QA to AQ: Shifting towards Agile Quality

Joseph Yoder
www.refactory.com
www.teamsthatinnovate.com

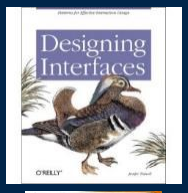
Who are your system's Quality Stakeholders?

- Users of the system
- Executives / System sponsors
- Internal quality stakeholders:
  - Developers
  - Database admins
  - Business process experts
  - Corporate compliance guys

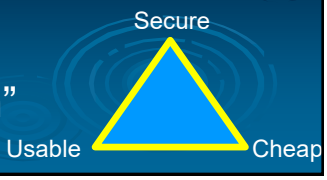
Artist important/boring	Scientist true/false
Designer cool/uncool	Engineer good/bad

Design is about Tradeoffs

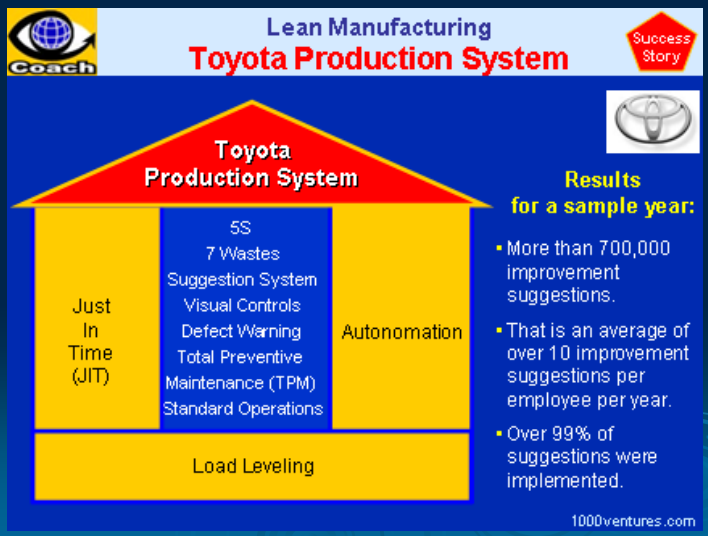
- Usability and Security often have *orthogonal* qualities
  - **Designing Interfaces: Patterns for Effective Interaction Design**
  - **Security Patterns: Integrating Security and Systems Engineering**



- Performance vs Small Memory
- Quality of being "Good Enough"



Lean by Toyota (TPS)



Lean Development

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

A ***lean*** organization ***understands*** customer ***value*** and focuses its core processes to continuously increase it

Ultimate goal is to provide ***perfect value*** to customer and business

Just in Time Practice – don't do something before you need it

Continuous Improvement

"*Retrospectives are Key!!!*"

1) What worked well that we do not want to forget for future iterations?
2) What should we do differently?

*there should be **no finger pointing**…*
do not focus on negative things

3) What still puzzles us?

Things we can't answer

4) What did you learn?
5) Hopes for the next iteration/release

Many Agile Teams "**only**" Focus on Functional Requirements

Functional:

- How do I …?
- Validate user stories work as advertised
  - "As a reviewer I want to add a note to a chart"
  - Compute the charge for an invoice
- Validate boundary conditions
  - Can I add more than one note at the same place?
  - Are excess charges computed correctly?

Non-functional Requirements

Accessibility	Reliability
Compatibility	Safety
Efficiency	Scalability
Effectiveness	Security
Extensibility	Stability
Maintainability	Supportability
Performance	Usability

Other terms for *non*-functional requirements: "constraints", "quality attributes", and "quality of service requirements"

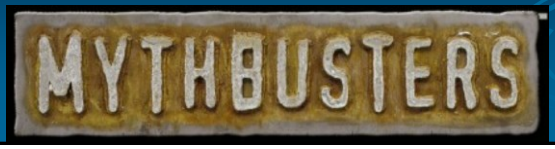
Qualities are usually described by "ilities" as seen in *non*-functional requirements…but quality can also focus on how well functional requirements are met

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…



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
- Monitoring Qualities
- Agile QA Tester
- Spread the Quality Workload
- Shadow the Quality Expert
- Pair with a Quality Advocate

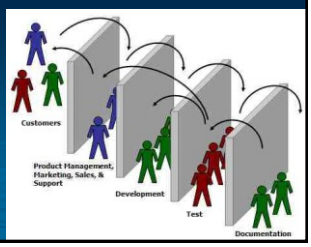
Identifying Qualities
- Finding the Qualities
- Agile Quality Scenarios
- Quality Stories
- Measureable System Qualities
- Fold-out Qualities
- Agile Landing Zone
- Recalibrate the Landing Zone
- Agree on Quality Targets

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

- ***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***



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

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…

While Re

chite ts and QA works closely with the tea
ntegrated during the day to day develo m nt

ngage in Architecture & QA activities much earli

orks with whole-team including product owner o
nderstanding and defining qualities

Assists teams with definition & validation of important quality requirements

Proactive working closely & coordinates between business, management and developers

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 (SLAs …)

…get involved late in the software lifecycle

Agile QA

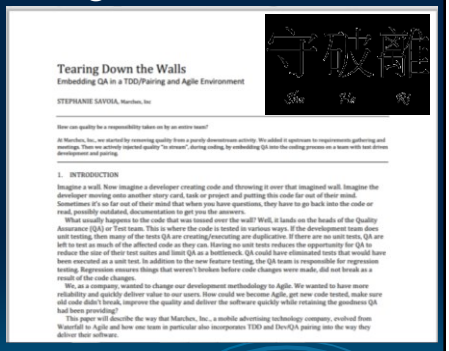
Integrated with day to day development

Proactive, engage in QA activities much earlier

Work closely with business, product owner, architects and devs to understand, define, and validate quality requirements

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

SEE WHAT'S NEW ON OUR WEBSITE

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

Build quality into your project rhythms

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

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

so

CHOOSE THE MOST RESPONSIBLE MOMENT

Qualify the Roadmap

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

- **As systems qualities are a key factor in the success of any product, how can agile teams include these qualities as part of the roadmap and overall timeline?**
- **While developing and evolving the product feature roadmap, also plan for when system qualities should be addressed. Be sure to "*Plan for Responsible Moments*" to know when important system qualities should be implemented**

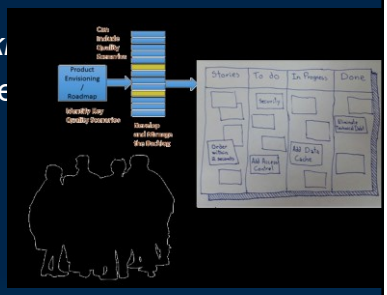
Find Essential Qualities

System qualities are often overlooked or simplified until late in the development process causing delays and extensive refactoring and rework

- **How can agile teams understand and prioritize essential qualities for an evolving system?**
- **Have team meetings at opportune times throughout the development process with stakeholders to brainstorm the important qualities to be considered for the system**

Qualify the Backlog

"Things come to me pretty regularly
There is never a shortage or a back
— Duncan She



- **How can agile developers better understand the scope of the work that needs to be done, especially when it comes to understanding, implementing and testing system qualities*?***
- **Create and add specific quality items to the backlog, these items can include *Quality Scenarios*, *Quality Stories*, and *Fold-Out Qualities***

Quality Checklists

What are the expectations for system quality? Different for new functionality but probably known for "standard" functionality that is familiar!

- **It is hard to ensure quality is being met if you do not know the expectations**
- **Develop a checklist that includes expectations for desired system qualities that are common across the system and should be consistently met. This checklist can be reviewed by the team to ensure that qualities are met before features are released and verified by the team as part of quality assurance**

Qualify Checklist Example

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 been implemented throughout the code

- ✓ ☐ All possible exceptions have been handled appropriately
- ✓ ☐ The code has been checked for memory leaks
- ✓ ☐ All dead code has been removed
- ✓ ☐ All unit tests have been run without error
- ✓ ☐ Unit tests have been written for all new code

Qualify Checklist Example

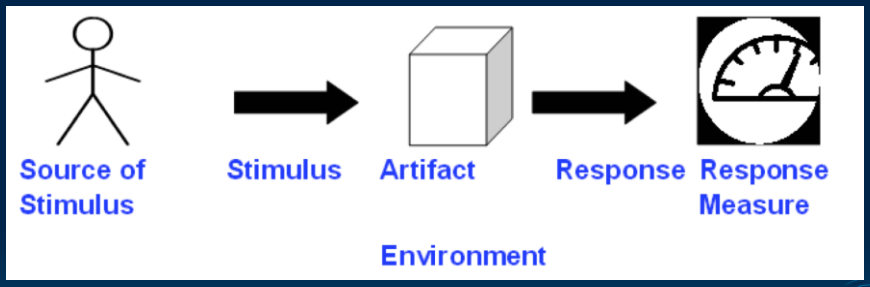
Performance

- ✓ ☐ 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 stored procedure or query takes more than 500 ms to return data

Database

- ➢ ☐ No temporary tables are created explicitly or implicitly in stored procedures or database queries
- ➢ ☐ There are no hard coded constants in any stored procedures, no passwords stored unencrypted, etc.….

Quality Scenarios, Quality Stories, and Fold-Out Qualities



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

Finding Even More Qualities: Fold-out Qualities

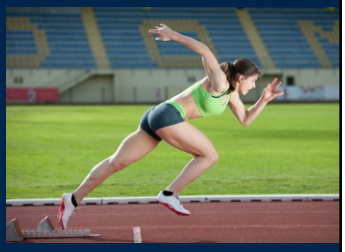
"As a customer I want to to place an order using my credit card."

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?
- …

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

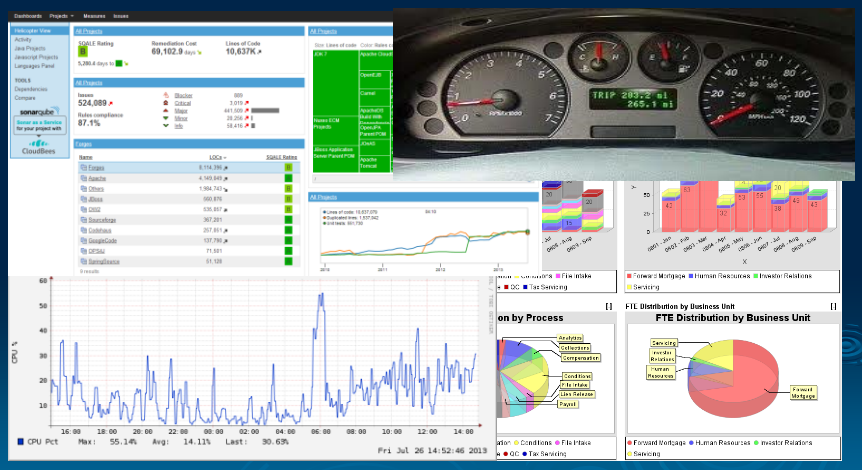


- ***How can you incorporate these other non-functional requirements into your system as needed?***
- ***Therefore, take time to focus on your software's non-functional qualities and devote a sprint to measuring and improving one or more of your system's qualities***

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



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
- 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
- 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
- Quality Checklists

Potential Patterns

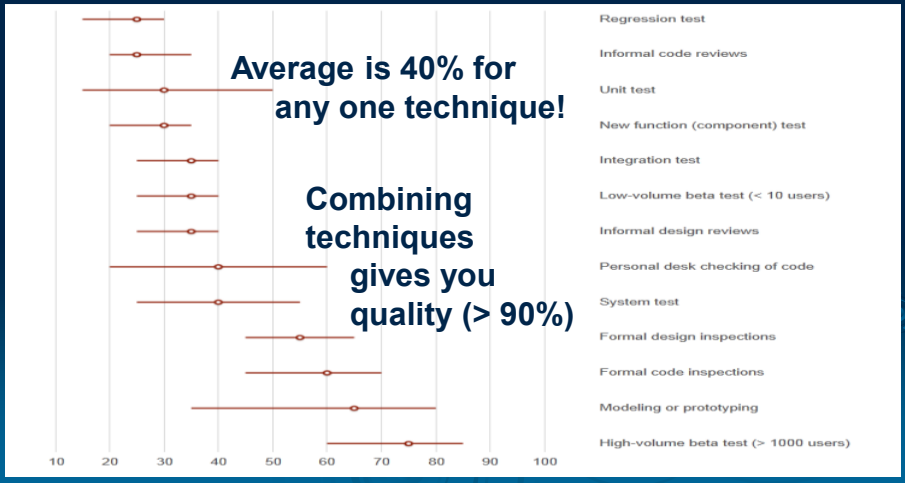
- Exploit Your Strengths
- Value Quality
- Everyone has QA responsibilities
- Grow the Team
- Architecture Runway
- Quality Debt related to Technical Debt
- Define Quality Acceptance Criteria
- Making Quality Debt Visible and How to Manage
- Getting the Agile Mindset
- Experiment to Learn
- Responsible Moments
- Continuous Inspection
- Quality Risk Assessment
- Quality Tests
- Automate First
- Share the Quality Load

Becoming Agile at Quality

- 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
- ***Deliver System Quality while Delivering System Functionality***
- 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"

Other Techniques for Improving Quality

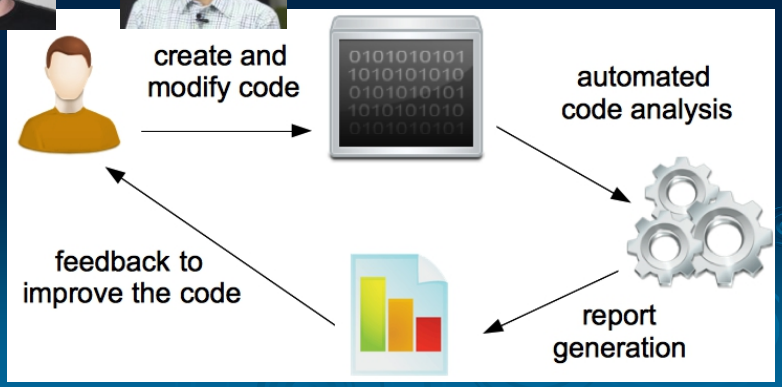
Steve McConnell
http://kev.inburke.com/kevin/the-best-ways-to-find-bugs-in-your-code/



Continuous Inspection

Asian PLoP 2014 Paper



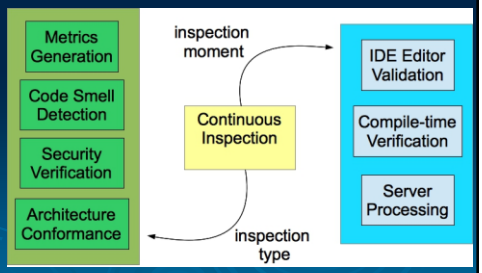


Continuous Inspection

Asian PLoP 2014 Paper

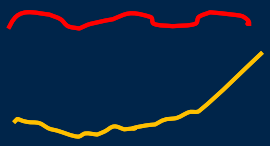
- *Code Smell Detection*
- *Metrics (Test Coverage, Cyclomatic complexity, Technical Debt, Sizes, …)*
- *Application Security Checks*
- *Architectural Conformance*

Automate where you can!!!

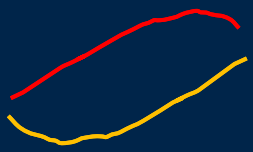


Agile Quality Benefits

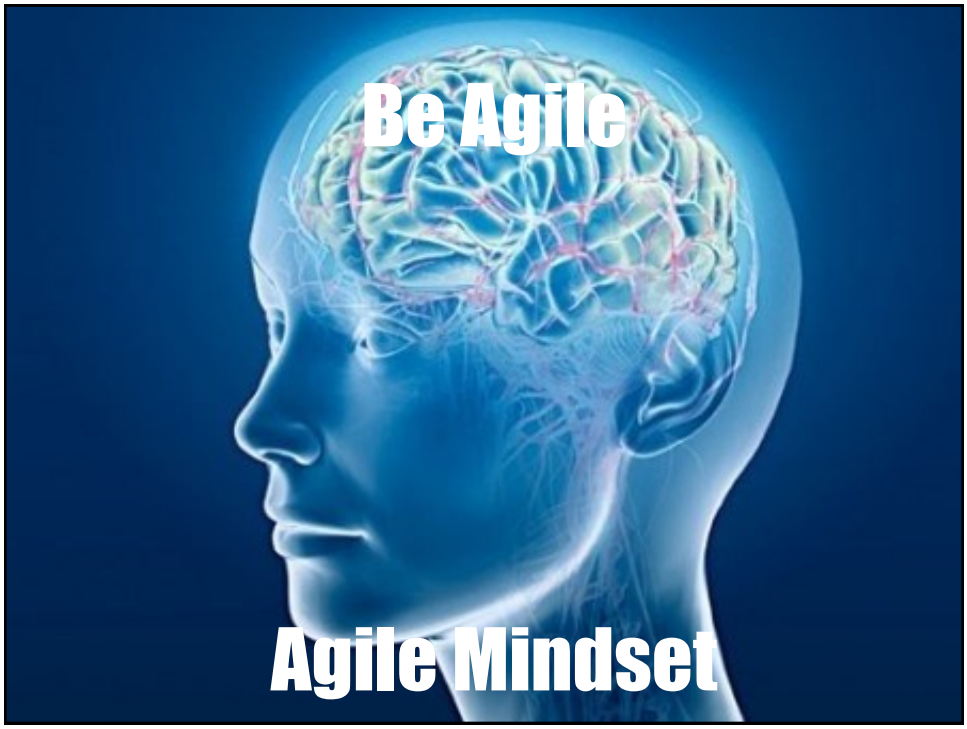
Visibility



Effectiveness



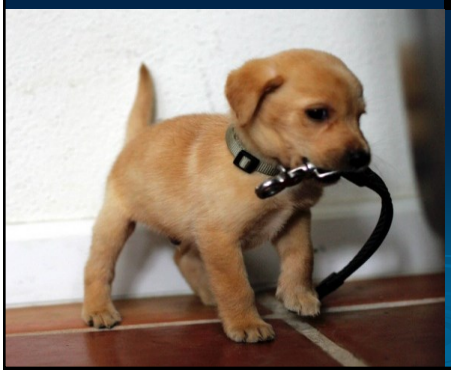
Architectural Risk



Dogmatic

Synonyms: **bullheaded, dictative**, **doctrinaire**, **fanatical**, **intolerant**

Antonyms: amenable, flexible, manageable

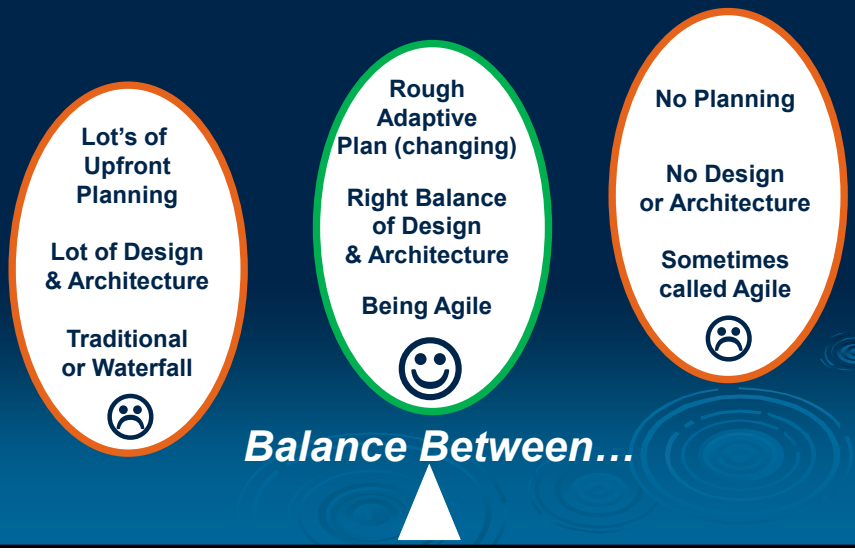


Pragmatic

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

Antonyms: idealistic, unrealistic

Being Pragmatic



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"

Resources

- Agile Myths: agilemyths.com
- The Refactory: www.refactory.com
- Joe's website: joeyoder.com
- TeamsThatInnovate: (www.teamsthatinnovate.com)
- Pragmatic TDD :
  - refactory.com/training/test-driven-development
  - http://adaptiveobjectmodel.com/2012/01/what-is-pragmatic-tdd/
- Being Agile at Quality blog:
  - www.adaptiveobjectmodel.com/2015/04/qa-to-aq-shifting-towards-agile-quality

Arigatou Gozaimasu!!!

joe@refactory.com
Twitter: @metayoda

Interactive Session

Whole Team

Agile Landing Zones

Quality Scenarios

Quality Radiator

LET'S SEE HOW THESE PATTERNS RELATE …