Agile Quality Management

Panagiotis Sfetsos, PhD
Assistant Professor,
Department of Informatics,
Alexander Technological Educational Institution
E-mail: sfetsos@it.teithe.gr
Web Page: http://aetos.it.teithe.gr/~sfetsos/
Objectives:

- Agile Manifesto Values - Agile Development
- What is quality?
- Does Quality Management have a place in agile methods?
- Deming’s Quality Improvement Cycles
- Best Agile practices
- Empirical findings on Quality in Agile Practices
- Human issues in Agile QM
- Key success factors
The emergence of the **Agile Methods** (1/2)

The Standish group, 1994:

<table>
<thead>
<tr>
<th>Project Challenged Factors</th>
<th>% of Responses</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Lack of User Input</td>
<td>12.8%</td>
</tr>
<tr>
<td>2. Incomplete Requirements &amp; Specifications</td>
<td>12.3%</td>
</tr>
<tr>
<td>3. Changing Requirements &amp; Specifications</td>
<td>11.8%</td>
</tr>
<tr>
<td>4. Lack of Executive Support</td>
<td>7.5%</td>
</tr>
<tr>
<td>5. Technology Incompetence</td>
<td>7.0%</td>
</tr>
<tr>
<td>6. Lack of Resources</td>
<td>6.4%</td>
</tr>
<tr>
<td>7. Unrealistic Expectations</td>
<td>5.9%</td>
</tr>
<tr>
<td>8. Unclear Objectives</td>
<td>5.3%</td>
</tr>
<tr>
<td>9. Unrealistic Time Frames</td>
<td>4.3%</td>
</tr>
<tr>
<td>10. New Technology</td>
<td>3.7%</td>
</tr>
<tr>
<td>Other</td>
<td>23.0%</td>
</tr>
</tbody>
</table>
The emergence of the Agile Methods (2/2)

The Standish group, 1994:

<table>
<thead>
<tr>
<th>Project Success Factors</th>
<th>% of Responses</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. User Involvement</td>
<td>15.9%</td>
</tr>
<tr>
<td>2. Executive Management Support</td>
<td>13.9%</td>
</tr>
<tr>
<td>3. Clear Statement of Requirements</td>
<td>13.0%</td>
</tr>
<tr>
<td>4. Proper Planning</td>
<td>9.6%</td>
</tr>
<tr>
<td>5. Realistic Expectations</td>
<td>8.2%</td>
</tr>
<tr>
<td>6. Smaller Project Milestones</td>
<td>7.7%</td>
</tr>
<tr>
<td>7. Competent Staff</td>
<td>7.2%</td>
</tr>
<tr>
<td>8. Ownership</td>
<td>5.3%</td>
</tr>
<tr>
<td>9. Clear Vision &amp; Objectives</td>
<td>2.9%</td>
</tr>
<tr>
<td>10. Hard-Working, Focused Staff</td>
<td>2.4%</td>
</tr>
<tr>
<td>Other</td>
<td>13.9%</td>
</tr>
</tbody>
</table>

Total:

- Success rate = 16.2%
- Challenged projects = 52.7%
- Impaired (cancelled) = 31.1%
Agile Manifesto values in practice

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan
What does Agile development mean in practice?

- Iterative and incremental development
- “Fits just right” process
- Continuous testing and validation
- Simple planning and design
- Rapid response to change
- Close collaboration
- Empowered and self-organized teams
- Frequent delivery of working software
  
  **Measure of progress**

---

**Graphs**

1. Risk vs. Time: Agile project vs. Waterfall project
2. Cost of change vs. Time
Agile means re-evaluate your best practices in a rapid fashion...

- Test Driven Development (TDD) or Test First Development + Refactoring
- Continuous Integration
- Customer participation
- Pair Programming
- Shared Code Ownership
# Agile Software Development (3/3)

<table>
<thead>
<tr>
<th></th>
<th><strong>Agile Methods</strong></th>
<th><strong>“Traditional” Methods</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Project size</strong></td>
<td>Small, Medium</td>
<td>Large</td>
</tr>
<tr>
<td><strong>Approach</strong></td>
<td>Adaptive</td>
<td>Predictive</td>
</tr>
<tr>
<td><strong>Success Measurement</strong></td>
<td>Business Value</td>
<td>Conformation to plan</td>
</tr>
<tr>
<td><strong>Management Style</strong></td>
<td>Decentralized</td>
<td>Autocratic</td>
</tr>
<tr>
<td><strong>Perspective to Change</strong></td>
<td>Change Adaptability</td>
<td>Change Sustainability</td>
</tr>
<tr>
<td><strong>Culture</strong></td>
<td>Leadership-Collaboration</td>
<td>Command-Control</td>
</tr>
<tr>
<td><strong>Documentation</strong></td>
<td>Low</td>
<td>Heavy</td>
</tr>
<tr>
<td><strong>Emphasis</strong></td>
<td>People-Oriented</td>
<td>Process-Oriented</td>
</tr>
<tr>
<td><strong>Cycles</strong></td>
<td>Numerous</td>
<td>Limited</td>
</tr>
<tr>
<td><strong>Domain</strong></td>
<td>Unpredictable/Exploratory</td>
<td>Predictable</td>
</tr>
<tr>
<td></td>
<td>Predictable</td>
<td></td>
</tr>
<tr>
<td><strong>Upfront Planning</strong></td>
<td>Minimal</td>
<td>Comprehensive</td>
</tr>
<tr>
<td><strong>Team Size</strong></td>
<td>Small/Creative</td>
<td>Large</td>
</tr>
<tr>
<td><strong>Quality</strong></td>
<td>Built-in from the beginning</td>
<td>Split in the devel. phases</td>
</tr>
<tr>
<td><strong>Quality Assurance</strong></td>
<td>Upstream in the SDP</td>
<td>End Product</td>
</tr>
<tr>
<td><strong>Return on Investment</strong></td>
<td>Early in Project</td>
<td>End of Project</td>
</tr>
</tbody>
</table>
What is Quality?

- The totality of characteristics of a product or service that bear on its ability to satisfy stated and implied needs (ISO 8402)

- The degree to which a system, component, or process meets specified requirements and customer/user needs or expectations (IEEE)

- Both definitions are focused on satisfying the customer's need for the software product
Why is Quality Important?

- **Customers expect it**
  - Focus on the Customer (expectations, requirements)

- **We need to get better to compete**
  - Teamwork (Working together to reach goals and solve problems)
  - Continuous Improvement

- **Need to be able to measure our progress**
  - Measuring Success
Quality Management Goals (1/3)

The 8 principles of **ISO 9001:2000**...

- Customer satisfaction
- Leadership
- Involvement of People
- Process approach
- Systemic approach
- Continuous improvement
- Decisions based on facts
- Relationship with suppliers mutually beneficiary
The ISO/IEC 12207

Agile Methods and the Development Process Area...

Planning game or sprint planning
Requirements definition activities

Test Driven Development, Continuous Integration, Customer involvement, Pair Programming

Testing and Implementation activities
Quality Management Goals (3/3)

The ISO/IEC 9126

Software Product in the Agile Methods...

<table>
<thead>
<tr>
<th>Quality Model</th>
<th>External Metrics</th>
<th>Internal Metrics</th>
<th>Quality in Use Metrics</th>
</tr>
</thead>
</table>

Does Quality Management, as in “traditional”, have a place in agile methods?

Validation
- Is the software matching the requirements?
- Can we trace from requirements to implementation?

Verification
- Are the classes matching the specification?
- Are we using good design and good practices?

Are we building the right software?
Are we building the software right?
Deming’s Quality Improvement Cycles (1/2)

**PLAN**
- Establish **the objectives and processes** necessary to deliver results in accordance with the expected output.

**DO**
- Implement **the new processes**. Often on a small scale if possible.

**CHECK / STUDY**
- Measure **the new processes** and compare the results against the expected results to ascertain any differences.

**ACT**
- Analyze the differences to determine their cause. Determine where to apply changes that will include **improvement**.
The Agile version of Deming's Quality Improvement Cycles (2/2)

Iterative and Incremental Quality Improvement Cycles

**Plan**
- Release and Iteration / Sprint Planning

**Do**
- A complete Iteration / Sprint

**Check / Study**
- Concurrent testing
- Continuous Integration
- Daily stands-up
- Iteration/Sprint reviews
- Retrospectives

**Act**
- Implement lessons learned from Retrospectives
- Team Adaptation
# Agile Quality Assurance

## QA in the Agile Software Development Life Cycle

<table>
<thead>
<tr>
<th>Requirement &amp; Design QA</th>
<th>Techniques / Practices</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- Simple planning and design</td>
</tr>
<tr>
<td></td>
<td>- User Stories</td>
</tr>
<tr>
<td></td>
<td>- Acceptance Tests and UAT</td>
</tr>
<tr>
<td></td>
<td>- Release/Iteration planning</td>
</tr>
<tr>
<td></td>
<td>- Architectural spike</td>
</tr>
<tr>
<td></td>
<td>- System Metaphor</td>
</tr>
<tr>
<td></td>
<td>- On-site customer feedback</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Implementation and Code QA</th>
<th>Techniques / Practices</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- Unit tests</td>
</tr>
<tr>
<td></td>
<td>- Refactoring</td>
</tr>
<tr>
<td></td>
<td>- Acceptance Tests and UAT</td>
</tr>
<tr>
<td></td>
<td>- CRC-cards</td>
</tr>
<tr>
<td></td>
<td>- Pair Programming</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Integration and System QA</th>
<th>Techniques / Practices</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- Unit tests coverage</td>
</tr>
<tr>
<td></td>
<td>- Acceptance Tests and UAT</td>
</tr>
<tr>
<td></td>
<td>- Continuous Integration</td>
</tr>
<tr>
<td></td>
<td>- Common Codebase</td>
</tr>
<tr>
<td></td>
<td>- Customer Feedback</td>
</tr>
</tbody>
</table>
On-time system delivery, to spec, and within budget and schedule

- A working product increment

QA team and PM responsible for “quality”

- “Whole team” approach - “Built-in”

QA team engaged in the testing phase (end of product)

Development team (testers included), upstream in the SDP
## “Traditional” vs. Agile (2/2)

### Differences in Quality

<table>
<thead>
<tr>
<th>Traditional</th>
<th>Agile</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Split quality (in each development phase)</strong></td>
<td><strong>Built-in quality (from the beginning)</strong></td>
</tr>
<tr>
<td><strong>QA team responsible for “quality”</strong></td>
<td><strong>“whole team” approach</strong></td>
</tr>
<tr>
<td>- Formal methods</td>
<td>- Self-directed team responsible for quality, testers included</td>
</tr>
<tr>
<td>- Quality Police</td>
<td>- Pair/team collaboration,</td>
</tr>
<tr>
<td>- No specs, can’t test</td>
<td>- Day-to-day activities</td>
</tr>
<tr>
<td><strong>Quality Assurance</strong> - end product</td>
<td><strong>Quality Assurance</strong> - upstream in the SDP (from the beginning)</td>
</tr>
</tbody>
</table>
How Have Agile Approaches Affected the Quality of Systems Deployed?

- 48% Much Higher
- 14% No Change
- 29% Somewhat Higher
- 6% Somewhat Lower
- 3% Much Lower

Source: Ambler ‘Agile Adoption Rate Survey’ of over 4200 Dr. Dobb’s subscribers, March 2008
Third-party research suggests even wider adoption:

Have you adopted any Agile techniques?

- "No" 31%
- "Yes" 69%

In a recent study conducted by the Agile Journal, it was determined that 88% of companies, many with over 10,000 employees, are using or evaluating Agile practices on their projects.

Which of the agile practices should be implemented to improve software quality?

Source: Ambler 'Agile Adoption Rate Survey' of over 4200 Dr. Dobb's subscribers, March 2008
What is TDD?

TDD isn’t about testing…
…it’s about programming! (specification, design, testing and coding)

Writing simpler, clearer and more robust and more maintainable code!

“Before you write code, think about what it will do. Write a test that will use the methods you haven’t even written yet.”

No code should go into production unless it has associated tests…
…Catch bugs before they are shipped to your customer.
A test is not something you “do”, it is something you “write” and run once, twice, etc.

- It is a piece of code
- Testing is therefore “automated”
- Repeatedly executed, even after small changes

The cycle of **Red** - **Green** - **Refactor**
Testing in all phases and levels...

- Unit testing
- Acceptance testing
- Integration testing
- Regression testing
- Functional testing
- System testing

Empirical findings

- The defects were decreased from 5% - 45%

**TDD benefits:**

- shortens the programming feedback loop
- provides detailed specification (tests)
- improves quality (design and code)
- provides concrete evidence that your software works
- ensures that your design is clean
- provides an interface before algorithm
- supports evolutionary development
- encapsulates learning
- enhances intermediate stability
- enhances confidence in code changes
Pair Programming

Continual design and code review process

- Reduction of the defects
- Improvement of design and code quality
- Higher product quality
- Faster cycle time
- Enhanced learning - Knowledge transfer
- Enhanced trust/teamwork
- More communication
- Collective code ownership

Empirical findings

- Design and code quality improvement varying from 15% up to 65%

Customer interaction and feedback

Customer involvement in the development phases...

Planning
- Scenarios and Features
- Cost estimation
- Acceptance testing

Implementation
- Beta versions
- Usability data and metrics
- Acceptance and Functional Testing

Testing
- Prioritize testing
- Incorporate necessary changes or reprioritize features

Release
- Implement lessons learned from Retrospectives
- Team Adaptation
Empirical Findings

A systematic literature review, evaluating quality approaches and metrics, in Agile practices, according to ISO/IEC 12207 and ISO/IEC 9126 standards:

**TDD**
- The defects were decreased from 5% - 45%
- Improvement of external quality (TFD + Refactoring)
- Decreased fault rates in Industry (than in Academia)

**Pair Programming**
- Design and code quality improvement varying from 15% up to 65%

**Other Practices**
- Quality improvement for combined practices including Planning Game or Sprint Planning, Refactoring and On-site Customer.

Human issues in Agile QM

- Teams as adaptive ecosystems (roles and tasks depend on developers)
- Communication and Collaboration (most critical success factors)
- Human factors must be identified, understood, controlled, predicted or manipulated.
- Organizations and managers must capitalize on developers potential talents and strengths

◊ Values
◊ People
◊ Teams
◊ Culture
**Key success factors**  

**Enhance:**

- **Agile Quality Planning & Execution**
  - “just enough, just-in-time (JIT)”
  - **Planning levels (Quality criteria):** for User stories, Iteration and Release planning, End product
  - **Visibility:** Release tracking, iteration tracking, and story tracking

- **Code quality**
  - Test-Driven Development (TDD)
  - Refactoring to improve existing design
  - Coding with design patterns
  - Pair programming
  - Shared code responsibilities
Key success factors (2/5)

Enhance:

- Test-Driven Development, Not Defect-Driven
  - TDD is a skill
  - Efficiently and effectively testing
  - Pairing and shared testing code responsibilities, refactoring, attention to design
  - Add more tests if they add value
  - Just Do It - Insist on achieving 100% coverage
  - Measure and Review
  - Make Results Visible

- Customer & User perspective
  - The customer must be always available
  - Business people and the project team must work together
  - User involvement is necessary
Key success factors (3/5)

Enhance:

- **Continuous Integration** (maintaining quality all the time)
  - Test Driven Integrations
  - Single source repository
  - Frequent automated builds and tests

- **Agile Verification and Validation**

  **Validation** *(Are we building the right product?)*
  - Acceptance Test-Driven Development (ATDD): to help define the design and code
  - All User Acceptance Tests should be passing (accept the story, the iteration and the release - quality criteria).

  **Verification** *(Are we building the product correctly?)*
  - A quality development process
  - Unit testing verifies at the code level
  - Functional testing (and other tests) verifies product at higher level
  - Frequent automated builds and tests
Enhance:

Team building
- Self Organizing vs. role or title based
- Empowered to make decisions vs. decisions by outside authority
- Decisions to be consensus-driven vs. leader-driven
- Committed to success as a team vs. success at any cost
- Motivated by trust vs. fear or anger
- Maintain constructive disagreement vs. damaging conflict

Agile metrics
- Discuss the importance of quality with the team
- Measure results at the team level
- Identify simple metrics that can be collected early
- Track the value delivered to customer or business
- Track only as long as it adds value
- Ensuring that quality is not treated as an event, but has become a way for life for the team
Key success factors (5/5)

**Agile Quality Assurance People must:**
- Write automated specifications
- Collaborate with developers
- Continuously run all Acceptance Tests
- Be at the front of the Test Driven Development

**Agile leadership**
- Fix values and culture
- Embrace change - adjust methods to the projects
- Create best teams
- Influence team decisions - set movable boundaries
- Continuously reflect, inspect, and adapt
Test-Driven Development: By Example

Kent Beck
Addison-Wesley, 2003
ISBN 0-321-14653-0

Test-driven development: A Practical Guide

Dave Astels
Prentice-Hall/Pearson Education, 2003
ISBN 0-13-101649-0
Refactoring: Improving the Design of Existing Code

Martin Fowler
Addison-Wesley, 1999

Agile Software Development Quality Assurance

Author(s)/Editor(s): I. Stamelos, P. Sfetsos
IGI Global, 2007
ISBN13: 9781599042169
ISBN10: 1599042169
The Quality Puzzle...

Thank you for your attention!