QA to AQ: Shifting towards Agile Quality

Joseph Yoder
www.refactory.com

Rebecca Wirfs-Brock
www.wirfs-brock.com
Introducing Joseph

Founder and Architect, The Refactory, Inc.
Pattern enthusiast, author and Hillside Board President
Author of the Big Ball of Mud Pattern
Adaptive systems expert (programs adaptive software, consults on adaptive architectures, author of adaptive architecture patterns, metadata maven, website: adaptiveobjectmodel.com)
Agile enthusiast and practitioner
Business owner (leads a world class development company)
Consults and trains top companies on design, refactoring, pragmatic testing
Amateur photographer, motorcycle enthusiast, enjoys dancing samba!!!
Introducing Rebecca

President, Wirfs-Brock Associates

Agile enthusiast (involved with experience reports since 1st agile conference, Director Agile Experience Reports Program, board president Agile Open Northwest)

First engineering job in Quality Assurance

Pattern enthusiast, author, and Hillside Board Treasurer

Old design geek (author of two object design books, inventor of Responsibility-Driven Design, advocate of CRC cards, hot spot cards, & other low-tech design tools, IEEE Software design columnist)

Consults and trains top companies on agile architecture, responsibility-driven design, enterprise app design, design storytelling, pragmatic testing

Runs marathons!!!
Making Quality Happen

- Budget time for quality discussions and quality testing
- During envisioning and requirements gathering, identify core qualities
- Engineer for Quality and Testing
  - Architect for Quality
- Determine appropriate times when qualities can be delivered and tested
Who are your system’s Quality Stakeholders?

- Users of the system
- Executives / System sponsors
- Internal quality stakeholders:
  - Developers
  - Database admins
  - Business process experts
  - Corporate compliance guys
Systems Development Life Cycle (SDLC)

Life-Cycle Phases

**Initiation**
- Begins when a sponsor identifies a need or an opportunity.
- Concept Proposal is created.
- Defines the scope or boundary of the concepts.

**System Concept Development**
- Develops a Project Management Plan and other planning documents.
- Provides the basis for acquiring the resources needed to achieve a solution.

**Planning**
- Analyses user needs and develops user requirements.
- Create a detailed Functional Requirements Document.
- Transforms detailed requirements into complete, detailed Systems Design Document.
- Focuses on how to deliver the required functionality.

**Requirements Analysis**
- Converts a design into a complete information system.
- Includes acquiring and installing systems environment; creating and testing databases; preparing test case procedures; preparing test files, coding, compiling, refining programs; performing test readiness review and procurement activities.

**Design**
- Includes implementation preparation, implementation of the system into a production environment, and resolution of problems identified in the Integration and Test Phases.
- Conducted by Quality Assurance staff and users.
- Produces Test Analysis Reports.

**Development**
- Describes that developed system conforms to requirements as specified in the Functional Requirements Document.
- Includes Post-Implementation and In-Process Reviews.

**Integration and Test**
- Operations & Maintenance
- Disposition
- Describes end-of-system activities, emphasis is given to proper preparation of data.
Agile Design Values

Core values:
- Design Simplicity
- Communication
- Continuous Improvement
- Teamwork/Trust
- Satisfying stakeholder needs
- Building Quality Software

Keep learning

Lots of Testing!!!
Some Agile Myths

- System Qualities easily added with an evolving architecture
- We can easily adapt to changing requirements (new requirements)
- You can change the system fast!!!
- Don’t worry about the performance, scalability, security, usability … until functionality is working…

MYTHBUSTERS
Design is about Tradeoffs

- Usability and Security often have orthogonal qualities
  - Designing Interfaces: Patterns for Effective Interaction Design
  - Security Patterns: Integrating Security and Systems Engineering

- Performance vs Small Memory
  - Quality of being “Good Enough”
BECOMING AGILE AT QUALITY
“QUALITY IS NOT AN ACT, IT IS A HABIT...”
—ARISTOTLE
Agile Teams

Cross Functional
Good Communication
Focus on Stakeholders Needs
Incrementally deliver working software
Adapt to Change as needed
Collaborative and Self Organizing
Whole Team working together
Agile Quality Teams

“Whole Team”

Architects and QA works closely with the team integrated during the day to day development
Engage in Architecture & QA activities much earlier
Works with whole-team including product owner on understanding and defining qualities
Assists teams with definition & validation of important quality requirements
Proactive working closely & coordinates between business, management and developers
Tearing Down the Walls
aka “Breaking Down Barriers”

Physical Barriers, Cultural Differences
Language/Communication, Background
Expertise, Lack of Time, Us and Them
Mentality

- How can agile teams remove the barriers and become more agile at quality?

- Tear down the walls through various actions: include QA early on; make them part of the sprints, embed them in the teams
Embedding QA with Team aka “Pair with a Quality Advocate”

- Great experience report at Agile 2014

- AgileAlliance.org

Experience Report posted: Tearing Down the Walls: Embedding QA in a TDD/Pairing and Agile Environment by Stephanie Savoia
Architecture Roles and Activities

Traditional Architects
Independent from development
Keepers of the overall vision of the architecture
Enforcers who…
  Certify compliance with corporate architecture standards
...get involved on an “as needed” basis in the software lifecycle

Agile Architects
More integrated with day to day development
Stewards for ongoing sustainable development
Mitigate architecture risks
Work with business, product owner, QA and devs to define and improve the evolving architecture
Establish good practices and pay attention to details
QA Roles and Activities

Traditional QA
Independent group
Gatekeepers who...

- Understand testing well and know how to specify and validate system qualities
- Certify app functionality based upon contracts and requirements
- ...get involved late in the software lifecycle

Agile QA
Integrated with day to day development
Proactive, engage in QA activities much earlier
Work closely with business, product owner, architects and devs to understand, define, and validate quality requirements
Who will lead?
Who contributes?

- Big Teams vs Small Teams?
- How does the Architect and QA interact with development?
Patterns for Being Agile at Quality

Core Patterns
- Breaking Down Barriers
- Integrating Quality into your Agile Process

Becoming Agile at Quality
- Whole Team
- Quality Focused Sprints
- QA Product Champion
- Agile Quality Specialist
- Monitoring Qualities
- Agile QA Tester
- Spread the Quality Workload
- Shadow the Quality Expert
- Pair with a Quality Advocate

Identifying Qualities
- Finding the Qualities
- Agile Quality Scenarios
- Quality Stories
- Specify Measureable Values or Qualities
- Fold-out Qualities
- Agile Landing Zone
- Recalibrate the Landing Zone
- Agree on Quality Targets

Making Qualities Visible
- System Quality Dashboard
- System Quality Radiator
- Qualify the Roadmap
- Qualify the Backlog
- Quality Chart
Build quality into your project rhythms
Some decisions are too important to leave until The Last Responsible Moment

so

CHOOSE THE MOST RESPONSIBLE MOMENT
Architecture Risk Reduction Tools for Larger Projects and Programs

- Grooming and vetting project/product road maps and timelines
- Agile Landing zones
- Architecture spikes
- Risk reduction backlogs
- Set-based design
How Quality Scenarios and Quality Testing Fit Into An Agile Process

Product Envisioning / Roadmap

- Identify Key Quality Scenarios

Can Include Quality Scenarios

- Develop and Manage the Backlog

Incorporate Feedback

- Plan a Sprint
- Run a Sprint

Include relevant quality tasks

Quality Testing

Functional Acceptance Testing

Deploy to Stakeholders
Test-Driven Development Should Include Quality

Requirements Envisioning (days/weeks/...)

Architecture Envisioning (days/weeks/...)

Iteration 0: Envisioning

Iteration Modeling (hours)

Model Storming (minutes)

Fast TDD (hours)

Iteration n: Development

Conceptual Modeling

What Qualities are important?

a little bit of modeling then a lot of coding, think about what qualities might fit into the “Sprint”
“Users initiate 1,000 order transactions per minute under normal operations; transactions are processed with an average latency of 2 seconds.”
“Users initiate 1,000 order transactions per minute under normal operations; transactions are processed with an average latency of 2 seconds”
## Possible Performance Scenario Values

<table>
<thead>
<tr>
<th>Portion of Scenario</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Source</strong></td>
<td>External systems, users, components, databases</td>
</tr>
<tr>
<td><strong>Stimulus</strong></td>
<td>Periodic events, sporadic or random events (or a combination)</td>
</tr>
<tr>
<td><strong>Artifact</strong></td>
<td>The system’s services, data, or other resources</td>
</tr>
<tr>
<td><strong>Environment</strong></td>
<td>The state the system can be in: normal, overloaded, partial operation, emergency mode…</td>
</tr>
<tr>
<td><strong>Response</strong></td>
<td>Process the event or event sequence and possibly change the level of service</td>
</tr>
<tr>
<td><strong>Response Measure</strong></td>
<td>Times it takes to process the arriving events (latency or deadline by which event must be processed), the variation in this time, the number of events that can be processed within a particular time interval, or a characterization of events that cannot be processed (missed rate, data loss)</td>
</tr>
</tbody>
</table>
“A known, authorized user transfers money between accounts. The user is later identified as an embezzler by the institution they belong to and the system then restores funds to the original account.”
## Possible Security Scenario Values

<table>
<thead>
<tr>
<th>Portion of Scenario</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Source</strong></td>
<td>A human or another system. May be identified (correctly or not) or be unknown.</td>
</tr>
<tr>
<td><strong>Stimulus</strong></td>
<td>An attack or an attempt to break security by trying to display information, change information, access system services, or reduce system availability.</td>
</tr>
<tr>
<td><strong>Artifact</strong></td>
<td>The system’s services or data.</td>
</tr>
<tr>
<td><strong>Environment</strong></td>
<td>The system might be online or offline, connected to or disconnected from the network, behind a firewall or open to the network.</td>
</tr>
<tr>
<td><strong>Response</strong></td>
<td>Authenticates user; hides identity of the user; blocks or allows access to data and/or services; records access/modification attempts by identity; stores data in encrypted format; recognizes unexplained high demand and informs a user or other system, or restricts availability.</td>
</tr>
<tr>
<td><strong>Response Measure</strong></td>
<td>Time/effort/resources required to circumvent security measures with probability of success; probability of detecting attack; probability of identifying individual responsible for attack or access/modification; time/effort/resources to restore data/services; extent to which data/services are damaged and/or legitimate access denied.</td>
</tr>
</tbody>
</table>
“A developer adds support for a new service code to the system by adding the service code to the definitions table and modifying the UI to make it available to users. The modification is made with no data schema changes.”
“An unknown sensor sends a report. The system stores the raw data in the unknown sensor database (to potentially be processed or purged later) and logs the event.”
Finding Even More Qualities: Fold-out Qualities

“As a customer I want to to place an order using my credit card.”

Quality-related acceptance criteria that can be attached to specific functional user stories

- Usability: Can I cancel my order? When?
- Security: Does the system retain credit information? If so, can I control how that information is retained?
- Security: Is credit information securely transmitted?
- Security: Is it protected from unauthorized access?
- Performance: How fast can I place an order and receive confirmation? When there are lots of users?
- Availability: What happens if the credit card service is unavailable?
- …
You can’t test warm and fuzzy…

“It should be easy to place an online order”

TURN VAGUE STATEMENTS INTO
CONCRETE MEASURABLE
ACTIONS
“80% of novice users should be able to place an order in under 3 minutes without assistance”

or

“80% of novice users should be able to place an order in under 3 minutes only using online help”
Agreeing on scenarios and target values

Some Options...

- Toss out a reasonable number, then discuss to come to a consensus
- Average informed individuals’ estimates
- Use an existing system as baseline
- Values for similar scenarios
- Benchmark working code
- ...

There is more than “pass” or “fail”

- **Landing Zone**: Lets you define a range of acceptable values
  - **Minimal**: OK, we can live with that
  - **Target**: Realistic goal, what we are aiming for
  - **Outstanding**: This would be great, if everything goes well
Product Landing Zones

- Key criteria defined at the beginning
- New criteria incrementally added
- Values recalibrated based on ongoing measurements/tradeoffs
Quality Focused Sprints

Features don’t make a viable system; rather a viable system is accomplished by focusing on features accompanied by paying attention to system qualities.

- How can you incorporate these other non-functional requirements into your system as needed?

- Therefore, take time to focus on your software’s non-functional qualities and devote a sprint to measuring and improving one or more of your system’s qualities.
Focus on Quality

- QA can gather / organize quality scenarios in collaboration with the development team
- Identify architecture risks and evolve architecture to incorporate important qualities
- Additional quality scenarios can be gleaned from Service Level Agreements (SLAs)
- Include relevant quality tests as part of each sprint
- Test important qualities early
- Automate “easy” quality tests
Make Qualities Visible

- Include quality scenarios for dev & testing in your backlog
- Maintain a separate quality scenario backlog
- Include quality and functional acceptance tests as acceptance criteria for releases
- Identify Architecture Tasks
- Part of the Roadmap
- Quality Radiators
Monitor Qualities—Build An Operational Dashboard
Sustainable Architecture
Sustainable Architecture

➢ Stewardship
  - Follow through
  - Ongoing attention
  - Not ignoring the little things that can undermine our ability to grow, change and adapt our systems
Test Architecture

Agile projects have a huge focus on test automation, however to enable more low level testing, such as at unit or component level tests, the architecture should provide for testability. Sometimes there needs to be additional interfaces defined specifically to support testing qualities.

- **How can you define how architectural components should be tested?**

- **Define the test approach for each kind of component, considering its scope, technique, kind of test and tools that are going to be used**
Sustaining an Architecture

- Minimize architectural debt: Support the ability to change/adapt what needs to change
- Make what is too difficult, time consuming, or tedious easier
- Decide at the most responsible moment, not the last possible moment
- Learn and evolve

Keep the system “livable” for its users and developers.
Agile Values Drive Architectural Practices

- Do something. Don’t debate or discuss architecture too long
- Do something that buys you information
- Prove your architecture ideas
- Reduce risks
- Make it testable
- Prototype realistic scenarios that answer specific questions
- Incrementally refine your architecture
- Defer architectural decisions that don’t need to be immediately made

Do something! Prove & Refine.
Indicators You’ve Paid Enough Attention to Architecture

- Defects are localized
- Stable interfaces
- Consistency
- Developers can easily add new functionality
- New functionality doesn’t “break” existing architecture
- Few areas that developers avoid because they are too difficult to work in
- Delivering System Qualities while delivering System Functionality
TESTING SYSTEM QUALITIES
Many Agile Teams “only” Focus on Functional Testing

Functional:

- How do I …?
- Tests user stories work as advertised
  - “As a reviewer I want to add a note to a chart”
  - “Compute the charge for an invoice”
- Tests boundary conditions
  - Can I add more than one note at the same place?
  - Are excess charges computed correctly?
Most testers spend the majority of their time writing functional tests.

...BUT THERE’S A LOT MORE TO TEST THAT YOUR SOFTWARE WORKS AS ADVERTISED.
System Quality Tests

- How does the system handle…?
  - system load …? number of add note transactions/minute under normal load
  - system support for…? simultaneously updating charts, reliability, …
  - usability…? ease of locating and selecting notes

- Tests that emphasize architecture capabilities and tangible system characteristics
Testing System Qualities

Qualities to consider…

- Usability
- Security
- Performance
- Scalability
- Internationalization
- Availability
- Flexibility
- Accessibility
- Location
- Regulation
Quality Assurance Teams

Understands testing well and knows how to specify and validate system qualities

Certify the functionality of the application based upon the contract and requirements

Many QA groups work independently from the software team

Usually involved late in the process and not a lot of communication with team

Usually not part of the Agile team…
Testing System Qualities

- Some require production or near-production environments
  - Load and performance tests
  - Complex quality stories involving interactions with several systems
- Some require extensive setup
- Some are “easy” and can be part of an automated quality test suite
Two Types of Quality Tests

Simple Measurements

- A measurement of a quality that (ideally) can be automated (performance, load, ...)

“Users initiate 1,000 order transactions per minute under normal operations; transactions are processed with an average latency of 2 seconds.”

Complex Quality Stories

- A complex set of actions that need to be verified as working. Hard to automate and might take a lot of setup (security, usability...)

“A known, authorized user transfers money between accounts. The user is later (next day) identified as an embezzler by the institution they belong to and the system then restores funds to the original account.”
When to test?

- Determine Important Qualities that should be considered early
  - Involve Product owner, architect, tech and QA leads in deciding when / how they should be tested
- Some qualities take several Sprints to develop
- Test qualities at most responsible moment
  - Do not wait until the end!!!
- Some Sprints may focus primarily on Qualities
Quality Testing Cycle for TDD

1. Identify and Write Quality Scenarios
2. Write Code and Tests
3. Clean up Code (Refactor/review/rework)
4. Check all Tests
   - all tests succeed
5. Verify Quality Scenarios
   - quality scenario fails
6. Ready to Release?
   - Ship it!!!
7. Check all Tests Succeed
   - 1 or more tests fail
Be Agile
Be Agile
Agile Mindset
Continuous Inspection

Asian PLoP 2014 Paper
Continuous Inspection

Asian PLoP 2014 Paper

**Code Smell Detection**

Metadata (Test Coverage, Cyclomatic Complexity, Technical Debt, Sizes, …)

Application Security Checks

Architectural Conformance

Automate where you can!!!
Other Techniques for Improving Quality

Steve McConnell

Average is 40% for any one technique!

Combining techniques gives you quality (> 90%)
Agile Quality Driven Development Is...

- **Practical.** Incrementally add and test for important system qualities at the “most responsible moments”, test early and often!!!

- **Thoughtful.** What system qualities need to be delivered & when? Who should write quality tests? When should you architect & test for qualities?

- **Realistic.** You only have so much time and energy so focus on the “Essential Qualities”
Summary

- Quality doesn’t just happen. It needs to be thought about and carefully considered.
- If you don’t pay attention to system qualities they can be hard to achieve at the last moment.
- Quality Scenarios are easy to write and read.
- Agile Quality is a “whole team” effort.
- Architecture is important to delivering System Qualities and it is critical to address issues at the “most responsible moments.”
Dogmatic

Synonyms: bullheaded, dictative, doctrinaire, fanatical, intolerant

Antonyms: amenable, flexible, manageable

Pragmatic

Synonyms: common, commonsense, logical, practical, rational, realistic, sensible

Antonyms: idealistic, unrealistic
Resources

- Agile Myths: agilemyths.com
- Wirfs-Brock Associates: www.wirfs-brock.com
- The Refactory: www.refactory.com
- Joe’s website: joeyoder.com
- TeamsThatInnovate (www.teamsthatinnovate.com)
- Our Pragmatic TDD Course:
  - refactory.com/training/test-driven-development
  - wirfs-brock.com/pragmatictestdrivendevdevelopment.html
- Being Agile at System Qualities workshop:
  - wirfs-brock.com/being_agile_about_system_qualities.html
  - www.adaptiveobjectmodel.com/2015/04/qa-to-aq-shifting-towards-agile-quality
- Landing Zone blogs:
  - wirfs-brock.com/blog/2011/07/20/introducing-landing-zones/
  - wirfs-brock.com/blog/2011/08/16/who-defines-or-redefines-landing-zone-criteria/
  - wirfs-brock.com/blog/2011/08/05/landing-zone-targets-precision-specificity-and-wiggle-room/
- Introducing Pragmatic TDD:
  - wirfs-brock.com/blog/2011/09/23/what-is-pragmatic-testing-all-about/
  - http://adaptiveobjectmodel.com/2012/01/what-is-pragmatic-tdd/
Thanks!!! Questions???

rebecca@wirfs-brock.com
Twitter: @rebeccawb

joe@refactory.com
Twitter: @metayoda