Mike Cohn - background

- 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. The role of leadership
5. Patterns of agile adoption

Why Transitioning to Agile Is Hard
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

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.
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.

Organizations are unpredictable, living systems.

- Traditional view of the organization is as a machine
- We can disassemble it, study its parts, put it back together
- Once we set it in motion, it will continue in motion
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
Each paired statement below and on the next slide describes either the traditional or CAS view of how to change an organization.

<table>
<thead>
<tr>
<th></th>
<th>Traditional view</th>
<th>CAS view</th>
</tr>
</thead>
<tbody>
<tr>
<td>Behavior is predictable and controllable</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Behavior is unpredictable and uncontrollable</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Direction is determined through emergence and by many people</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Direction is determined by a few leaders.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Every effect is also a cause</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Every effect has a cause</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Relationships are directive

Relationships are empowering

Responsiveness to the environment is the measure of value

Efficiency and reliability are measures 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
Discuss these questions

Looking back on the previous two slides, circle the item in each pair that is most closely aligned with agile.

1. Are those views the predominant views in your organization today?

2. If not, what problems do you expect to encounter while transitioning?

ADAPTING to Agile Development
Awareness that the current approach isn’t working
Desire to change
Ability to work in an agile manner
Promote early successes to build momentum and get others to follow
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?
   • Why?
2. Thinking about ADAPT, what might you try?
Individual and group change

• All individuals will need to move through the Awareness, Desire, and Ability stages
• But will do so at different rates
• Early adopters and leaders:
  • Use the Promote stage to build Awareness and Desire in later adopters
  • Need to Transfer the impact of agile to groups like Human Resources or the transition will fail

Tools for building...

Awareness
• Communicate
• Establish a vision
• Narrow the focus
• Metrics
• Run a pilot
On the following slides, identify at least five ways to build desire, create ability, promote the transition, and transfer its implications.
• My thoughts on desire, ability, transfer and promote are on the following pages.
• You don’t want to see them before you think about your own.
Tools for building...

**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

**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 transition process
Decide how pervasive to go with agile—development only or full company.

Identify which issues agile can solve or help with.

Transition backlog:
- Quarterly: 3-4 goals
  - Meet weekly to execute, monthly to plan, quarterly to strategize
  - Equivalent to daily standups
  - Equivalent to iterations
  - Equivalent to releases

- Monthly
- Weekly
  - Discuss progress
  - Remove impediments

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

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?

Your transition team

1. Who might desperately want the transition to fail?
   • Why?
   • What might you be able to do to prevent them from sabotaging the change?
2. Who might want to be on the transition team who shouldn’t be on the team?
   • Why?
What hidden agendas will people bring to the transition team?
   • What can you do to counter (or make use of) those hidden agendas?
What can you do to handle snakes who need to be on the team?
Leading an agile transition

- Transition 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)

---

### 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
Containers

- Enlarge or shrink teams
- Enlarge or shrink the responsibility boundary of teams
- Change team membership
- Create new teams or groups

Differences

- Don’t require consensus
  - Creativity comes from tension
  - Quiet disagreement is not as good as fierce debate that leads to behavior change
- Ask hard questions
  - Then expect teams to find solutions
Transforming exchanges

- Encourage communication between teams and groups
  - Who isn’t talking that should?
- Add or remove people from exchanges
  - Change reporting relationships
  - Relocate people
- Encourage learning

You are the ScrumMaster or project manager...

- The next set of slides describes some teams with some trouble spots. Think about how you might help them by changing their Containers, amplifying or dampening Differences, or changing their Exchanges.
- For each case, identify at least one thing you’d do.
- Note whether you are tweaking their Container, Differences, or Exchanges. (You might be affecting more than one.)
The team consists of four developers, two testers, a database engineer and you. The developers and testers are not working well together. Developers work in isolation until two days are left in the iteration. They then throw the code “over the wall” to the testers.

The team is failing to deliver potentially shippable software at the end of each iteration. None of the items they start are 100% finished. Their close but work is always left to be done in the next iteration.

The team seems to be consistently undercommitting during iteration planning. They finish the work they commit but it doesn’t seem like much. The product owner hasn’t complained yet but you’re worried she will soon.

Your organization has 20 different agile teams. Each team has its own testers who are starting to go in different directions in terms of preferred tools and approaches.
Jeff, a senior developer, is very domineering. During iteration planning the team defers to him on every decision even though he is a horrible estimator. You notice the glances that other team members exchange when he suggests very low estimates on some tasks.

You are responsible for two teams. Team members on one discuss all sides of various issues before making a decision. This has been working well. On the other team, discussions drag on endlessly because they pursue absolute consensus in all cases.
Six patterns, three decisions

- Technical Practices First
  - or
  - Iterative First
- Start Small
  - or
  - All In
- Stealth Mode
  - or
  - Public Display of Agility

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

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

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

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

Patterns of agile adoption
Discuss these questions:
- Which of these techniques have you used in the past?
- Was the transition successful?
- If not, would a different pattern have helped?
- What advice would you give to someone about to use one of these patterns you’ve used in the past?
- What pattern would you prefer to use in the future? What conditions would you like to be true for you to use that pattern?

Sunday, November 9, 2008
Three expansion patterns

1. Split and Seed

2. Grow and split
Internal coaching

- Attend planning meeting
- Attend 2 daily scrums per week
- Spend 4 hours with the team per sprint

Give coaches specific duties such as:

Two mix-in patterns

- Impending doom
- Requirements first
Impending Doom

**Advantages**
- It can shock the team out of complacency
- Admitting that a project is headed toward disaster can free the team to experiment
- It can help overcome a lot of resistance
- The transition can be quick

**Disadvantages**
- It isn't always an option
- A big change in a time of trouble can increase stress on the team

**Useful when**
- A project is on its way to failure unless dramatic action is taken
- Apathy has set in among team members

Requirements First

**Advantages**
- Starting with agile requirements makes it hard to avoid being agile later
- It makes introducing other practices easier

**Disadvantages**
- You have to wait until the right project is ready to start
- Starting the project takes longer than it should

**Useful when**
- There is general agreement on what to build
- You are starting a new project or restarting a failed project
- You have the discipline and skill to do this quickly
### Upcoming public classes

<table>
<thead>
<tr>
<th>Date</th>
<th>What</th>
<th>Where</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jan 27–28</td>
<td>Certified ScrumMaster Agile Estimating and Planning</td>
<td>Dallas</td>
</tr>
<tr>
<td>Jan 29</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feb 18–19</td>
<td>Certified Scrum Product Owner (with Ken Schwaber)</td>
<td>Boulder</td>
</tr>
<tr>
<td>Mar 31–Apr</td>
<td>Certified ScrumMaster Agile Estimating and Planning</td>
<td>Seattle</td>
</tr>
<tr>
<td>1 Apr 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>May 12</td>
<td>Effective User Stories Certified ScrumMaster Agile Estimating and Planning</td>
<td>Orlando</td>
</tr>
<tr>
<td>May 13–14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>May 15</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Other classes in London, Oslo and Stockholm if you’re up for a longer trip.

Information and registration at [www.mountaingoatsoftware.com](http://www.mountaingoatsoftware.com)

---

**Mike Cohn**

mike@mountaingoatsoftware.com

[www.mountaingoatsoftware.com](http://www.mountaingoatsoftware.com)

(720) 890–6110 (office)

(303) 810–2190 (mobile)