An Agile Practice Framework
for Scaling Agile Adoption
in an Enterprise

by Brad Appleton

April 23, 2013
Agenda

• Background
• A Framework
• Mindset & Values
• Principles
• Practices & Practice-Areas
• Adoption
• Scaling
Background

A quick overview
Overview

Have an existing community-of-practice of agile coaches, developers

- A few have been part of agile efforts that started between 1999-2003
- More have been part of efforts since >=2007
- The “third wave” started around 2008-2010
- Had strong sr. mgmt sponsorship in late 2000’s
- Still have sr. mgmt support for Agile & Lean
- (but visibility & momentum has waned)
Waves of Adoption & Support

First wave of agile adoption efforts (1999-2003) were grass-roots practitioner-driven efforts that started bottom-up at the individual & team-level.

As the number of teams and interest grew, this gave rise to a second-wave (2003-2007) at the organization-level with top-down sponsorship/acceptance.
Waves of Adoption & Support [cont.]

The Third-wave (2008-2012) took an *outside-in* approach, using a “*middle-management sandwich*” to help grow more agile coaches and teams, but experienced some *new challenges* ...

- Many teams & practitioners were new to agile
  - Less pre-existing expertise in agile principles & practices
- Tailoring communication messages & meanings for multiple audiences
  - New practitioners, middle-management, and senior management
  - Each needs different emphasis on different sets of things at different levels of detail
How are adoption efforts going?

Some teams are further along in their progress than others

- Some are “doing agile” or even “being agile”
- Many others are still struggling to get more than just minor improvement
- Still more have stalled due to loss of momentum or critical-mass *(some have even regressed backward)*

*What can be done to get back on track?!*
What is the barrier/plateau?

Progress (and motivation) has been slowing

1. To make important cultural and mind-set changes
2. To fully realize the significance of key principles, technical practices and technical debt
3. To escalate & overcome external impediments
4. To adequately grow automation capability + test-coverage.

No clear milestones other than “initial state” and “desired state” – the gap is too intimidating!
An Agile Framework?

Challenges, Goals & Objectives
Evolution -vs- Revolution

Agile Transformation: *Revolutionary*
- Getting an entire organization to change it’s culture and practices, together. *(Very often top-down!)*
- Learn “The whole enchilada” and begin training and adoption of (most) all the practices at once.

Incremental Transition: *Evolutionary*
- Getting a few key people & teams at a time to learn and adopt a practice or two at a time. *(Very often bottom-up!)*
- Gradually building competence with a handful of practices and teams before trying a new one.

*Existing organizational-culture and business-context determines which approach is acceptable!*
Doing Agile versus Being Agile

**Being Agile:** *Internalizing the mindset, values, and principles* then applying the right practices and tailoring them to different situations as they arise.

**Doing Agile:** Executing the practices *as closely as possible* to “as prescribed” description, and trying to “inspect and adapt” to remove impediments to achieving the “as prescribed” execution.

- **Don’t “Do” Agile. Be Agile!**
  – Alan Kelly

- **Stop ‘Doing Agile’ Start ‘Being Agile’!**
  – Jim Highsmith
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.

http://www.halfarsedagilemanifesto.org/
WANTED: more adoption guidance and progress milestones (and scaling)

Looking for a Framework …

That synthesizes existing/proven frameworks

- Practices from the full range of methods:
  - XP, Scrum, Lean, FDD, AUP

- Team-Level Agility:
  - Disciplined Agile Delivery (DAD) — Ambler & Lines
  - Distributed Scrum

- Scaling:
  - Scaled Agile Framework (SAFe) — Leffingwell, et.al.
  - Enterprise Agile Delivery — Alan Brown, et.al.

- Learning and Mastery:
  - Dreyfus model, other models,
Three Major Methodologies

Individual practices from agile (A), lean (L), and unified (U) methodologies fit well with certain IT cultures and not others. Lines on the map show the sweet spot of each approach. Use this map to identify those practices that work together best at your organization.

- Agile practices’ sweet spot
- Lean practices’ sweet spot
- Unified process practices’ sweet spot

Data: Application of Diagnosing and Changing Organizational Culture, Cameron and Quinn, 2006
Other important requirements

• Not looking to do top-down “transformation” of a whole program or organization

• Start small (individual teams) and help them incrementally grow their agility “up” and “out”

• Striving for high-cohesion within teams and loose-coupling between teams

• Existing frameworks (e.g., DAD, SAFe) are good as long as proceed in an evolutionary fashion
An Agile Practice Framework

Agile Scaling

Agile Results

Agile Practices

Agile Principles

Agile Values

Agile Adoption
The Agile Practice Framework

• The Agile Mindset
• 4 Agile Values
• 6x2 Agile Principles
• 6 Agile Practice Areas
• 6 Stages of Agile Adoption/Evolution
• 6 Levels of Agile Scaling
The “Agile” mindset

Agile is a mindset

- Defined by values
- Guided by principles
- Manifested through many different practices

—Ahmed Sidky
Values, Principles & Practices

Values bring purpose to practices.

Practices bring accountability to values.

Principles bridge values and practices by guiding behavior.

― Kent Beck, Extreme Programming Explained (2nd edition)
Which comes first?

“Practices by themselves are barren. Unless given purpose by values, they become rote.”

—Kent Beck, Extreme Programming Explained (2nd edition)

“Is it better to spend your time understanding principles or studying practices? We observe that the best results come from combining the two approaches. Copying practices without understanding the underlying principles has a long history of mediocre results.”

—Mary & Tom Poppendieck, Implementing Lean Software Development, Chapter 2 - Principles
Agile Values (from the Agile Manifesto)

The Agile Manifesto establishes a set of **four values** that are **people-centric** and **results-driven**:

- **Individuals & Interactions** over **Processes & Tools**
- **Working Software** over **Comprehensive Documentation**
- **Responding to change** over **Following a plan**
- **Customer Collaboration** over **Contract Negotiation**

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

Agile Principles are Guiding Themes

- Continuous Delivery of Customer-Value and Working Software Defines Progress
- Welcome Change – Reflect and Adapt
- Deliver Working Software Frequently at a Sustainable Pace
- Collaborate Daily Across Functions and Convey Information via Conversation
- Align, Support & Trust … Self-Organizing Teams
- Continuous Technical Excellence and Simplicity is Essential
Values and Principles

- Continuous Value thru Working Software
- Welcome Change -- Reflect and Adapt
- Individuals & Interactions
- Responding to change
- Working Software
- Customer Collaboration
- Continuous Simplicity & Technical Excellence
- Collaborate Daily via Conversation
- Deliver Frequently at a Sustainable Pace
- Align, Support & Trust Self-Organizing Teams
Grouping Practices into Practice-Areas

• Each of the 10-12 initial agile practices are really several smaller practices put together
• Each one of the smaller (“canonical”) practices is often easier to learn and assimilate
• But the full-list of all the “canonical” practices can be very intimidating!
• Grouping them into practice-areas makes them easier to present, explain, learn and digest
• Use the values and principles as a guide for how to divide into practice-areas
Grouping Values & Principles into ...

- Continuous Customer-Value
- Align, Support & Trust
- Self-Organizing Teams
- Continuous Technical Excellence
- Welcome Change
- Individuals & Interactions
- Working Software
- Simplicity is Essential
- Working Software Defines Progress
- Responding to change
- Customer Collaboration
- Deliver Frequently
- Reflect and Adapt
- Collaborate Daily Across Functions
- Convey Information via Conversation
- Sustainable Pace
Agile Practice-Areas

Project & Program Management

Programming, Design & Architecture

Testing & Specification

Integration, Build & Release

Collaboration & Leadership

Business Analysis & Value Management
Viewing Practices by Practice-Area

• Even for a given practice-area, the full list of practices can still be very daunting.

• Plus some “stepping stone” practices are often necessary to transition to a specific practice.

  – E.g., being able to build daily (or even weekly) prior to integrating continuously
<table>
<thead>
<tr>
<th>Practice Area</th>
<th>Practices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project &amp; Program Management</td>
<td>Release Plan, Iteration Plan, Velocity Tracking, Buffer Management, Sprint Goals &amp; PSIs, Iteration Review, Task/Kanban Board, Release Train (ART), etc.</td>
</tr>
<tr>
<td>Collaboration &amp; Leadership</td>
<td>Team Coaching/Facilitation, Pairing, Standups, Retrospectives, Collective Ownership, Customer Proxy, Definition of DONE, Cross-Functional Teams, etc.</td>
</tr>
<tr>
<td>Integration, Build &amp; Release</td>
<td>Continuous Integration, Daily (Staged CI) Build, Continuous Inspection, Deployment Pipeline, Continuous Delivery, etc.</td>
</tr>
<tr>
<td>Testing &amp; Specification</td>
<td>Automated Testing, Concurrent Testing, Acceptance Criteria, ATDD, Specification by Example, Test-Driven Requirements, etc.</td>
</tr>
<tr>
<td>Business Analysis &amp; Value Mgmt</td>
<td>Agile Product Mgmt, Backlog Mgmt, MVP/MMF, Stories &amp; Spikes, Epics &amp; Themes, Value Streams, WSJF Prioritization, Impact Mapping, Agile UX Design, etc.</td>
</tr>
</tbody>
</table>
Adoption Stages/Milestones

- Still need to breakdown adoption efforts into many smaller milestones
- This breaks-up the long journey of agile adoption and continuous improvement into “baby steps”
- *Needed for overall adoption and by practice-area!*
- Each step/stage represents an important transition to a higher-level of skill mastery, and understanding
- Identifies the important stages where teams often get stuck and don’t progress further (or regress)
- Teams can strive for the next attainable near-term milestone without losing sight of desired destination
# Stages of Agile Adoption/Evolution

<table>
<thead>
<tr>
<th>#</th>
<th>Stage Name</th>
<th>Skill Level</th>
<th>Expected Results / Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.</td>
<td>Regressive</td>
<td>Legacy</td>
<td>$\leq 0$</td>
</tr>
<tr>
<td>1.</td>
<td>Exploring Agility</td>
<td>Beginner</td>
<td>$0.10X - 0.25X$</td>
</tr>
<tr>
<td>2.</td>
<td>Emerging Agility</td>
<td>Advanced Beginner</td>
<td>$0.25X - 1X$</td>
</tr>
<tr>
<td>3.</td>
<td>Principled Agility</td>
<td>Intermediate</td>
<td>$1X - 2X$</td>
</tr>
<tr>
<td>4.</td>
<td>Disciplined Agility</td>
<td>Proficiency</td>
<td>$2X - 4X$</td>
</tr>
<tr>
<td>5.</td>
<td>Transforming Agility</td>
<td>Mastery</td>
<td>$4X - 10X$</td>
</tr>
<tr>
<td>6.</td>
<td>Transcending Agility</td>
<td>Supremacy</td>
<td>???</td>
</tr>
</tbody>
</table>
### Stages of (Team) Learning & Mastery for Skill Acquisition/Execution

<table>
<thead>
<tr>
<th>Stage</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>My Proposal</strong></td>
<td>Exploring</td>
<td>Emerging</td>
<td>Principled</td>
<td>Disciplined</td>
<td>Transformed</td>
</tr>
<tr>
<td>Dreyfus Model</td>
<td>Novice</td>
<td>Advanced Beginner</td>
<td>Competent</td>
<td>Proficient</td>
<td>Expert</td>
</tr>
<tr>
<td>Agile Skills Project</td>
<td>Questioning</td>
<td>Learning</td>
<td>Practicing</td>
<td>Journeying</td>
<td>Mastering / Contributing</td>
</tr>
<tr>
<td>ThoughtWorks*</td>
<td>Neutral / Chaotic</td>
<td>Collaborative</td>
<td>Operating</td>
<td>Adapting</td>
<td>Innovating</td>
</tr>
<tr>
<td>AgileFluency</td>
<td>One Star (Focus on Value)</td>
<td>Two Star (Deliver Value)</td>
<td>Three Star (Optimize Value)</td>
<td>Four Star (Optimize for Systems)</td>
<td></td>
</tr>
<tr>
<td>Sidky/SAMI*</td>
<td>Collaborative</td>
<td>Evolutionary</td>
<td>Integrated</td>
<td>Adapting</td>
<td>Encompassing</td>
</tr>
<tr>
<td>ICAgile Model</td>
<td>Associate/Apprentice</td>
<td>Professional</td>
<td>Expert</td>
<td>Master</td>
<td></td>
</tr>
<tr>
<td>Shu-Ha-Ri</td>
<td>Shu</td>
<td>Ha</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Integration, Build & Release

- 0: Big-Bang Integration
- 1: Incremental Integration
- 2: Daily Build
- 3: Continuous Integration
- 4: Continuous (Quality) Inspection
- 5: Continuous Delivery
- 6: Continuous Feedback
### Stages of Mastery by Skill-set

<table>
<thead>
<tr>
<th>Regressive</th>
<th>Exploring</th>
<th>Emerging</th>
<th>Principled</th>
<th>Disciplined</th>
<th>Transformed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline for Improvement</td>
<td>.1X – .25X Improvement</td>
<td>.25X – 1X Improvement</td>
<td>1X - 2X Improvement</td>
<td>2X - 4X Improvement</td>
<td>4X - 10X Improvement</td>
</tr>
<tr>
<td><strong>Project Mgmt</strong></td>
<td>Predictive Control</td>
<td>Scrummerfall</td>
<td>Incremental</td>
<td>Water-Scrum-Fall</td>
<td>Iterative &amp; Incremental</td>
</tr>
<tr>
<td><strong>Design &amp; Architecture</strong></td>
<td>BDUF</td>
<td>Clean Coding</td>
<td>“Just Enough, Just-in-Time”</td>
<td>Validated Learning</td>
<td>Test-First &amp; Refactoring</td>
</tr>
<tr>
<td><strong>Collaboration &amp; Leadership</strong></td>
<td>Self-Serving</td>
<td>Self-Motivating</td>
<td>Self-Awareness</td>
<td>Self-Reliant</td>
<td>Self-Disciplined</td>
</tr>
<tr>
<td><strong>Integration, Build, Release</strong></td>
<td>“Big Bang” Integration</td>
<td>Incremental Integration</td>
<td>Daily Build</td>
<td>Continuous Integration</td>
<td>Continuous Inspection</td>
</tr>
<tr>
<td><strong>BA &amp; Value Mgmt</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Levels of Agile Scaling

<table>
<thead>
<tr>
<th>#</th>
<th>Name of Level</th>
<th>Scope of Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.</td>
<td>Individual</td>
<td>personal</td>
</tr>
<tr>
<td>1.</td>
<td>Team</td>
<td>interpersonal – across people</td>
</tr>
<tr>
<td>2.</td>
<td>Project/Feature</td>
<td>across places/sites and phases/functions</td>
</tr>
<tr>
<td>3.</td>
<td>Product-Line</td>
<td>across product-lines and their projects/features</td>
</tr>
<tr>
<td>4.</td>
<td>Program</td>
<td>across products &amp; projects for a system/solution</td>
</tr>
<tr>
<td>5.</td>
<td>Portfolio</td>
<td>across systems/solutions and their programs</td>
</tr>
<tr>
<td>6.</td>
<td>Enterprise</td>
<td>across businesses, markets, and their portfolios</td>
</tr>
</tbody>
</table>
## Agile Practice Scaling

<table>
<thead>
<tr>
<th>Level</th>
<th>Exploring</th>
<th>Emerging</th>
<th>Principled</th>
<th>Disciplined</th>
<th>Transformed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Team</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project / Feature</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Product Line</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Program</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Portfolio</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
An Agile Practice Framework for Scaling Agile Adoption in an Enterprise

Brad Appleton
blog.bradapp.net

An Agile Framework

Agile Values

Agile Principles

Agile Practices

Agile Results

Agile Scaling
An Agile Practice Framework for Scaling Agile Adoption in an Enterprise

Brad Appleton
blog.bradapp.net

APLN Chicago
Collaboration & Leadership

Continuous Value thru Working Software

Project & Program Mgmt

Deliver Frequently at a Sustainable Pace

Align, Support & Trust Self-Organizing Teams

Individuals & Interactions

Responding to change

Customer Collaboration

Welcome Change -- Reflect and Adapt

Business Analysis & Value Mgmt

Exploring
Emerging
Principled
Disciplined
Transformed

Programming, Design, Architecture

Integration, Build/Release

Team

Project/Feature
Product-Line
Program
Portfolio

APLN Chicago
April 23, 2013
An Agile Framework
Thank You!

Shokran
Gracias
Juspajaraña
Teşekkürler
Salamat
Xie xie
Bedankt
Khawn khun
Obrigado!
Arigato
Díky
Mahalo
Mahalo
Spacibo
Vielen Dank
Grazie
Köszönettel
Thank You!
References


- **Doing Agile versus Being Agile**
  - Doing agile isn’t the same as *being* agile, Bob Hartman, [http://www.slideshare.net/lazygolfer/doing-agile-isn’t-the-same-as-being-agile](http://www.slideshare.net/lazygolfer/doing-agile-isn’t-the-same-as-being-agile)
  - Stop doing Agile! Start being Agile!, Jim Highsmith, [http://www.youtube.com/watch?v=cZsB8-7K13o](http://www.youtube.com/watch?v=cZsB8-7K13o)

- **Disciplined Agile Delivery,** [www.disciplinedagiledelivery.com](http://www.disciplinedagiledelivery.com)

- **Scaled Agile Framework,** [scaledagileframework.com](http://scaledagileframework.com)

- **Distributed Scrum,** [www.distributedscrum.com](http://www.distributedscrum.com)


- **Practical Approach to Large-Scale Agile Development**
  - [http://www.amazon.com/dp/0321821726](http://www.amazon.com/dp/0321821726)
  - [http://www.infoq.com/articles/practical-large-scale-agile](http://www.infoq.com/articles/practical-large-scale-agile)

- **James Shore on Large-Scale Agile,** [http://www.jamesshore.com/Blog/Large-Scale-Agile.html](http://www.jamesshore.com/Blog/Large-Scale-Agile.html)
References [continued]


- **Models of Learning and Mastery:**
  - [http://www.agileskillsproject.org/skill-levels](http://www.agileskillsproject.org/skill-levels)
  - [http://martinfowler.com/articles/agileFluency.html](http://martinfowler.com/articles/agileFluency.html)
  - [http://martinfowler.com/bliki/ShuHaRi.html](http://martinfowler.com/bliki/ShuHaRi.html)
  - [http://alistair.cockburn.us/Shu+Ha+Ri](http://alistair.cockburn.us/Shu+Ha+Ri)

- **Continuous Inspection:**


BACKUP Slides

Supplementary material and work-in-progress
What is Agile Software Development?

“An iterative and incremental (evolutionary) approach to software development
- performed in a highly collaborative manner
- by self-organizing teams with "just enough" ceremony
- producing high quality software in a cost effective and timely manner
- to meet the changing needs of its stakeholders.”
  — Scott Ambler, IBM Practice Leader, Agile Development

“Ultimately, Agility is about:
- Embracing change rather than attempting to resist it
- Focus on talent and skills of individuals and teams.”
  — James Highsmith, Cutter Consortium
Importance of Technical Practices

The defining characteristic of software is that it is *soft*.  

• The easier software is to change, the easier it is to achieve any of its other required characteristics.  

• Both *architectural integrity and process effectiveness* will drive the cost of change.  

Therefore, *agility* is not solely an attribute of a process. It is equally, if not more, *an attribute of good design*.  

• *When* architectures are resilient and platforms support change automation and measurement,  

• *then* projects can optimize their resource investments ... and steer toward better outcomes for all stakeholders.  

— *Walker Royce*, *Measuring Progress and Quality Honestly*
Programming, Design & Architecture

- 0: BDUF
- 1: Clean Coding
- 2: Incremental Design ("Just Enough, Just-In-Time")
- 3: Validated Learning (Automated Unit-Testing)
- 4: Emergent Design (Test-First & Refactoring)
- 5: Supple Design
- 6: Sustainable, Evolutionary Architecture
Project & Program Management

- **0: Conformance to Plan** *(Predict & Control)*
- **1: Scrummerfall** *(Timeboxed with velocity & burn-up/down)*
- **2: Incremental** *(Including Two-Level Planning)*
- **3: Water-Scrum-Fall** *(Up-front reqts/arch with test-phases)*
- **4: Iterative & Incremental* 
- **5: Adaptive & Evolutionary** *(Scrum & Kanban)*
- **6: Scrumban/Kanban & Lean* 

**-regressive**
- **Exploring**
- **Emerging**
- **Principled**
- **Disciplined**
- **Transformed**
• Cadence
• Flow
• Pull
• Visibility & Transparency
• Seeing WIP & Waste
• Measuring Feedback & Cycle-time
Team Leadership & Collaboration

- **0: Self-Serving** *(Command & Control)*
- **1: Self-Motivating** *(Accepting Responsibility)*
- **2: Self-Awareness** *(Empathy & Openness)*
- **3: Self-Reliant** *(Trusting, Attentive & Conflict Resolution)*
- **4: Self-Discipline** *(Committed & Accountable)*
- **5: Self-Organizing** *(Reflective & Responsive)*
- **6: Self-Renewing** *(Reinvention of self)*
## Integration, Build & Release

<table>
<thead>
<tr>
<th></th>
<th>&quot;Big Bang&quot; Integration</th>
<th>Incremental Integration</th>
<th>Daily Build</th>
<th>Continuous Integration</th>
<th>Continuous Inspection</th>
<th>Continuous Delivery</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regressive</td>
<td>Exploring (.1X – .25X)</td>
<td>Emerging (.25X – 1X)</td>
<td></td>
<td>Principled (1X - 2X)</td>
<td>Disciplined (2X - 4X)</td>
<td>Transformed (4X - 10X)</td>
</tr>
<tr>
<td><strong>Building</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Integrating</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Environment/Staging</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Testing</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Reporting</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Collaborating</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Disciplined Agile Delivery

**Identification and Prioritization**
- Initial vision and funding
- Initial modeling, planning, and organization

**Inception**
- Identify, prioritize, and select projects
- Initial architectural vision

**Construction**
- Highest-priority work items
- Iteration planning session to select work items and identify work tasks for current iteration
- Funding
- Feedback

**Transition**
- Iteration review & retrospective: Demo to stakeholders, determine strategy for next iteration, and learn from your experiences
- Release solution into production
- Operate and support solution in production

### Roadmap

<table>
<thead>
<tr>
<th>Inception</th>
<th>Construction</th>
<th>Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>One or more short iterations</td>
<td>Many short iterations producing a potentially consumable solution each iteration</td>
<td>One or more short iterations</td>
</tr>
<tr>
<td>Stakeholder consensus</td>
<td>Project viability (several)</td>
<td>Production ready</td>
</tr>
<tr>
<td>Proven architecture</td>
<td>Sufficient functionality</td>
<td>Delighted stakeholders</td>
</tr>
</tbody>
</table>
Disciplined Agile Delivery

Scott Ambler & Mark Lines, 
*Disciplined Agile Delivery*
http://disciplinedagiledelivery.com/

**Figure 1.4** A lean version of the DAD lifecycle
6 F’s

1. **Focus** *(on Value)*
2. **Fast** *(at creating and responding to change)*
3. **Flexible** *(in adopting and adapting to changes)*
4. **Frequent**
5. **Flow**
6. **Feedback**

- Value == Focus on value
- Value-Stream == Fast ... Across the Value-Stream
- Flow
- Pull == Feedback?
Foundations of Agile Teaming

Self Organization

Self Discipline

Self Awareness

Scott Ambler & Mark Lines, Disciplined Agile Delivery
http://disciplinedagiledelivery.com/
5 dysfunctions of a Team

1. Absence of Trust
2. Fear of Conflict
3. Lack of Commitment
4. Avoidance of Responsibility
5. Inattention to Results

Elements of Self-Organization from Nonaka and Takeuchi

A group possesses a self-organizing capability when it exhibits three conditions

1. Autonomy
2. Self-transcendence
3. Cross-fertilization

Levels of Learning & Feedback Loops

Context

- How do we decide what is right?
  - Triple-Loop Learning
    (Transformational: Changing the Culture)

Assumptions

- Are we doing the right things?
  - Double-Loop Learning
    (Reframing: Changing the Process)

Actions

- Are we doing things right?
  - Single-Loop Learning
    (Information: Following the Process)

Results
### Types of Complexity

<table>
<thead>
<tr>
<th>Complexity</th>
<th>Low</th>
<th>High</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dynamic</strong></td>
<td>Cause and effect are close together in space and time.</td>
<td>Cause and effect are far apart in space and time.</td>
</tr>
<tr>
<td><strong>Focus is on various parts or the whole system?</strong></td>
<td>Solutions can be found by testing and fixing one part at a time.</td>
<td>Solution can be found only when situation is understood systemically, taking account of the interrelationships among the parts and the functioning of the system as whole.</td>
</tr>
<tr>
<td><strong>Generative</strong></td>
<td>Future is familiar and predictable.</td>
<td>Future is unfamiliar and unpredictable.</td>
</tr>
<tr>
<td><strong>Solutions are planned or emergent?</strong></td>
<td>Solutions from the past or other places can be repeated or replicated.</td>
<td>Solutions cannot be calculated in advance based on what has worked in the past. Emergent solutions have to be worked out as situations unfold.</td>
</tr>
<tr>
<td><strong>Social</strong></td>
<td>People involved have common assumptions, values, rationales and objectives.</td>
<td>People involved look at things very differently.</td>
</tr>
<tr>
<td><strong>Solutions come from leaders or from participants?</strong></td>
<td>A leader or expert can propose a solution with which everyone agrees.</td>
<td>Solutions cannot be given by authorities; the people involved must participate in creating and implementing solutions.</td>
</tr>
</tbody>
</table>
Cynefin Framework

Complex
- Probe
- Sense
- Respond
- Emergent

Complicated
- Sense
- Analyze
- Respond
- Good Practice

Chaotic
- Act
- Sense
- Respond
- Novel

Simple
- Sense
- Categorize
- Respond
- Best Practice
Software Development as a Complex System with Adaptive Control

Perceived Scope

\[ \sum \]

Error = Perceived − Actual Scope

Influence

Effort (Knowledge)

Complex Adaptive System

Acceptance Tests

Realized Scope (Working Software)

feedforward

feedback
Software Development as a Complex System with Adaptive Control

- When demystified, we have what is referred to in Control Theory as an Optimal Control Problem.
- Specifically, management beyond just the application of people serves as the project initiation document controller within a closed-loop, socio-technical system.
- When one learns about the sub-branch of the General Systems Theory called Control Systems Engineering, it becomes very clear that the true universal that must be in place to yield an optimal delivery is negative feedback at the correct frequency, with adaptive structure and gain scheduling applied as the system's environment changes.
- This was what the folks who articulated the Agile Manifesto and various Agile methods, albeit anecdotally and tacitly, where trying to say.

-- Mark Kennaley, SDLC 3.0: Beyond a Tacit Understanding of Agile
(see http://blogs.pmi.org/blog/voices_on_project_management/2010/11/can-agile-conquer-the-physics.html#comment-55595)
Agility and Rapid Response

Responding quickly and effectively to change is easiest to do when you can minimize the following:

– the cost of knowledge-transfer between individuals
– the amount of knowledge that must be captured in intermediate artifacts
– the duration of time between making a project decision, and exploring its results to learn the consequences of implementing that decision

“The team can be more effective in responding to change if it can reduce the elapsed time between making a decision to seeing the consequences of that decision, and reduce the cost of moving information between people.”
—James Highsmith

“One of several reasons why agile techniques are so effective is that they reduce the feedback cycle between the generation of an idea (perhaps a requirement or a design strategy) and the realization of that idea. This not only minimizes the risk of misunderstanding, it also reduces the cost of addressing any mistakes.”
—Scott Ambler
Requirements for Attaining Agility

In order to be Agile in responding to change, the following are necessary ...

1. Quick Customer feedback
2. Plans must be quick+easy to change
3. Team Communication must be quick+effective
4. Artifacts (Specs/Design/Code) must be quick+easy to change
5. Validation must be quick+easy to perform
6. Processes must be quick+easy to change

This applies to EACH feedback loop at each level!