Scaling Agile Testing Using The TMMi

Thomas M. Cagley Jr.
Agenda

• History
• What Agile brings to the table
• TMMi, a framework for verification and validation
• Why TMMi is useful for scaling Agile testing
• Case Study: TMMi as a tool in scaling testing process
Dangerous History

• Studies by Capers Jones indicate that IT has more named specialties than any other profession. Each specialty represents a boundary and potentially a silo!

• Agile has primarily been practiced at the team level, making Agile on large projects difficult.

• Infrastructure and coordination are needed to scale Agile without overhead.

Blow up the silos!
Agile Manifesto

Customer Collaboration
- Frequent comm.
- Close proximity
- Regular meetings
- Multiple comm. channels
- Frequent feedback
- Relationship strength

Individuals & Interactions
- Leadership
- Boundaries
- Empowerment
- Competence
- Structure
- Manageability/Motivation

Working Software
- Clear objectives
- Small/feasible scope
- Acceptance criteria
- Timeboxed iterations
- Valid operational results
- Regular cadence/interval

Responding to Change
- Org. flexibility
- Mgt. flexibility
- Process flexibility
- System flexibility
- Technology flexibility
- Infrastructure flexibility

Contracts
- Contract compliance
- Contract deliverables
- Contract change orders

Processes
- Lifecycle compliance
- Process Maturity Level
- Regulatory compliance

Documentation
- Document deliveries
- Document comments
- Document compliance

Project Plans
- Cost Compliance
- Scope Compliance
- Schedule Compliance

Rico 2014
Agile Delivers Faster

• Agile delivers value at the end of every sprint or iteration.
• Value builds and compounds on each iteration.
• Waterfall delivers value upon release or implementation.

If the market changes during a waterfall project, zero value may be delivered.
Lean Provides Focus And Flow
Team Level Agile And Quality

Quality Focus:
• Cross-functional teams, including testers
• Pairing is equivalent to peer reviews
• Refactoring simplifies code
• Continuous builds ensures integration
• Automated testing builds knowledge
Verification and Validation are Important

• Survey says…
  – Confident that Testers (internal and external) are NOT "best in class," more than 95% of organizations
  – Poor monitoring and reporting
  – Reinventing the process and template wheel
  – Wasted test execution time
  – Rework
  – No idea what their rework was on a per-sprint basis
  – Software released with too many defects
  – Lack of adequate software quality assurance program
  – Poor requirement practices
  – Poor management of stakeholders
  – Poor configuration management of the code and documentation
  – Project with an “improbable” likelihood of success
  – Cost of Quality

12% of project wasted
1 week per project
up to 30%
60-80% of dev
44% of people surveyed
32% of organizations
38% of organizations
 costs 60%
20% of project test budget
45% of defects
68% of projects
15-25% of Sales

77% of managers aware that bad decisions have been made given lack of access to accurate information.
Cost to fix defect at end of lifecycle 470-880 times greater.
Delivering: Test First

Test First
• Increased communication
• Better requirements
• Better code
• Fosters team culture
• Change is required

Test Last
• Status quo communication
• Testing apartheid
• No change required
Evidence: Agile Cost of Quality

Product Rather Than Project

- Agile value and principles for team guidance
- Scrum for team discipline, control, and roles
- Extreme Programming (xP) for technical structure and focus
- Portfolio Kanban to control portfolio flow and WIP
- Lean principles to identify bottlenecks and constraints as work transitions for the backlog to delivery
What Is The TMMi

What is the TMMi standard reference model?

• A staged assessment model
• Aligns with the CMMI model
• Compliant with SPICE - ISO 15504
• Can be used to support ITIL and others
• Lower levels can be used in isolation
• Follows similar assessment and ratings process as CMMI

TMMi does not replace your testing processes, but rather organizes and structures what you do!
Goal Of The TMMi

• Purpose of the TMMi
  – Provide a standard reference model for all industries
  – Identify testing strengths and best practices
  – Improve integrated test effectiveness and efficiency
  – Identify issues and risks
  – Identify a program of test process improvements
  – Provide public governance, measurement, and accreditation
TMMi: Staged Model Structure

- **Independent model**
- **Integrated best practices derived from over 14 quality and test models**
- **Underpinned by Risk Management and Monitoring & Control**
- **Addresses all test levels**
- **Focused on moving organisations from defect detection to defect prevention**
- **Fit for purpose – Appropriate process in the appropriate places**
- **Is the de facto International Standard to measure test maturity**

The 5 levels of TMMi

- **Level 1 - Initial**
  - Chaotic process
  - Dependant on heroes
  - No understanding of the cost of quality

- **Level 2 - Managed**
  - Test policy and strategy
  - Test planning
  - Test monitoring and control
  - Test design and execution
  - Test environment

- **Level 3 - Defined**
  - Test organisation
  - Test training programme
  - Test life cycle and integration
  - Non-functional testing
  - Peer reviews

- **Level 4 - Management and Measurement**
  - Test measurement
  - Software quality evaluation
  - Advanced peer reviews

- **Level 5 - Optimisation**
  - Test process optimisations
  - Quality control
  - Defect prevention

**Focused on testing ONLY**
Using A Reference Model To Scale Agile Testing

• A Test Reference Model is a predefined structure for verification of a test process
• Substantial efforts made to improve the quality of products
• Despite quality improvement approaches, the software industry remains far from effective and efficient
• Testing accounts for between 30% and 50% of total project costs
Why is TMMi Useful to Scale Agile

Attributes of the TMMi
• Not a rigid framework
• Covers all of the quality lifecycle
• Identifies areas of good practices, weaknesses, and gaps in the processes
• Uses good practices, which are appropriate to the way the people operate
• Is focused on defect prevention rather than defect detection
• Uses “Do you have it? Are you using it? Does it work?” assessment technique

Agile Manifesto
We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

1. Individuals and interactions over processes and tools
2. Working software over comprehensive documentation
3. Customer collaboration over contract negotiation
4. Responding to change over following a plan
Case Study

- Large multinational organization
- Several large internal value streams (product related)
- Each application was pursuing its own development process
- Dates were being missed
- Implementation full of horror stories
- Products failed to integrate well
- Quality was … not good

Original State
TMMi Assessment
TMMi Test Planning
Self-Assessment Questions

Test Planning
• **Do you perform product risk assessment?**
  Strongly Agree – Agree – Neutral – Disagree – Strongly Disagree

• **Do you have a test approach that is based on identified risks?**
  Strongly Agree – Agree – Neutral – Disagree – Strongly Disagree

• **Do you have an established test estimating method?**
  Strongly Agree – Agree – Neutral – Disagree – Strongly Disagree
Prescription: Retool

• Implement Scaled Agile
  – SAFe
  – TDD
  – Capability Teams
• Community of Practice
  – TMMi
  – Verification and Validation
  – ATDD
Scale to the Program Level

- Self-organizing, self-managing team-of-Agile-teams
- Working, system increments every two weeks
- Aligned to a common mission via a single backlog
- Common sprint lengths and estimating
- Face-to-face release planning cadence for collaboration, alignment, synchronization, and assessment
- Value Delivery via **Features and Benefits**
Work Flow: Program Layer Deliverables

- **Feature Definition:** In-depth understanding of the requirements for a piece of client-valued functionality.

- **Architecture Plan:** Architecture needed to deliver identified features including UI—User Interface, PD—Problem Domain (business logic), DM—Data Management, and SI—Systems Interfaces.

- **Release Plan:** Epics, features, stories, and architectural components required to deliver the planned features grouped by sprint (this plan continually evolves based on reprioritization and velocity changes). All types of releases should be noted in the release plan.

- **Program Relationship Map:** The relationship between applications and features within the product.

- **Test Architecture:** Overall strategy for verification and validation

- **Program Backlog**
Team-Level Agile

- Deliver “done” software on cadence
- Empowered cross-functional team
- Guided by high-level standards and architecture
- Teams deliver value using user stories prioritized by the product owner
Teams: Stable Capability Scrum Teams

Value (Work) Flow

Capability Team

Product Owner
Scrum Master
Business Analyst
Developer
Developer
Developer / Tester
Tester
Tester
Work Flow: Iteration Layer Deliverables

- **Team Charter**: Roles, responsibilities, and rules.
- **Capabilities and Dependencies**: Team skills and capabilities and dependencies.
- **Implemented Data Model**: Documented data characteristics.
- **Implemented Architecture**: Documented product architecture.
- **Done**: Functional code, commented, tested (unit and integration), with all required release notes and design documents.
- **Performance Test Results**: Documented test results.
- **Functional Test Results**: Documented test results.
Implemented TTD Using TMMi Structure

- Embed testing into the development approach.
- Provide system team for integration (and integration testing).
- Implement TDD/ATDD.
- TDD
  - Test Policy and Strategy
  - Test Planning
  - Scrum – Monitoring and Control
  - Feedback / Peer Reviews
  - Test Cycle
Suggested Implementation Notes

• TDD, ATDD, or BDD are usually not where most organizations start with Agile.
• Weigh the costs and benefits carefully.
  – Many of the benefits and criticisms of TDD and other test-first techniques can be quantified but others are less tangible.
  – The affect of TDD in your organization and environment may yield different results.
• Pilot TDD with a normal team to validate your ROI model.
• Do not forget to train!!!!
  If you don’t pilot, just remember to temper your interpretation of the outside data when you are weighing your interpretation of costs and benefits.
Case Study Results

- Coordinated, predicted delivery of functionality
- Fully tested releases
- Improved quality (fewer delivered defects)
- Improved customer satisfaction

Current State
## Continuing Evolution: Agile Maturity Curve

<table>
<thead>
<tr>
<th>Improvising</th>
<th>Practicing</th>
<th>Streamlined</th>
<th>Governed</th>
<th>Matured</th>
</tr>
</thead>
</table>
| • Short Iterations  
• Delivery of working software  
• Reviews, Demos, and Retrospectives  
• Prioritized Product Backlog  
• Daily Personal Team-Level Interactions (effective and tool supported)  
• Manual Testing  
• Builds Once or Twice Per Week  
• Configuration and Defect Tracking Tools  
• Coding Standards  
• Periodic Governance Meetings | • Iteration Goal  
• Full-time leader/SCRUM Master  
• Tools for Release and Iteration Management  
• Build Scripts, Daily Builds  
• Code Refactoring  
• Tracking of Velocity and Burn-down | • Clear Definition of Done  
• Task Boards, Stickies, and Video Conferences  
• Confidence in Estimates  
• Technical Debt and Static Analysis  
• Confidence in Estimates  
• Reusable Test Data  
• Automated Unit Testing  
• Release Management  
• Automated Testing  
• Defect Analysis | • Knowledge Management  
• Long-Term Product Roadmap  
• Advanced Metrics (Build, Code Quality, Testing) | • Decisions on inclusion of Advanced Techniques (TDD, Agile modeling)  
• More Than 60% of Tests Automated  
• Builds Several Times A Day |

Raja Bavani 2012
5 Steps to World-Class Testing Performance

1. Know your testing performance goals
   – Understand what is important to the organization

2. Assess current testing performance against your goals
   – You need to know where you are before building a map to where you want to be

3. Develop a plan and close the gap
   – Consider a framework/proven methods
   – Identify repeatable processes

4. Measure, Measure, Measure
   – Decisions based on fact are always better than assumptions or guessing

5. Continuously improve your process
   – Determine what is working, what isn’t, and what could be working even better
Tom Cagley, CFPS, CSM, CTFL
Vice President
DCG Software Value

t.cagley@softwarevalue.com
(440) 668-5717

Software Process and Measurement Podcast
http://www.spamcast.net (or iTunes)

Software Process and Measurement Blog
http://tcagley.wordpress.com