KEY CONCEPTS OF AGILE
Ahmed Sidky, Ph.D. (aka Dr. Agile)
Ahmed Sidky

- Co-Author of “Becoming Agile”
- Director of Agile Services as TenPearls
- Over 10 years of dev and delivery experience
- Masters in Software Engineering
- Ph.D in Agile Adoption - Virginia Tech
- Agile Educator, Coach and Consultant
- Frequent Presenter at Conferences
- Recently the Program Chair of Agile 2009
A Word About TenPearls

- Process Automation, Optimization
- Outsourcing
- Software Products and Services
- Technology and IT Consulting
Worldwide Locations

London
United Kingdom

Dubai
United Arab Emirates

Virginia
United States of America

Cairo
Egypt

Karachi
Pakistan
Some of Our Customers
Warming up ...

Five Volunteers, please :)
The... ‘relay race’ approach to product development... may conflict with the goals of maximum speed and flexibility. Instead a holistic or ‘rugby’ approach - where a team tries to go the distance as a unit, passing the ball back and forth - may better serve today’s competitive requirements.

Harvard Business Review, January 1986
Why Agile ...
- *Chasing the Rabbit* by Steven Spear

- Describes what sets high-velocity, market-leading organizations apart and explains how to lead the pack in your industry

- Toyota, Aloca, Pratt and Whitney, US Navy’s Nuclear Power Program and many many more.

- Structure and dynamics of high-velocity organizations
Managing the Functions as Part of the Process

They avoid “siloization”
Functional integration at all levels – everyday
Each piece of work be done with an eye to the larger process
Avoid phrases like “You do your job and I’ll do mine”
Chasing the Rabbit

Continually Improving the Pieces and the Process

Constantly experimenting and learning more about the work
Getting rid of the problem once and for all
Constantly modifying the way they work
Each piece of work be done in such a way as to bring problems to the attention of those who can best analyze and solve them
Do not encourage workarounds and firefighting
Avoid phrases like: “This will do for now” or “Don’t worry this happens all the time”
We wish were true
- The customer knows *exactly* what he wants
- The developers know *exactly* how to build it
- Nothing will change along the way

We have to live with
- The customer discovers what he wants
- The developers discover how to build it
- Many things change along the way
Traditional Approach to Software Development

“Try Harder, I am sure it will work !!!”

“Great, I think it worked”
The Reality of Software

<table>
<thead>
<tr>
<th>Predictable Manufacturing (Defined Process)</th>
<th>New Product Development (Empirical Process)</th>
</tr>
</thead>
<tbody>
<tr>
<td>It is possible to first complete specifications, and then build.</td>
<td>Rarely possible to create upfront unchanging and detailed specs.</td>
</tr>
<tr>
<td>Near the start, one can reliably estimate effort and cost.</td>
<td>Near the beginning, it is not possible. As empirical data emerge, it becomes increasingly possible to plan and estimate.</td>
</tr>
<tr>
<td>It is possible to identify, define, schedule, and order all the detailed activities.</td>
<td>Near the beginning, it is not possible. Adaptive steps driven by build-feedback cycles are required.</td>
</tr>
<tr>
<td>Adaptation to unpredictable change is not the norm, and change-rates are relatively low.</td>
<td>Creative adaptation to unpredictable change is the norm. Change rates are high.</td>
</tr>
</tbody>
</table>

Source: IBM Global Services – Dr. Christoph Steindle
The Cone of Uncertainty

The diagram illustrates the concept of the Cone of Uncertainty in software development. As time progresses from Initial Concept to Approved Product Definition to Requirements Complete to User Interface Design Complete to Detailed Design Complete to Software Complete, the variability of estimates decreases, reaching a steady state at Software Complete.

Key milestones in the development process are indicated along the timeline, showing how the uncertainty in estimates reduces as the project advances through different phases.

This diagram is a visual representation of how uncertainty in project estimates decreases as more details are defined and planned.
Dr. Winston W. Royce
The Waterfall Model

"I believe in this concept, but the implementation described above is risky and invites failure."
Quick History of Agile

Software Crisis (1960’s) Software intensive systems delivered late, over budget and do not meet the quality requirements

Solution attempt #1: **Structured Methods** (in 1970’s)
Solution attempt #2: **Object Oriented Methodologies**

Chronic Software Crisis (1990’s) Software intensive systems still delivered late, over budget and do not meet the quality requirements

Solution attempt #3: **Software process improvement**
Solution attempt #4: **Agile development methodologies**
Pre-Agile: Lightweight Methodologies

Extreme Programming (Kent Beck, Ward Cunningham, Ron Jeffries)
Scrum (Ken Schwaber and Jeff Sutherland)
Lean Software Development (Mary and Tom Poppendieck)
Crystal Methods (Alistair Cockburn)
Feature Driven Development (Jeff DeLuca)
Dynamic Systems Development Method (DSDM Consortium)
We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- **Individuals and interactions over** processes and tools
- **Working software 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.**
Agile at a Glance

- **Agile Practices**: Defines in detail how this is implemented in practice.
- **Agile Principles**: Defines a set of ways to meet the values.
- **Agile Values**: Defines the set of most important beliefs of what is truly important.
- **The need to respond to constant change**: The fundamental reason for a “new” paradigm.
Shifting Paradigms

The Cone of Uncertainty

Anticipation

Adaptation
Paradigm Shift

TRADITIONAL

Fixed
Requirements

Plan Driven

Estimated
Resources
Time

The Plan creates cost/schedule estimates

Paradigm Shift

Releases themes & feature intent drive estimates

The Plan creates cost/schedule estimates
Feature Usage

Source: Jim Johnson of the Standish Group, Keynote Speech XP 2002
Lean Principles

- Eliminate Waste (Removing non-value-adding wastes)
  - Waste is partially done software, and extra features,
  - “Churns” [Batch Productions] – Requirements, Test & Fix – usually sign of large inventory of partially done work

- Build in Quality
  - Build quality into the code from the start, not test it in later
  - Control the conditions so not to allow defects in the first place

- Create Knowledge
  - Generating new knowledge about the product through disciplined experimentation
  - Systematic learning throughout the development lifecycle.
  - Avoid “analysis paralysis”
### The Seven Wastes of Manufacturing

<table>
<thead>
<tr>
<th>Waste</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overproduction</td>
</tr>
<tr>
<td>Inventory</td>
</tr>
<tr>
<td>Extra Processing Steps</td>
</tr>
<tr>
<td>Motion</td>
</tr>
<tr>
<td>Defects</td>
</tr>
<tr>
<td>Waiting</td>
</tr>
<tr>
<td>Transportation</td>
</tr>
</tbody>
</table>

### The Seven Wastes of Software Development

<table>
<thead>
<tr>
<th>Waste</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overproduction = Extra Features</td>
</tr>
<tr>
<td>Inventory = Requirements</td>
</tr>
<tr>
<td>Extra Processing Steps = Extra Steps</td>
</tr>
<tr>
<td>Motion = Finding Information</td>
</tr>
<tr>
<td>Defects = Defects Not Caught by Tests</td>
</tr>
<tr>
<td>Waiting = Waiting, Including Customers</td>
</tr>
<tr>
<td>Transportation = Handoffs</td>
</tr>
</tbody>
</table>

Principles of Lean Thinking, Mary Poppendieck
Lean Principles

- Defer Commitment
  - Last responsible moment, Just-in-time
- Deliver Fast
- Respect People
  - Greatest Asset
  - People not resources
- Optimize the whole
  - Throughput optimization
- GAMBA, in essence the boss on the floor looking at what is going on.
1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

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

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

4. Business people & developers must work together daily throughout project.

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

6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.

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

9. Continuous attention to technical excellence & good design enhances agility.

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

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

12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.
- "Principles are underlying truths that don’t change over time or space.

- Practices are the application of principles to a particular situation

- Practices can and should differ as you move from one environment to the next and they also change as a situation evolves”

- I don’t like the notion of “best practices”

1 From the book: Implementing Lean Software Development; from Concept to Cash
Agile Practices

- Self Organizing Teams
- Release Planning
- Iteration Planning
- Deliver Frequently
- Time boxing
- Client-Driven Iterations
- Retrospectives
- Team Room

- Pair Programming
- Automated Unit Testing
- Test Driven Development
- Continuous Integration
- Refactoring
Agile - in a word

Minimum Process
Maximum Value
5: Encompassing
Establishing a vibrant and all-encompassing environment to sustain agility

4: Adaptive
Responding to change through multiple levels of feedback

3: Integrated
Developing high quality, working software in an efficient and integrated manner

2: Evolutionary
Delivering software early and continuously

1: Collaborative
Enhancing communication and collaboration
<table>
<thead>
<tr>
<th>Step</th>
<th>Agile Practices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 5</td>
<td>Encompassing</td>
</tr>
<tr>
<td>Step 4</td>
<td>Adaptive</td>
</tr>
<tr>
<td>Step 3:</td>
<td>Integrated</td>
</tr>
<tr>
<td>Step 2:</td>
<td>Evolutionary</td>
</tr>
<tr>
<td>Step 1:</td>
<td>Collaborative</td>
</tr>
<tr>
<td>Step 5: Encompassing</td>
<td>Embrace Change to Deliver Customer Value</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>Step 4: Adaptive</td>
<td></td>
</tr>
<tr>
<td>Step 3: Integrated</td>
<td></td>
</tr>
<tr>
<td>Step 2: Evolutionary</td>
<td></td>
</tr>
<tr>
<td>Step 1: Collaborative</td>
<td></td>
</tr>
</tbody>
</table>

**Principles can guide the population of the practices**
<table>
<thead>
<tr>
<th>Step 5</th>
<th>Encompassing</th>
<th>Embrace Change to Deliver Customer Value</th>
<th>Plan and Deliver Software Frequently</th>
<th>Human Centric</th>
<th>Technical Excellence</th>
<th>Collaboration with Business People</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Low Process Ceremony</td>
<td>Agile Project Estimation</td>
<td>Ideal Agile Physical Setup</td>
<td>Test Driven Development</td>
<td>Paired Programming</td>
<td>Frequent Face-to-face interaction between developers &amp; Users (Collocated)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 4</th>
<th>Adaptive</th>
<th>Client Driven Iterations</th>
<th>Smaller and More Frequent Releases (4-8 Weeks)</th>
<th>Adaptive Planning</th>
<th>Daily Progress Tracking Meetings</th>
<th>Collaborative, Representative, Authorized, Committed and Knowledgeable (CRACK) Customer Immediately Accessible</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Customer Satisfaction Feedback</td>
<td></td>
<td></td>
<td>Agile Documentation (from Agile Modeling)</td>
<td>Customer contract revolves around commitment of collaboration, not features</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>User Stories</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 3: Integrated</th>
<th>Risk Driven Iterations</th>
<th>Maintain a list of all remaining features (Backlog)</th>
<th>Self Organizing Teams</th>
<th>Frequent face-to-face communication between the team</th>
<th>Continuous Integration</th>
<th>Continuous Improvement (i.e. Refactoring)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Have around 30% of Cockburn Level 2 and Level 3 people on team</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Automated Unit Tests</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 2: Evolutionary</th>
<th>Evolutionary Requirements</th>
<th>Continuous Delivery (Incremental-Iterative development)</th>
<th>Planning at different levels</th>
<th>Software Configuration Management</th>
<th>Tracking Iteration through Working Software</th>
<th>No Big Design Up Front (BDUF)</th>
<th>Customer Contract reflective of Evolutionary Development</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Step 1: Collaborative</th>
<th>Reflect and tune Process</th>
<th>Collaborative Planning</th>
<th>Collaborative teams Empowered and Motivated Teams</th>
<th>Coding Standards</th>
<th>Knowledge Sharing Tools (Wikis, Blogs)</th>
<th>Task Volunteering not Task Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Monday, October 12, 2009
<table>
<thead>
<tr>
<th>Step 1: Collaborative</th>
<th>Step 2: Evolutionary</th>
<th>Step 3: Integrated</th>
<th>Step 4: Adaptive</th>
<th>Step 5: Encompassing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retrospectives</td>
<td>Evolutionary</td>
<td>Streamlining usability with development efforts</td>
<td>Client Driven Iterations</td>
<td>Low Process Ceremony</td>
</tr>
<tr>
<td>User Stories with Acceptance Tests</td>
<td>Requirements</td>
<td>Using Agile PM Tools</td>
<td>Measuring Customer satisfaction</td>
<td>Encompassing</td>
</tr>
<tr>
<td>Whole-team collaborative planning</td>
<td>Multi-level Planning (Releases and Iterations)</td>
<td>Automated Deployment</td>
<td>Story Maps for Release Planning</td>
<td>Low Process Ceremony</td>
</tr>
<tr>
<td>Group Estimation using planning poker</td>
<td>Time-boxed iterations</td>
<td>Configuration Management</td>
<td>Pair Programming</td>
<td>Plan and Deliver Software Frequently</td>
</tr>
<tr>
<td>Maintaining a Backlog</td>
<td>Steady Releases every 6 weeks</td>
<td>Automated Build and Deployment scripts</td>
<td>Test Driven Development</td>
<td>Human Centric</td>
</tr>
<tr>
<td>Daily Standups</td>
<td>Burn-up Charts</td>
<td>Refactoring</td>
<td>Collaborative, Representative, Authorized, Committed and Knowledgeable (CRACK) Customer Immediately Accessible</td>
<td>Technical Excellence</td>
</tr>
<tr>
<td>Empowered Cross-functional teams</td>
<td>Celebrating Success</td>
<td>Continuous Integration</td>
<td>Agile Contracting</td>
<td>Collaboration with Business People</td>
</tr>
<tr>
<td>Emotional Chart</td>
<td>Self Organizing Teams</td>
<td>One-click-builds</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task Volunteering</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Team Room with Information Radiators</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project Chartering</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer Commitment to work with Developing Team</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| Monday, October 12, 2009 | | | | 37

- **Step 1: Collaborative**
  - Retrospectives
  - User Stories with Acceptance Tests
  - Whole-team collaborative planning
  - Group Estimation using planning poker
  - Maintaining a Backlog
  - Daily Standups

- **Step 2: Evolutionary**
  - Evolutionary Requirements
  - Multi-level Planning (Releases and Iterations)
  - Time-boxed iterations
  - Steady Releases every 6 weeks
  - Burn-up Charts

- **Step 3: Integrated**
  - Streamlining usability with development efforts
  - Using Agile PM Tools
  - Automated Deployment
  - Configuration Management

- **Step 4: Adaptive**
  - Client Driven Iterations
  - Measuring Customer satisfaction
  - Story Maps for Release Planning
  - Post-development Documentation

- **Step 5: Encompassing**
  - Low Process Ceremony
  - Pair Programming
  - Test Driven Development
  - Collaborative, Representative, Authorized, Committed and Knowledgeable (CRACK) Customer Immediately Accessible