Agile(?) Quality Management

Software Engineering or Oxymoron
A Winsor Brown
AWBrown@sunset.usc.edu

Goals of Presentation
Help you avoid headaches struggling with terms
• What could "Agile Quality Management" mean?
• What is "Quality Management", anyway?
  – Where did it come from?
  – How can it go "Agile"?
• How might "Agile" fit with the top-of-the-line Quality Management approach, Fagan's Inspections?
• What possible fit is there with the "Agile Manifesto"?
You should learn about some of our lessons learned on the path to
• Lean Quality in MBASE for CS577
• Software Engineering truth and light
TaDotEM & Software Engineering

TaDotEM: "Toward A Definition Of The Engineering Method"¹
A fascinating article, which concludes with

• "The engineering method is the use of heuristics to cause the best change in a poorly understood situation within the available resources."

Or

• "The engineering method is the use of engineering heuristics."

Only for the skilled or trained?


¹ Dr. Billy V. Koen, The Bent of Tau Beta Pi, Spring 1985.

What do I Mean by Quality Management

The Quality tasks in QM
• Quality Assessment
• Quality Tracking
• Quality Improvement

With pre-requisites
• Configuration Management
• Early Defect Finding (Identification) mechanisms
• Defect and Effort Data submittal
Back to the Future: The "Software Crisis"

The SEI formed to address the "Software Crisis", at least for DoD Software.

An accepted paradigm (heuristic?): Vary only two with constant functionality/capability

Cost Quality Schedule

CMM as process assessment framework: Productivity focus?

- Level 2 of CMM addressed planning of Cost and Schedule
- Level 3 of CMM added Quality: Improvement through
  - process focus  
  - training
  - Peer Reviews
  [MP says Fagan's Inspections are the best for L4 and L5]
- Level 4 (Managed) called for Quantitative Process Management
- (Optimizing) called for Defect Prevention
Data for Quality Management

Sufficiency of early measurement activities
- FI: Defect description, location & classification (MWE/Mm)
- Fagan’s Process/Quality Improvement Cycle: Defect Summary Report & Detailed Defect Categorization

Extrapolations in use
- ANAMAM: Make it ‘easier’ for "participants" to gather/use data
  Effect: make it data gathering more formal
- NB: ANAMAM = Another Now-Absorbed Major Aircraft Manufacturer
- CS577: Assess process adherence

CS577 MBASE Defect Reporting Concepts

CS577: Still More information
- Details to assess process adherence
- Result: Productivity $ because Overhead ↑

Agile Methods & MBASE Practices? Lessons Learned:
- Agile/Lean approach characteristics:
  Productivity ↑ by use of tacit knowledge AND
  Reduced overhead (people vs. process) and frustration

Given
- Why metrics
- What Fagan did
- CS577: Software Engineering I and II
  Large Scale Software Engineering project course
- CS577: Schedule as an Independent Variable, but
  - Still want easily gathered raw data.
  - Enough data so post-facto generation of details possible
New CS577 MBASE Defect Reporting Concepts

Range of Defect identification and reporting mechanisms

- One at a time: gnats problem reporting system
- Multiple issues/problems found by a single reviewer:
  - Agile Artifact review (only two types of forms: Issues/Concern and Defect List)
  - Agile Internal/Informal Review: Three types of forms
  - Agile Formal Review: Three different types of forms
  - Internal/Informal Review: Four different types of bigger forms
  - Formal Review: Four different types of bigger forms
  - Fagan's Inspection: Five different types of forms

Lessons Learned Along the Way

PK via Manifesto Principle:

*individuals and interactions* over processes and tools

For Quality Management in CS577 (Software Engineering I & II) which teaches MBASE

- Different information relevant for different types of reviews
- Subjective definition of a "defect":
  - Who decides? Only the "author": every change made
  - Call all identifications from review activity an "issue"

From Quality Improvement World

- Push on Productivity, get Productivity impact;
  Push on Quality, get a multiplicative impact on quality
- Defect prevention and early defect detection:
  - Increases Quality AND decreases [total] Cost
  - Requires an investment
  - Seems to require/imply "process"
"Agile QM" [Boehm]

More efficiently achieving Quality by sharing tacit knowledge

- Risk-driven
  - Importance of scenarios
  - Criticality of defects
  - Risk of late defect detection
  - Risk of missing resources
    (test data, drivers, tests, oracles, etc.)

- Efficient
  - Risk-driven PBRs
  - As early as possible
  - Automated aides
  - Essential plans

Agile QM Challenges

Avoiding scalability problems

- Hierarchical QM : Stakeholder IPTs : Rqts., Arch.
  - Buddy Reviews, Pair Programming
    Code

Rapidly achieving shared vision

- Team building (WinWin Rqts. negotiation, prototyping, group planning exercises)

Avoiding critical defect leakage ...

Making cost effective quality investments ...

© 2002 USC Center for Software Engineering AgileQualityManagement--ARR2002v1.doc – 11 of 35 v1.0 - 03/12/02
Agile QM Challenges (cont.)

Avoiding critical defect leakage
- Risk-driven plans and procedures
  - Peer Review Plans ensuring early detection of defects
  - QM Plans ensuring just-in-time test data, drivers, oracles, tools, facilities, etc.

Making cost effective quality investments
- Requires knowing the "costs"
- Requires knowing the return on the investment
[Defect Reduction Opportunity Tree helping to address?]

Understanding both QM and Agile Approaches:
mapping for understanding and opportunity, for example

<table>
<thead>
<tr>
<th>QM Function</th>
<th>XP</th>
<th>Agile QM</th>
<th>Normal QM</th>
<th>Heavy QM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reqts. validation</td>
<td>Story discussions, test cases</td>
<td>Thorough validation of critical reqts.; Light review of non-critical reqts.</td>
<td>...</td>
<td>Every reqt. complete, consistent, traceable, testable; etc.</td>
</tr>
<tr>
<td>Detailed Design verification</td>
<td>Pair programming</td>
<td>CDRs replaced by risk-driven peer reviews; pair prog'g an option?</td>
<td>So-so CDRs; informal in-process reviews or walkthroughs</td>
<td>Extensive peer reviews; Exhaustive CDRs</td>
</tr>
</tbody>
</table>
Agile Methods, The CMM, & Sw. Eng: Agile Quality Management

Could organizations practicing Agile Methods claim to be Level 5?

SEI Level 5 (Optimizing) called for Defect Prevention

- Agile methods approaches highlighted above seem to do that

SEI CMM Level 4 (Managed) called for Quantitative Process Management

- But Agile methods put people first?
- Never [explicitly] got the quantitative data

Agile(?) Quality Management
Some Details

Software Engineering or Oxymoron
A Winsor Brown
AWBrown@sunset.usc.edu
Back to the Future: The "Software Crisis"

The SEI formed to address the "Software Crisis", at least for DoD Software

An accepted paradigm (heuristic?): Vary only two with constant functionality/capability

Cost

Quality

Schedule

Is there a Quality*Functionality constant for software development?

Can Productivity be an independent variable?

- Productivity $\propto 1$/Cost
- Productivity $\propto 1$/Schedule
- Productivity $\propto$ Quality
  - Early removal increases Productivity: shrinks the Hidden Factory
  - Prevention reduces defect injection

One view of SEI's approach: They initially pushed(?) on Productivity, but Quality was not far behind

- Level 2 of CMM addressed planning of Cost and Schedule
- Level 3 of CMM added Quality:
  - Improvement through process focus
  - Improvement Peer Reviews
    [MP says Fagan's Inspections are the best for L4 and L5]
  - Improvement through Training

SEI CMM Level 4 (Managed) called for Quantitative Process Management

SEI Level 5 (Optimizing) called for Defect Prevention
Data for Quality Management

Sufficiency of early measurement activities

- FI: Defect description, location & classification (MWE/Mm)

Extrapolations in use

- ANAMAM: Make it 'easier' for "participants" to gather/use data
  Effect: make it data gathering more formal
  NB: ANAMAM = Another Now-Absorbed Major Aircraft Manufacturer
- CS577: Assess process adherence

Earliest(?) Quantification Steps – Sufficient?

Michael Fagan's "Defect Free Software Development" course (© 1985)

- 90% Defect detection/reduction BEFORE test when coupled with process execution requirements
  [Come see the DR OT demonstration]
- Course used simple defect data recording instrument with
  - defect description,
  - location, and
  - classification as MWE/Mm
- Course exercises, done on real software, reported simple information back to the class:
  - summary of defects; size of artifact; review rates
  - inspection effort; expected time to fix
Data for Quality Management (cont.)
Earliest(?) Quantification Steps – Sufficient?

Michael Fagan's "Defect Free Software Development" course (© 1985)

- Reflection on above: SE operational metrics to allow ROI calculation and inspection closed loop control (given some heuristics also covered in the class)

- Fagan also covered "quality management"
  - Lead Inspector (for Inspection process management)
  - For causal analysis
    - Defect Summary Report
    - Detailed report with break down by "type" (similar to those later seen in ODC)
What Organizations Extrapolated

ANAMAM: Make data gathering more formal
- to help(?!) guide participants
- for regularity to ease metrics application/use
- Added* or expanded^ forms for
  - *Invitations/Announcements
  - *Preparation
  - *Defect log
  - ^Detailed defect type summary
  - ^Summary report

CS577: Still More information
- Details to assess process adherence
- Result: Productivity ↓ because Overhead ↑

CS577 MBASE Defect Reporting Concepts

Agile Methods and MBASE Practices?
Lesson's along the way: Agile Methods
- Agile/Lean approach characteristics:
  - Productivity ↑ by use of tacit knowledge AND
  - Reduced overhead (people vs. process) and frustration

Given
- Why metrics
- What Fagan did
- CS577: Software Engineering I and II
  - Large Scale Software Engineering project course
- CS577: Schedule as an Independent Variable, but
  - Still want easily gathered raw data.
  - Enough data so post-facto generation of details possible
New CS577 MBASE Defect Reporting Concepts

Range of Defect identification and reporting mechanisms

- One at a time\(^2\): gnats problem reporting system
- Multiples found by a single reviewer: Agile Artifact Review (only two types of forms: Issues/Concern and Defect List\(^6\)
- Agile Internal/Informal Review: Three types of forms
- Agile Formal Review: Three different types of forms
- Internal/Informal Review: Four different types of bigger forms
- Formal Review: Four different types of bigger forms
- Fagan's Inspection: Five different types of forms\(^2\)

\(^2\) Examples follow


---

New CS577 MBASE Defect Reporting Concepts (cont.)

gnats Problem Reporting System: chance problems

![Report problems - CodeCount](Image)

Please provide details of the problem.

- Classification:
  - [ ] Other
  - [ ] Issue
  - [ ] Concern
  - [ ] Bug

- Class:
  - [ ] Software bug
  - [ ] Other issues

- Nature:
  - [ ] Free text

- Additional information:
  - [ ] Free text

New CS577 MBASE Defect Reporting Concepts (cont.)

Agile Artifact Review (cont.)

<table>
<thead>
<tr>
<th>Project</th>
<th>Review #</th>
<th>Artifact</th>
<th>MBASE Phase/level</th>
<th>Review Date(s)</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Type of review</th>
<th>MBASE Phase/level</th>
<th>Review Date(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Independent Review</td>
<td>Major</td>
<td>Number of Unavoidable defects</td>
</tr>
<tr>
<td>Minor</td>
<td>Number of Avoidable defects</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Problems/Issues</th>
<th>Location(s)</th>
<th>Description</th>
<th>Classification</th>
<th>Nature of defect</th>
<th>Location of fix</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Major</td>
<td>Major</td>
<td>Major</td>
<td>Major</td>
<td>Major</td>
<td>Major</td>
<td>Major</td>
</tr>
<tr>
<td>Minor</td>
<td>Minor</td>
<td>Minor</td>
<td>Minor</td>
<td>Minor</td>
<td>Minor</td>
<td>Minor</td>
</tr>
</tbody>
</table>

© 2002 USC Center for Software Engineering AgileQualityManagement-ARR2002v1.doc - 28 of 35 v1.0 - 03/12/02
### New CS577 MBASE Defect Reporting Concepts (cont.)

#### Fagan's Inspection for CS577 Forms (cont.)

<table>
<thead>
<tr>
<th>Project</th>
<th>Moderator</th>
<th>Inspection #</th>
<th>Date</th>
</tr>
</thead>
</table>

**Type of inspection**

<table>
<thead>
<tr>
<th>Defects found</th>
<th>Minor</th>
<th>Major</th>
<th>No. of Open Issues</th>
</tr>
</thead>
</table>

**Estimated completion date**

<table>
<thead>
<tr>
<th>Total time estimated for review by author</th>
<th>Total time expended during review by author</th>
</tr>
</thead>
</table>

#### DEFECTS

<table>
<thead>
<tr>
<th>#</th>
<th>Location (C)</th>
<th>Description</th>
<th>Classifications</th>
<th>Severity (Phenomena, Change Code, etc.)</th>
<th>Location of correction(s)</th>
<th>Date of Fix</th>
<th>Comments</th>
</tr>
</thead>
</table>

Lessons Along the Way

PrN via Manifesto Principle: *individuals and interactions* over processes and tools

- Using people effectively achieves maneuverability, speed, and cost savings.
- People can transfer ideas faster by talking face to face than by writing and reading documents.
- A few designers sitting together can produce a better design than each could produce alone.
- When developers talk with customers and sponsors, they can iron out difficulties ... in ways not possible when they are not working together.

Clear concern about schedule
Lessons Along the Way (cont.)

For Quality Management in CS577 (Software Engineering I & II) which teaches MBASE

- Different information relevant for different types of reviews
- Subjective definition of a "defect":
  - Who decides? Only the "author": every change made
  - Call all identifications from review activity an "issue"

Quality Improvement World

Push on Productivity, get Productivity impact
Push on Quality, get a multiplicative impact on quality

Defect prevention and early defect detection:

- Increases Quality AND decreases [total] Cost
- Requires an investment
- Seems to require/imply "process"
Orthogonal Defect Classification (ODC)

- Subjective definition of a "defect":
  - Who decides? Only the "author": every change made
  - Call all identifications from review activity an "issue"

- Defect Signature:
  - Organization (ecosystem) specific
  - Only used within organization for comparison
  - An possibly closed loop planning
  - Detailed defect types very much like Fagan's initial

- The "trigger" is what made a problem visible
  - Good for possibly improving detection efficiency
  - But probably help find new way detection methods