Tearing Down the Walls
Embedding QA in a TDD/Pairing and Agile Environment

A little about me

A little bit of background

From Waterfall to Agile

Agile evolution - TDD, Pairing, Embedded QA

Questions?
Stephanie Savoia – Marchex, Inc., Seattle, WA

▪ History:
  – Involved in quality for the last 12 years – both QC and QA, including legal systems field. Applied Discovery, LexisNexis
  – Have experience morphing QC processes into QA methodologies.

▪ Current:
  – Technical Lead in QA with Marchex, Inc., a mobile advertising technology company
  – Expanded my Agile experience with my current team:
    • Test-driven development
    • Agile and XP Pairing
    • Pairing on this team is not only Dev/Dev but also QA/Dev.
My history at Marchex
The way Marchex was…

- **Waterfall**
  - Documentation before coding

- **Siloed teams**
  - Mentally and physically

- **Development (Dev)**
  - Daily check ins
  - Toss the code over the wall to QA
  - Move on to the next thing
  - Try to remember what I did when QA has questions

- **Quality Assurance (QA)**
  - Receive daily builds
  - Questions for developers
  - Bug discovery/reporting
  - Regression testing
Transitioning to Agile – Go! ... um ...

- **2010: We’re “Going Agile!”**
  - I hear crickets
  - Learned on our own (painfully)
  - Scrum boards, daily stand ups, 2 week iterations and backlogs

- **Other teams implemented Agile-Fall**
  - Boards, Stand ups, Iterations and Backlogs
  - Actually shorter waterfall sessions

- **Confusion!**
  - From exhaustive documentation to no documentation
  - Two Scrum boards
  - Two retrospectives
  - Inconsistent, unpredictable & low quality releases
Transitioning to Agile – 1st try; not bad

- **Good!**
  - Eliminate physical separation
    - Consolidate into one location
  - Surfaced several underlying problems
    - People still not partnering
    - Hiding in cubes to avoid conversations

- **But still – big problems!**
  - Poor cross-team communication
    - Bottlenecks, Time crunches / overtime, QA not asking for help
  - **STILL** Not Partnering
  - Different goals
    - Dev wanted to create
    - QA wanted to prevent problems in prod
  - Fear
    - Dev think learning test principles = More work
    - QA doesn’t want to lose turf
  - Duplication
Becoming more Agile

1. 2011: Change in team leadership and methodology
   - Introduced Extreme Programming (XP)
   - Team size at beginning of 2011 - 24
     • Dev – 15
     • QA – 9

2. Remove self made roadblocks
   - 1 Scrum board
     • Cards still had QA and Dev estimates
   - 1 Retro
     - Move from Quality Control to Quality Assurance
       • QA added to meetings prior to coding
         - Requirements gathering
         - Risk assessment
         - Design
       • Use QA's domain knowledge to alert possible issues early
Results!

- **Organizational - Evaluate Roles and adjust**
  - 3 QA Manager Roles Eliminated; all retained, chose new positions
  - SDET position in our group eliminated; most retained, moved to other groups
  - Created Support Team

- **Product and Team – Quality and job satisfaction**
  - High quality weekly releases
  - Happy SDETs on their new teams
  - QA Engineers still in a state of fear and stress
    - Fear of change
      - Losing the comfort of their processes
      - Didn’t mind testing after coding
      - Influencing quality earlier in the process did not interest them
      - Didn’t mind not having a voice until late in the process and it was expensive to fix

- **Change QA’s methodology**
  - Measurement
  - Customer / User Advocacy
  - Use the information!
How we continued to evolve

- **Measurement**
  - High Quality Software is:
    - Low cost to maintain
    - Few production issues
    - Happy customers
    - Solves the actual problem
  - Thresholds and quantifiable measurements
    - Ensure “low”, “few”, “happy” and “solves” are occurring

- **Advocacy**
  - Know who your customers/users are
  - Internal Customer/User
    - Talking to and shadowing the users
    - Bring findings back to team
      - Requirements/Acceptance Criteria
      - Requests
      - Solidifies ‘Done’ for a story
How we continued to evolve (cont’d)

- Advocacy (cont’d)
  - External customer/user
    - Have preferred workflows
    - Identify complication early on
    - Empathy interviews
      - Uncover insights, beliefs and values
    - Work with Business to identify external customers
      - 60 – 90 minute interview
      - Incentives
      - Pencil and paper demos

- Use the information
  - Anywhere BEFORE coding
    - Design, planning, story writing, estimation, etc….
  - Continue during coding
  - Test after coding
    - Exploratory, Scenario/workflow and integration
We kept going! Building quality into the product

- Test-driven development (TDD)
  - Run existing unit tests
  - Write a failing test
  - Add/Edit the code
  - Re-run the new test
    - Edit code until new test passes
  - Run all of the tests again
    - Ensures existing tests are not failing after code changes
  - If tests fail?
    - Fix code/test and repeat until the tests pass
  - DEPLOY!
  - The test is now a part of our regression suite
We kept going! (cont’d)

- **QA job changes**
  - Sitting with developers
    - Designing tests; Developers write all automation
    - No distinct QA tests
    - No separate test repository
    - No test duplication
  - What we gained
    - Better acceptance criteria
    - Better scenario/workflows
    - Advocacy
    - Fewer problems in production
  - Exposed those not able to have an open mind
  - Entered Q1 of 2012 with
    - 13 Developers and 1 QA
  - Team merges in Q1 of 2012
    - Stable number of 25 Developers and 5 QA
    - Some teams have QA
    - Some teams have no QA
      - 1 uber QA
How I was embedded

- Role
  - Direct member of a pair
    - Talk through the story before coding
    - What tests will be written
    - Asking questions early and often
    - Solidify acceptance criteria
    - Code review
    - Missing assertions
    - Data profiling
    - UI rendering and inspection

- Any interesting learnings from our first project like this?
  - Took 6 weeks from requirements gathering to completion, instead of 6 months
  - Less developer rework
  - QA deeper understanding of system mechanics
  - QA deeper understanding of system expectations
  - QA can question decisions as they’re being made
  - It can be done, and it’s fun!
How I was embedded (cont’d)

- **How is the quality?**
  - Minimal rework
  - Improved quality as a mindset
    - Pairs change when a new card is started
    - Integration testing in QA with production data
    - Deploys

- **Tools**
  - Retrospectives
    - Last day of iteration
    - What worked
    - What didn’t work
    - Patterns
    - Action items
Summary

- **Team make up:**
  - 4 to 5 devs and 1 QA
  - Patience & Encouragement
  - Very low churn
  - Predictability
    - Velocity
    - Forecasting

- **Trust in engineering**
  - Entire team believes in the approach
  - Everyone cares about the quality
  - Question and remind, but trust

- **Trust in leadership**
  - Believes in approach
    - No methodology succeeds without leadership’s belief in it
Summary

- Has the transition to Agile and pairing worked?
  - Painful in the beginning
  - Team new to warehousing
  - Incremental value
    - Not an entire product
    - Constant feedback
      - Redirection if needed
  - Internal analysts using our data
  - UIs accessing our data via our API
  - Code quality improvements
    - Appropriate test coverage
      - No test metric
      - Trust in team, user feedback, performance

- Who can do this?
  - All QA
    - Testers
    - SDETs
Appendix

(More Details!)
The way Marchex was…

- **Waterfall**
  - Documentation before coding
    - Functional Specification Document (FSD)
      - Written by Business Team and reviewed by Dev & QA
    - Outline Systems Design (OSD)
      - Written by Developers and reviewed by Biz & QA
    - Test Plan
      - Written by QA and reviewed by Biz and Dev

- **Siloed teams**
  - Mentally and physically
    - 2 buildings a block apart
      - Padded meetings for walking to the other building
    - No partnerships between Dev and QA
The way Marchex was (cont’d)

- **Development (Dev)**
  - Daily check ins
  - Toss the code over the wall to QA to test
  - Move on to the next thing
  - Try to remember what I did when QA has questions

- **Quality Assurance (QA)**
  - Receive daily builds
    - 1 to 3 per day
  - Questions for developers
    - Is it supposed to work this way?
  - Bug discovery/reporting
  - Regression testing
    - Things still work
Transitioning to Agile – Go! … um …

- **2010: We’re “Going Agile!”**
  - I hear crickets
    - No transition plan

- **Learning on our own**
  - Implemented Scrum boards
  - Added Daily Stand ups
  - 2 Week Iterations
  - Backlogs

- **Other teams implemented Agile-Fall**
  - Boards, Stand ups, Iterations and Backlogs
  - Actually shorter waterfall sessions
    - Code in one iteration
    - Test some in current iteration and some in next
Transitioning to Agile (cont’d)

- **Confusion!**
  - From exhaustive documentation to no documentation
  - Two Scrum boards – One for Dev and one for QA
  - Two retrospectives – One for Dev and one for QA
  - Inconsistent, unpredictable & low quality releases

- **Eliminate physical separation**
  - Consolidate into one location

- **Mental Separation**
  - More visible with possible increases
    - People still not partnering
    - Hiding in cubes to avoid conversations
Transitioning to Agile (cont’d)

- **Communication**
  - Poor between stakeholders, developers and QA
    - Bottlenecks
    - Time crunches / Overtime
    - QA not asking for help

- **Different goals**
  - Dev wanted to create
  - QA wanted to prevent problems in prod
  - Fear
    - Dev think learning test principles = More work
    - QA doesn’t want to lose turf
  - Eliminated test duplication
Becoming more Agile

- **2011: Change in team leadership and methodology**
  - Extreme Programming (XP)
    - Small team to work on a single project with 1 week iterations

- **Team size 24**
  - Beginning of 2011
    - Dev – 15
    - QA – 9

- **Remove self made roadblocks**
  - 1 Scrum board
    - Still had QA and Dev estimates on stories
  - 1 Retro
  - Move from Quality Control to Quality Assurance
    - QA added to meetings prior to coding
      - Requirements gathering
      - Risk assessment
      - Design
    - Use QA’s domain knowledge to alert possible issues as early as possible
Becoming more Agile (cont’d)

- Evaluate Roles and adjust
  - 3 QA Manager Roles Eliminated
    - All staff retained and chose their new positions
      - 1 SDET
      - 1 Sr. QA
      - 1 Technical Program Manager
  - SDET in our group eliminated
    - Most staff retained and moved to other groups
      - 1 went to DevOps then became a Dev
      - 1 left company
      - 2 other teams as SDETs
  - Create Support Team
    - Light SDET type work creating tools
    - Address support tickets
    - Escalate tickets to Dev when needed
Becoming more Agile (cont’d)

- Organizational Change Results
  - High quality weekly releases
  - Happy SDETs on their new teams
  - QA Engineers still in a state of fear and stress
    - Fear of change
      - Losing the comfort of their processes
      - Didn’t mind testing after coding
      - Influencing quality earlier in the process did not interest them
      - Didn’t mind not having a voice until late in the process and it was expensive to fix

- Change QA’s methodology
  - Measurement
  - Advocacy
  - Use the information
How we evolved

- **Measurement**
  - High Quality Software is:
    - Low cost to maintain
    - Few production issues
    - Happy customers
    - Solves the actual problem
  - Thresholds and quantifiable measurements
    - Ensure “low”, “few”, “happy” and “solves” are occurring
  - Example:
    - “happy” and “few” = lower numbers of support tickets

- **Advocacy**
  - Know who your customers/users are
  - Done in conjunction with PMs or UX
  - Done on your own
How we evolved (cont’d)

- **Advocacy**
  - Internal Customer/User
    - Talking to the users
    - Shadowing their use of the tool(s)
    - Understand their job functions, constraints and goals
    - What do they really need
    - Bring findings back to team
      - Requirements
      - Requests
      - Acceptance criteria
      - Solidifies ‘Done’ for a story
      - Gives users a voice
How we evolved (cont’d)

- Advocacy (cont’d)
  - External customer/user
    - Have preferred workflows
    - Identify complication early on
    - Empathy interviews
      - Team of 2-3
        - Interviewer
        - Note taker
        - Observer
    - Does not have to be performed by a User Experience (UX) team
      - Any combination of:
        - Project/Program manager
        - Developer
        - QA
How we evolved (cont’d)

- External customer/user (cont’d)
  - Empathy interview goals
    - Uncover insights
    - Beliefs
    - Values
  - Work with Business to identify external customers
    - Come to us
    - Go to them – preferred
      - Better learning opportunity
    - 60 – 90 minute interview
    - Incentives
    - Pencil and paper demos
      - Demo finished software later
How we evolved (cont’d)

- Use the information
  - Anywhere BEFORE coding
    - Design discussions
    - Planning
    - Story writing
    - Estimation
  - Continue during coding
  - Test after coding
    - Done by QA or Dev
    - Exploratory
    - Scenario/workflow driven
    - Integration
We kept going: Building quality into the product

- **Test-driven development**
  - Run existing unit tests
  - Write a failing test
  - Add/Edit the code
  - Re-run the new test
    - Edit code until new test passes
  - Run all of the tests again
    - Ensures existing tests are not failing after code changes
  - If tests fail?
    - Fix code and repeat until the tests pass
  - DEPLOY!
    - Regression suite
We kept going! (cont’d)

- **QA job changes**
  - Sitting with developers
    - Designing tests
      - Unit
      - Workflow
      - Integration
      - System
    - Developers write all automation
      - No SDETs in our process
  - No formal QA tests
  - No separate test repository
  - No test duplication
    - 50% duplication rate
    - Waste of time and resources
We kept going! (cont’d)

- QA job changes
  - What we gained
    - Better acceptance criteria
      - Shadowing
    - Better scenario/workflows
      - Talking to end users
    - Advocacy
      - Early in process
    - Exposed those not able to have an open mind
  - Entered Q1 of 2012 with
    - 13 Developers and 1 QA
  - Team merges in Q1 of 2012
    - Stable number of 25 Developers and 5 QA
    - Some teams have QA
    - Some teams have no QA
      - 1 uber QA
How I was embedded

- **Role**
  - Direct member of a pair
    - Talk through the story before coding
    - Tests to be written
      - Unit
      - Integration
      - Continuous integration tool
    - Asking questions early and often
    - Solidify acceptance criteria
    - Code review
    - Missing assertions
    - Data profiling
      - Source > Stage > ETL > Destination
    - UI rendering and inspection
    - Write ETLs
How I was embedded (cont’d)

- **How did I end up on this team?**
  - It’s my choice
    - Stay on UI team?
    - Move to back end team?
      - I chose a new and unfamiliar challenge

- **Any interesting learnings?**
  - Our very first end to end ETL job
    - Took 6 weeks to:
      - Gather requirements
      - Create the fact and dimension tables
      - Perform data validation together
    - How was this different from traditional back and forth testing?
      - Daily builds to test and provide feedback on
      - Took months instead of weeks
How I was embedded (cont’d)

- **How is the quality?**
  - Minimal rework
  - Improved quality as a mindset
    - Pairs change when a new card is started
    - Integration testing in QA
      - With production data
        - Find data anomalies and edge cases
    - Verify the deploys together
    - Found bugs are estimated and prioritized

- **Retrospectives**
  - Last day of iteration
  - What worked
  - What didn’t work
  - Patterns
    - Improvement
    - Standardizing
  - Action items
Our team makeup

- 4 to 5 devs
- 1 QA
- Patience & Encouragement
- Very low churn
  - 1 left company…. And came back
  - 1 asked to be on team from within
  - 1 left industry
  - 1 contractor converted to full hire
  - 1 new hire
- Predictability
  - Velocity
  - Forecasting
Trust and leadership

- Trust in engineering
  - Entire team believes in the approach
  - Everyone cares about the quality
  - Question and remind, but trust

- Trust in leadership
  - Believes in approach
    - No methodology succeeds without leadership’s belief in it
Summary

- Has the transition to Agile and pairing worked?
  - Painful in the beginning
  - Team new to warehousing
  - Incremental value
    - Not entire product
    - Constant feedback
      - Redirection if needed
  - Internal analysts using our data
  - UIs getting our data via our API
  - Code quality improvements
    - Appropriate test coverage
      - No test metric
      - Trust in team, user feedback, performance

- Who can do this?
  - All QA
    - Testers
    - SDETs