Being Agile about Qualities
"Values, Practices & Patterns"

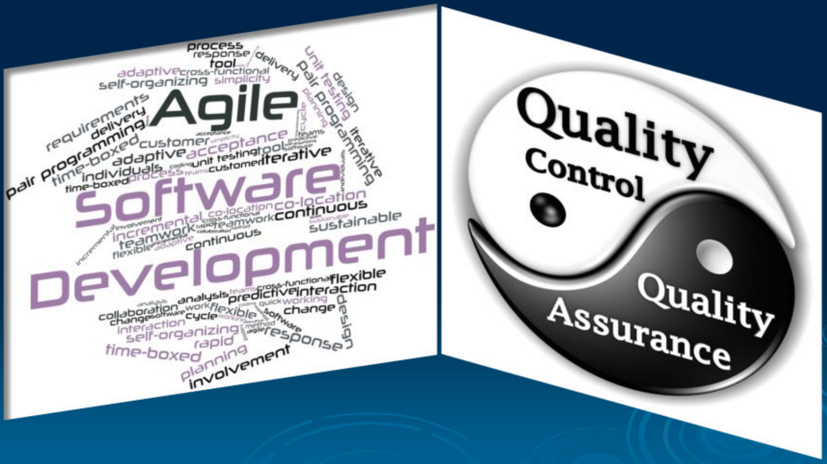
Joseph W. Yoder
Teams That Innovate
The Refactory, Inc.
Twitter: @metayoda
joe@refactory.com
http://www.refactory.com
http://www.teamsthatinnovate.com

Waseda University

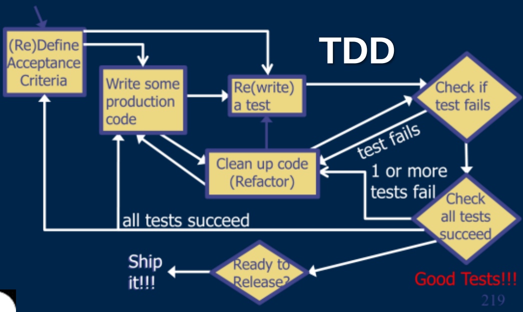
Core Ideas / Takeaways

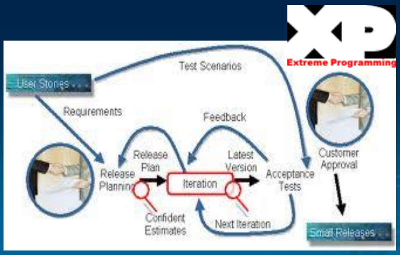
- Patterns and practices
- Values drive practice
- Quality-related activities
- Roles QA and architects play
- Call To Action (steps you can take)

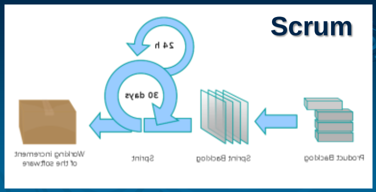
Agile & Quality



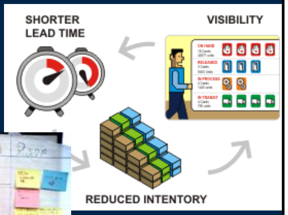
Agile Practices







Kanban (看板)
Signboard / Billboard



The basic principles of Kanban

- Limit Work in Process (WIP)
- Pull value through (WIP)
- Make progress visible
- Increase throughput
- Fixed Kanban Backlog
- Quality is part of the processed (internal)

Continuously monitor the above to improve!!!
Is this similar to a Retrospective?

Lean Development

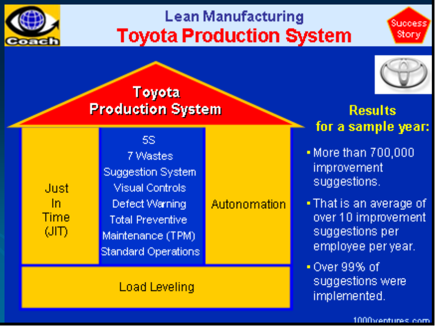
Increase Value, ***Reduce Waste (Muda)***, Improve Flow, Quality, …

Understands customer ***value*** and focus continuously to increase it

Provide ***perfect value*** to customer and business

Just in Time Practice

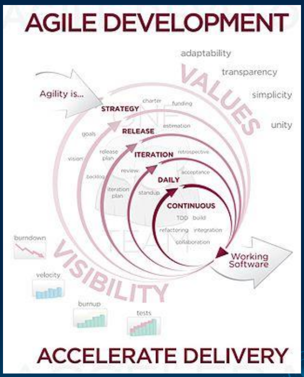
Learn and ***Improve…***



Results for a sample year:

- More than 700,000 improvement suggestions.
- That is an average of over 10 improvement suggestions per employee per year.
- Over 99% of suggestions were implemented.

Agile == Lean?



Early Agilest were influenced by Lean, but…

- Many get stuck in the process
- Many Misconceptions about Agile

Agile/Lean Design Values

- Core values:
  - **Design Simplicity**
  - **Quick Feedback**
  - **Communication**
  - **Continuous Improvement**
  - **Teamwork/Trust**
  - **Satisfying stakeholder needs**
  - ***Building Quality Software***
- **Keep Learning**
- **Lots of Testing!!!**

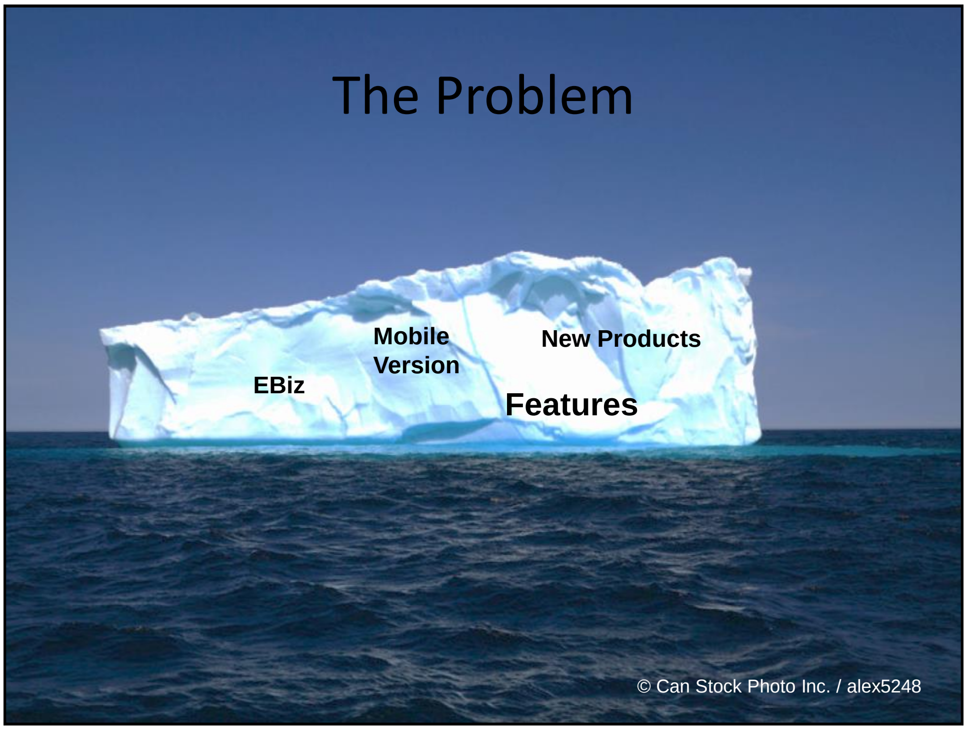
Continuous Improvement

"Retrospectives are Key!!!"

Small Steps we can take - next sprint!!!

architecture quality can be invisible

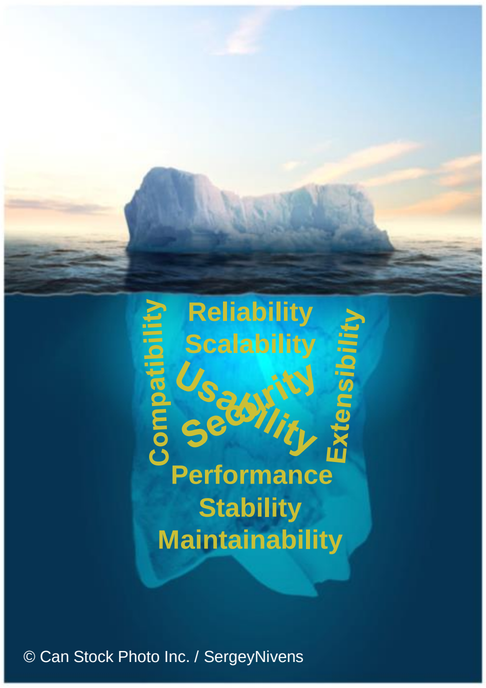
…especially when the spotlight is on



The Problem

The Solution

© Can Stock Photo Inc. / Freezingpicture



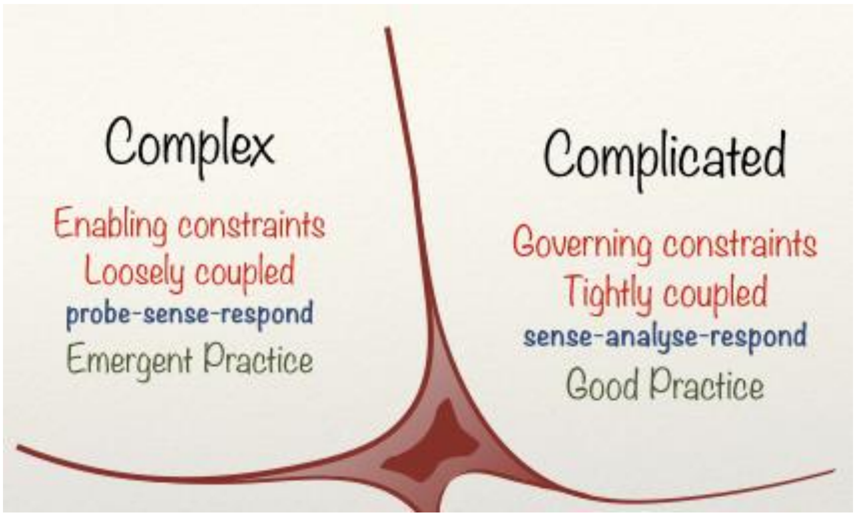
© Can Stock Photo Inc. / SergeyNivens

What's below the waterline?

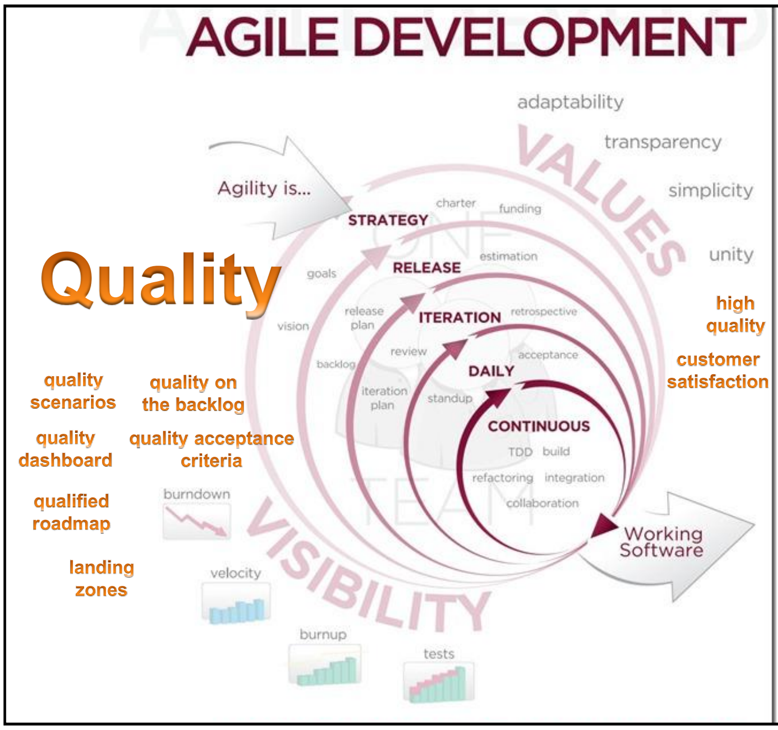
all those "ilities" we can't ignore

…

Complex vs Complicated Systems (Cynefin Framework)



"Cynefin as of 1st June 2014" by Snowded - Own work. Licensed under CC BY-SA 3.0 via Commons - https://commons.wikimedia.org/wiki/File:Cynefin_as_of_1st_June_2014.png#/media/File:Cynefin_as_of_1st_June_2014.png



Quality

quality scenarios

quality on the backlog

quality dashboard

quality acceptance criteria

qualified roadmap

landing zones

high quality

customer satisfaction

positive team

sustainable delivery

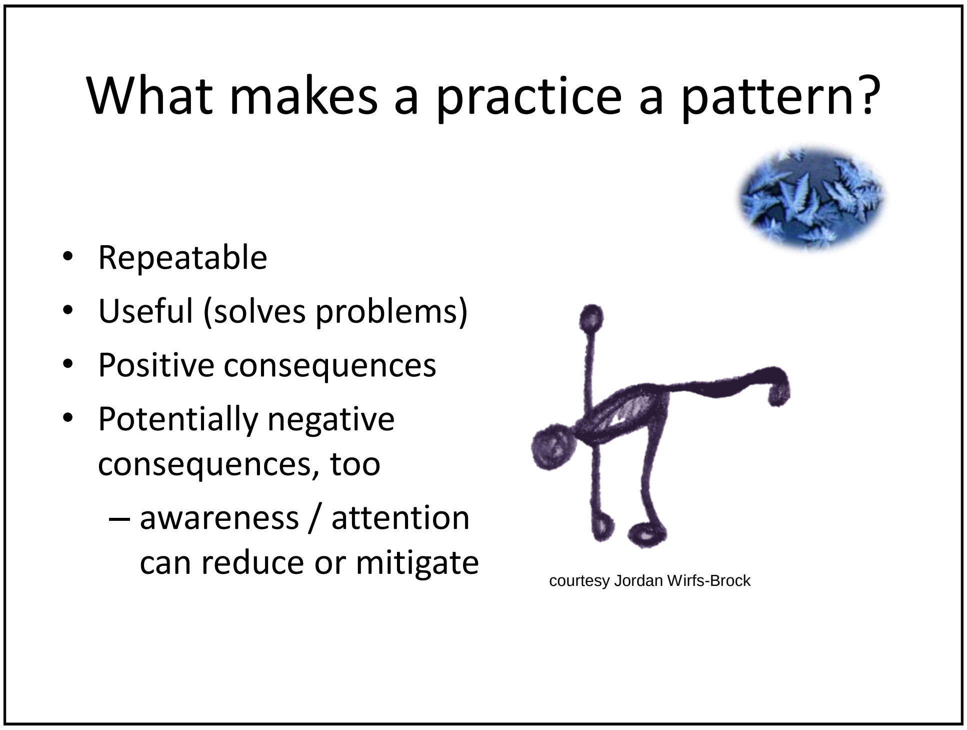
paying attention

dependability

Values Drive Practices

Practices and Patterns That Add Value

Successfully applied in different contexts



What makes a practice a pattern?

- Repeatable
- Useful (solves problems)
- Positive consequences
- Potentially negative consequences, too
  - awareness / attention can reduce or mitigate

courtesy Jordan Wirfs-Brock

BECOMING AGILE AT QUALITY

"Quality is not an act,
it is a habit…"
—Aristotle

Patterns for Being Agile at Quality

Core Patterns
- Breaking Down Barriers
- Integrate Quality

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

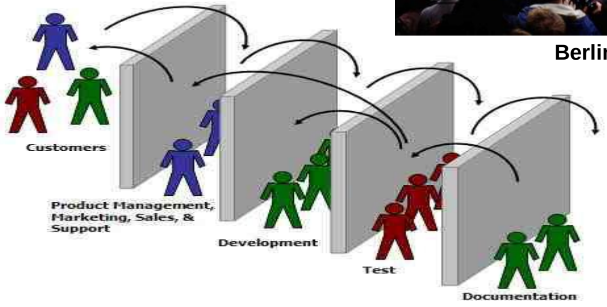
Identifying Qualities
- Finding the Qualities
- Agile Quality Scenarios
- Quality Stories
- Measureable System Qualities
- Fold-out Qualities
- Agile Landing Zone
- Recalibrate the Landing Zone
- Agree on Quality Targets

Making Qualities Visible
- System Quality Dashboard
- System Quality Radiator
- Qualify the Roadmap
- Qualify the Backlog
- Automate First
- Quality Checklists

Tearing Down the Walls
aka “Breaking Down Barriers”

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

Berlin Wall



Agile Quality Teams
“Whole Team”

- Architects and QA work closely with the product or program teams
- Whole team works at understanding, defining, delivering, and verifying system qualities

Some decisions and actions are too important to leave until The Last Responsible Moment

SO

CHOOSE THE MOST RESPONSIBLE MOMENT

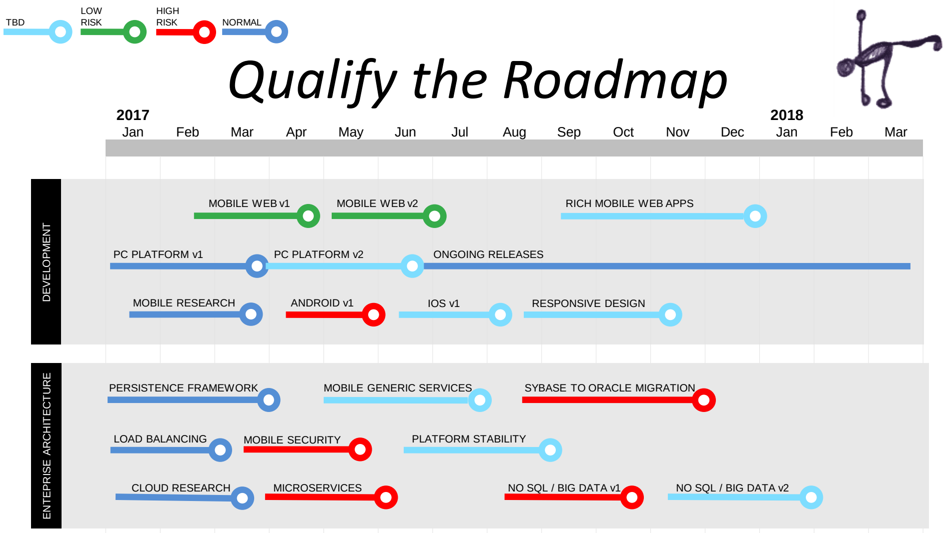
How do you

FIND RESPONSIBLE MOMENTS?

Qualify the Roadmap

"All you need is the plan, the roadmap, and the courage to press on to your destination"
— Earl Nightingale

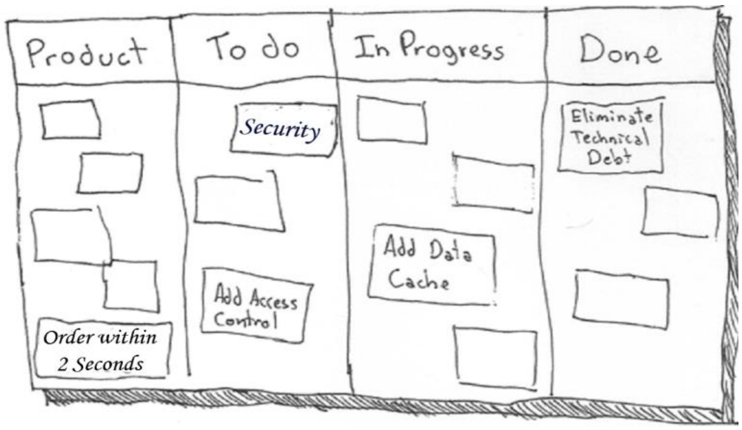
Qualify the Roadmap



DASHBOARD

DELIVERY	BUDGET	RESOURCE	ARCHITECTURE	DEPENDENCIES	RISKS	ISSUES	ON RADAR
Delays expected to Version 1	Budget will need bolstering in Q2 2017	All resources on track.	Persistence Framework Load-Balancing. Cloud	Partnerships and services all in place and on track.	**COMPETITOR** E Corp – new product.	**DELIVERY** Tech issues	**AUG 2017** New mobile opportunity
					ARCHITECTURE Performance Platform stability	**ARCHITECTURE** Migration Security	**OCT 2017** Re-evaluate NO SQL strategy

Qualify the Backlog



You can add backlog items for quality scenarios,
system quality-related architecture work… yes, you can

Make Architecture Work Visible and Explicit

	Visible	Invisible
Positive Value	Visible Feature	Invisible Architectural Feature
Negative Value	Visible Defect	Technical Debt

Color your backlog—Phillipe Kruchten

http://philippe.kruchten.com/2013/12/11/the-missing-value-of-software-architecture/

Fold-out Qualities

Quality-related acceptance criteria attached to user stories

"As a customer I want to place an order using my credit card…."

Security: Use 256 bit SSL encryption….

Security: Is credit information retained? Do I have control over this?

Performance: How fast can I place an order and receive confirmation?

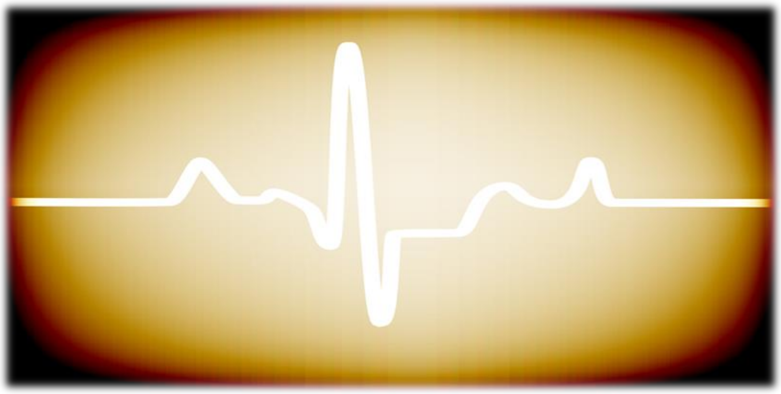
Performance: Order time < 2 seconds

Usability: Can I cancel my order? When?

"Acceptable means done with quality"

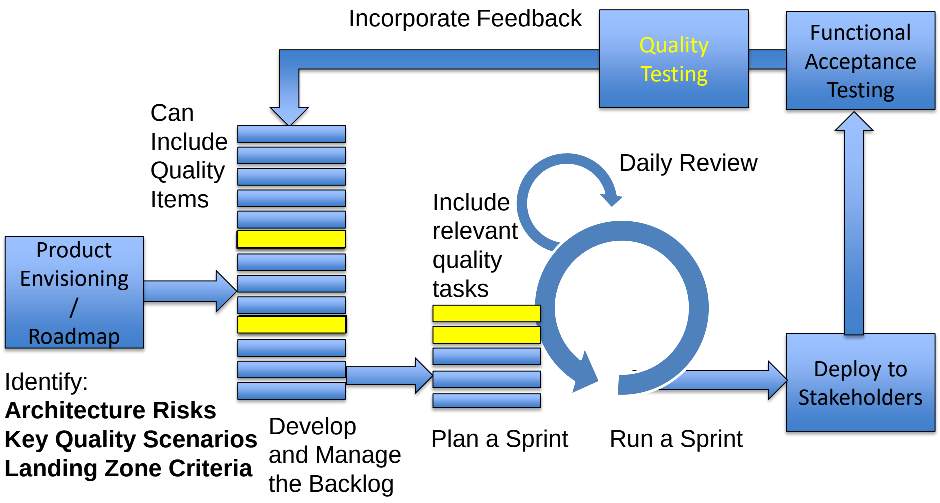
HOW SYSTEM QUALITY WORK CAN FIT INTO YOUR RHYTHMS

Build architectural quality into your project rhythms



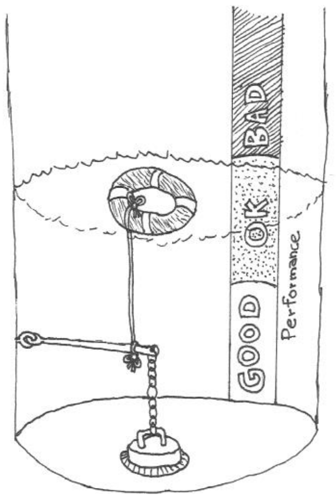
"QUALITY IS NOT AN ACT, IT IS A HABIT."
—ARISTOTLE

How Quality Fits Into An Agile Process



Define Architecture Triggers

- Conditions that cause architecture investigation/ tasks
  - Quality target no longer met
  - Code quality metrics violations
  - …
- Have broad system impact

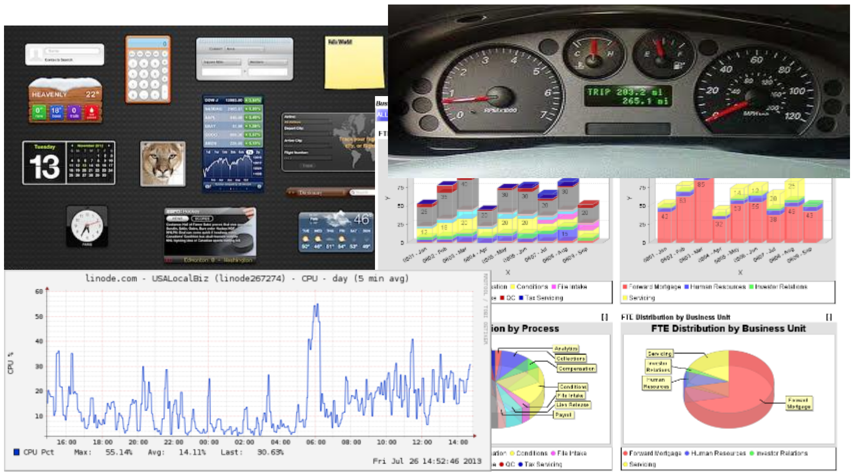


Architecture Spikes & Explorations

- Answer deep questions / offers potential architecture solutions
- Not as tactical as an XP Design Spike
- Visible and bounded

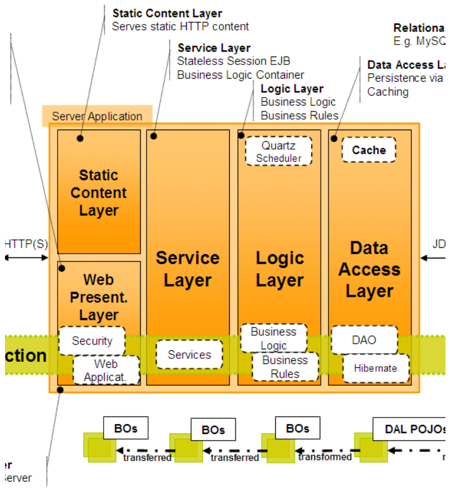
ONGOING QUALITY ACTIVITIES

Monitor System Qualities—Build An Operational Dashboard



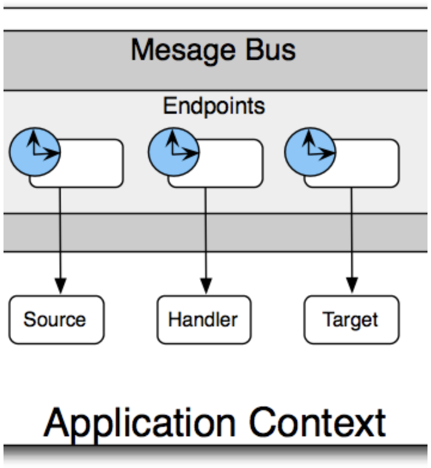
Incrementally Test Key Components' Performance

- Identify key pathways and critical components
- Test components as they arrive to access performance
- Use mocks, stubs, and auto-responders to simulate missing components



Test Infrastructure To Verify Architecture Assumptions

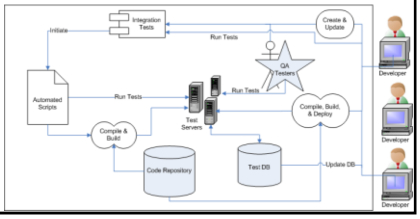
- Benchmark early, then track
- Example:
  - Push/pull response times
  - Msg creation rates with >1 publisher
  - Consumption rates
  - Effects of adding msg dispatchers



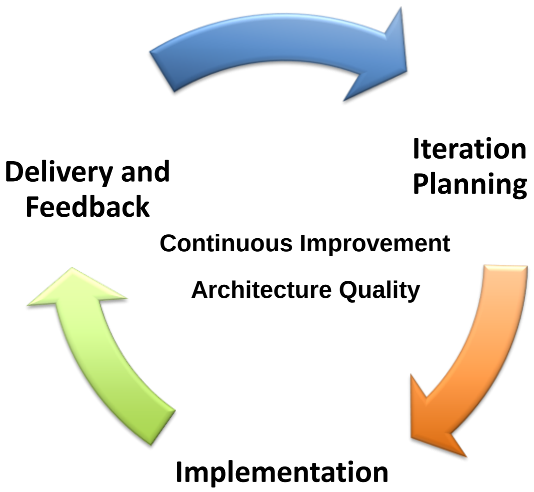
Example: Message Bus Performance

Testing Overall System Qualities

- Some are "easy" and can be part of a frequently run automated quality test suite
- Some require "extensive" setup
- Some require near-production environments
  - Load and performance tests
  - Complex quality scenarios involving interactions with several systems/services



Periodically Re-Evaluate Architecture Risks

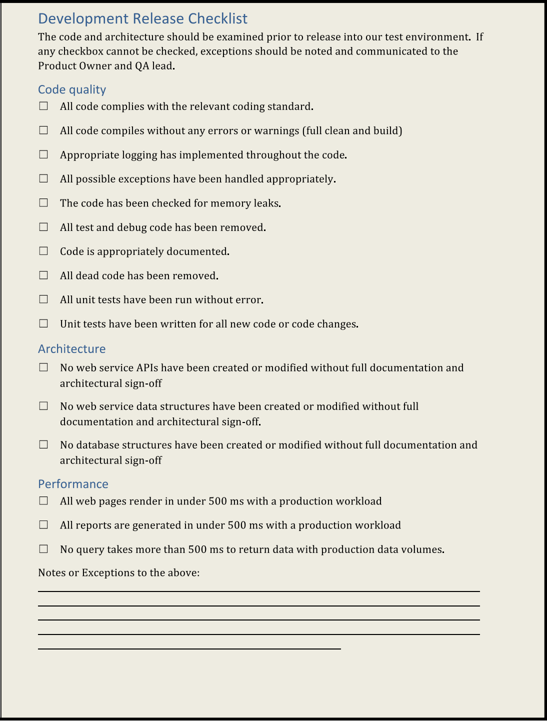


PAUSE POINTS HELP EVALUATE RISK

Quality Focused Checklists

- Release Checklists*
  - Agreed upon checklist for quality and major architecture concerns
- Use at pause points
  - sprint planning, release planning, …

*Thanks, James Thorpe for sharing your company's checklist



Development Release Checklist

The code and architecture should be examined prior to release into our test environment. If any checkbox cannot be checked, exceptions should be noted and communicated to the Product Owner and QA lead.

Code quality

- ☐ All code complies with the relevant coding standard.
- ☐ All code compiles without any errors or warnings (full clean and build)
- ☐ Appropriate logging has implemented throughout the code.
- ☐ All possible exceptions have been handled appropriately.
- ☐ The code has been checked for memory leaks.
- ☐ All test and debug code has been removed.
- ☐ Code is appropriately documented.
- ☐ All dead code has been removed.
- ☐ All unit tests have been run without error.
- ☐ Unit tests have been written for all new code or code changes.

Architecture

- ☐ No web service APIs have been created or modified without full documentation and architectural sign-off
- ☐ No web service data structures have been created or modified without full documentation and architectural sign-off.
- ☐ No database structures have been created or modified without full documentation and architectural sign-off

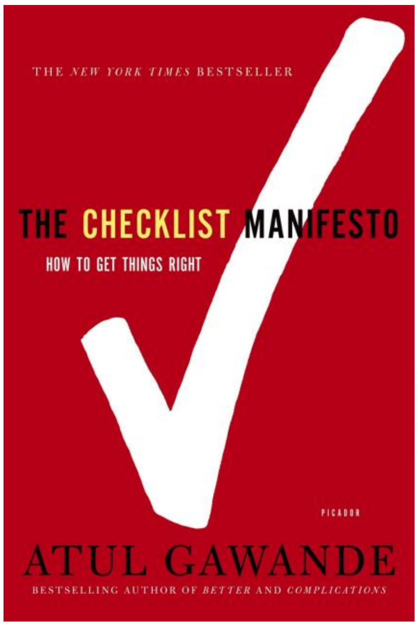
Performance

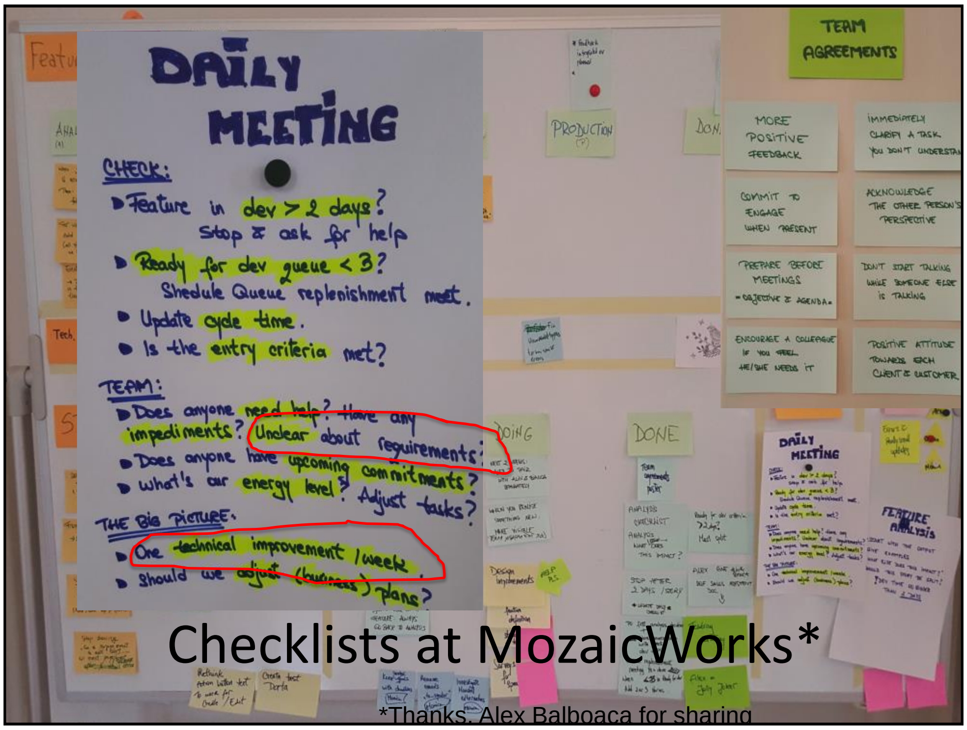
- ☐ All web pages render in under 500 ms with a production workload
- ☐ All reports are generated in under 500 ms with a production workload
- ☐ No query takes more than 500 ms to return data with production data volumes.

Notes or Exceptions to the above:

Two Kinds of Checklists

1. Read-review
2. Do-confirm


THE NEW YORK TIMES BESTSELLER
THE CHECKLIST MANIFESTO
HOW TO GET THINGS RIGHT
PICADOR
ATUL GAWANDE
BESTSELLING AUTHOR OF BETTER AND COMPLICATIONS


DAILY MEETING

CHECK:
- Feature in dev > 2 days? Stop & ask for help
- Ready for dev queue < 3? Shedule Queue replenishment meet.
- Update cycle time.
- Is the entry criteria met?

TEAM:
- Does anyone need help? Have any impediments? Unclear about requirements?
- Does anyone have upcoming commitments?
- what's our energy level? Adjust tasks?

THE BIG PICTURE:
- One technical improvement / week
- Should we adjust (business) plans?

TEAM AGREEMENTS
- MORE POSITIVE FEEDBACK
- IMMEDIATELY CLARIFY A TASK YOU DON'T UNDERSTAND
- COMMIT TO ENGAGE WHEN PRESENT
- ACKNOWLEDGE THE OTHER PERSON'S PERSPECTIVE
- PREPARE BEFORE MEETINGS - OBJECTIVE & AGENDA.
- DON'T START TALKING WHILE SOMEONE ELSE IS TALKING
- ENCOURAGE A COLLEAGUE IF YOU FEEL HE/SHE NEEDS IT
- POSITIVE ATTITUDE TOWARDS EACH CLIENT & CUSTOMER

Checklists at MozaicWorks*

*Thanks, Alex Balboaca for sharing

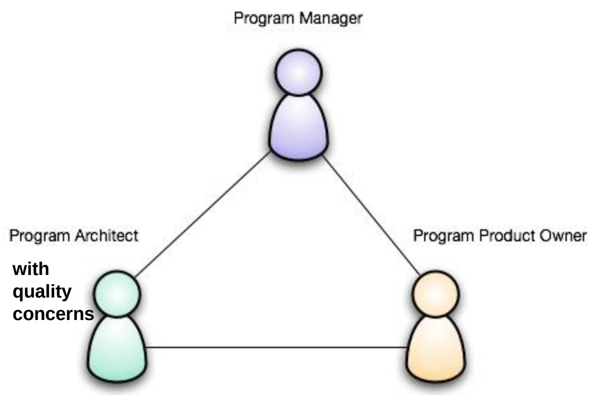
ROLES AND WHOLE TEAM DEDICATION

Who will lead? Who contributes?

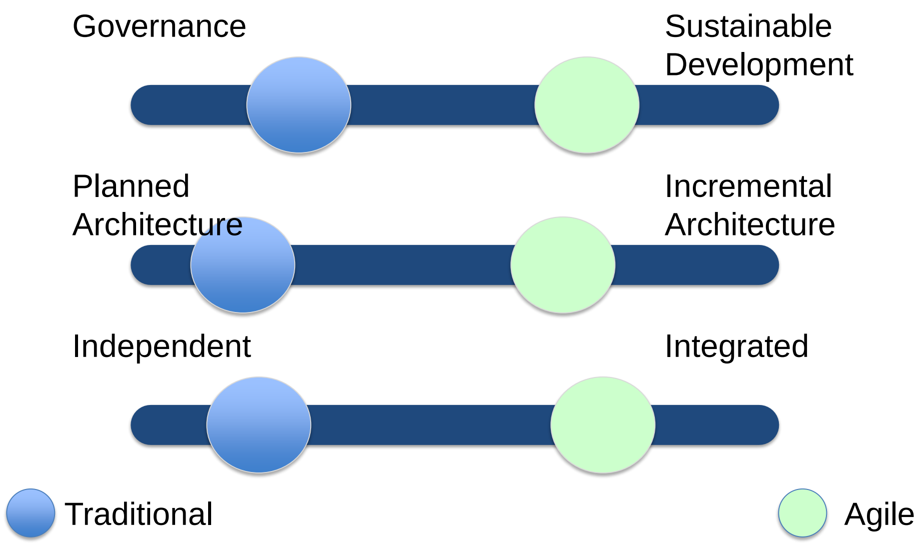
- Big teams vs. small teams????

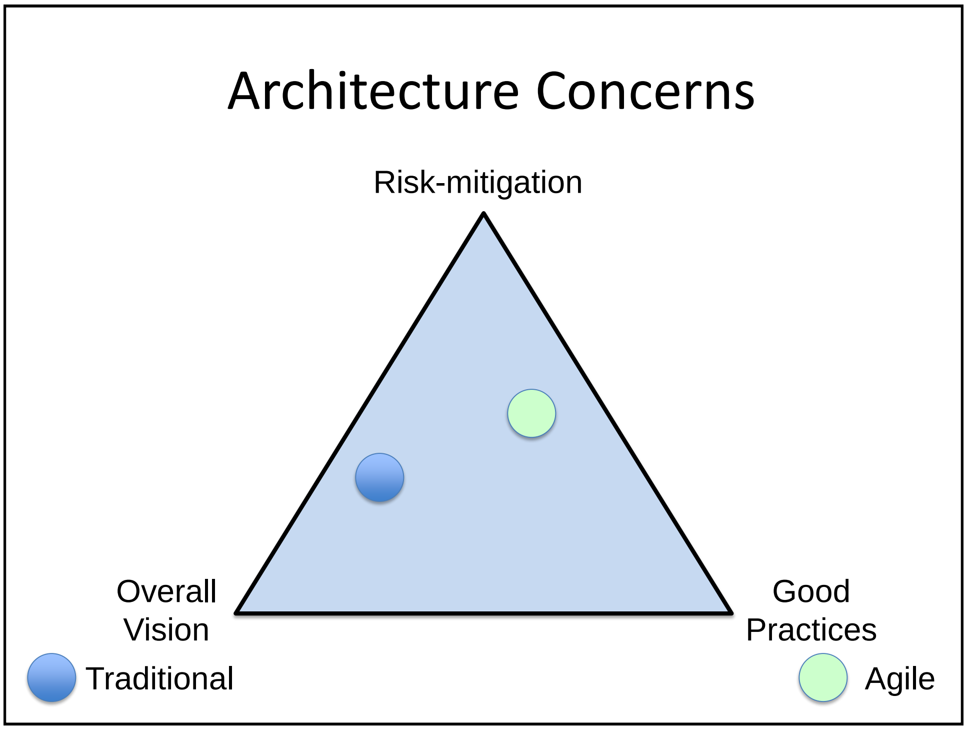
- Does system quality get the attention it needs?

How Product and Program Management and Architects Interact

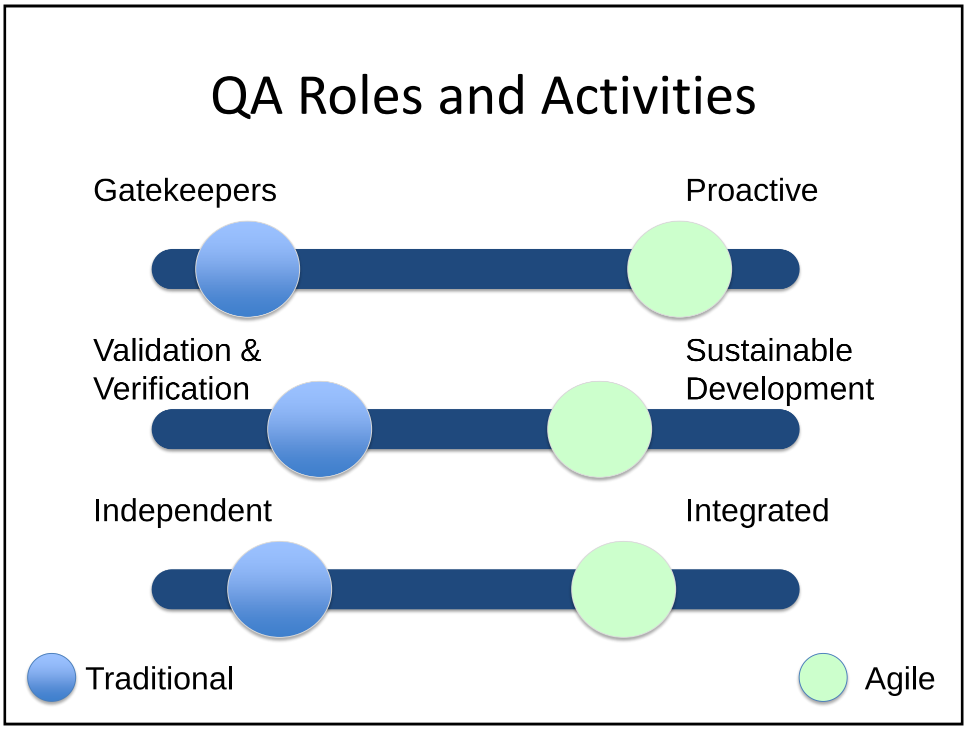


Architecture Roles and Activities



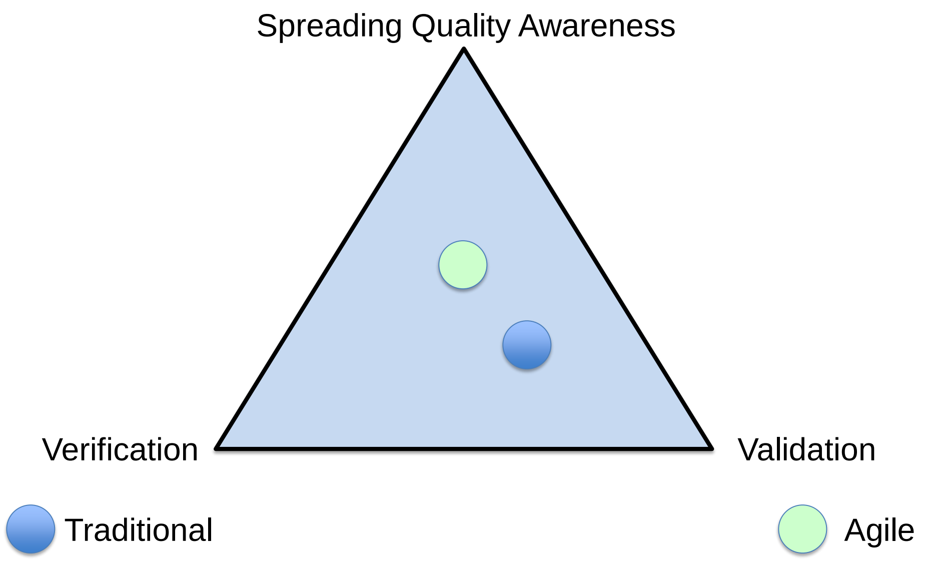


Architecture Concerns



QA Roles and Activities

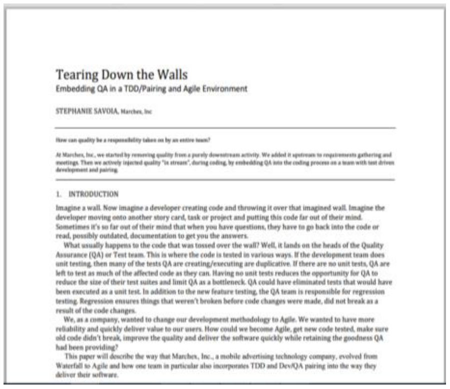
QA Focus



Embedding QA with Team

aka "Pair with a Quality Advocate"

Great experience report at Agile 2014



Tearing Down the Walls

Embedding QA in a TDD/Pairing and Agile Environment

STEPHANIE SAVOIA, Marchex, Inc

How can quality be a responsibility taken on by an entire team?

At Marchex, Inc., we started by removing quality from a purely downstream activity. We added it upstream to requirements gathering and meetings. Then we actively injected quality "in stream", during coding, by embedding QA into the coding process on a team with test driven development and pairing.

1. INTRODUCTION

Imagine a wall. Now imagine a developer creating code and throwing it over that imagined wall. Imagine the developer moving onto another story card, task or project and putting this code far out of their mind. Sometimes it's so far out of their mind that when you have questions, they have to go back into the code or read, possibly outdated, documentation to get you the answers.

What usually happens to the code that was tossed over the wall? Well, it lands on the heads of the Quality Assurance (QA) or Test team. This is where the code is tested in various ways. If the development team does unit testing, then many of the tests QA are creating/executing are duplicative. If there are no unit tests, QA are left to test as much of the affected code as they can. Having no unit tests reduces the opportunity for QA to reduce the size of their test suites and limit QA as a bottleneck. QA could have eliminated tests that would have been executed as a unit test. In addition to the new feature testing, the QA team is responsible for regression testing. Regression ensures things that weren't broken before code changes were made, did not break as a result of the code changes.

We, as a company, wanted to change our development methodology to Agile. We wanted to have more reliability and quickly deliver value to our users. How could we become Agile, get new code tested, make sure old code didn't break, improve the quality and deliver the software quickly while retaining the goodness QA had been providing?

This paper will describe the way that Marchex, Inc., a mobile advertising technology company, evolved from Waterfall to Agile and how one team in particular also incorporates TDD and Dev/QA pairing into the way they deliver their software.

AgileAlliance.org

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

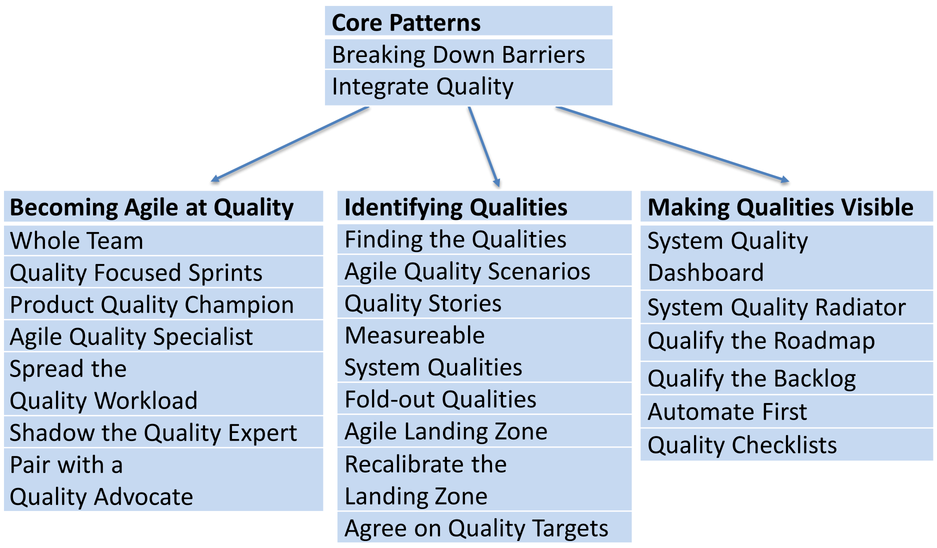
Shadow the Quality Expert

aka “Spread the Quality Expertise”

“Tell me and I forget, teach me and I remember, involve me and I learn” — Benjamin Franklin

As organizations grow, need to grow and evolve quality expertise …
Many organizations lack the resources fulfill their Quality needs …

Patterns for Being Agile at Quality

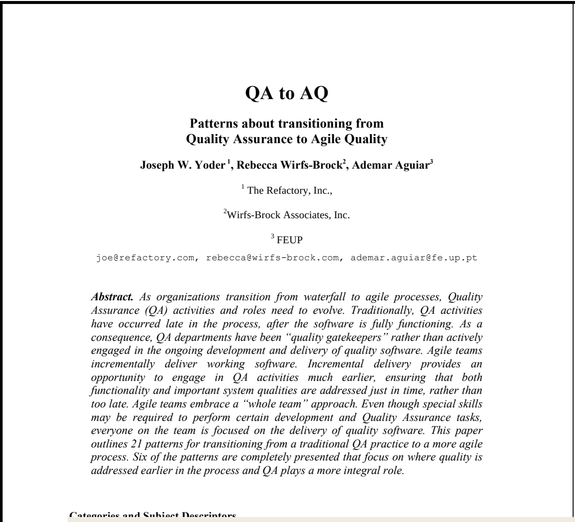


Core Patterns
- Breaking Down Barriers
- Integrate Quality

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

Identifying Qualities
- Finding the Qualities
- Agile Quality Scenarios
- Quality Stories
- Measureable System 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
- Automate First
- Quality Checklists



QA to AQ

Patterns about transitioning from Quality Assurance to Agile Quality

Joseph W. Yoder[1], Rebecca Wirfs-Brock[2], Ademar Aguiar[3]

[1] The Refactory, Inc.,

[2]Wirfs-Brock Associates, Inc.

[3] FEUP

joe@refactory.com, rebecca@wirfs-brock.com, ademar.aguiar@fe.up.pt

Abstract. As organizations transition from waterfall to agile processes, Quality Assurance (QA) activities and roles need to evolve. Traditionally, QA activities have occurred late in the process, after the software is fully functioning. As a consequence, QA departments have been "quality gatekeepers" rather than actively engaged in the ongoing development and delivery of quality software. Agile teams incrementally deliver working software. Incremental delivery provides an opportunity to engage in QA activities much earlier, ensuring that both functionality and important system qualities are addressed just in time, rather than too late. Agile teams embrace a "whole team" approach. Even though special skills may be required to perform certain development and Quality Assurance tasks, everyone on the team is focused on the delivery of quality software. This paper outlines 21 patterns for transitioning from a traditional QA practice to a more agile process. Six of the patterns are completely presented that focus on where quality is addressed earlier in the process and QA plays a more integral role.

QA to AQ: Patterns about transitioning from Quality Assurance to Agile Quality, AsianPLoP 2014

QA to AQ Part Two: Shifting from Quality Assurance to Agile Quality, PLoP 2014

QA to AQ Part Three: Shifting from Quality Assurance to Agile Quality "Tearing Down the Walls", SugarLoafPLoP 2014

QA to AQ Part Four: Shifting from Quality Assurance to Agile Quality "Prioritizing Qualities and Making them Visible", PLoP 2015

QA to AQ Part Five: Being Agile At Quality "Growing Quality Awareness and Expertise", AsianPLoP 2016

QA to AQ Part Six: Shifting from Quality Assurance to Agile Quality "Enabling and Infusing Quality", To appear at PLoP 2016

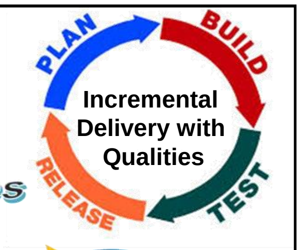
Continuous Inspection: A Pattern for Keeping your Code Healthy and Aligned to the Architecture, AsianPLoP 2014

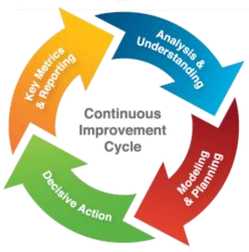
Patterns to Develop and Evolve Architecture in an Agile Project, PLoP 2016

…PATTERNS FOR TRANSITIONING FROM TRADITIONAL TO AGILE QA AND AGILE ARCHITECTURE

Copies available off our websites.

Roles & Responsibilities



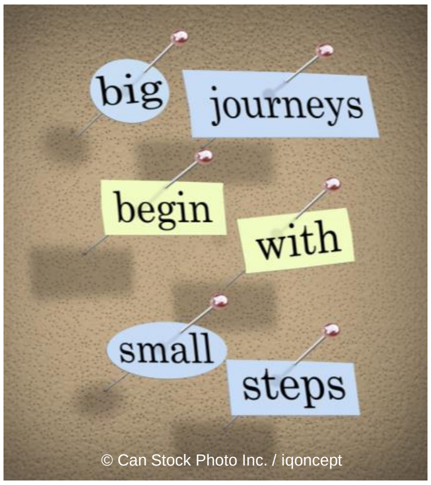


OUR QUALITY VALUES CALL TO ACTION

(CC) by muffinn on Flickr

Where do you start?

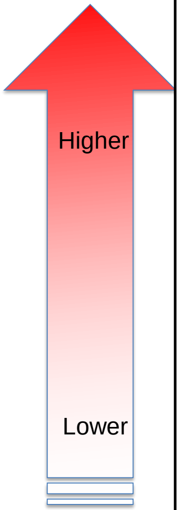
- Monitor qualities
- Pick some low hanging fruit
  - Make goals visible
  - Colorize your backlog
  - Create quality-related checklists
- Spread attention to system quality throughout teams
- Depends on where you are and where the pain is…

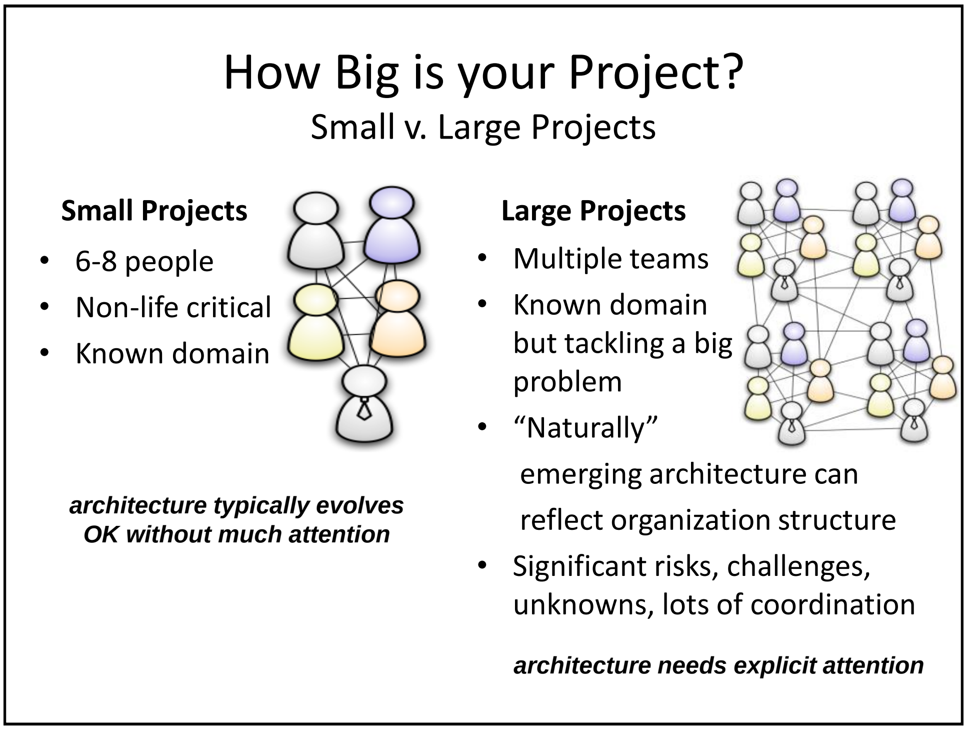


How Much Architecture Risk do you Have?

- New architecture, new product, new market, new technologies
- Transforming an existing product
- Evolving a product
- Feature extensions on a "stable" architecture

"the more risk, the more attention you need to pay to architecture"





How Big is your Project?

Small v. Large Projects

Small Projects

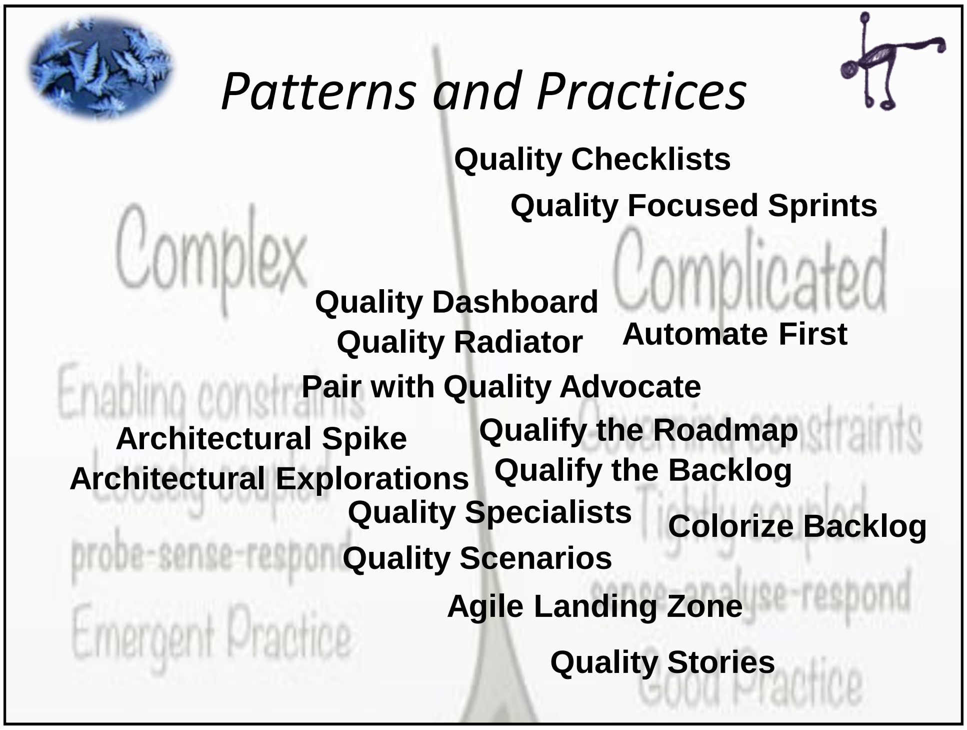
- 6-8 people
- Non-life critical
- Known domain

architecture typically evolves OK without much attention

Large Projects

- Multiple teams
- Known domain but tackling a big problem
- "Naturally" emerging architecture can reflect organization structure
- Significant risks, challenges, unknowns, lots of coordination

architecture needs explicit attention



Patterns and Practices

Complex
Complicated
Enabling constraints
Governing constraints
Loosely coupled
Tightly coupled
probe-sense-respond
sense-analyse-respond
Emergent Practice
Good Practice

Quality Checklists

Quality Focused Sprints

Quality Dashboard

Quality Radiator

Automate First

Pair with Quality Advocate

Architectural Spike

Qualify the Roadmap

Architectural Explorations

Qualify the Backlog

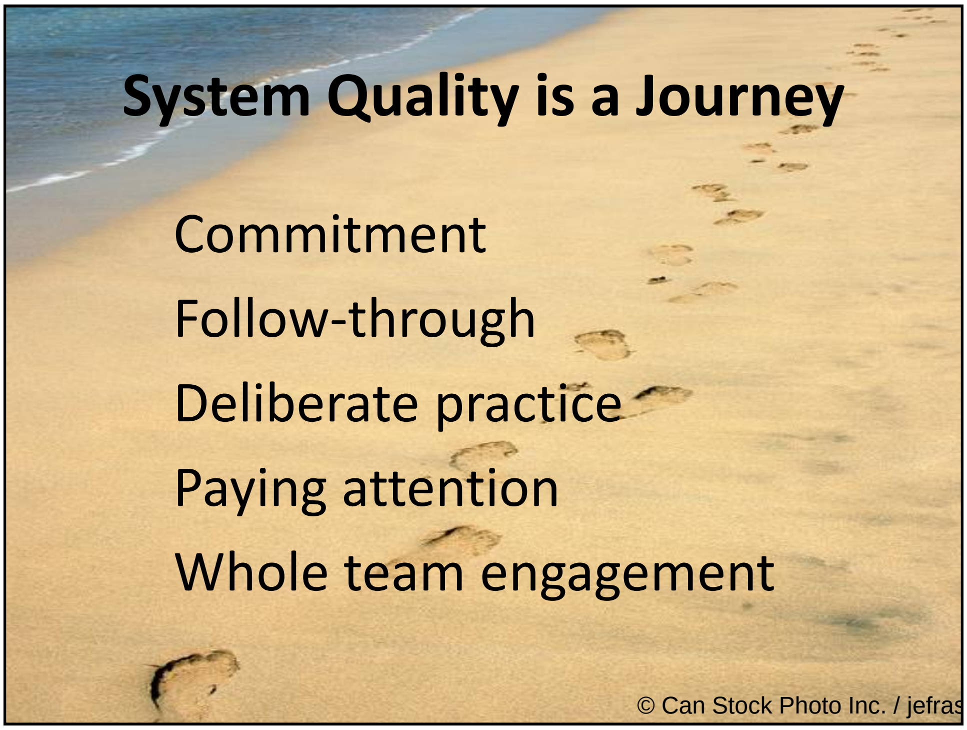
Quality Specialists

Colorize Backlog

Quality Scenarios

Agile Landing Zone

Quality Stories



System Quality is a Journey

- Commitment
- Follow-through
- Deliberate practice
- Paying attention
- Whole team engagement

Additional Resources

- The Hillside Group (patterns community): Hillside.net
- Being Agile at System Qualities workshop:
  - www.adaptiveobjectmodel.com/2015/04/qa-to-aq-shifting-towards-agile-quality
- Agile Myths: agilemyths.com
- The Refactory (www.refactory.com)
- Teams That Innovate (www.teamsthatinnovate.com)
- Pragmatic TDD :
  refactory.com/training/test-driven-development
  http://adaptiveobjectmodel.com/2012/01/what-is-pragmatic-tdd

joe@refactory.com
Twitter: @metayoda

www.joeyoder.com
www.refactory.com