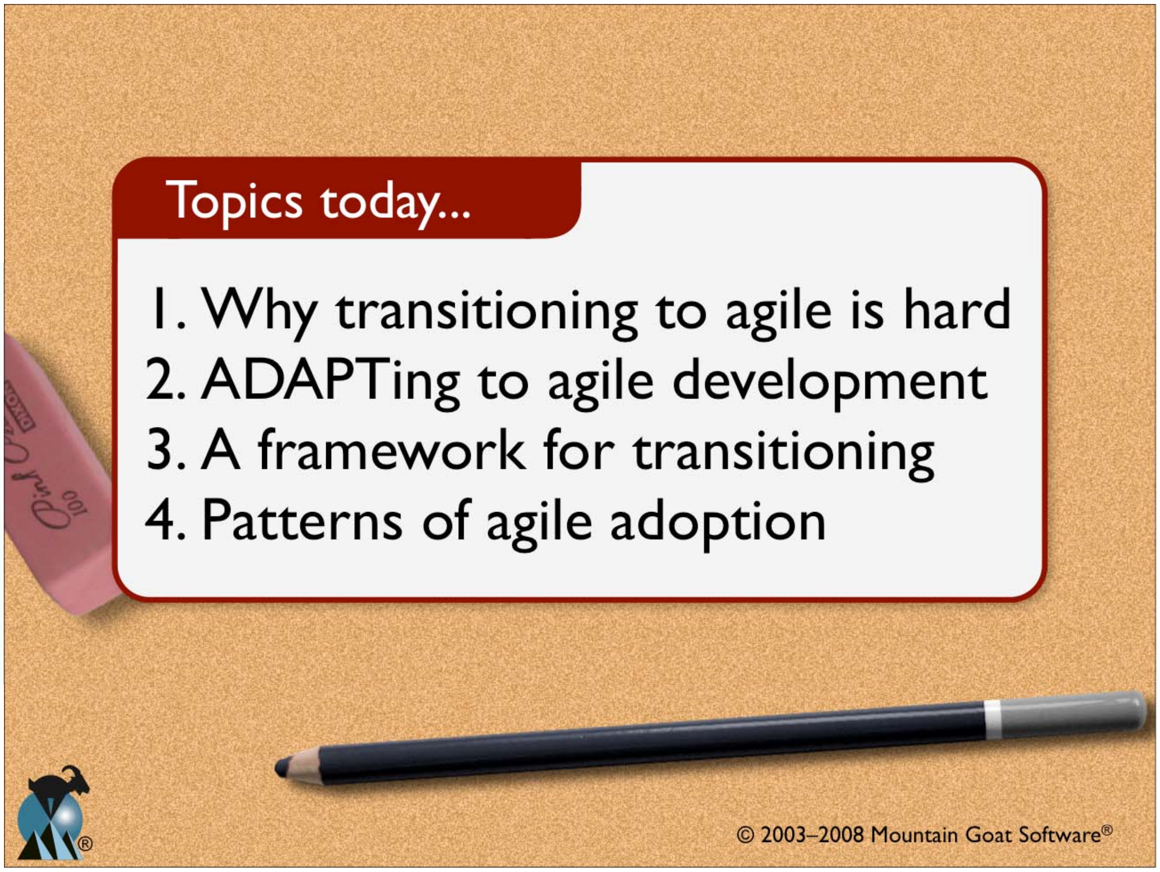


# Mike Cohn - background

The Addison-Wesley Signature Series
A Kent Beck Signature Book
User Stories Applied
For Agile Software Development
Mike Cohn
Foreword by Kent Beck
Robert C. Martin Series
Agile Estimating and Planning
Mike Cohn

Agile coach and trainer

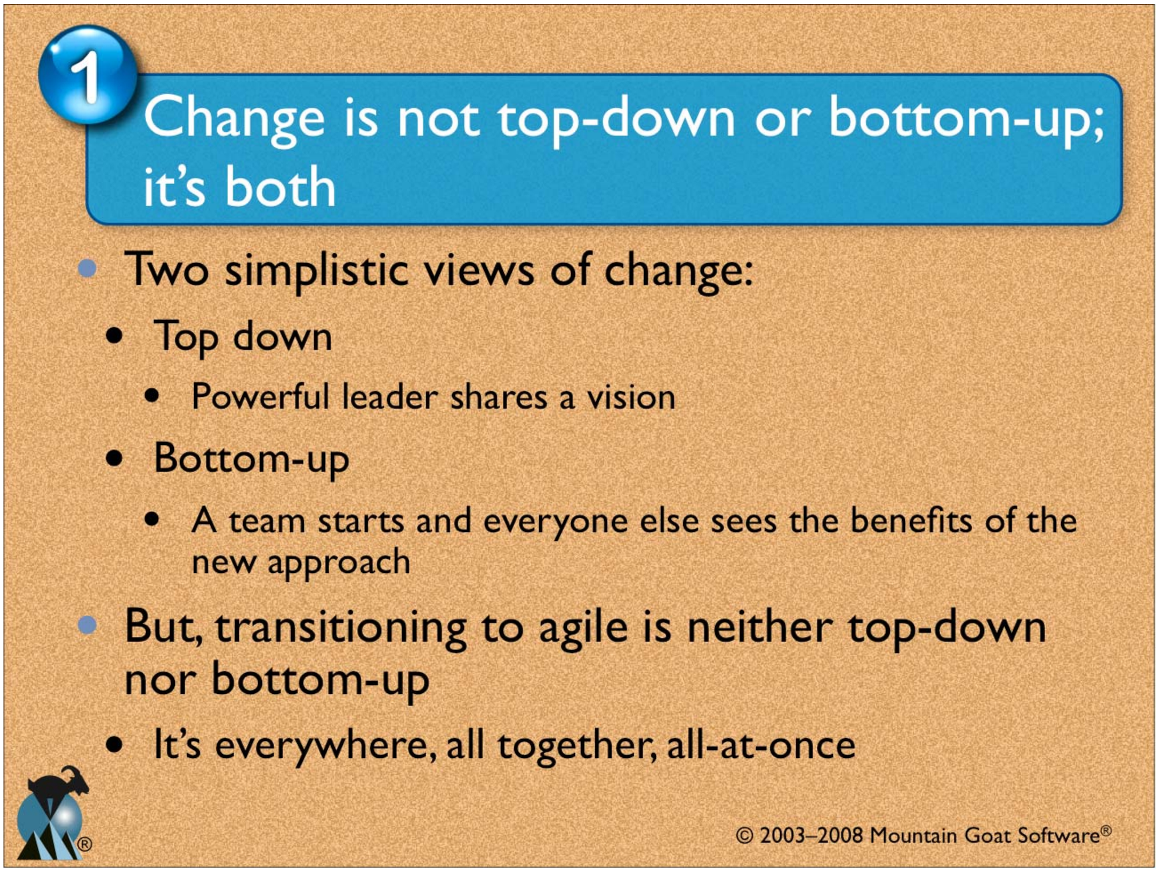
- Founding member and director of Agile Alliance and Scrum Alliance
- Founder of Mountain Goat Software
- Ran my first Scrum project in 1995
- Typical programmer to manager etc. progression



## Topics today...

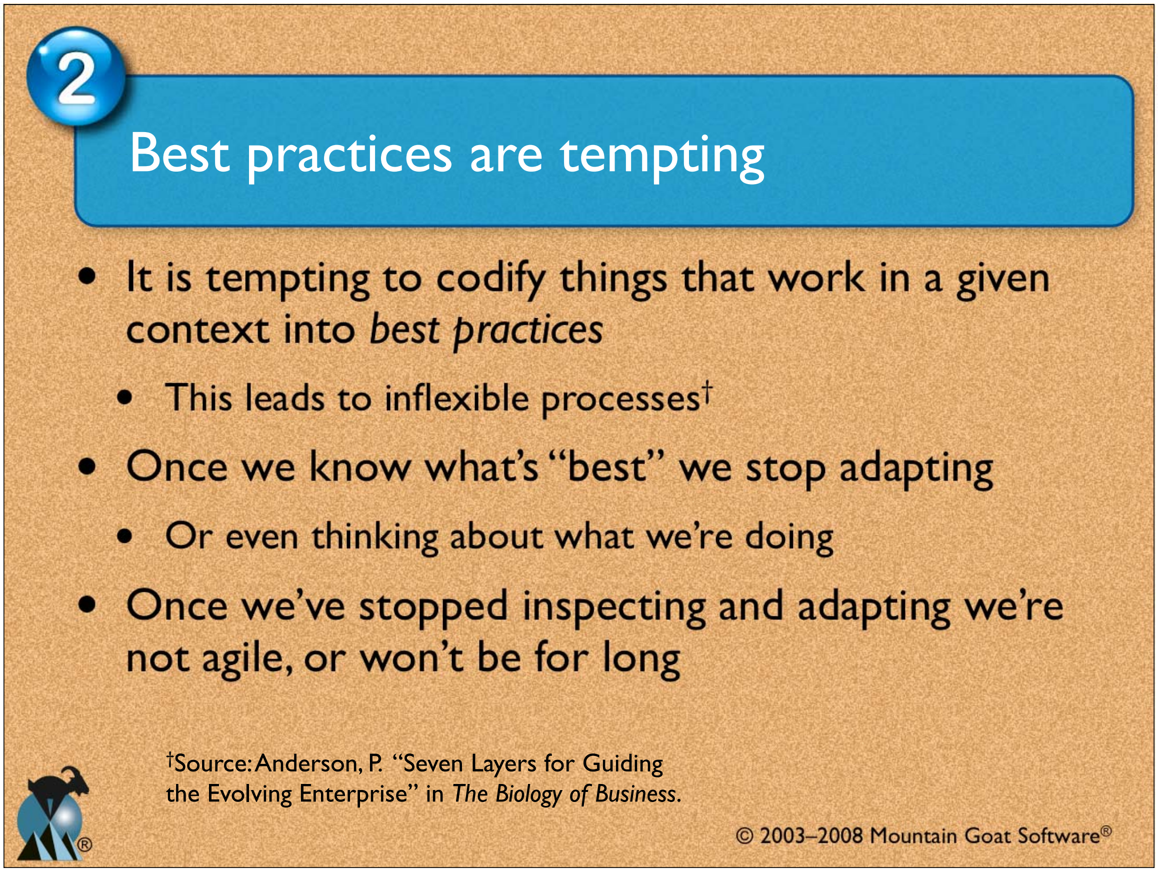
1. Why transitioning to agile is hard
2. ADAPTing to agile development
3. A framework for transitioning
4. Patterns of agile adoption

## Why Transitioning to Agile Is Hard



## 1 Change is not top-down or bottom-up; it's both

- Two simplistic views of change:
  - Top down
    - Powerful leader shares a vision
  - Bottom-up
    - A team starts and everyone else sees the benefits of the new approach
- But, transitioning to agile is neither top-down nor bottom-up
  - It's everywhere, all together, all-at-once

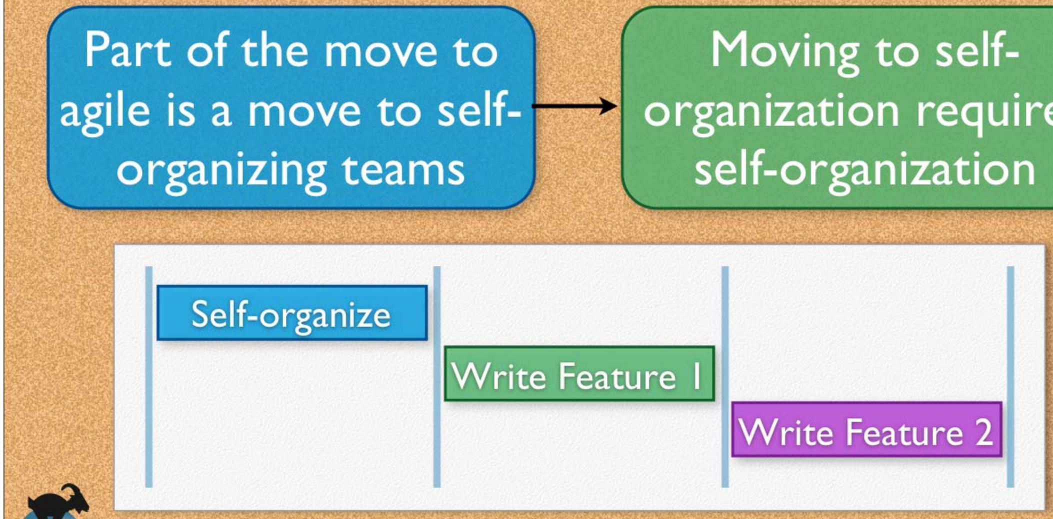


## 2 Best practices are tempting

- It is tempting to codify things that work in a given context into *best practices*
  - This leads to inflexible processes†
- Once we know what's "best" we stop adapting
  - Or even thinking about what we're doing
- Once we've stopped inspecting and adapting we're not agile, or won't be for long

†Source: Anderson, P. "Seven Layers for Guiding the Evolving Enterprise" in *The Biology of Business*.

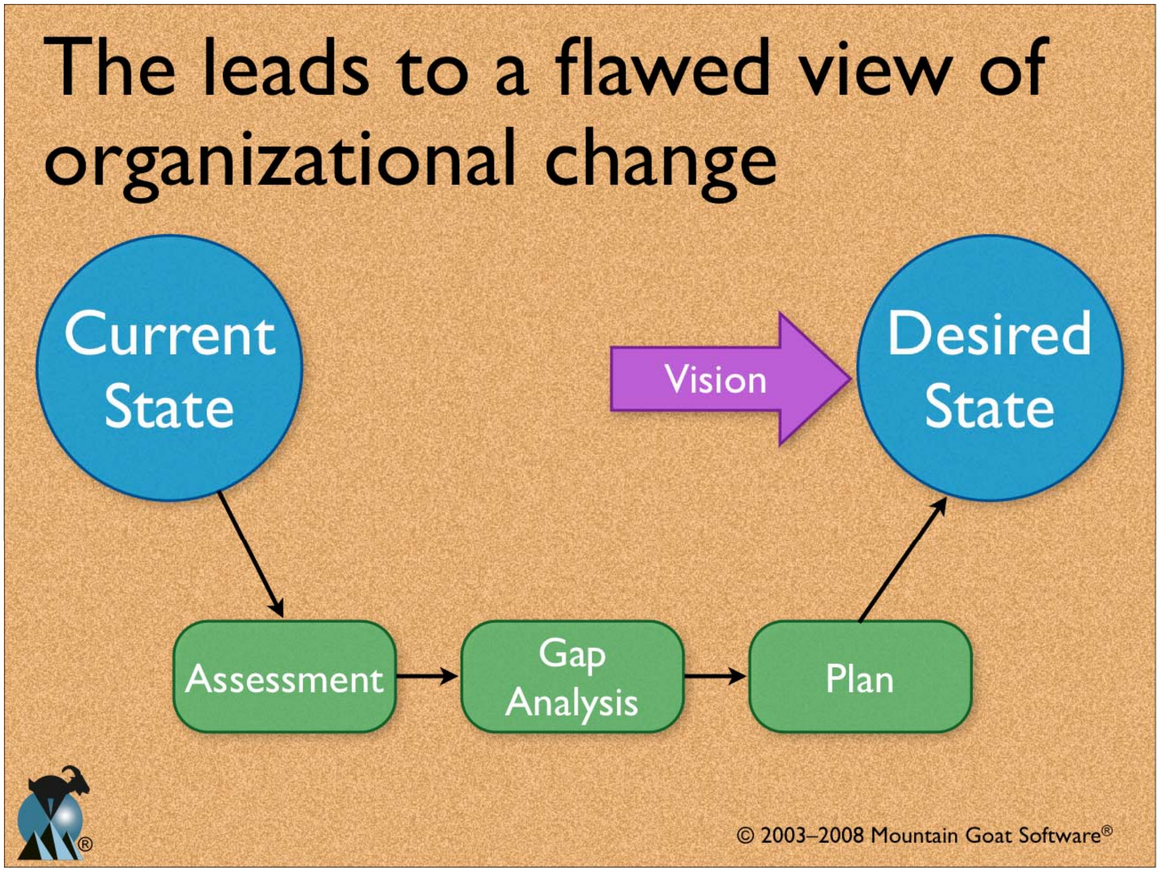
## 3 The transition process must be congruent with the development process



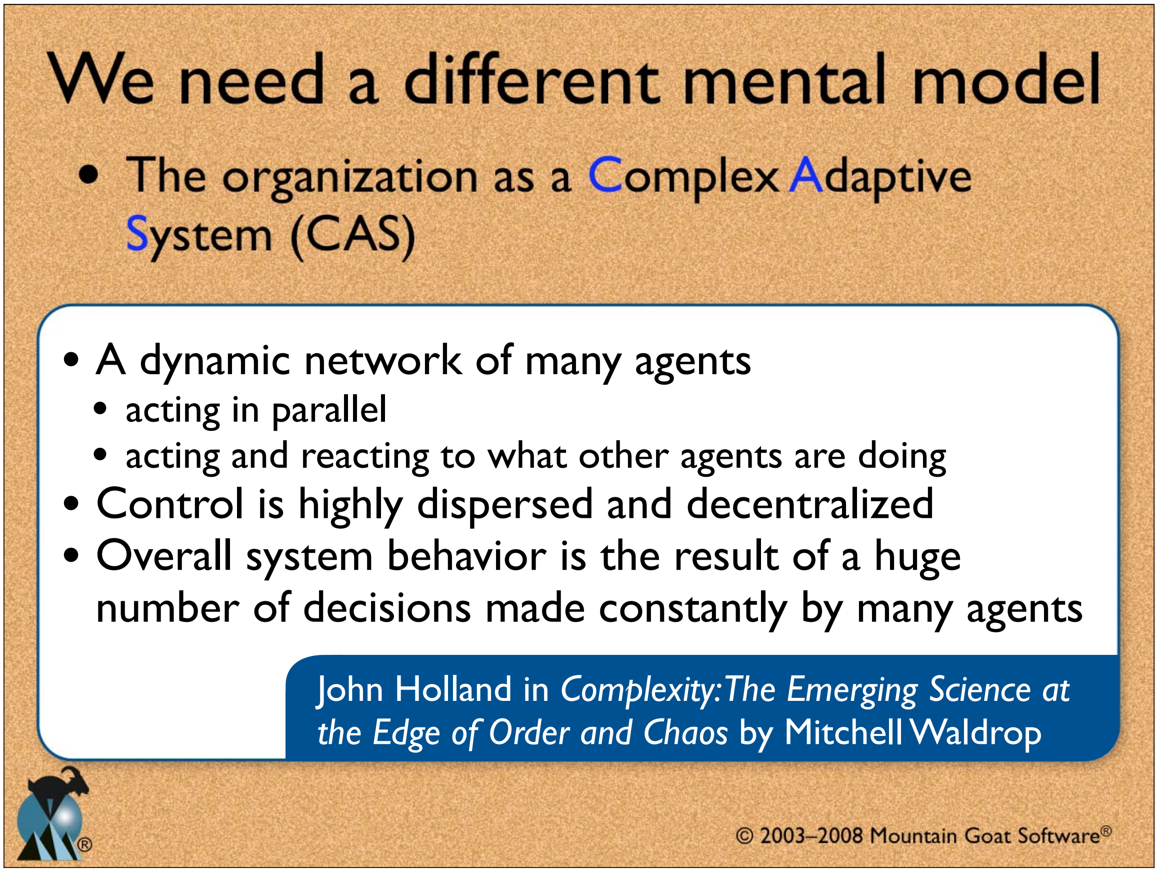
Part of the move to agile is a move to self-organizing teams → Moving to self-organization requires self-organization

## 4 Organizations are unpredictable, living systems

- Traditional view of the organization is as a machine
  - We can dissemble it, study its part, put it back together
  - Once we set it in motion, it will continue in motion



# The leads to a flawed view of organizational change



# We need a different mental model

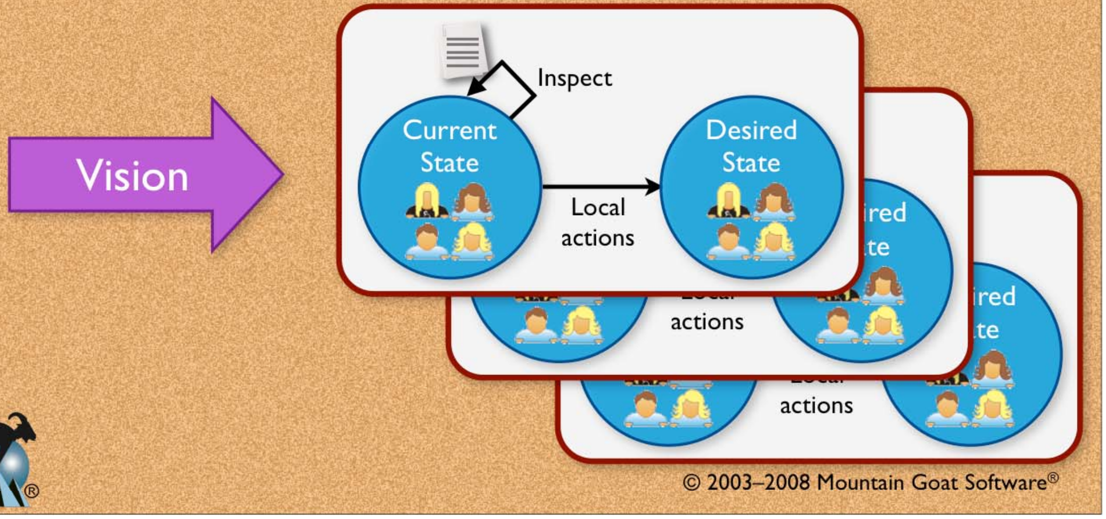
- The organization as a Complex Adaptive System (CAS)

- A dynamic network of many agents
  - acting in parallel
  - acting and reacting to what other agents are doing
- Control is highly dispersed and decentralized
- Overall system behavior is the result of a huge number of decisions made constantly by many agents

John Holland in *Complexity:The Emerging Science at the Edge of Order and Chaos* by Mitchell Waldrop

# Local goals and gaps

- Local agents (individuals, project teams, discipline coworkers) identify local gaps based on their local goals



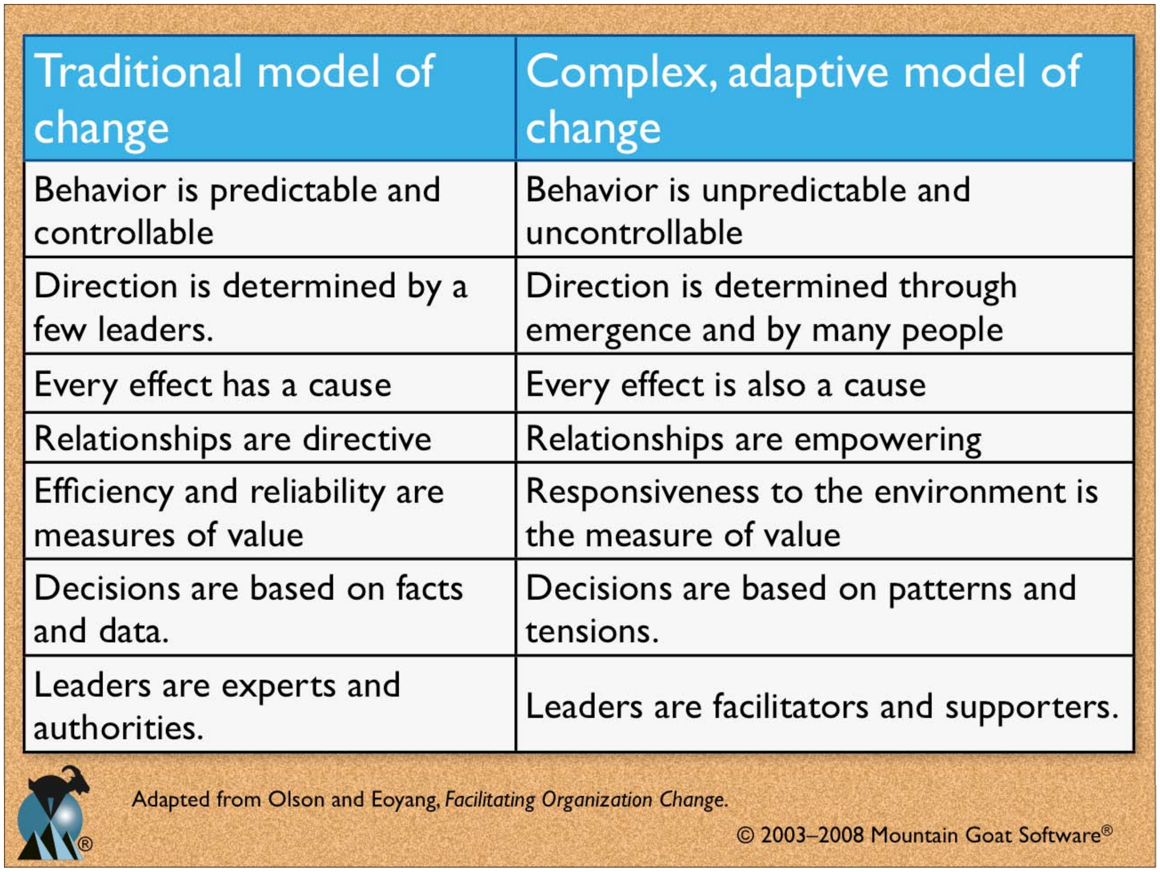
# Differing views of success

## Newtonian view

Success =
closing the gap with the desired state

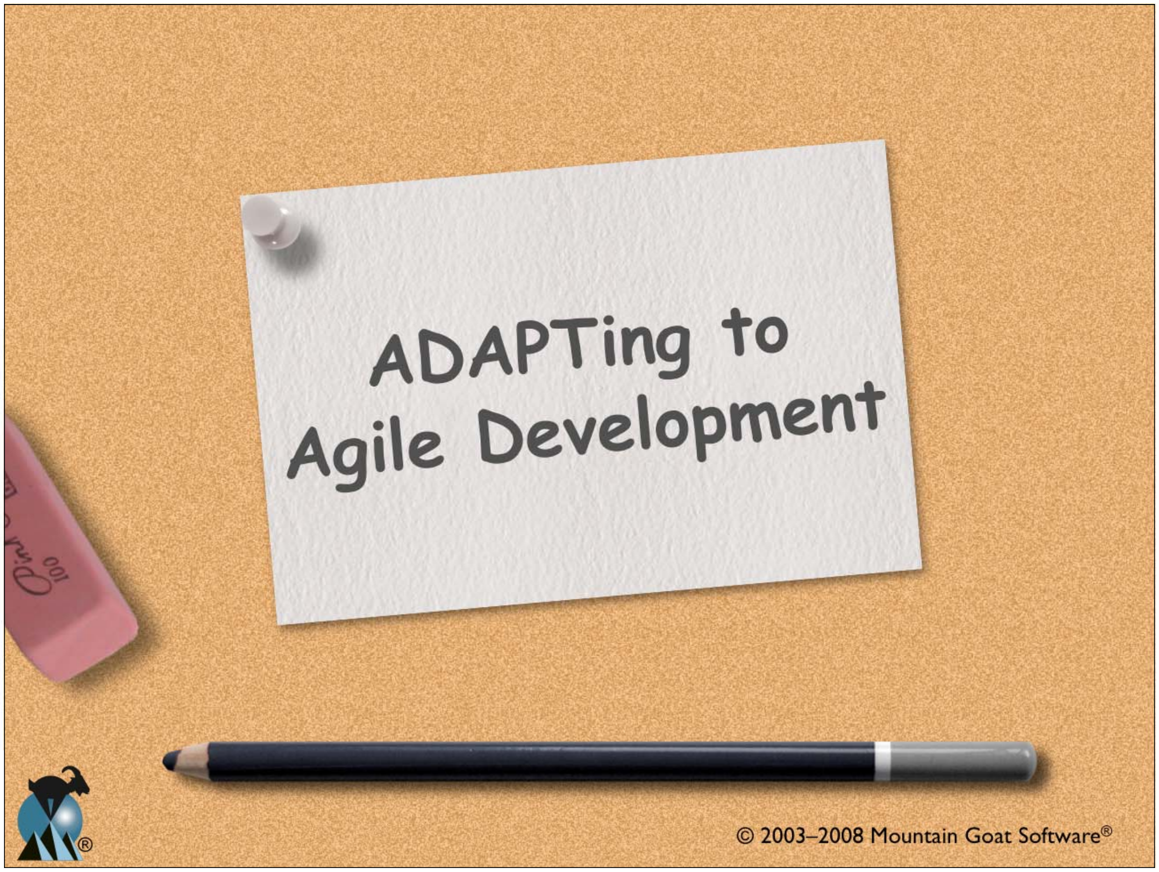
## CAS view

Success =
achieving a good fit with the environment

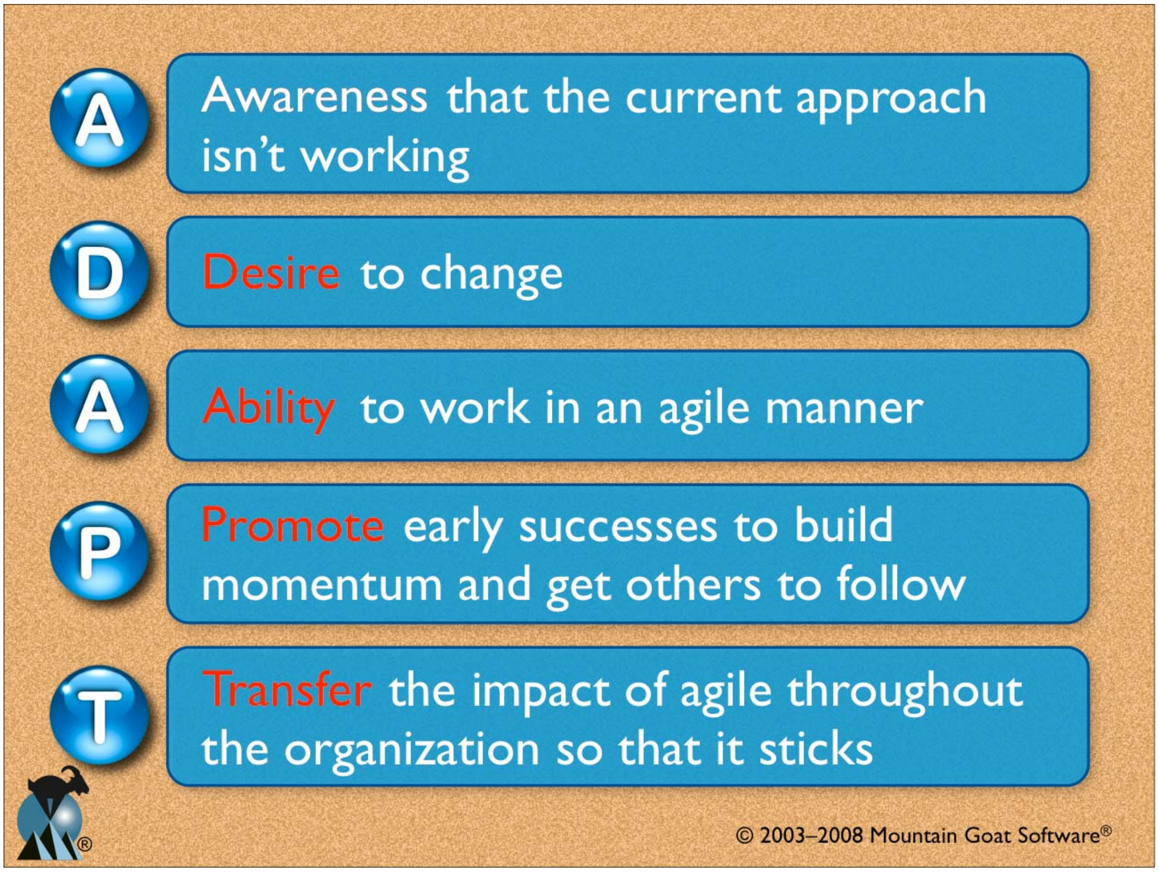


| Traditional model of change | Complex, adaptive model of change |
|---|---|
| Behavior is predictable and controllable | Behavior is unpredictable and uncontrollable |
| Direction is determined by a few leaders. | Direction is determined through emergence and by many people |
| Every effect has a cause | Every effect is also a cause |
| Relationships are directive | Relationships are empowering |
| Efficiency and reliability are measures of value | Responsiveness to the environment is the measure of value |
| Decisions are based on facts and data. | Decisions are based on patterns and tensions. |
| Leaders are experts and authorities. | Leaders are facilitators and supporters. |

Adapted from Olson and Eoyang, *Facilitating Organization Change*.



ADAPTing to Agile Development



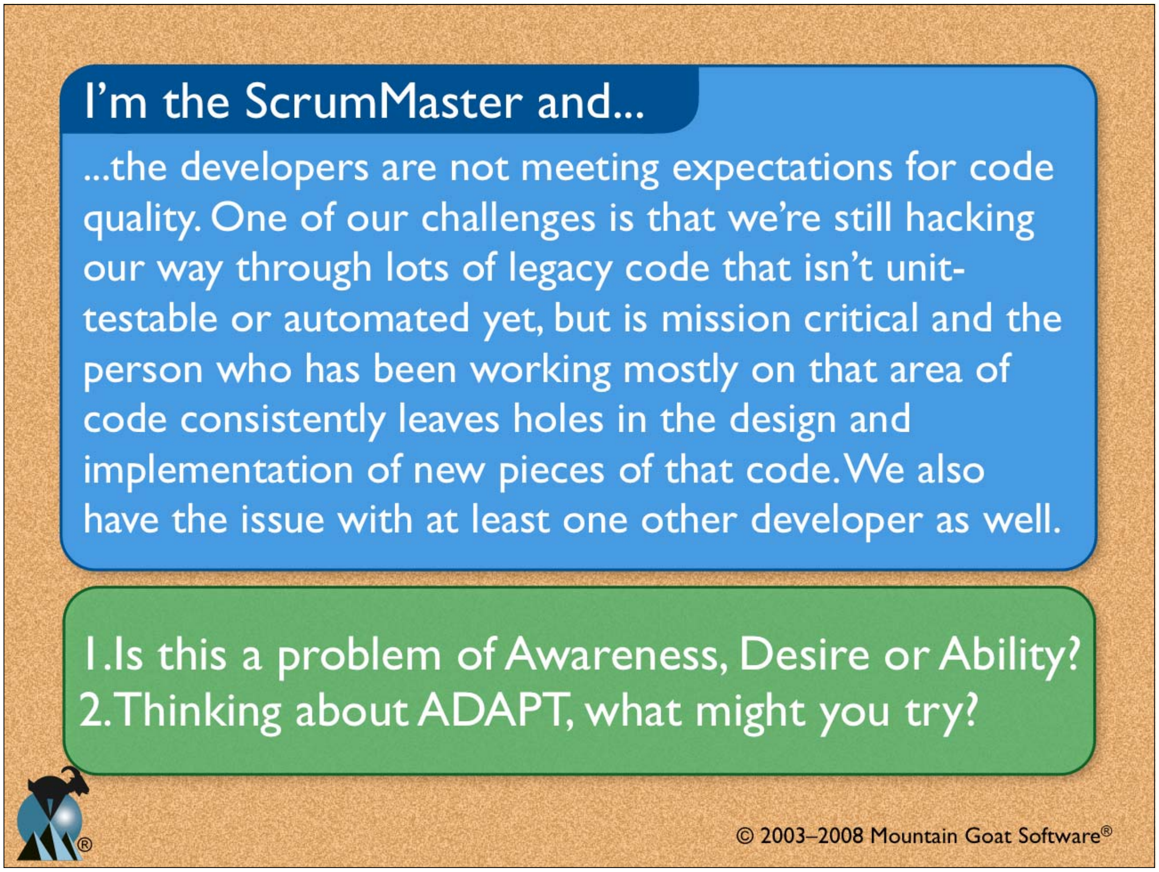
A — Awareness that the current approach isn't working

D — Desire to change

A — Ability to work in an agile manner

P — Promote early successes to build momentum and get others to follow

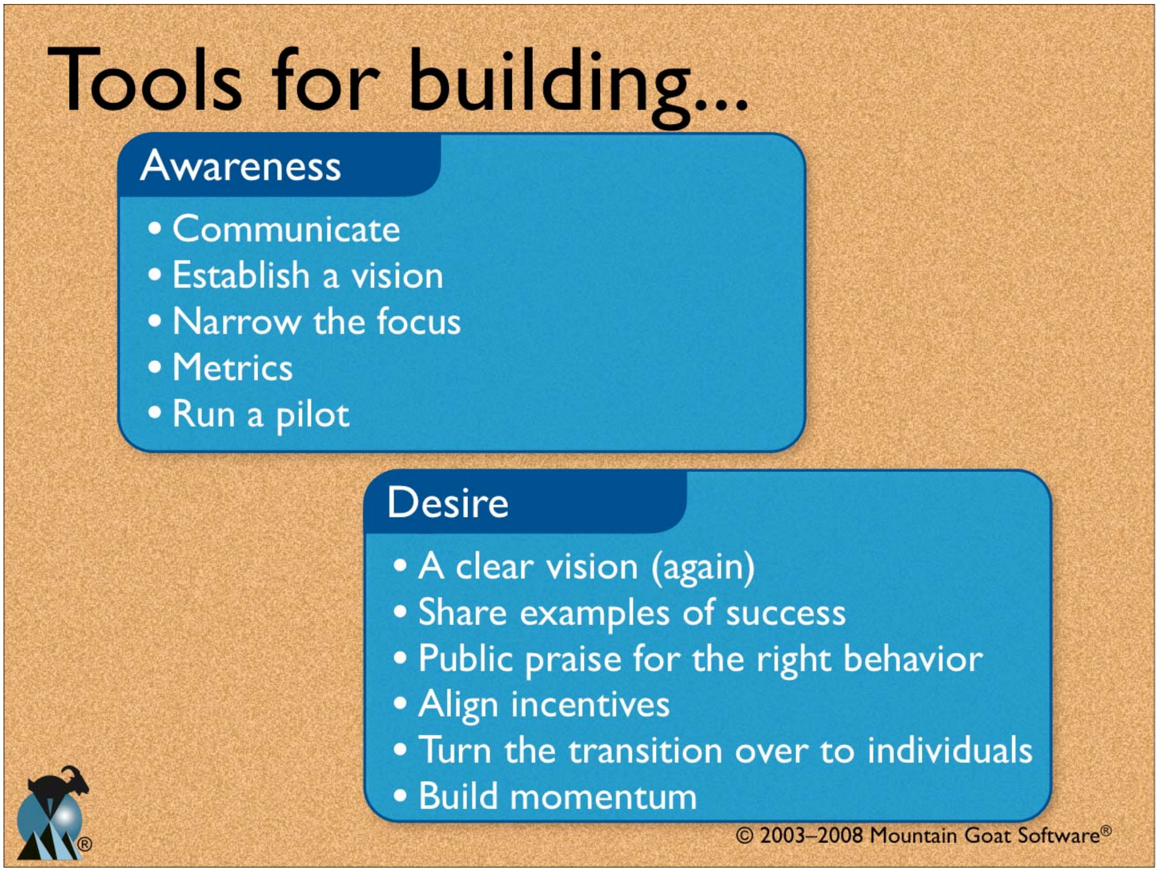
T — Transfer the impact of agile throughout the organization so that it sticks



## I'm the ScrumMaster and...

...the developers are not meeting expectations for code quality. One of our challenges is that we're still hacking our way through lots of legacy code that isn't unit-testable or automated yet, but is mission critical and the person who has been working mostly on that area of code consistently leaves holes in the design and implementation of new pieces of that code. We also have the issue with at least one other developer as well.

1. Is this a problem of Awareness, Desire or Ability?
2. Thinking about ADAPT, what might you try?



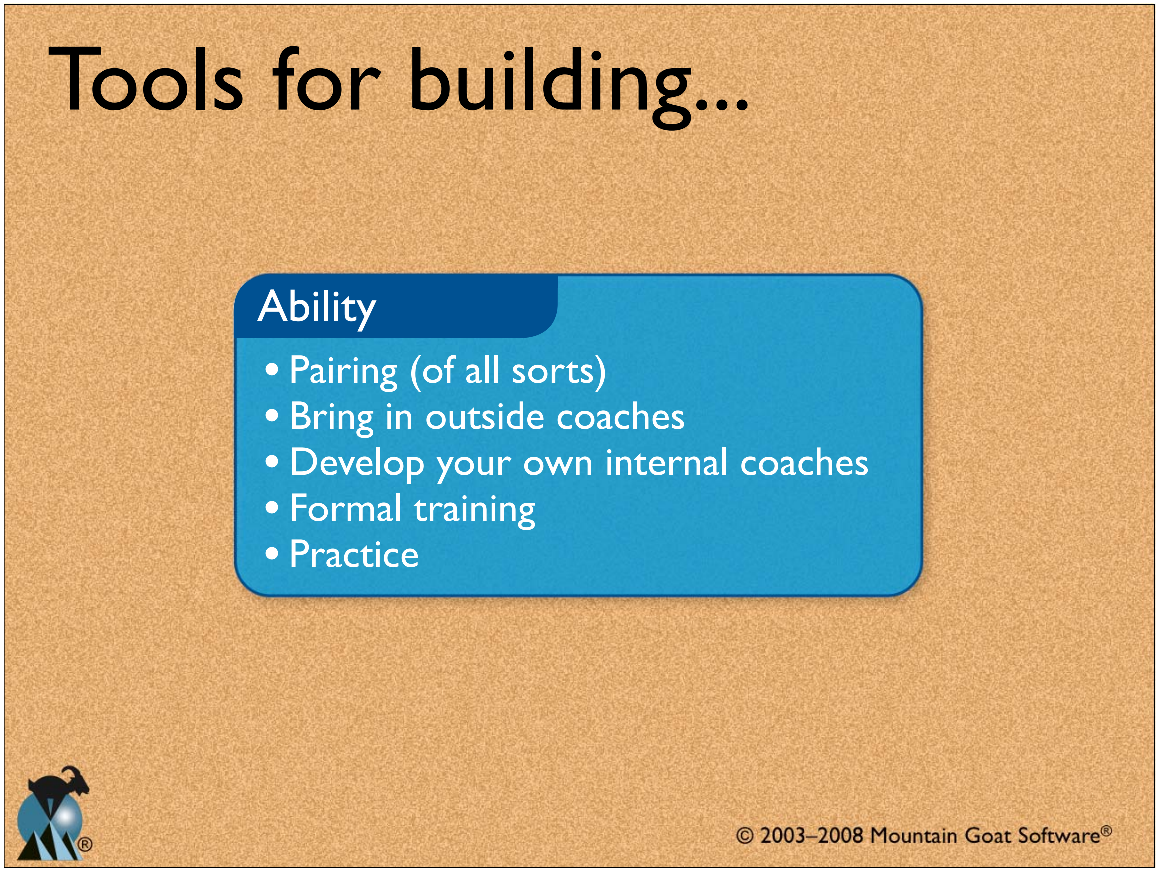
# Tools for building...

## Awareness

- Communicate
- Establish a vision
- Narrow the focus
- Metrics
- Run a pilot

## Desire

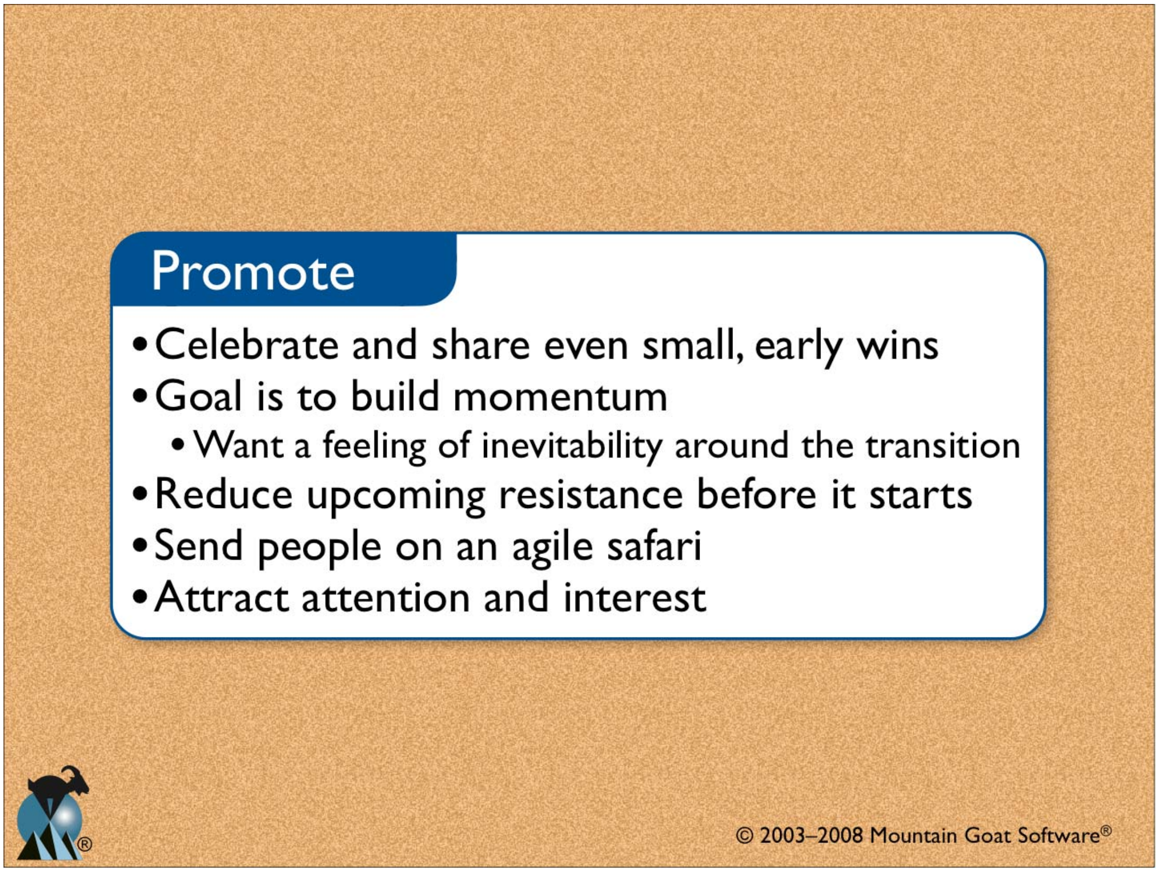
- A clear vision (again)
- Share examples of success
- Public praise for the right behavior
- Align incentives
- Turn the transition over to individuals
- Build momentum



# Tools for building...

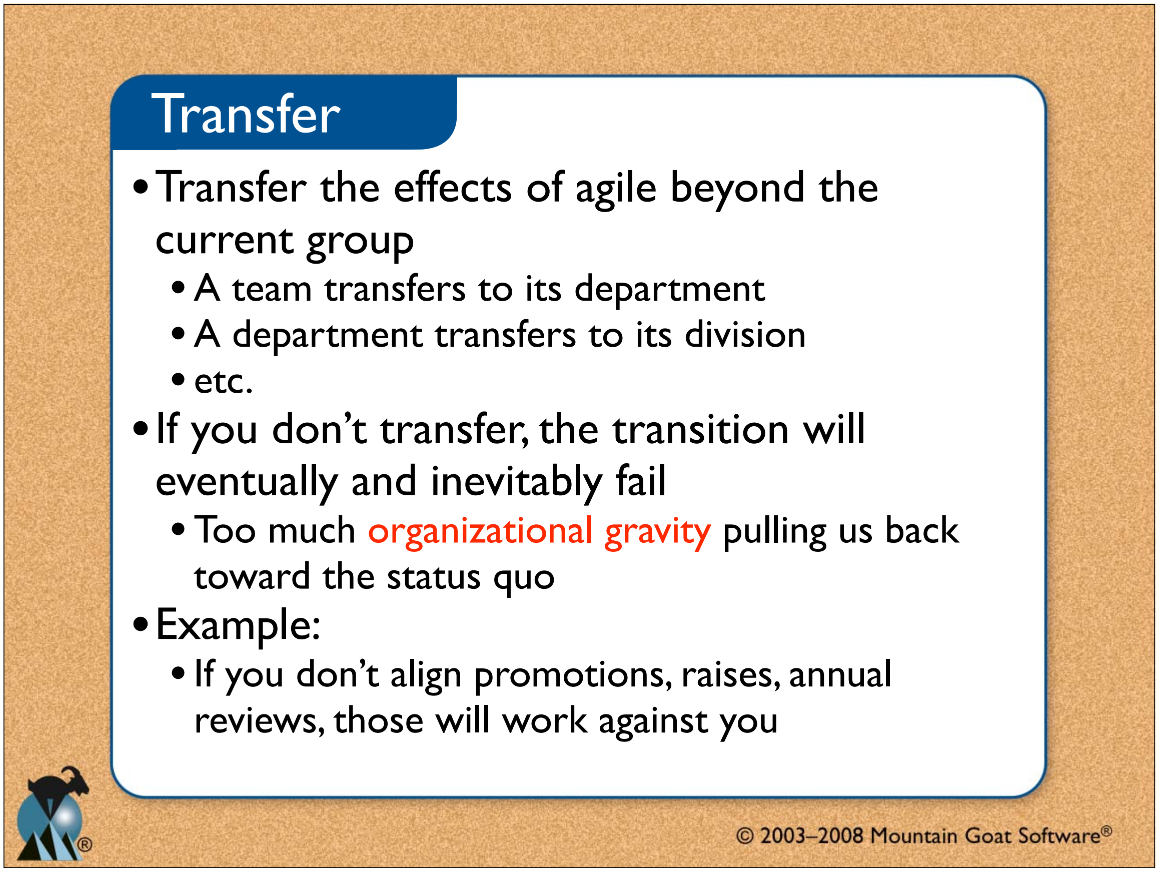
## Ability

- Pairing (of all sorts)
- Bring in outside coaches
- Develop your own internal coaches
- Formal training
- Practice



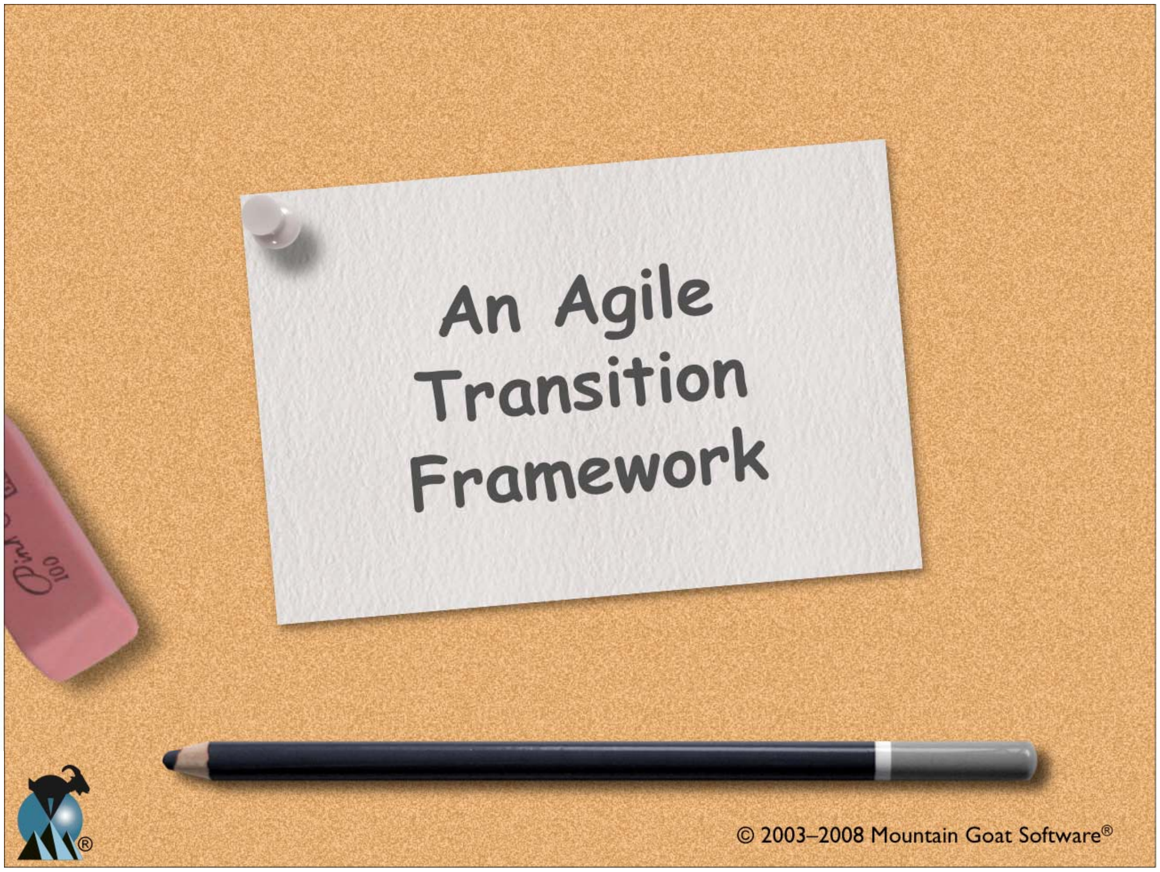
## Promote

- Celebrate and share even small, early wins
- Goal is to build momentum
  - Want a feeling of inevitability around the transition
- Reduce upcoming resistance before it starts
- Send people on an agile safari
- Attract attention and interest

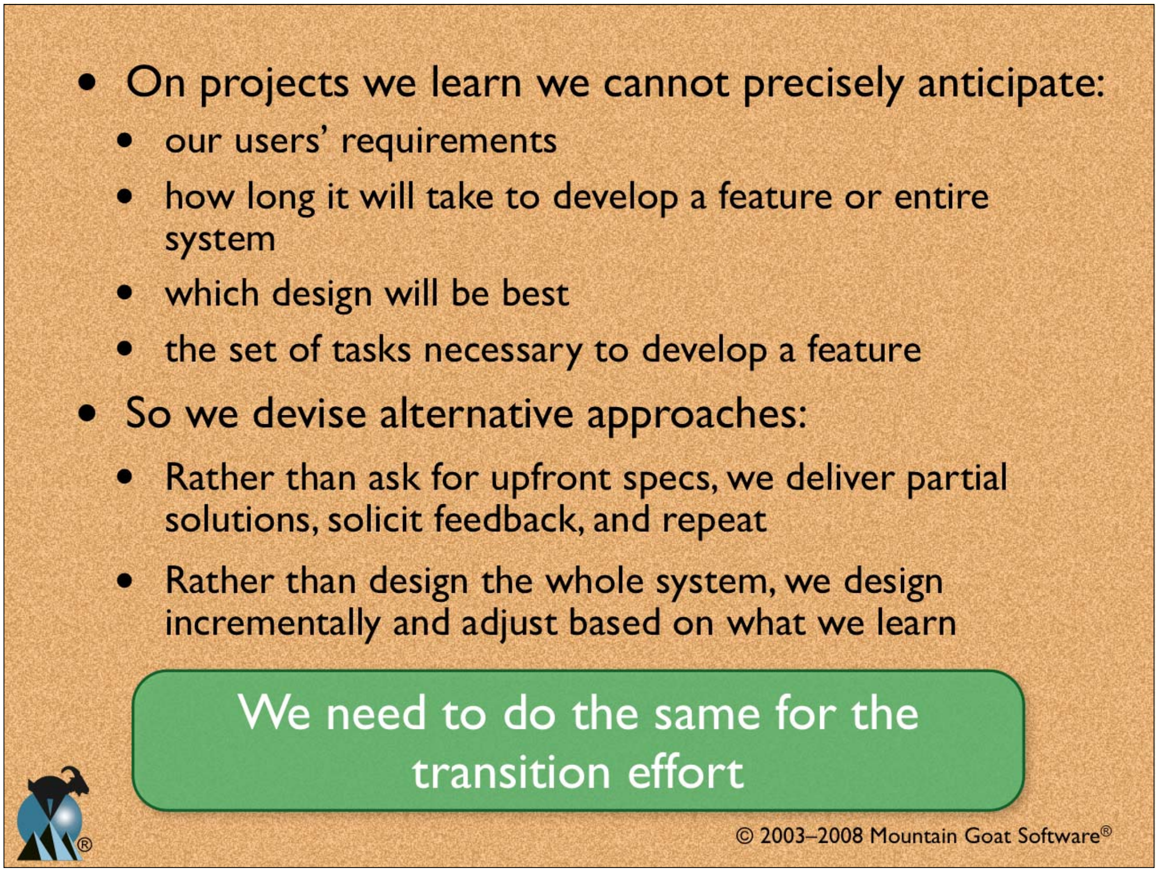


## Transfer

- Transfer the effects of agile beyond the current group
  - A team transfers to its department
  - A department transfers to its division
  - etc.
- If you don't transfer, the transition will eventually and inevitably fail
  - Too much organizational gravity pulling us back toward the status quo
- Example:
  - If you don't align promotions, raises, annual reviews, those will work against you

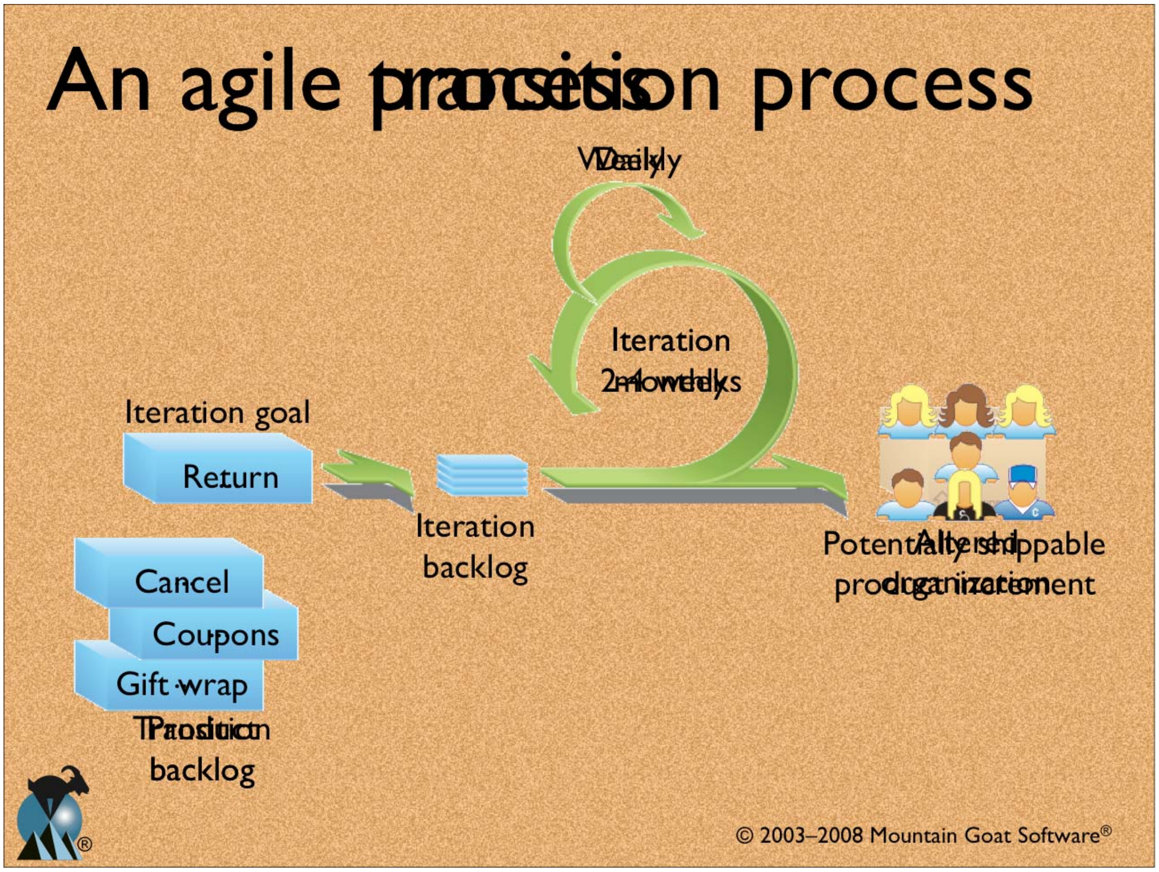


# An Agile Transition Framework



- On projects we learn we cannot precisely anticipate:
  - our users' requirements
  - how long it will take to develop a feature or entire system
  - which design will be best
  - the set of tasks necessary to develop a feature
- So we devise alternative approaches:
  - Rather than ask for upfront specs, we deliver partial solutions, solicit feedback, and repeat
  - Rather than design the whole system, we design incrementally and adjust based on what we learn

**We need to do the same for the transition effort**



# An agile process
# An agile transition process

Daily
Weekly

Iteration
2-4 weeks
1 month

Iteration goal

Return

Iteration backlog

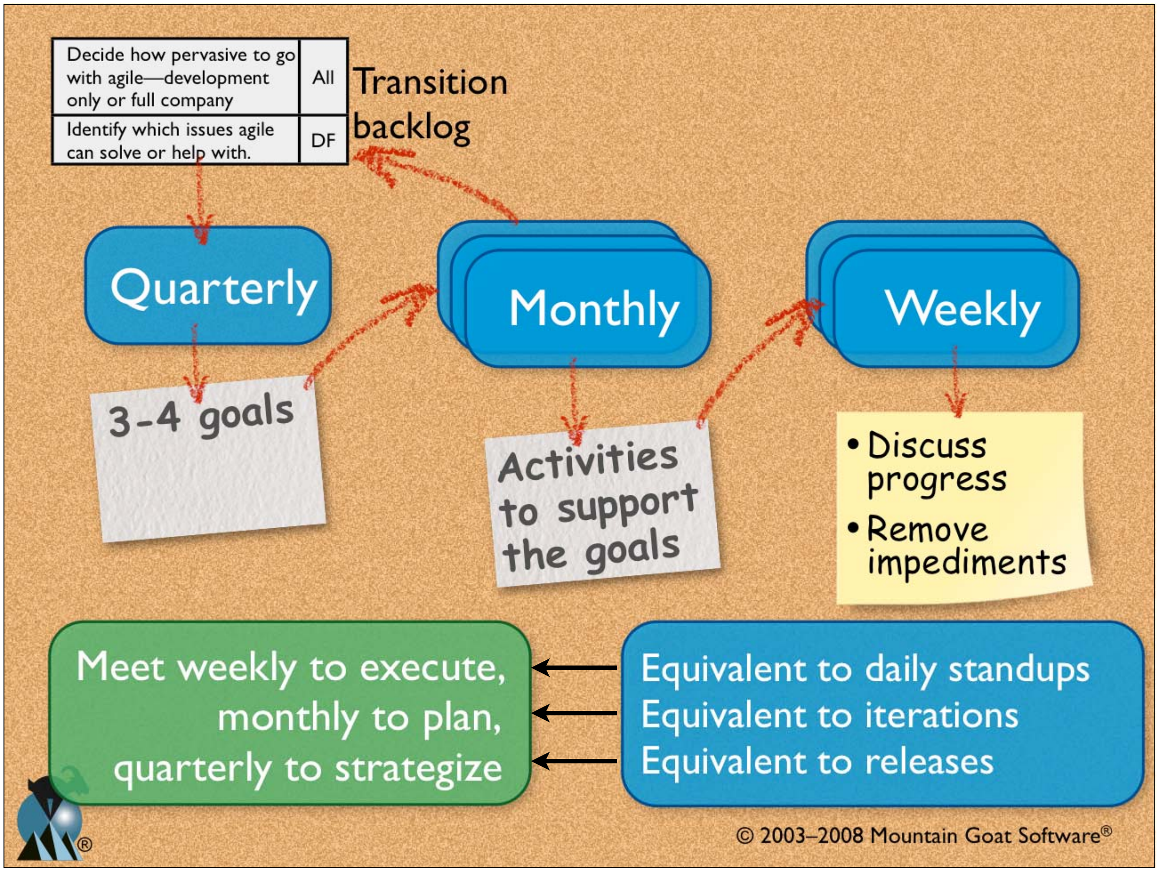
Cancel

Coupons

Gift wrap

Product backlog
Transition backlog

Potentially shippable product increment
Altered organization



| | |
|---|---|
| Decide how pervasive to go with agile—development only or full company | All |
| Identify which issues agile can solve or help with. | DF |

Transition backlog

Quarterly

3-4 goals

Monthly

Activities to support the goals

Weekly

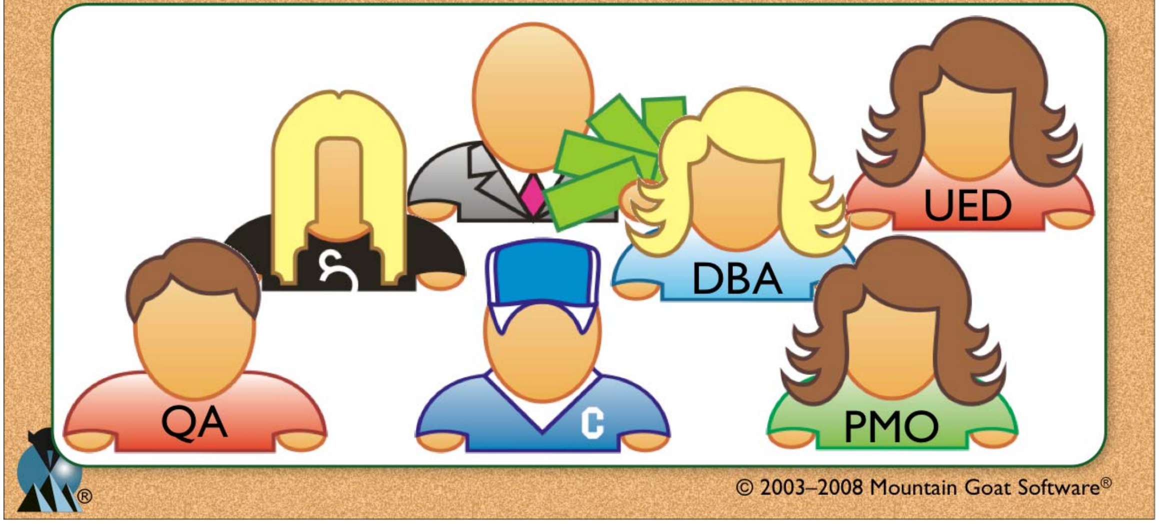
- Discuss progress
- Remove impediments

Meet weekly to execute,
monthly to plan,
quarterly to strategize

Equivalent to daily standups
Equivalent to iterations
Equivalent to releases

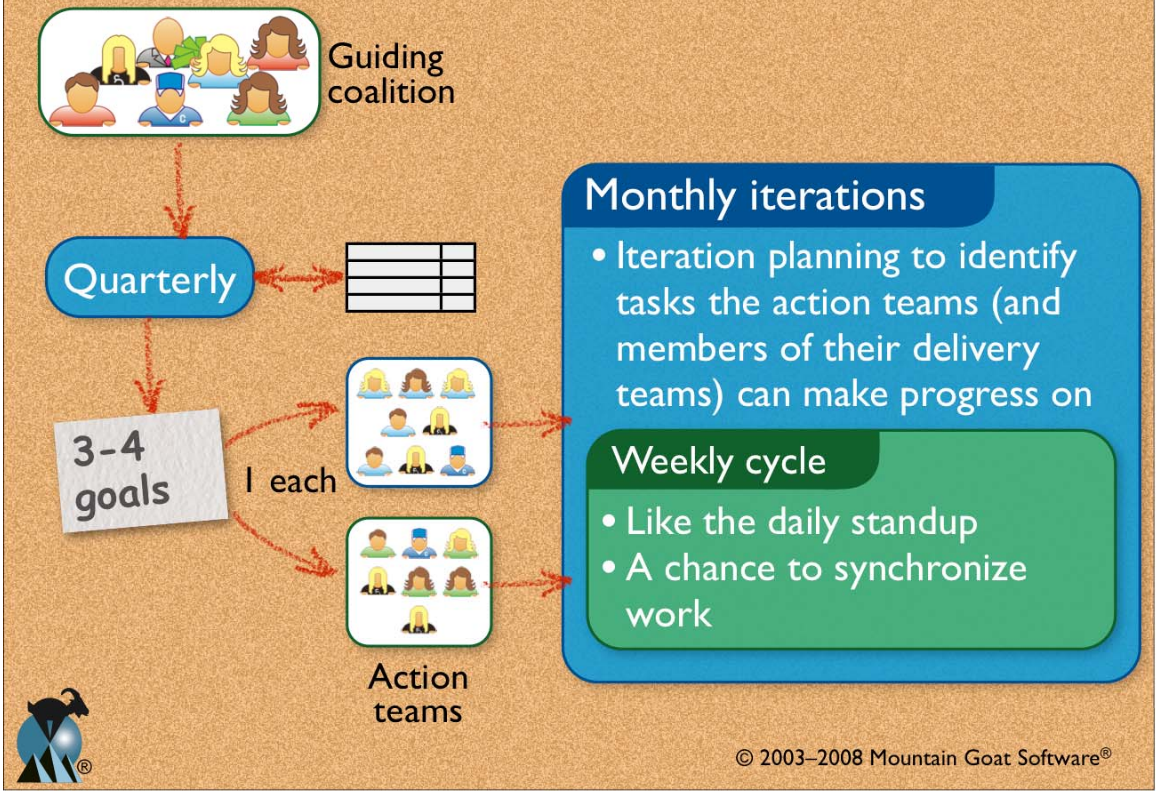
# Establish a "guiding coalition"

- Who?
  - Sponsor—senior person responsible for success
  - Area managers or leads who can make it happen



# Action teams

- Usually more than one at any time
  - Each focused on a different goal
- Organized around achieving specific organizational goals
  - e.g., test automation or user experience design
- Some teams in an organization will be organic
  - Individuals notice something needs to be achieved
- Others will be formally-sponsored
  - Guiding coalition puts someone in charge of achieving a goal that hasn't been picked up
  - Usually best only if an organic team doesn't form



Guiding coalition

Quarterly

3-4 goals

1 each

Action teams

## Monthly iterations

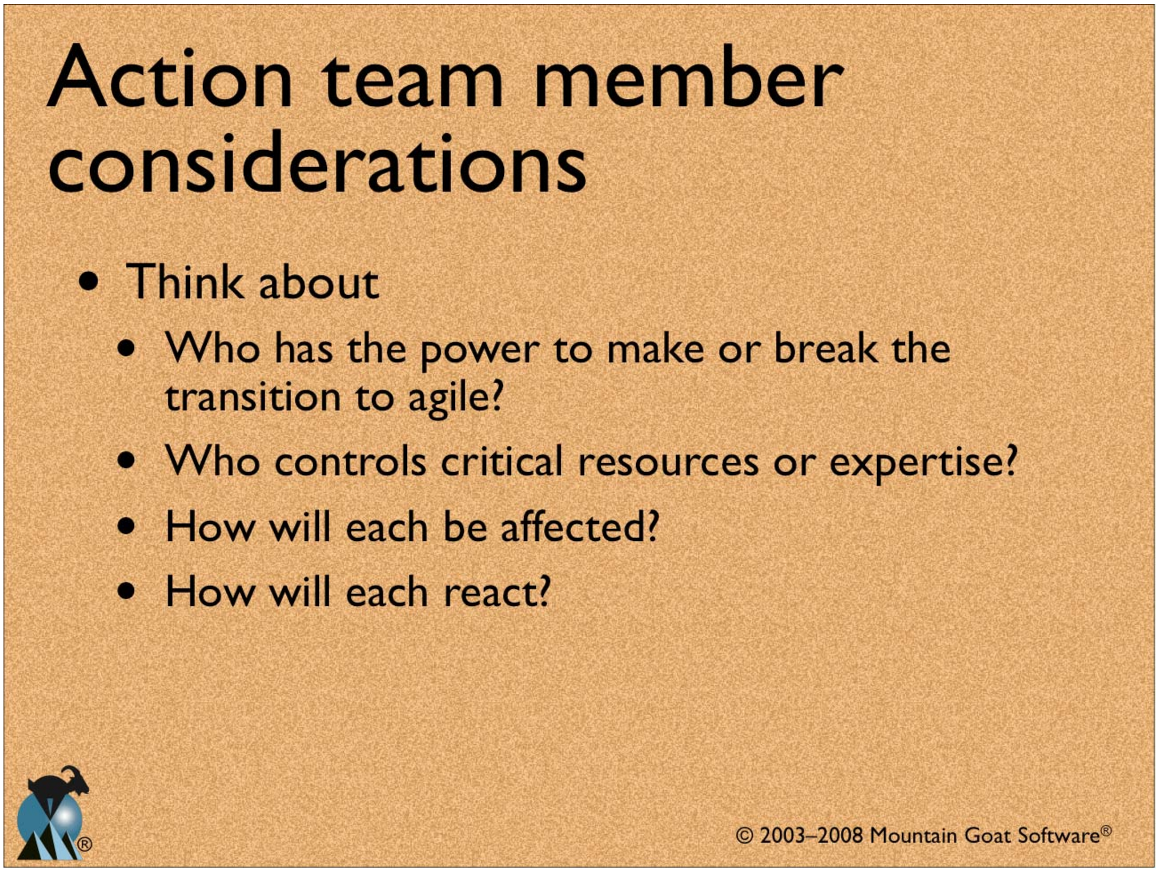
- Iteration planning to identify tasks the action teams (and members of their delivery teams) can make progress on

### Weekly cycle

- Like the daily standup
- A chance to synchronize work

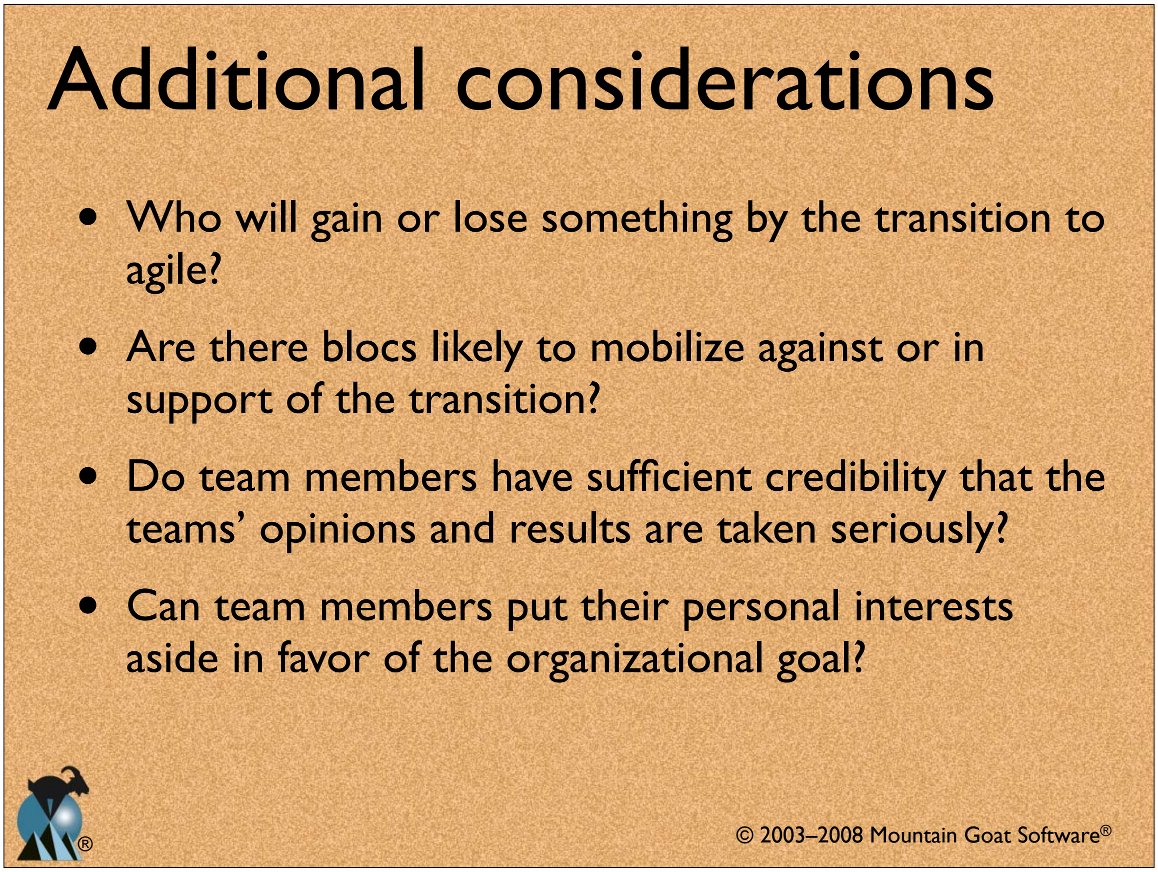
# Action team members

- Try to form these teams organically
  - Possible with a point person to start the team
  - True product owner for the team is the guiding coalition
  - But this starting person acts as a combination day-to-day product owner and ScrumMaster
- Initial membership
  - Start with 1-3 members who "get it"
  - Ask each of those members to pick 1-2 more



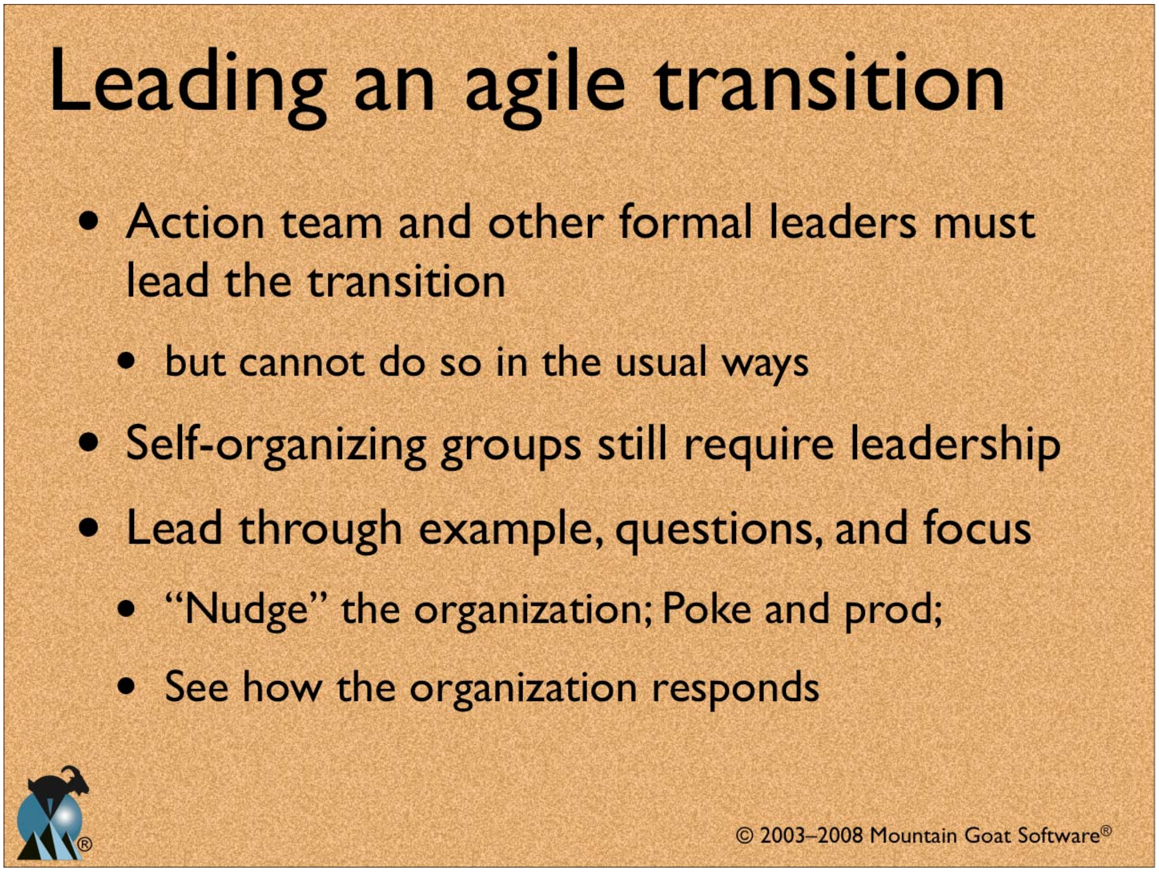
## Action team member considerations

- Think about
  - Who has the power to make or break the transition to agile?
  - Who controls critical resources or expertise?
  - How will each be affected?
  - How will each react?



## Additional considerations

- Who will gain or lose something by the transition to agile?
- Are there blocs likely to mobilize against or in support of the transition?
- Do team members have sufficient credibility that the teams' opinions and results are taken seriously?
- Can team members put their personal interests aside in favor of the organizational goal?



# Leading an agile transition

- Action team and other formal leaders must lead the transition
  - but cannot do so in the usual ways
- Self-organizing groups still require leadership
- Lead through example, questions, and focus
  - “Nudge” the organization; Poke and prod;
  - See how the organization responds

# Pre-requisites of self-organization

## Container

- A boundary within which self-organization occurs
  - Company, project, team, city, role, nationality

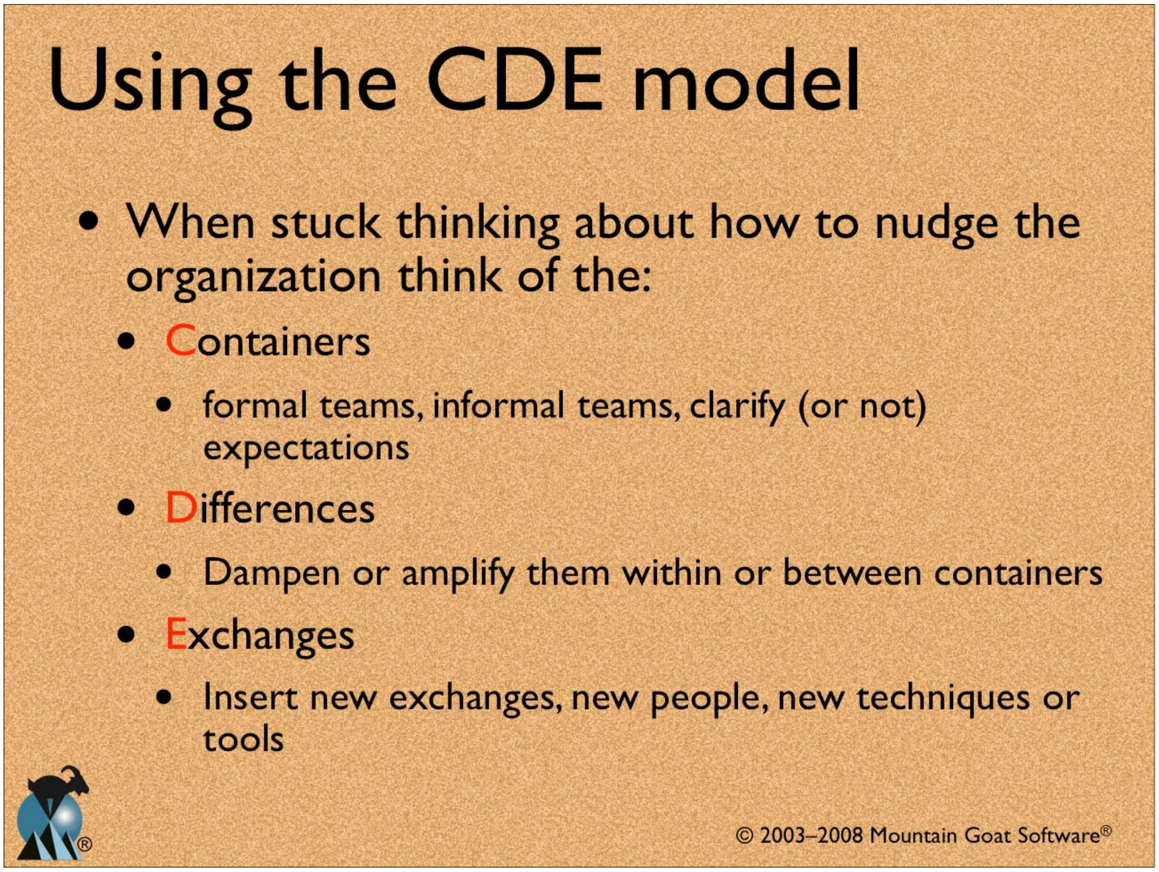
## Differences

- There must be differences among the agents acting in our system
  - Technical knowledge, domain knowledge, education, experience, power, gender

## Transforming Exchanges

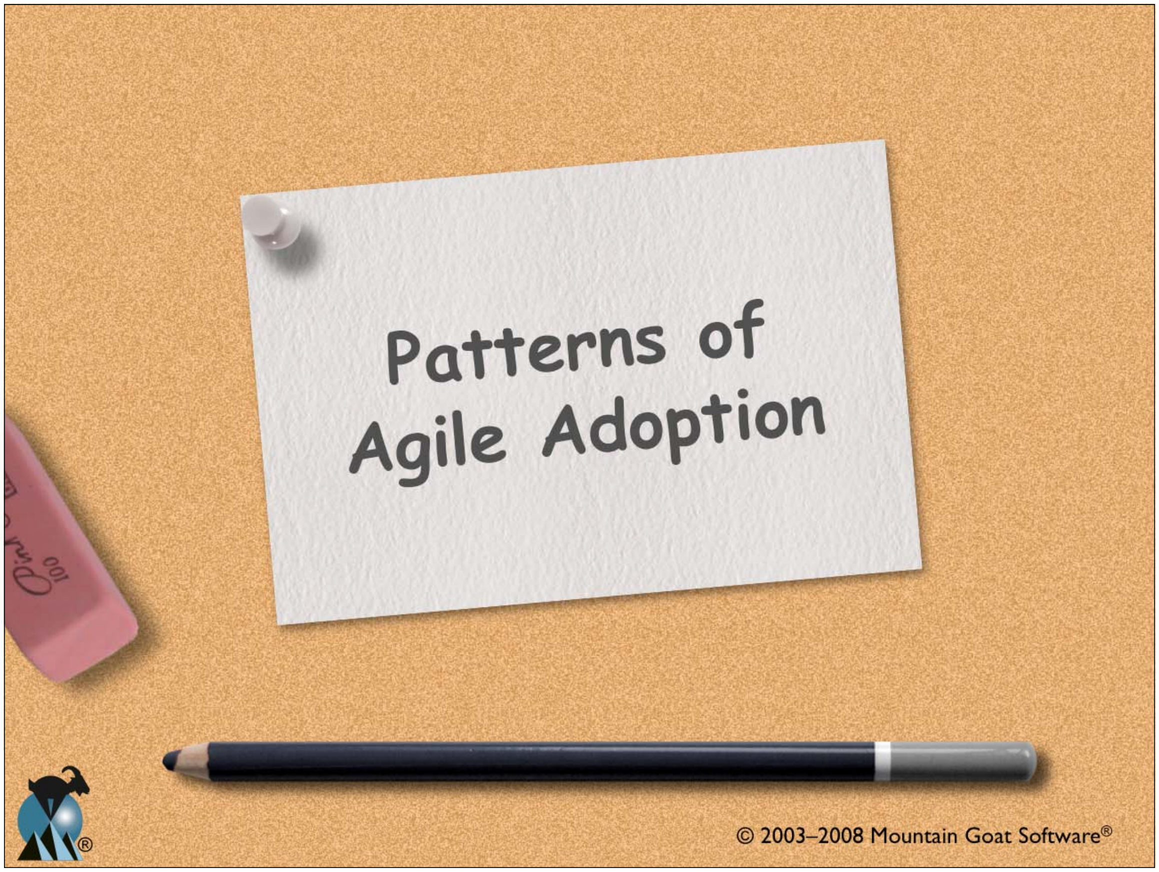
- Agents in the system interact and exchange resources
  - Information, money, energy (vision)

Glenda Eoyang: *Conditions for Self-Organizing in Human Systems*

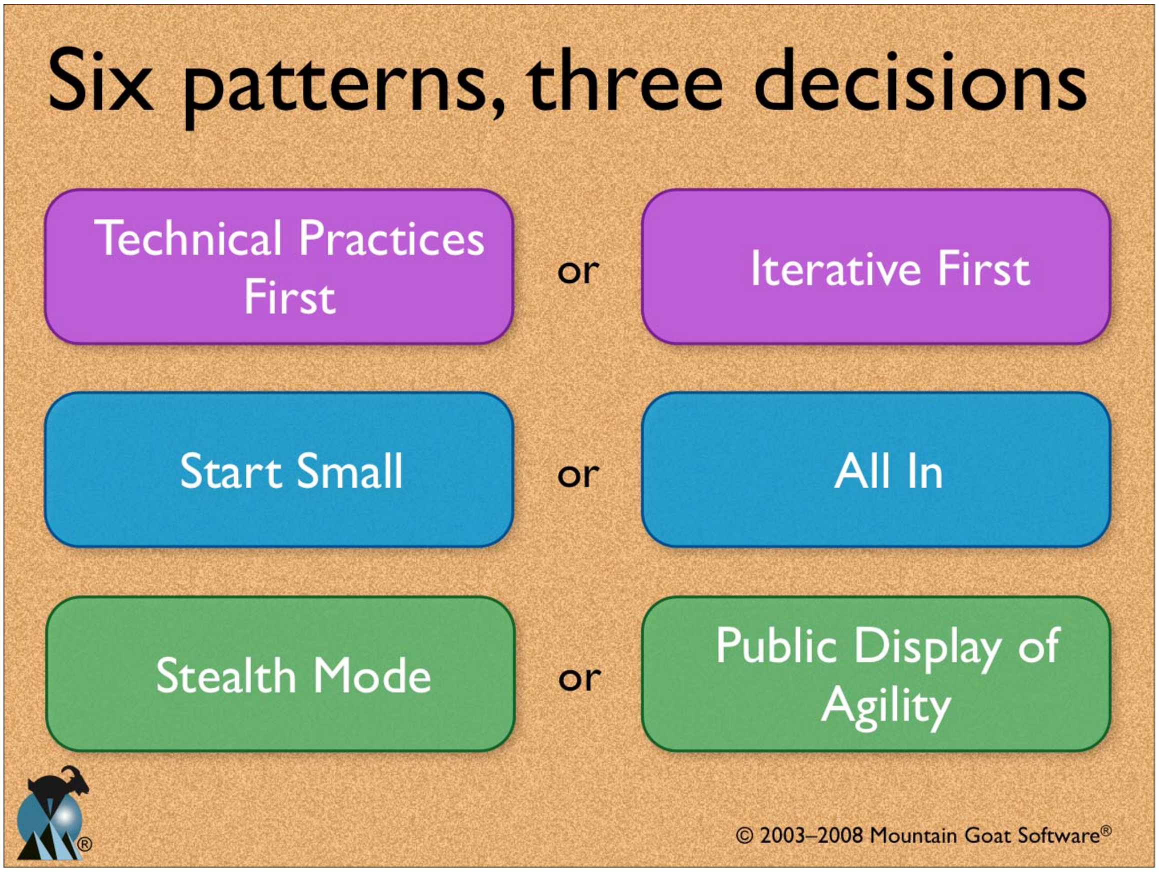


# Using the CDE model

- When stuck thinking about how to nudge the organization think of the:
  - Containers
    - formal teams, informal teams, clarify (or not) expectations
  - Differences
    - Dampen or amplify them within or between containers
  - Exchanges
    - Insert new exchanges, new people, new techniques or tools



# Patterns of Agile Adoption

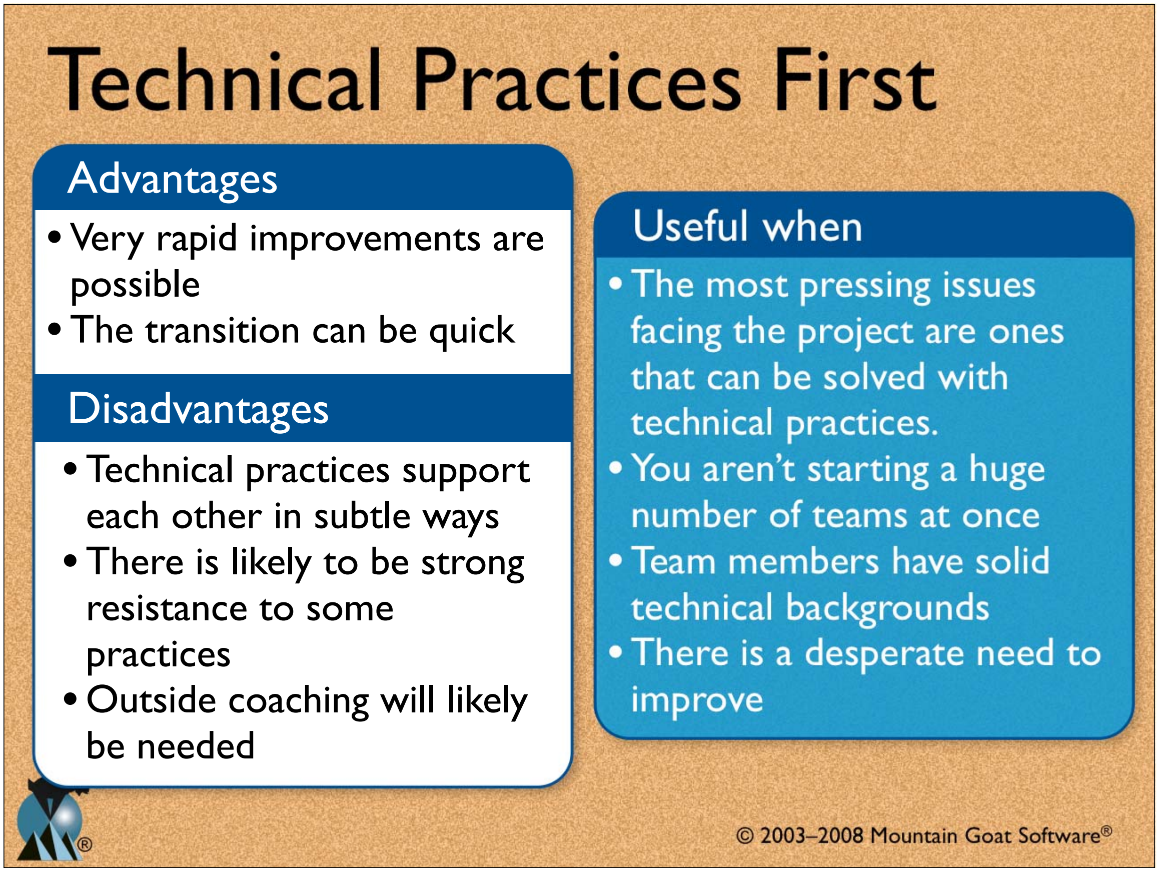


# Six patterns, three decisions

Technical Practices First or Iterative First

Start Small or All In

Stealth Mode or Public Display of Agility



# Technical Practices First

## Advantages

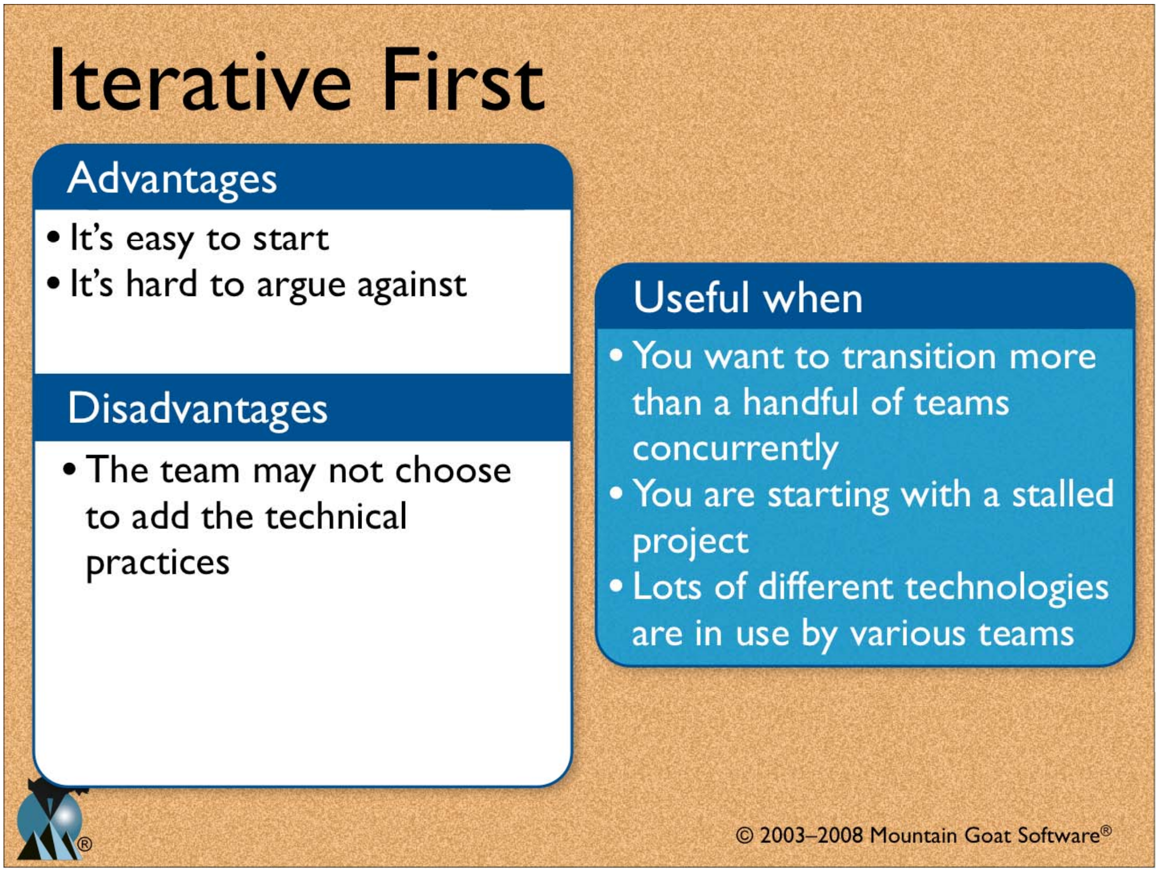
- Very rapid improvements are possible
- The transition can be quick

## Disadvantages

- Technical practices support each other in subtle ways
- There is likely to be strong resistance to some practices
- Outside coaching will likely be needed

## Useful when

- The most pressing issues facing the project are ones that can be solved with technical practices.
- You aren't starting a huge number of teams at once
- Team members have solid technical backgrounds
- There is a desperate need to improve



# Iterative First

## Advantages

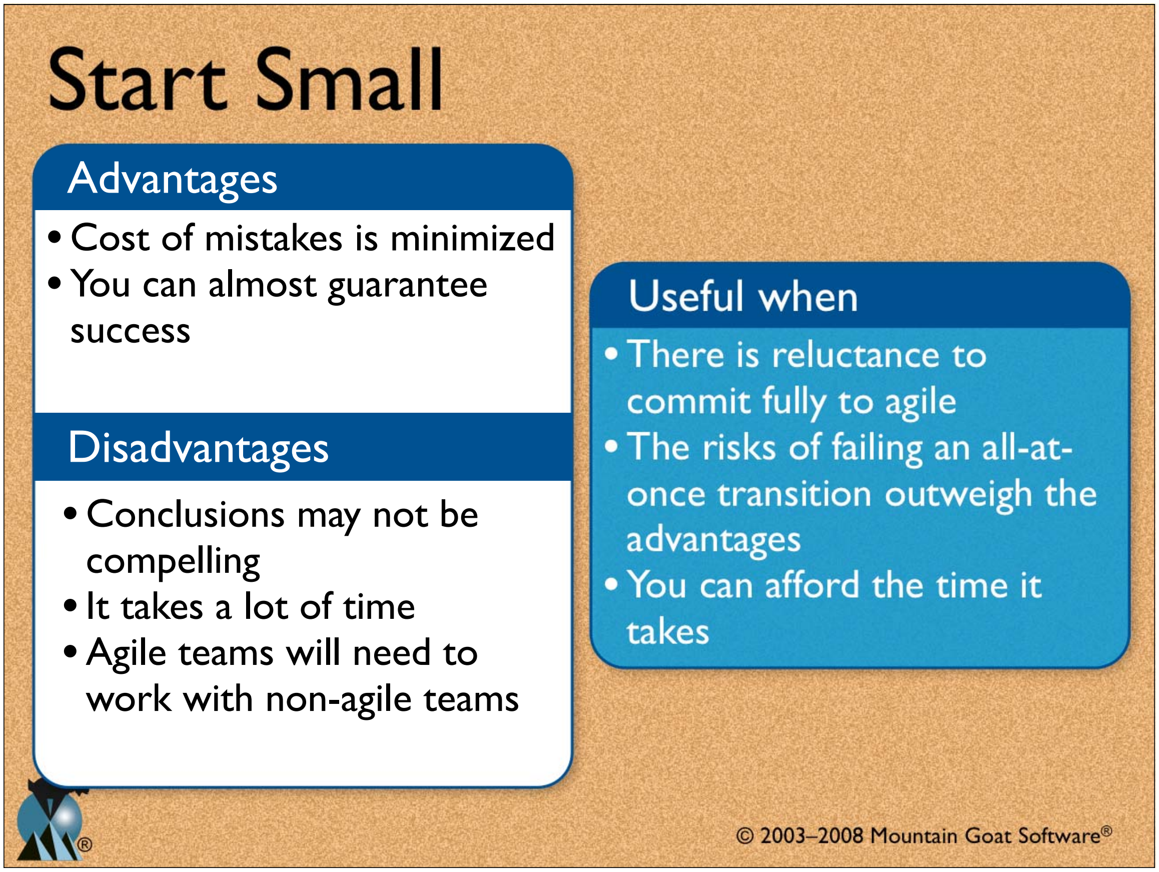
- It’s easy to start
- It’s hard to argue against

## Disadvantages

- The team may not choose to add the technical practices

## Useful when

- You want to transition more than a handful of teams concurrently
- You are starting with a stalled project
- Lots of different technologies are in use by various teams



# Start Small

## Advantages

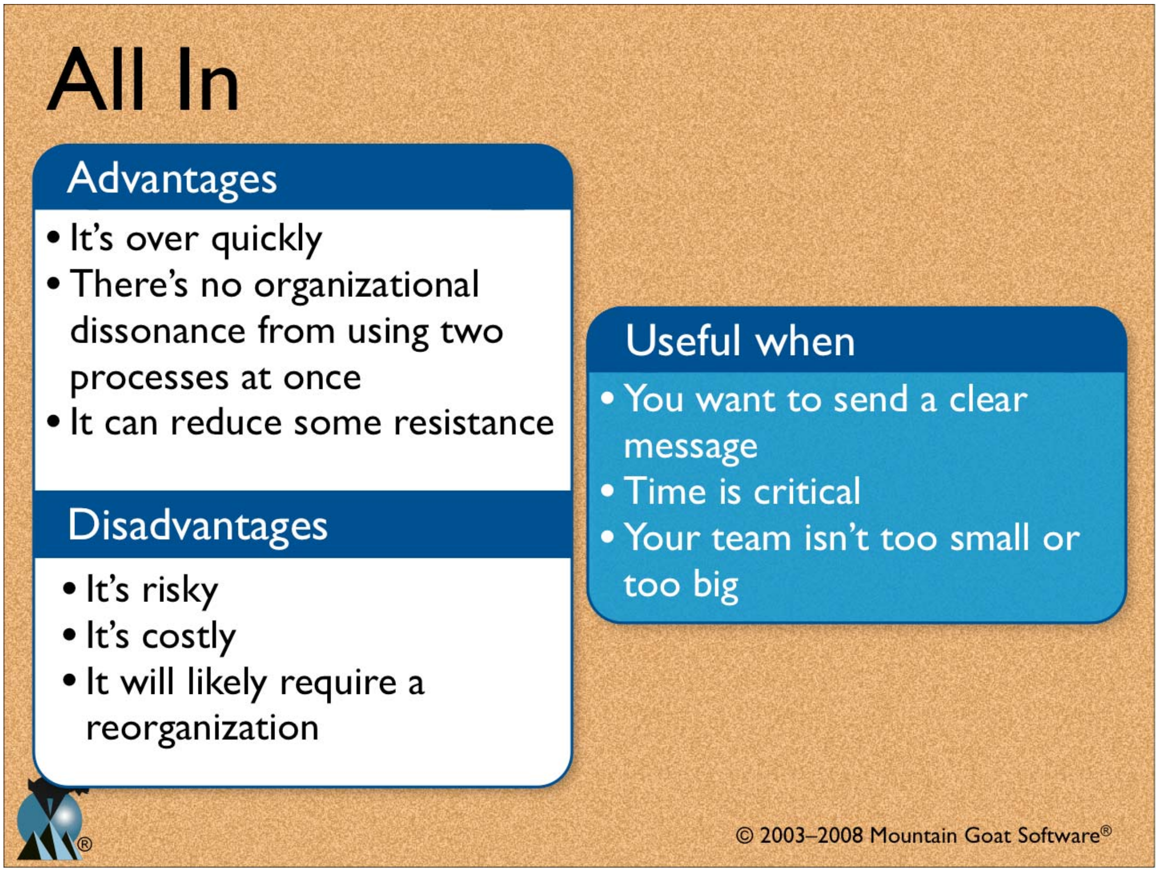
- Cost of mistakes is minimized
- You can almost guarantee success

## Disadvantages

- Conclusions may not be compelling
- It takes a lot of time
- Agile teams will need to work with non-agile teams

## Useful when

- There is reluctance to commit fully to agile
- The risks of failing an all-at-once transition outweigh the advantages
- You can afford the time it takes



# All In

## Advantages

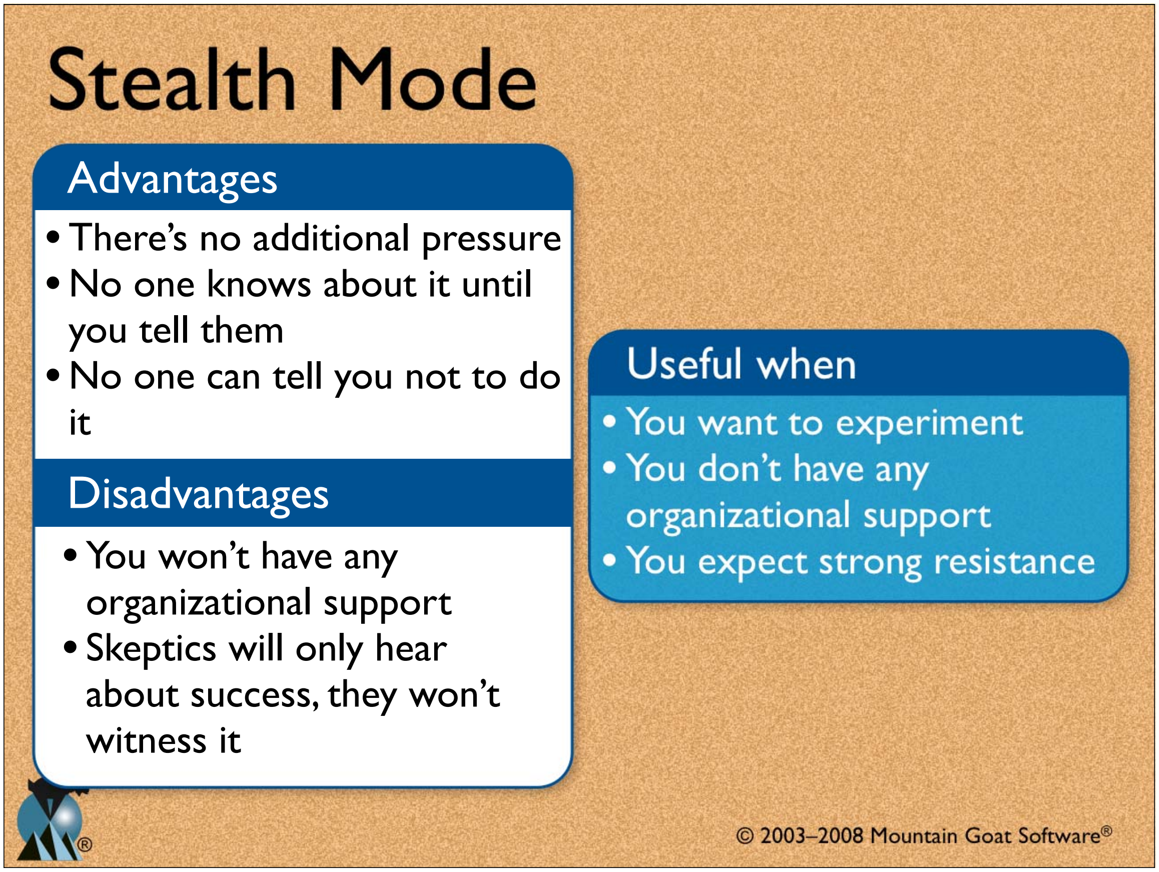
- It’s over quickly
- There’s no organizational dissonance from using two processes at once
- It can reduce some resistance

## Disadvantages

- It’s risky
- It’s costly
- It will likely require a reorganization

## Useful when

- You want to send a clear message
- Time is critical
- Your team isn’t too small or too big



# Stealth Mode

## Advantages

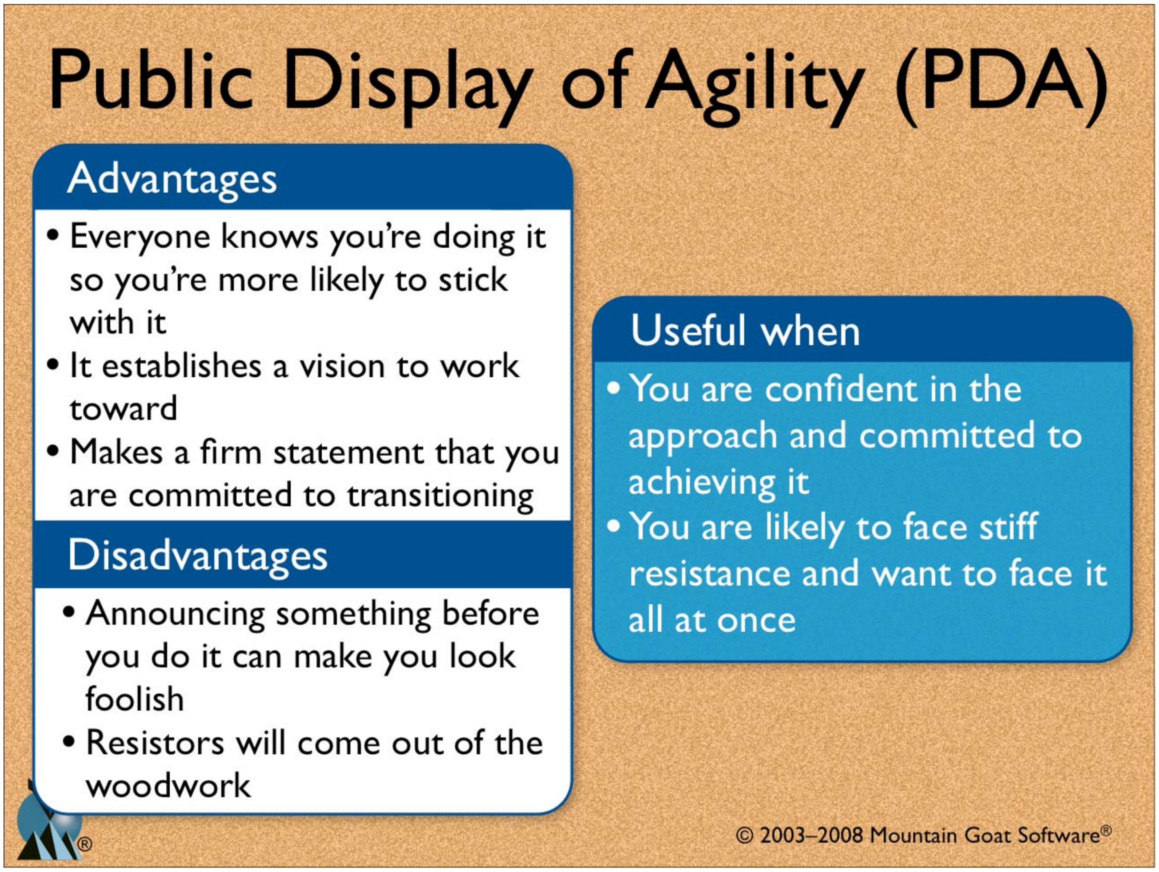
- There’s no additional pressure
- No one knows about it until you tell them
- No one can tell you not to do it

## Disadvantages

- You won’t have any organizational support
- Skeptics will only hear about success, they won’t witness it

## Useful when

- You want to experiment
- You don’t have any organizational support
- You expect strong resistance



# Public Display of Agility (PDA)

## Advantages

- Everyone knows you're doing it so you're more likely to stick with it
- It establishes a vision to work toward
- Makes a firm statement that you are committed to transitioning

## Disadvantages

- Announcing something before you do it can make you look foolish
- Resistors will come out of the woodwork

## Useful when

- You are confident in the approach and committed to achieving it
- You are likely to face stiff resistance and want to face it all at once

Robert C. Martin Series
Agile Estimating and Planning
Forewords by Jim Highsmith, Ag
Gabrielle ene ield , Director, Agi
The Addison Wesley Signature Series
A Kent Beck Signature Book
User Stories Applied
For Agile Software Development
Mike Cohn
Foreword by Kent Beck

Mike Cohn

mike@mountaingoatsoftware.com

www.mountaingoatsoftware.com

(720) 890−6110 (office)

(303) 810−2190 (mobile)

MOUNTAIN GOAT SOFTWARE