Business Value of Agile Methods

Benefits of Testing Early & Often

Dr. David F. Rico, PMP, ACP, CSM

Twitter: @dr_david_f_rico
Website: http://www.davidfrico.com
LinkedIn: http://www.linkedin.com/in/davidfrico
Facebook: http://www.facebook.com/profile.php?id=1540017424
Dave’s Agile Articles: http://davidfrico.com/agile-message.doc
Author Background

- DoD contractor with 28+ years of IT experience
- Large gov’t projects in U.S., Far/Mid-East, & Europe

Published six books & numerous journal articles
Adjunct at George Wash, UMBC, UMUC, Argosy
Agile Program Management & Lean Development
Specializes in metrics, models, & cost engineering
Six Sigma, CMMI, ISO 9001, DoDAF, & DoD 5000
Cloud Computing, SOA, Web Services, FOSS, etc.
Today’s Whirlwind Environment

- Overruns
- Attrition
- Escalation
- Runaways
- Cancellation

Global Competition

Demanding Customers

- Inefficiency
- High O&M
- Lower DoQ
- Vulnerable
- N-M Breach

Reduced IT Budgets

Obsolete Technology & Skills

81 Month Cycle Times

Redundant Data Centers

Overburdening Legacy Systems

system Complexity

Organization Downsizing

Reduced IT Security

Poor IT Security

Vague Requirements

Work Life Imbalance

Technology Change


Traditional Projects

- Big projects result in poor quality and scope changes
- Productivity declines with long queues/wait times
- Large projects are unsuccessful or canceled

Global Project Failures

- Challenged and failed projects hover at 67%
- Big projects fail more often, which is 5% to 10%
- Of $1.7T spent on IT projects, over $858B were lost

- Requirements defects are #1 reason projects fail
- Traditional projects specify too many requirements
- More than 65% of requirements are never used at all

What is Agility?

- **Agility** (ə-ˈji-lə-tē) Property consisting of quickness, lightness, and ease of movement; **To be very nimble**
  - The ability to create and **respond to change** in order to profit in a turbulent global business environment
  - The ability to **quickly reprioritize** use of resources when requirements, technology, and knowledge shift
  - A very **fast response** to sudden market changes and emerging threats by intensive **customer interaction**
  - Use of **evolutionary, incremental, and iterative delivery** to converge on an optimal customer solution
  - Maximizing **BUSINESS VALUE** with right sized, just-enough, and just-in-time processes and documentation

What are Agile Methods?

- **People-centric** way to create innovative solutions
- **Product-centric** alternative to documents/process
- **Market-centric** model to maximize business value

<table>
<thead>
<tr>
<th>Customer Collaboration</th>
<th>Contracts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequent comm.</td>
<td>Contract compliance</td>
</tr>
<tr>
<td>Close proximity</td>
<td>Contract deliverables</td>
</tr>
<tr>
<td>Regular meetings</td>
<td>Contract change orders</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Individuals &amp; Interactions</th>
<th>Processes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leadership</td>
<td>Lifecycle compliance</td>
</tr>
<tr>
<td>Boundaries</td>
<td>Process Maturity Level</td>
</tr>
<tr>
<td>Empowerment</td>
<td>Regulatory compliance</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Working Software</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clear objectives</td>
<td>Document deliveries</td>
</tr>
<tr>
<td>Small/feasible scope</td>
<td>Document comments</td>
</tr>
<tr>
<td>Acceptance criteria</td>
<td>Document compliance</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Responding to Change</th>
<th>Project Plans</th>
</tr>
</thead>
<tbody>
<tr>
<td>Org. flexibility</td>
<td>Cost Compliance</td>
</tr>
<tr>
<td>Mgt. flexibility</td>
<td>Scope Compliance</td>
</tr>
<tr>
<td>Process flexibility</td>
<td>Schedule Compliance</td>
</tr>
</tbody>
</table>

---


How Agile Works

- Agile requirements implemented in slices vs. layers
- User needs with higher business value are done first
- Reduces cost & risk while increasing business success

**Agile**

- Faster
- Early ROI
- Lower Costs
- Fewer Defects
- Manageable Risk
- Better Performance
- Smaller Attack Surface

**Traditional**

- Late
- No Value
- Cost Overruns
- Very Poor Quality
- Uncontrollable Risk
- Slowest Performance
- More Security Incidents

**Seven Wastes**

1. Rework
2. Motion
3. Waiting
4. Inventory
5. Transportation
6. Overprocessing
7. Overproduction

**MINIMIZES**

- JIT, Just-enough architecture
- Early, in-process system V&V
- Fast continuous improvement
- Scalable to systems of systems
- Maximizes successful outcomes

**MAXIMIZES**

- Myth of perfect architecture
- Late big-bang integration tests
- Year long improvement cycles
- Breaks down on large projects
- Undermines business success

What is Agile Testing?

- Traditional testing is a late, manual process
- Agile testing is an early and automated process
- The goal of agile testing is to deliver early and often

<table>
<thead>
<tr>
<th>Traditional Testing</th>
<th>Agile Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Combining source files</td>
<td>• Code is frequently checked in</td>
</tr>
<tr>
<td>• Combining software and environment</td>
<td>• Code is automatically retrieved</td>
</tr>
<tr>
<td>• Combining software and data</td>
<td>• Compilation is done automatically</td>
</tr>
<tr>
<td>• Combining software and tests</td>
<td>• Tests are done automatically</td>
</tr>
<tr>
<td>• Combining developers</td>
<td>• Code reports are generated</td>
</tr>
<tr>
<td></td>
<td>• Developers get instant feedback</td>
</tr>
<tr>
<td></td>
<td>• Code is automatically deployed or packaged for delivery</td>
</tr>
</tbody>
</table>

Basic—Test Driven Development

- Term coined by Kent Beck in 2003
- Consists of writing all tests before design
- Ensures all components are verified and validated

Advanced—Continuous Integration

- User needs designed & developed one-at-a-time
- Changes automatically detected, built, and tested
- System fully tested and deployed as changes occur

Thousands of Tests Continuously Executed

Early, Automated, Fast, Efficient, & Repeatable

Lean, Waste Free, Low WIP, No Deadlocked Test Queues

Builds
Database
Analysis
Testing
Reporting
Documentation
Deployment

Build
Integration
Server

Commits Changes
Commits Changes
Commits Changes

Build Status
Watches
Build Integration Server
Build Scripts

Version Control Server

Commits

Constant Readiness
State & CM Control

No More Late Big Bang Integration

Rapidly & Successfully Dev. Complex Systems

Builds
Database
Analysis
Testing
Reporting
Documentation
Deployment

Agile Testing Done Early & Often

- Eliminates big-bang integration in the 11th hour
- Creates a repeatable and reliable testing process
- Evaluates system-wide changes throughout project

Agile Testing Practices

- Agile testing consists of seven broad practices
- Includes automated builds, testing, inspections, etc.
- Also includes reporting, documentation, deployment, etc.

<table>
<thead>
<tr>
<th>Practice</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Building</td>
<td>Frequently assembling products and services to ensure delivery readiness</td>
</tr>
<tr>
<td>Database</td>
<td>Frequently generating/analyzing database schemas, queries, and forms</td>
</tr>
<tr>
<td>Inspections</td>
<td>Frequently performing automated static analysis of product/service quality</td>
</tr>
<tr>
<td>Testing</td>
<td>Frequently performing automated dynamic product and service evaluation</td>
</tr>
<tr>
<td>Feedback</td>
<td>Frequently generating automated status reports/messages for all stakeholders</td>
</tr>
<tr>
<td>Documentation</td>
<td>Frequently performing automated technical/customer document generation</td>
</tr>
<tr>
<td>Deployment</td>
<td>Frequently performing automated delivery of products/services to end users</td>
</tr>
</tbody>
</table>

Agile Testing Workflow

- Late big bang integration increases WIP backlog
- Agile testing early and often reduces WIP backlog
- Improves workflow and reduces WIP and lead times

Agile Testing Costs & Benefits

- Most agile testing tools are “free” open source
- A build server is no more than a commodity PC
- 10x more efficient/effective than traditional testing

- Free, Open Source Software
- $500 for a dedicated build machine
- 4 hours configuration time for new user
- 2 hours for an experienced user
- 20 minutes to set up a new project
- It becomes more valuable with use
- Less than half the cost of traditional testing

- 36% reduction in defect rate when integration/regression testing at each code check-in
- 90% reduction in bugs reaching QA
  - Major municipal gas utility
- 95% cut in cost of bugs
  - Large retail web site
- 90% cut in defect remediation cost
  - Global supplier of healthcare equipment
- Faster time-to-market
  - More features and higher quality
- Agility in the marketplace
  - Added new functionality 2 weeks before ship
- Confidence in the process
  - “Oozing Confidence”

Agile Testing Economics

- Traditional testing finds a defect in about 10 hours
- Manual code inspections find a defect in 1 hour
- Agile testing finds a defect every 6 minutes

<table>
<thead>
<tr>
<th>Activity</th>
<th>CoQ</th>
<th>Economics of Continuous Integration</th>
<th>Hours</th>
<th>ROI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continuous Integration</td>
<td>0.1</td>
<td>100 Defects x 70% Efficiency x 0.1 Hours</td>
<td>7</td>
<td>n/a</td>
</tr>
<tr>
<td>Code Inspections</td>
<td>1</td>
<td>30 Defects x 70% Efficiency x 1 Hours</td>
<td>21</td>
<td>300%</td>
</tr>
<tr>
<td>Testing</td>
<td>10</td>
<td>9 Defects x 70% Efficiency x 10 Hours</td>
<td>63</td>
<td>900%</td>
</tr>
<tr>
<td>Debugging</td>
<td>100</td>
<td>2.7 Defects x 70% Efficiency x 100 Hours</td>
<td>189</td>
<td>2,700%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tests</th>
<th>1 Hour</th>
<th>1 Day</th>
<th>1 Week</th>
<th>1 Month</th>
<th>3 Months</th>
<th>6 Months</th>
<th>1 Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>One</td>
<td>6</td>
<td>48</td>
<td>240</td>
<td>1,040</td>
<td>3,120</td>
<td>6,240</td>
<td>12,480</td>
</tr>
<tr>
<td>Three</td>
<td>18</td>
<td>144</td>
<td>720</td>
<td>3,120</td>
<td>9,360</td>
<td>18,720</td>
<td>37,440</td>
</tr>
<tr>
<td>Six</td>
<td>36</td>
<td>288</td>
<td>1,440</td>
<td>6,240</td>
<td>18,720</td>
<td>37,440</td>
<td>74,880</td>
</tr>
<tr>
<td>Twelve</td>
<td>72</td>
<td>576</td>
<td>2,880</td>
<td>12,480</td>
<td>37,440</td>
<td>74,880</td>
<td>149,760</td>
</tr>
</tbody>
</table>

Agile Cost of Quality (CoQ)

- Agile testing is **10x better than code inspections**
- Agile testing is **100x better than traditional testing**
- Agile testing is **done earlier “and” 1,000x more often**

Agile Testing Statistics

- Fewer builds leave in higher bug counts
- A high number of builds eliminates the defects
- Goal is to have as many, early builds as possible

How Google Tests Software

- Google early adopter of agile methods and Scrum
- Google also uses agile testing at enterprise scale
- 15,000 developers run 75+ million tests per day

- 15,000+ developers in 40+ offices
- 4,000+ projects under active development
- 5,500+ submissions per day on average
- Single monolithic code tree with mixed language code
- Development on one branch - submissions at head
- All builds from source
- 20+ sustained code changes per minute with 60+ peaks
- 50% of code changes monthly
- 75+ million test cases run per day

Micco, J. (2013). Continuous integration at google scale. Eclipse Con, Boston, MA.
Scaling Agile Testing

- Agile testing slows down with very large systems
- Slow testing slows integration and increases bugs
- Agile testing can speed back up with proper attention

<table>
<thead>
<tr>
<th>Wide Impact Tuning</th>
<th>Focused Impact Tuning</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Fast builds – less changes – more green</td>
<td>• More memory and CPUs</td>
</tr>
<tr>
<td>• Parallelization of test runs</td>
<td>• Parallelize builds</td>
</tr>
<tr>
<td>• ClearCase to subversion</td>
<td>• Replace 3rd party test libraries</td>
</tr>
<tr>
<td>• Pre-installing as much as possible</td>
<td>• Reduce/remove timeouts in tests</td>
</tr>
<tr>
<td>• Removal of randomness</td>
<td>• Select different tests</td>
</tr>
<tr>
<td>• Compilation in memory</td>
<td>• Refactor code &amp; components</td>
</tr>
<tr>
<td>• Installation starting parallel with system build</td>
<td>• Tune the network &amp; software</td>
</tr>
<tr>
<td></td>
<td>• Tune the database</td>
</tr>
</tbody>
</table>

General Agile Metrics

- Agile methods are based on traditional measures
- Velocity, burnup, and burndown are basic measures
- Top-notch shops use business value, Agile EVM, etc.

**Basic Metrics**
- Velocity
- Productivity
- Cycle time
- Effort
- Cost
- Schedule
- System quality
- Customer Satisfaction

**Burnup & Burndown**
- Epic burnup & burndown
- Feature burnup & burndown
- Release burnup & burndown
- Iteration burnup & burndown
- Weekly burnup & burndown
- Daily burnup & burndown

**Business Value**
- Business value per epic
- Business value per feature
- Business value per release
- Business value per iteration
- Business value per week
- Business value per day
- Business value per story

**Agile Earned Value**
- Story points per release
- Planned sprints per release
- Planned budget per release
- Current to planned sprint
- Current to planned points
- Planned release points
- Release points completed

**Advanced Measures**
- Customer collaboration
- Customer trust & loyalty
- Teamwork cohesion & quality
- Purpose, autonomy, & mastery
- Workflow capacity & throughput
- Workflow efficiency & reliability
- Organizational culture & agility
- Information systems flexibility

Agile Testing Metrics

- Agile testing also based on traditional metrics
- Agile testing measures include basic volumetrics
- Key metrics are RTF, code coverage, complexity, etc.

<table>
<thead>
<tr>
<th>Volumetrics</th>
<th>Running Tested Code</th>
<th>Code Coverage</th>
<th>Code Complexity</th>
<th>Defect Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of user stories</td>
<td>Running tested epics</td>
<td>Epic coverage</td>
<td>Epic complexity</td>
<td>Defects per epic</td>
</tr>
<tr>
<td>Number of story points</td>
<td>Running tested features</td>
<td>Feature coverage</td>
<td>Feature complexity</td>
<td>Defects per feature</td>
</tr>
<tr>
<td>Number of lines of code</td>
<td>Running tested releases</td>
<td>Release coverage</td>
<td>Release complexity</td>
<td>Defects per release</td>
</tr>
<tr>
<td>Number of function points</td>
<td>Running tested iterations</td>
<td>Iteration coverage</td>
<td>Iteration complexity</td>
<td>Defects per iteration</td>
</tr>
<tr>
<td>Number of check-ins</td>
<td>Running tested builds</td>
<td>Build coverage</td>
<td>Build complexity</td>
<td>Defects per build</td>
</tr>
<tr>
<td>Number of unit tests</td>
<td>Running tested user stories</td>
<td>Statement coverage</td>
<td>Cyclomatic complexity</td>
<td>Defects per user story</td>
</tr>
<tr>
<td>Number of builds</td>
<td></td>
<td>Function coverage</td>
<td>Actual complexity</td>
<td>Defects per story point</td>
</tr>
<tr>
<td>Number of deployments</td>
<td></td>
<td>Other coverage types</td>
<td>Other complexity types</td>
<td>Defects per other types</td>
</tr>
</tbody>
</table>

# Agile Cost & Benefit Analysis

- Costs based on avg. productivity and quality
- Productivity ranged from 4.7 to 5.9 LOC an hour
- Costs were $588,202 and benefits were $3,930,631

<table>
<thead>
<tr>
<th>Metric</th>
<th>Formula</th>
<th>Trad. Testing</th>
<th>Agile Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Costs</td>
<td>$(10,000 \div 5.4436 + 3.945 \times 10 \times 100) \times 100$</td>
<td>$588,202$</td>
<td>$233,152$</td>
</tr>
<tr>
<td>Benefits</td>
<td>$(10,000 \times 10.51 - 6,666.67 \times 9) \times 100 - 588,202$</td>
<td>$3,930,631$</td>
<td>$4,275,681$</td>
</tr>
<tr>
<td>B/CR</td>
<td>$\frac{3,930,631}{588,202}$</td>
<td>7:1</td>
<td>18:1</td>
</tr>
<tr>
<td>ROI</td>
<td>$\frac{(3,930,631 - 588,202)}{588,202} \times 100%$</td>
<td>567%</td>
<td>1,734%</td>
</tr>
<tr>
<td>NPV</td>
<td>$(\sum_{i=1}^{5} \frac{3,930,631}{5} \div 1.05^5) - 588,202$</td>
<td>$2,806,654$</td>
<td>$3,469,140$</td>
</tr>
<tr>
<td>BEP</td>
<td>$\frac{588,202}{(4,509,997 \div 588,202 - 1)}$</td>
<td>$88,220$</td>
<td>$12,710$</td>
</tr>
<tr>
<td>ROA</td>
<td>$\text{NORMSDIST}(2.24) \times \frac{3,930,631 - \text{NORMSDIST}(0.85) \times 588,202 \times \exp(-5% \times 5)}{}$</td>
<td>$3,504,292$</td>
<td>$4,098,159$</td>
</tr>
</tbody>
</table>

\[
d_1 = \left[ \ln \left( \frac{\text{Benefits}}{\text{Costs}} \right) + \left( \text{Rate} + 0.5 \times \text{Risk}^2 \right) \times \text{Years} \right] \div \text{Risk} \times \sqrt{\text{Years}}, \quad d_2 = d_1 - \text{Risk} \times \sqrt{\text{Years}}
\]

Studies of Agile Methods

- Dozens of surveys of agile methods since 2003
- 100s of Agile and CMMI case studies documented
- Agile productivity, quality, and cost better than CMMI


Benefits of Agile Methods

- Analysis of 23 agile vs. 7,500 traditional projects
- Agile projects are 54% better than traditional ones
- Agile has **lower costs** (61%) and **fewer defects** (93%)

Agile vs. Traditional Success

- Traditional projects succeed at 50% industry avg.
- Traditional projects are challenged 20% more often
- Agile projects succeed 3x more and fail 3x less often

Benefits of Organizational Agility

- Study of 15 agile vs. non-agile Fortune 500 firms
- Based on models to measure organizational agility
- Agile firms **outperform** non agile firms by up to 36%

Agile Enterprise Delivery Model

- Begins with a high-level product vision/architecture
- Continues with needs development/release planning
- Includes agile delivery teams to realize business value

Agile Adoption

- VersionOne found 80% using agile methods today
- Most are using Scrum with several key XP practices
- **Lean-Kanban** is a rising practice with a **24% adoption**

---

Agile Proliferation

- Number of CSMs have doubled to 200,000 in 2 years
- 558,918 agile jobs for only 121,876 qualified people
- 4.59 jobs available for every agile candidate (5:1)


## Agile Industry Case Studies

- 80% of worldwide IT projects use agile methods
- Includes regulated industries, i.e., DoD, FDA, etc.
- Agile now used for **safety critical systems**, FBI, etc.

<table>
<thead>
<tr>
<th>Industry</th>
<th>Org</th>
<th>Project</th>
<th>Purpose</th>
<th>Size</th>
<th>Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electronic Commerce</td>
<td>Google</td>
<td>Adwords</td>
<td>Advertising</td>
<td>20 teams, 140 people, 5 countries</td>
<td>1,838 User Stories, 6,250 Function Points, 500,000 Lines of Code</td>
</tr>
<tr>
<td>Shrink Wrapped</td>
<td>Primavera</td>
<td>Primavera</td>
<td>Project Management</td>
<td>15 teams, 90 people, Collocated</td>
<td>26,809 User Stories, 91,146 Function Points, 7,291,666 Lines of Code</td>
</tr>
<tr>
<td>Law Enforcement</td>
<td>FBI</td>
<td>Sentinel</td>
<td>Case File Workflow</td>
<td>10 teams, 50 people, Collocated</td>
<td>3,947 User Stories, 13,419 Function Points, 1,073,529 Lines of Code</td>
</tr>
<tr>
<td>U.S. DoD</td>
<td>Stratcom</td>
<td>SKIweb</td>
<td>Knowledge Management</td>
<td>3 teams, 12 people, Collocated</td>
<td>390 User Stories, 1,324 Function Points, 105,958 Lines of Code</td>
</tr>
</tbody>
</table>

Perceptions of Agile Methods

- Structure, reward, decision, staffing, leadership, etc.
- Top-down, individualism, regulation, compliance, etc.
- Focus on reforming acquisition & procurement system

<table>
<thead>
<tr>
<th>Type/Kind</th>
<th>Common DoD Agile Perceptions</th>
<th>Reality with Respect to Agile Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discipline</td>
<td>Undisciplined Cowboy Coding</td>
<td>• Rigorous process, plans, requirements, QA, CM, testing, documents etc.</td>
</tr>
<tr>
<td>Scalability</td>
<td>Only Applies Small Projects</td>
<td>• Used by 100, 500, 1,000, 10,000+ person person projects &amp; organizations</td>
</tr>
<tr>
<td>Domain</td>
<td>Only for Protoperational Systems</td>
<td>• Used in DoD, medical devices, avionics, autos, electronics, etc.</td>
</tr>
<tr>
<td>Management</td>
<td>Flexible Scope/Can't Use EVM</td>
<td>• Lightweight EVM model is used with its release planning methodology</td>
</tr>
<tr>
<td>Requirements</td>
<td>Doesn't Use Requirements</td>
<td>• Always begins with valuable, well-defined, &amp; prioritized requirements</td>
</tr>
<tr>
<td>Architecture</td>
<td>Spaghetti Code from Iterations</td>
<td>• Begins with lean architecture or create waste-free emergent design</td>
</tr>
<tr>
<td>Quality</td>
<td>No Documents/Unmaintainable</td>
<td>• Electronic plans, requirements, designs, tests, manuals, documents, etc.</td>
</tr>
<tr>
<td>Inspections</td>
<td>High CoQ from No Inspections</td>
<td>• One or two orders of magnitude more inspections &amp; tests performed</td>
</tr>
<tr>
<td>Security</td>
<td>Vulnerabilities from Hacking</td>
<td>• Security practices result in smaller attack surface &amp; fewer vulnerabilities</td>
</tr>
</tbody>
</table>

“Agility” has many **dimensions** other than IT
It ranges from **leadership** to **technological agility**
The focus of this brief is **program management agility**
Agile Recap

- Agile methods **DON’T** mean deliver it now & fix it later
- Lightweight, yet disciplined approach to development
- Reduced **cost, risk, & waste** while **improving quality**

<table>
<thead>
<tr>
<th>What</th>
<th>How</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flexibility</td>
<td>Use lightweight, yet disciplined processes and artifacts</td>
<td>Low work-in-process</td>
</tr>
<tr>
<td>Customer</td>
<td>Involve customers early and often throughout development</td>
<td>Early feedback</td>
</tr>
<tr>
<td>Prioritize</td>
<td>Identify highest-priority, value-adding business needs</td>
<td>Focus resources</td>
</tr>
<tr>
<td>Descope</td>
<td>Descope complex programs by an order of magnitude</td>
<td>Simplify problem</td>
</tr>
<tr>
<td>Decompose</td>
<td>Divide the remaining scope into smaller batches</td>
<td>Manageable pieces</td>
</tr>
<tr>
<td>Iterate</td>
<td>Implement pieces one at a time over long periods of time</td>
<td>Diffuse risk</td>
</tr>
<tr>
<td>Leanness</td>
<td>Architect and design the system one iteration at a time</td>
<td>JIT waste-free design</td>
</tr>
<tr>
<td>Swarm</td>
<td>Implement each component in small cross-functional teams</td>
<td>Knowledge transfer</td>
</tr>
<tr>
<td>Collaborate</td>
<td>Use frequent informal communications as often as possible</td>
<td>Efficient data transfer</td>
</tr>
<tr>
<td>Test Early</td>
<td>Incrementally test each component as it is developed</td>
<td>Early verification</td>
</tr>
<tr>
<td>Test Often</td>
<td>Perform system-level regression testing every few minutes</td>
<td>Early validation</td>
</tr>
<tr>
<td>Adapt</td>
<td>Frequently identify optimal process and product solutions</td>
<td>Improve performance</td>
</tr>
</tbody>
</table>

Conclusion

- Agility is the evolution of management thought
- Confluence of traditional and non-traditional ideas
- Improve performance by over an order of magnitude

Agile methods are ...

- Systems development approaches
- New product development approaches
- Expertly designed to be fast and efficient
- Intentionally lean and free of waste (muda)
- Systematic highly-disciplined approaches
- Capable of producing high quality systems
- Right-sized, just-enough, and just-in-time tools
- Scalable to large, complex mission-critical systems
- Designed to maximize business value for customers

“The world of traditional methods belongs to yesterday”

“Don’t waste your time using traditional methods on 21st century projects”

Books on ROI of SW Methods

- Guides to software methods for business leaders
- Communicates business value of software methods
- Rosetta stones to unlocking ROI of software methods

- [THE BUSINESS VALUE OF AGILE SOFTWARE METHODS](http://davidfrico.com/agile-book.htm)
- [ROI OF SOFTWARE PROCESS IMPROVEMENT](http://davidfrico.com/roi-book.htm)