

Refactory

Being Agile About Architecture

Joseph W. Yoder
The Refactory

Twitter: @metayoda
joe@refactory.com
<http://www.refactory.com>

Copyright 2021 Joseph Yoder & The Refactory, Inc.



JOE YODER

teaches & mentors
on agile/lean practices,
architecture, flexible systems,
clean design, patterns, refactoring,
& testing.



www.refactory.com
joe@joeyoder.com

Disclaimer

The **views** and **opinions** expressed in this talk are **mine** and **do not** necessarily **reflect** the official **policy** or position of any organization I might reference. **My opinions are my own!** Any **examples** or numbers **discussed** are used as examples only and **do not** necessarily **represent** real **scenarios**.

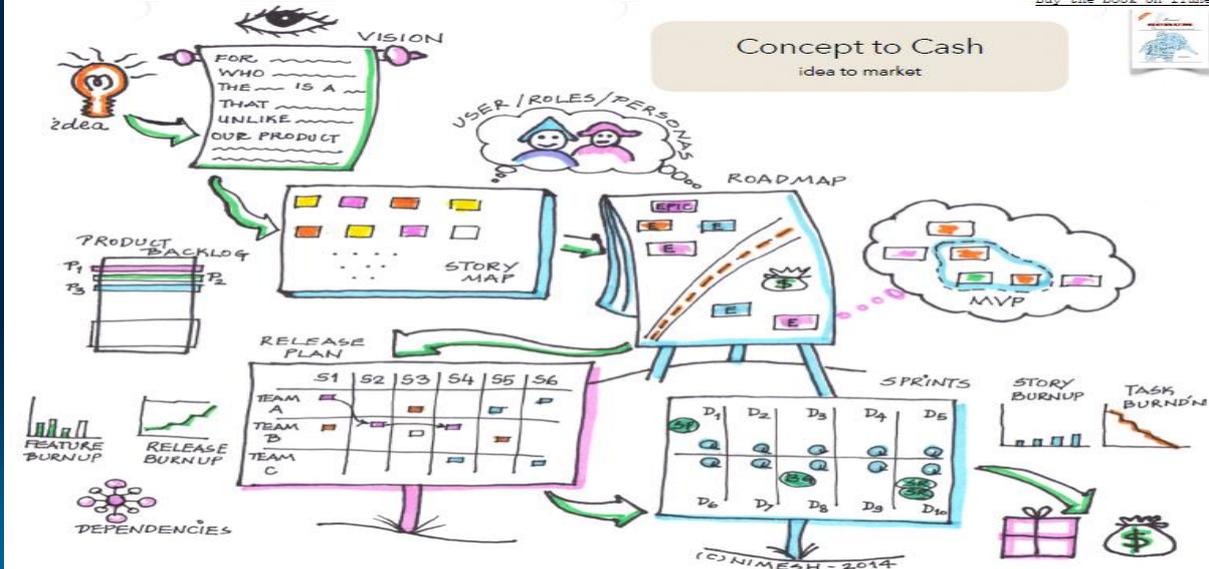
I am a **critically-thinking** human being, **my views** are always **subject to change**, **revision**, and **rethinking** at **anytime**. **Do not hold** me to them in **perpetuity**.

Agile Architecture Practices



**Collaborated with
Eduardo Guerra**

Agile/Lean Processes



Agile Principles and Practices

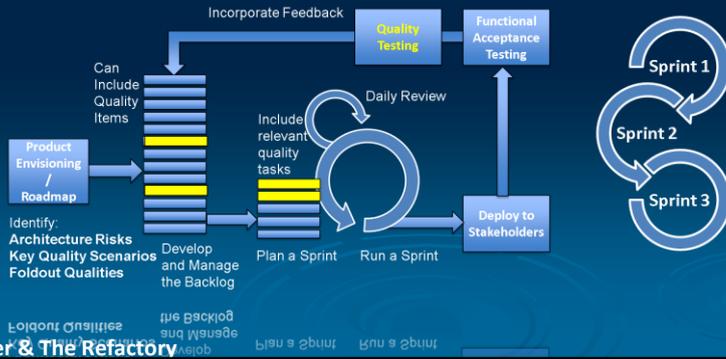
This collage represents Agile principles and practices. It features:

- A red circular stamp with the text 'CONTROL QUALITY CONTROL'.
- A pair of glasses with 'L F' and 'D 4' on the lenses.
- A group of diverse people giving a thumbs up.
- A hand holding a pen over a document with the word 'Change' written on it.
- A circular diagram with the text 'Incremental Delivery with Qualities' and the stages 'PLAN', 'BUILD', 'TEST', and 'RELEASE'.
- A circular diagram with three people icons and arrows.
- A gold award ribbon with the text 'Award for Excellence'.
- A document titled 'Testing' with a green checkmark and a red 'X'.
- A document titled 'Refactoring' with the text 'Supporting the Design of Existing Code' and an illustration of a person climbing a ladder.
- The word 'FEEDBACK' in large, colorful, multi-colored letters.



Agile/Lean Myths

KISS
Simple
is Best



© 2019 Joseph Yoder & The Refactory

www.agilemyths.com



Values Drive Practices



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!!!**



Patterns!



What is a Pattern?

Patterns can be thought of “**Good Practices**”

Proven Solutions to Repeating Problems

Proven Practices to Repeating Situations

Embody Experiences of What Works...

...and What Doesn't Work

Captures or Describes Knowledge of Experts

Embody “Quality” Attributes for
Solutions to specific Designs

THE INTERNET of THINGS

Machine Learning

Big Data

system
Recommend
recommender



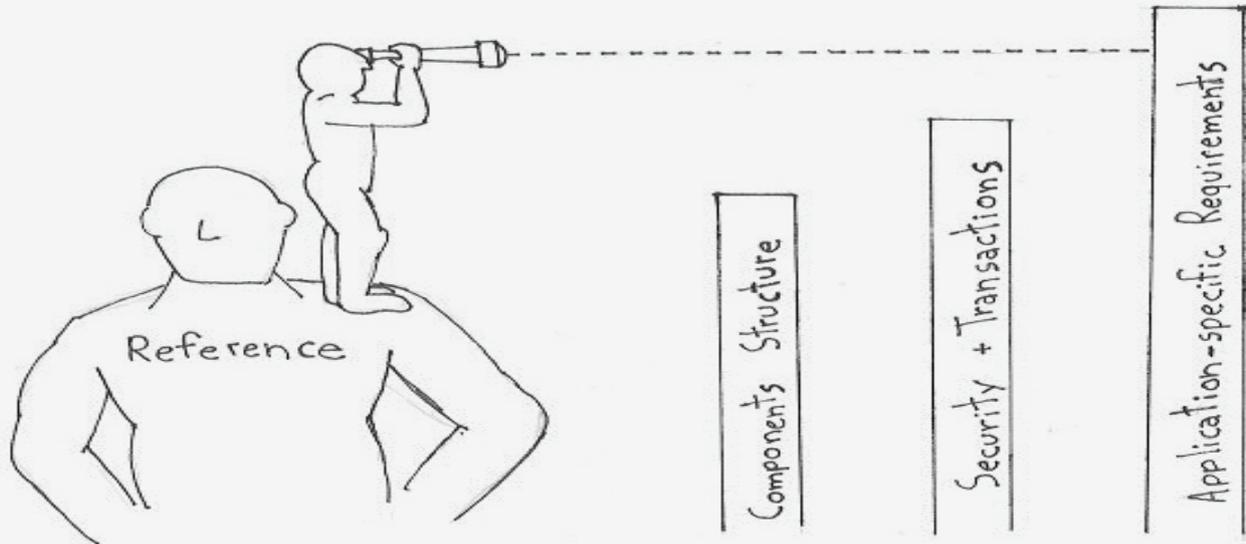
Have a sustainable architecture
Start the project fast, being agile



to start the
architecture

**Climbing on the
Shoulders of Giants**

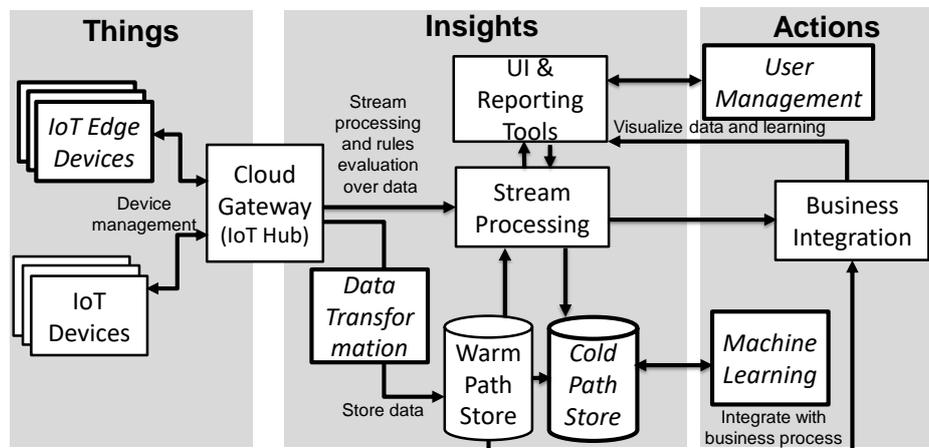
Climbing on the Shoulders of Giants



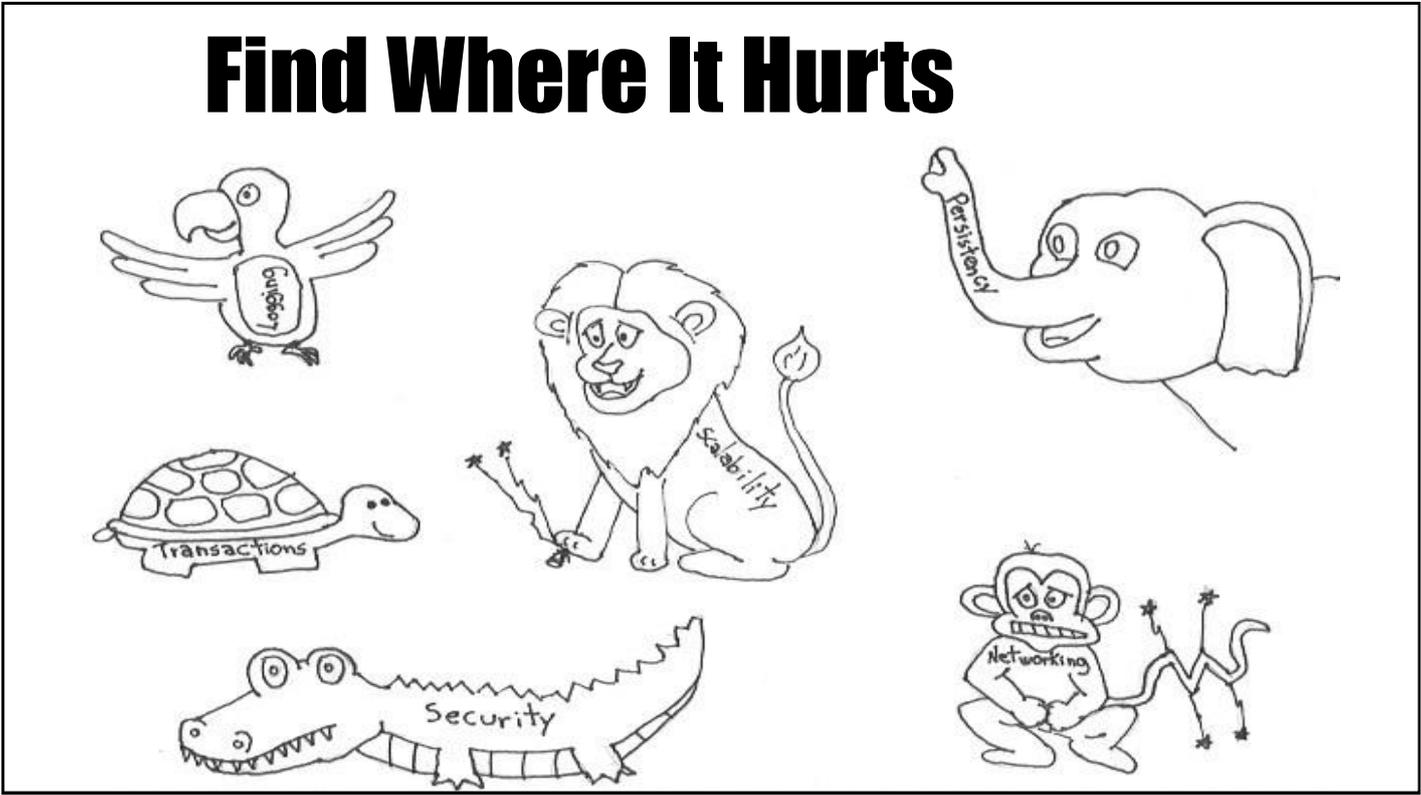
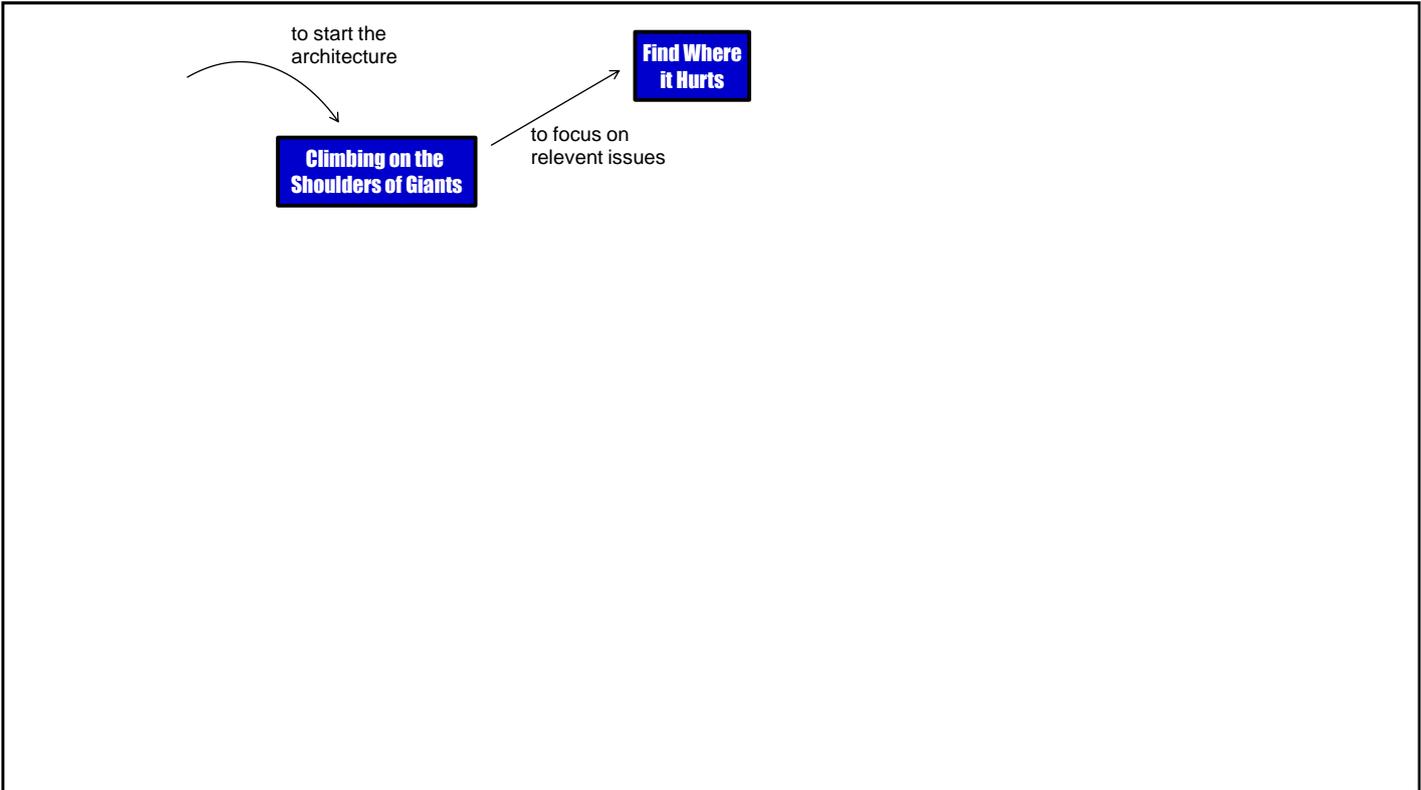
Build on a Reference Architecture

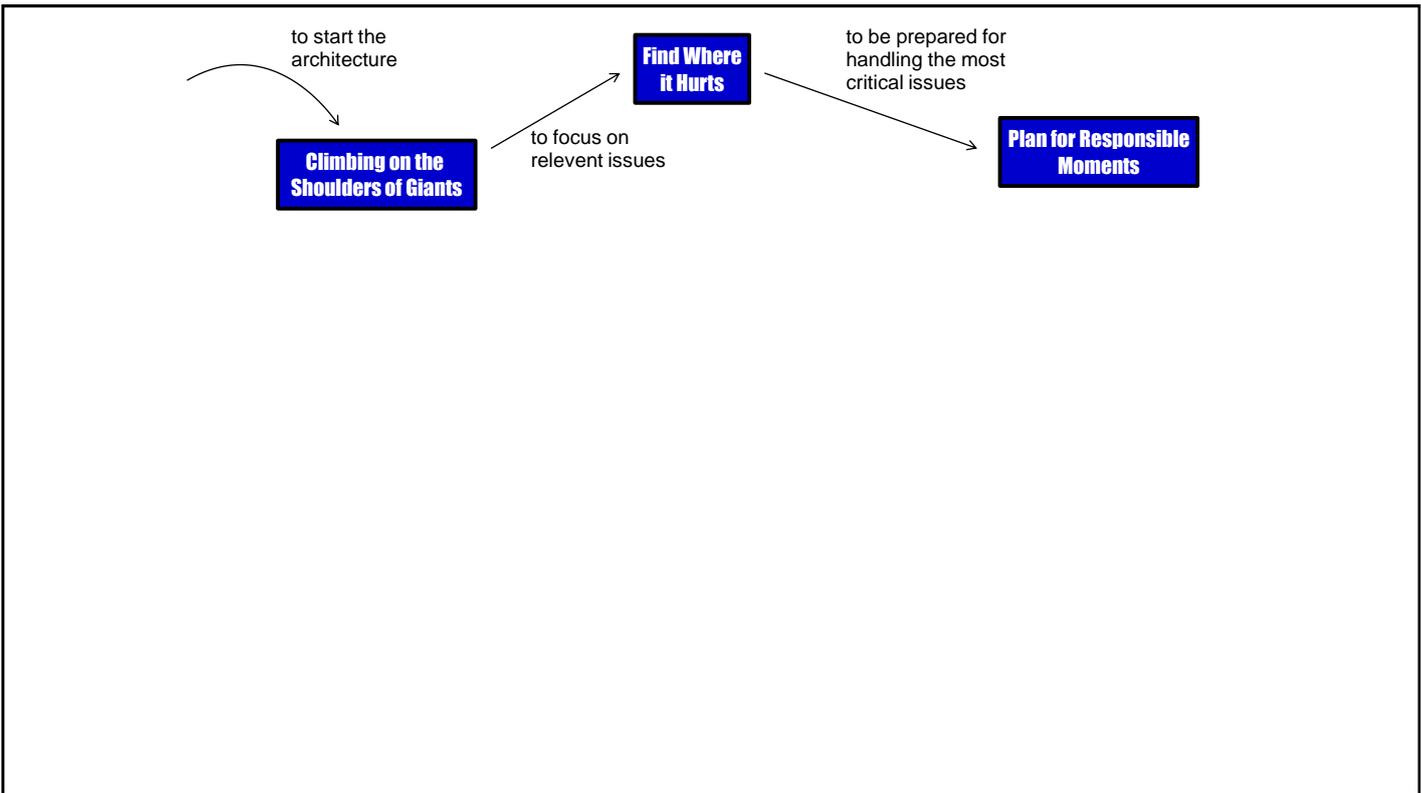
IoT Reference Architecture

[Azure IoT]

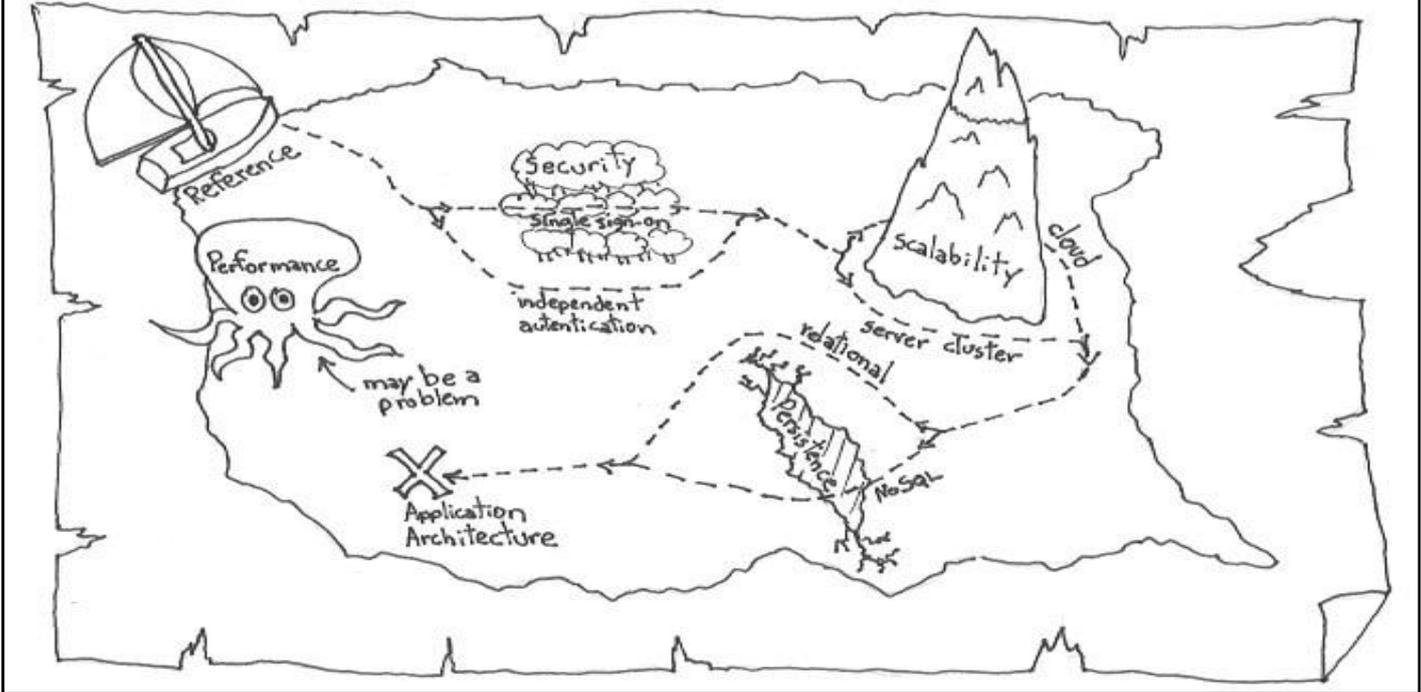


Microsoft, Azure IoT Reference Architecture, <https://aka.ms/iotrefarchitecture> — 2018





Plan for Responsible Moments



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

so

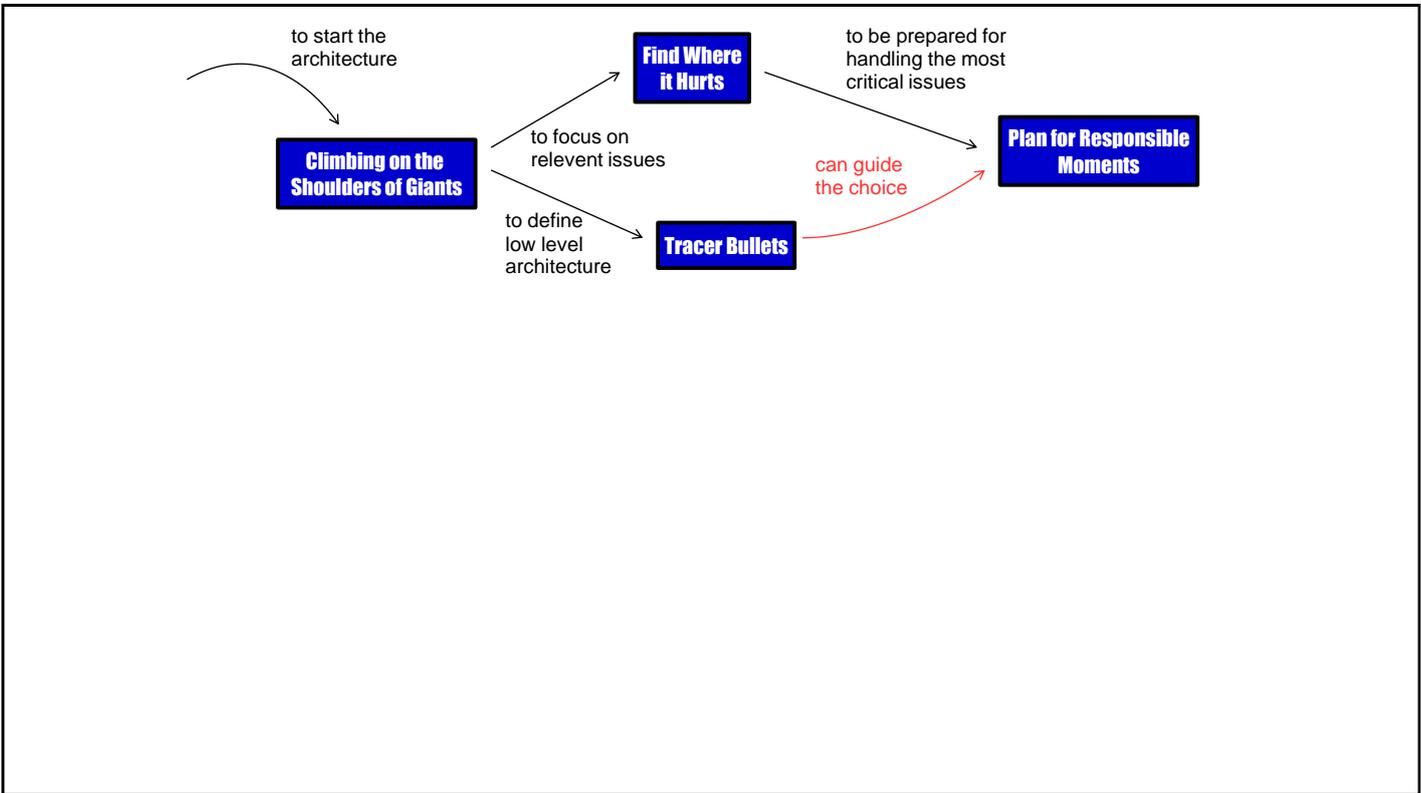
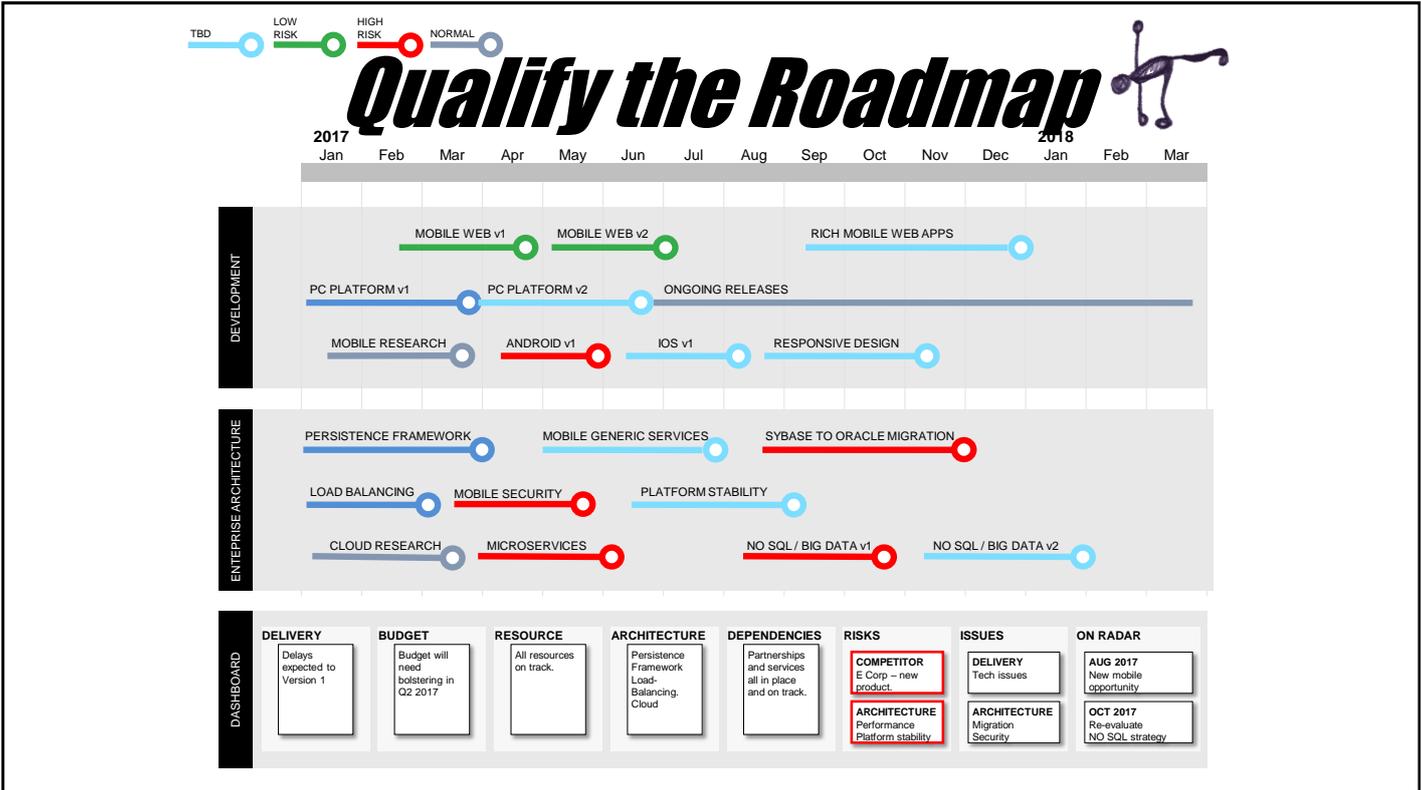
Choose the Most Responsible Moment

Qualify the Roadmap with Architecture Decisions

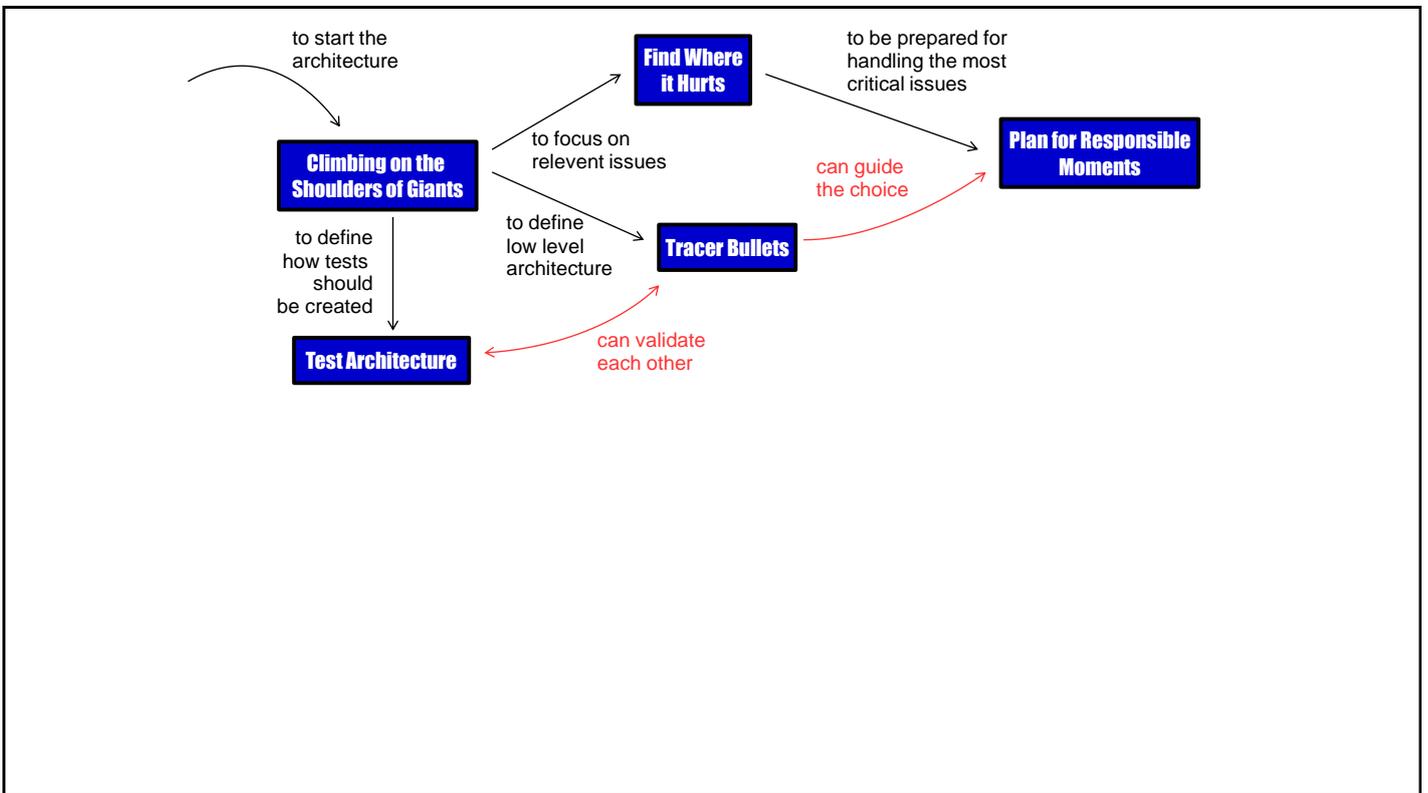
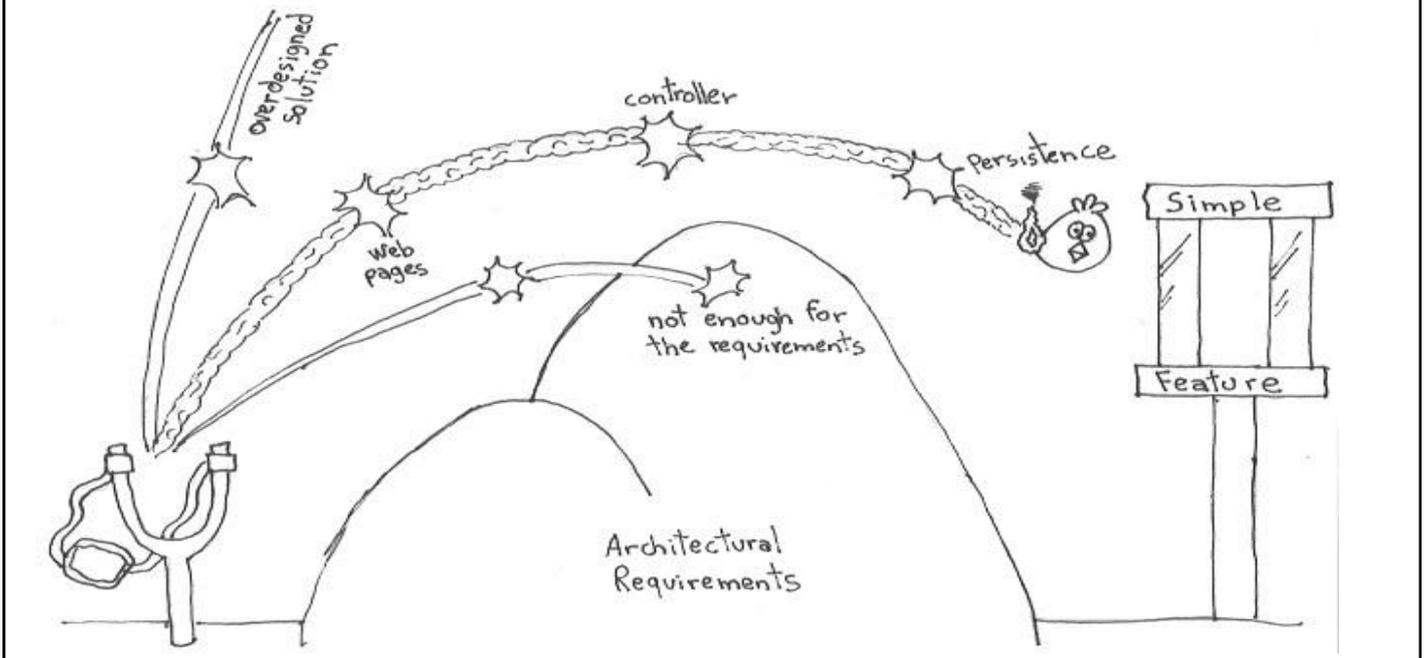
“All you need is the plan, the roadmap, and the courage to press on to your destination”

— Earl Nightingale

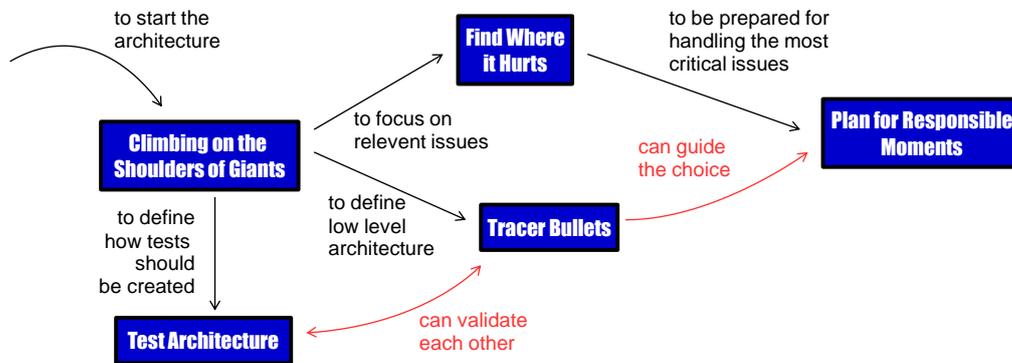
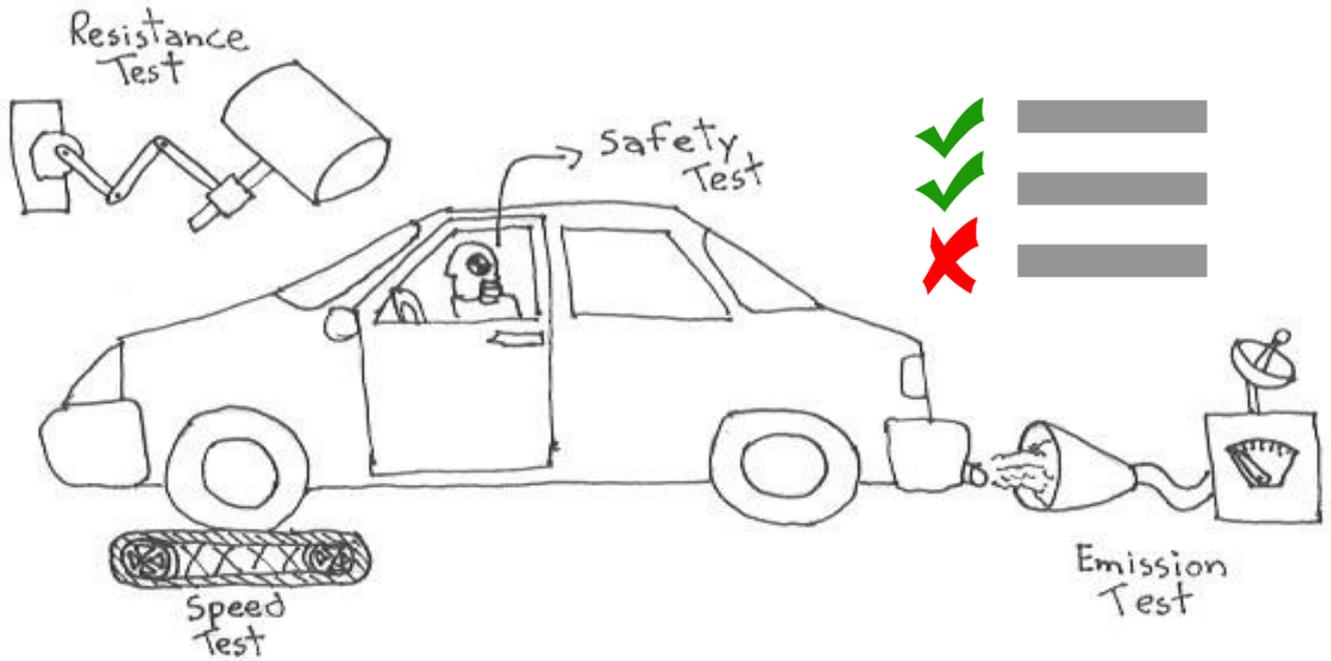




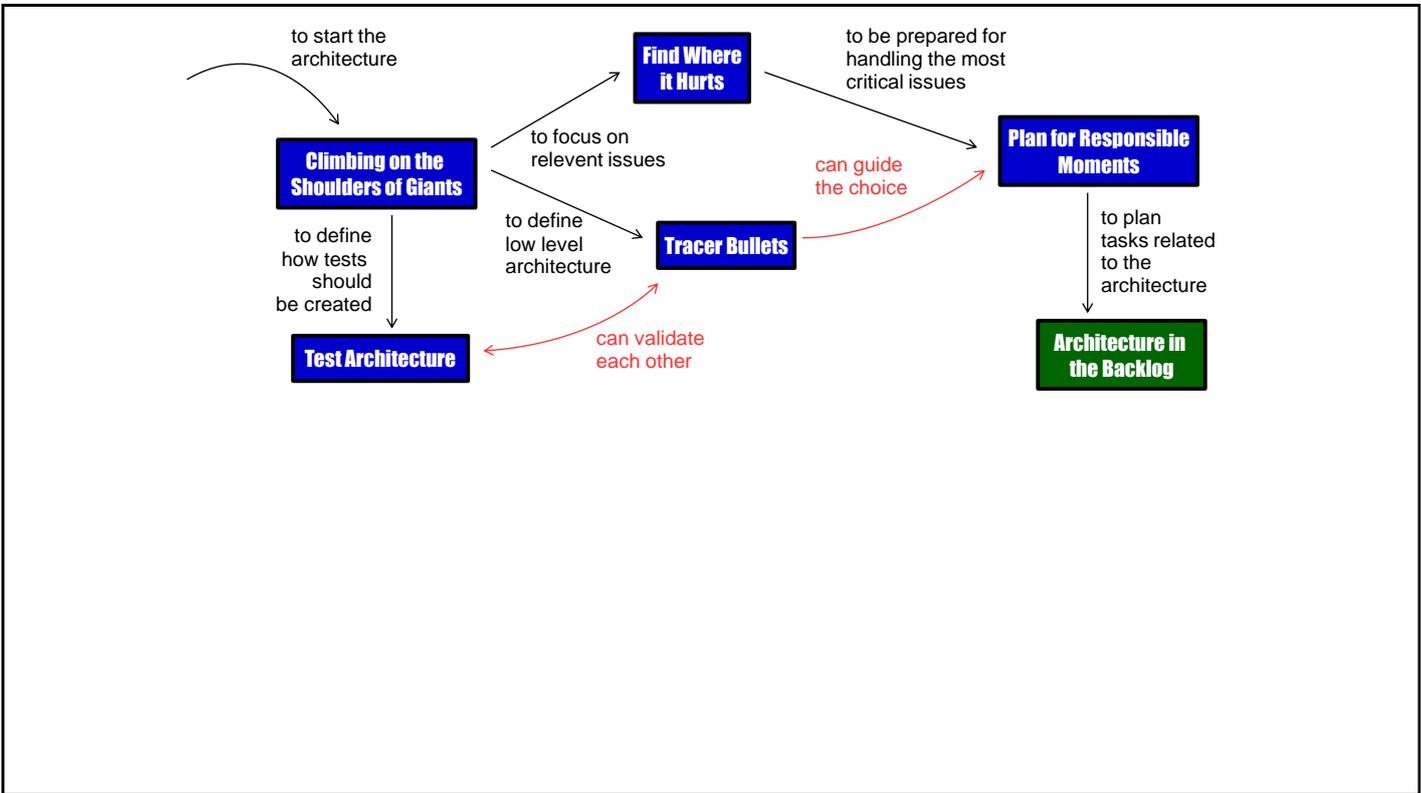
Tracer Bullets



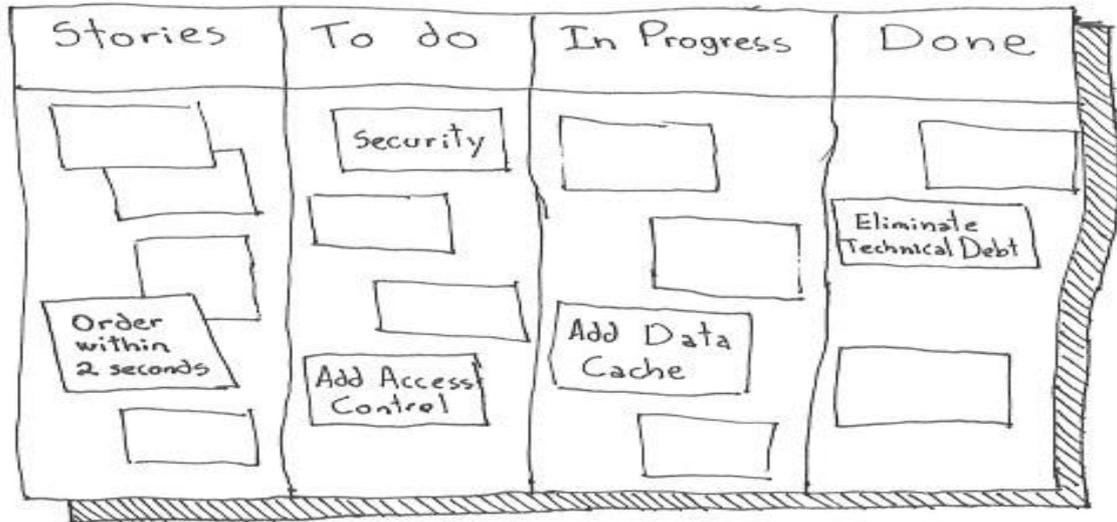
Test Architecture



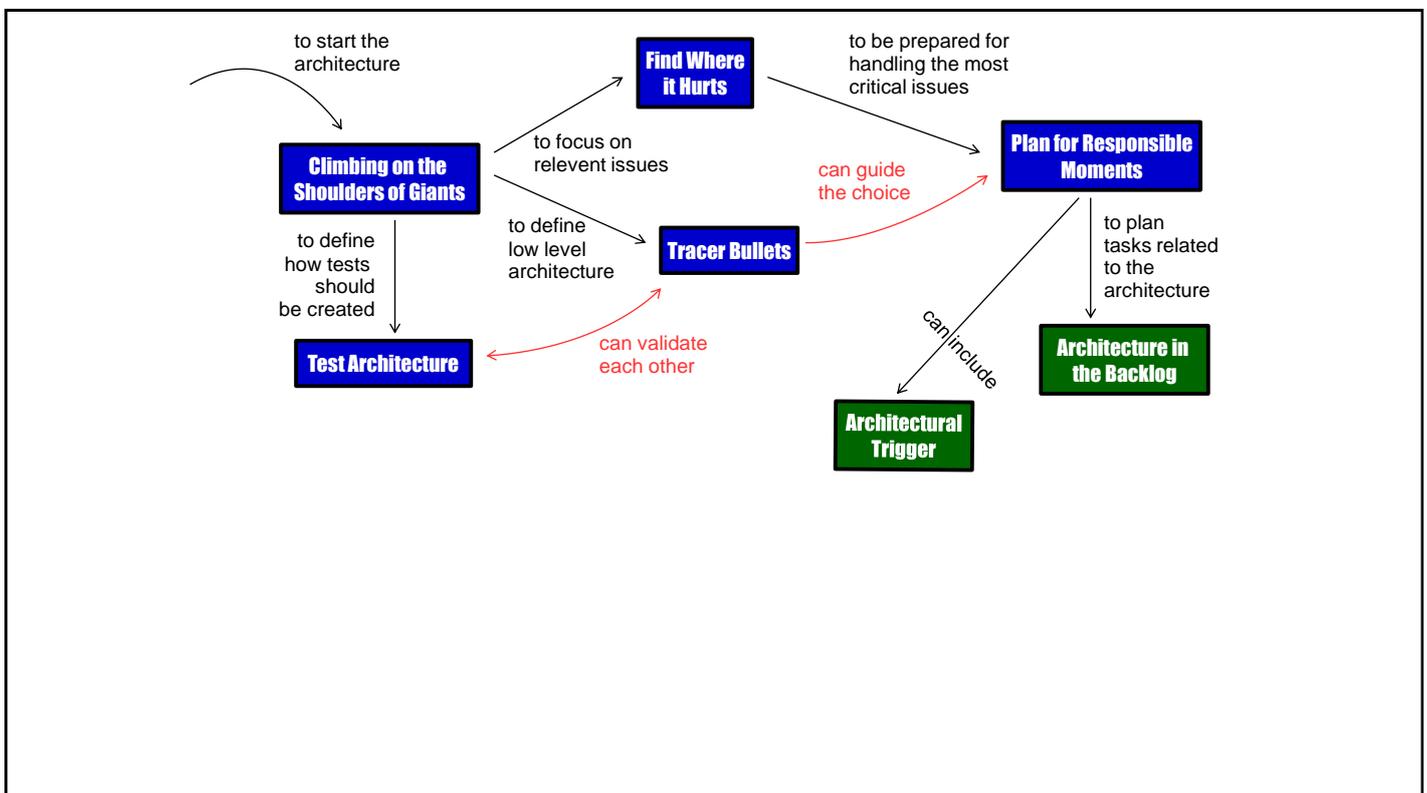
Note these can be done throughout the project...



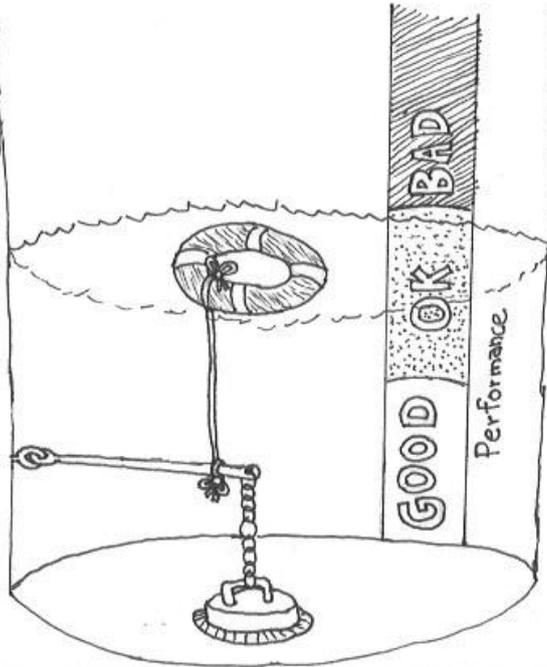
Architecture in the Backlog



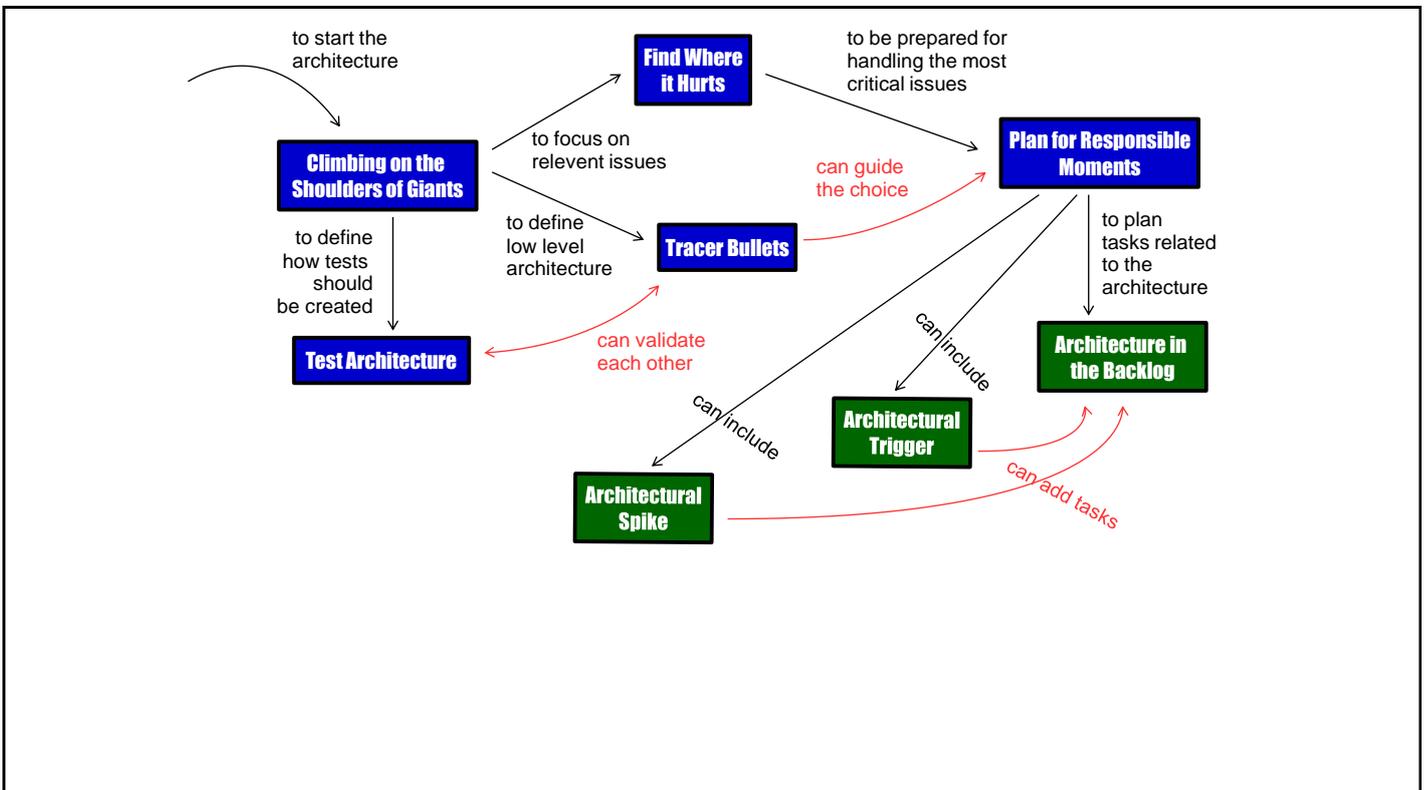
You can add backlog items for technical debt and quality-related architecture work... "yes, you can"



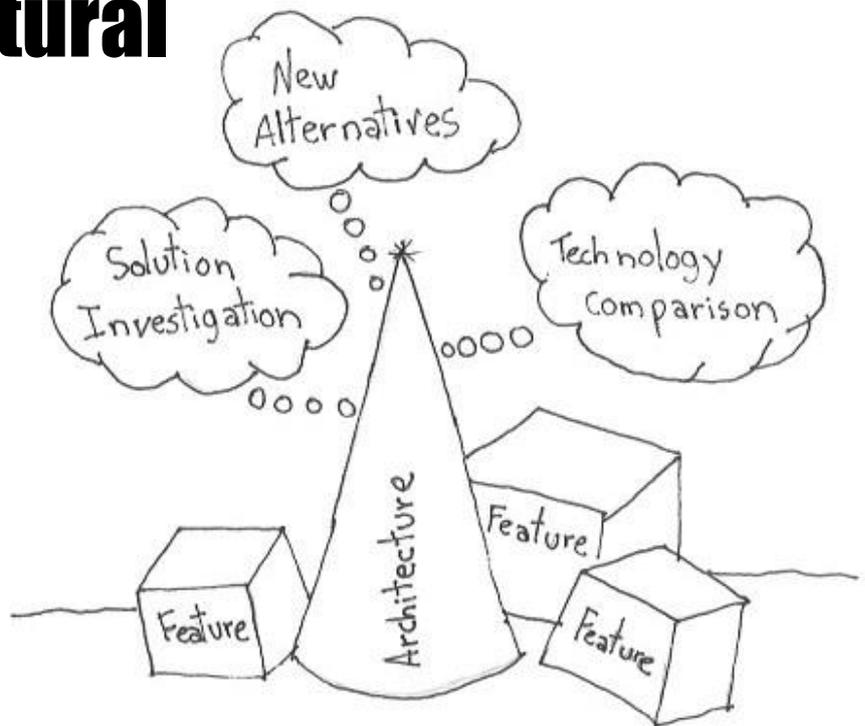
Architectural Trigger



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



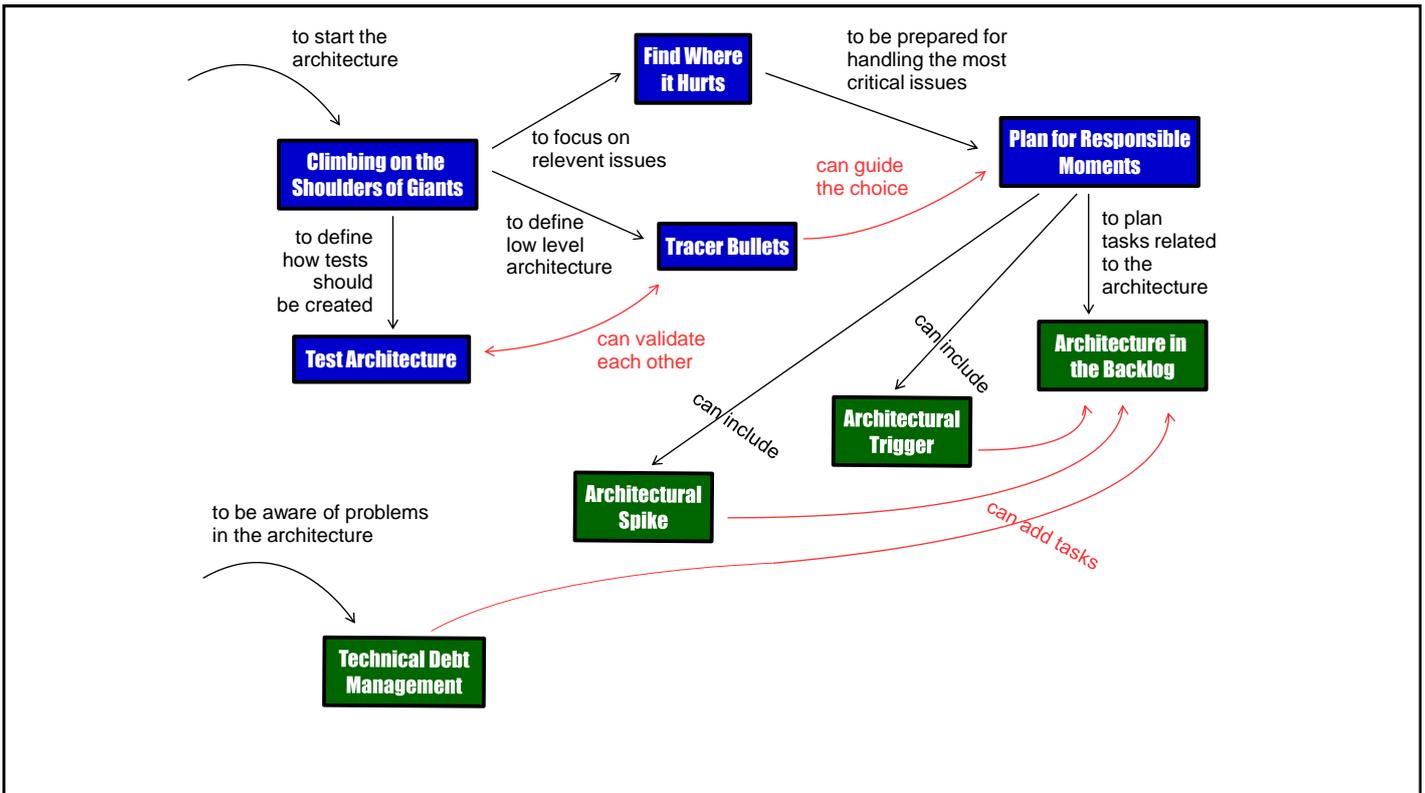
Architectural Spike



Architecture Spikes & Explorations

- Answer deep questions / offers potential architecture solutions
- Not as tactical as an XP Design Spike
- Visible and bounded
- Can be a Sprint!!!



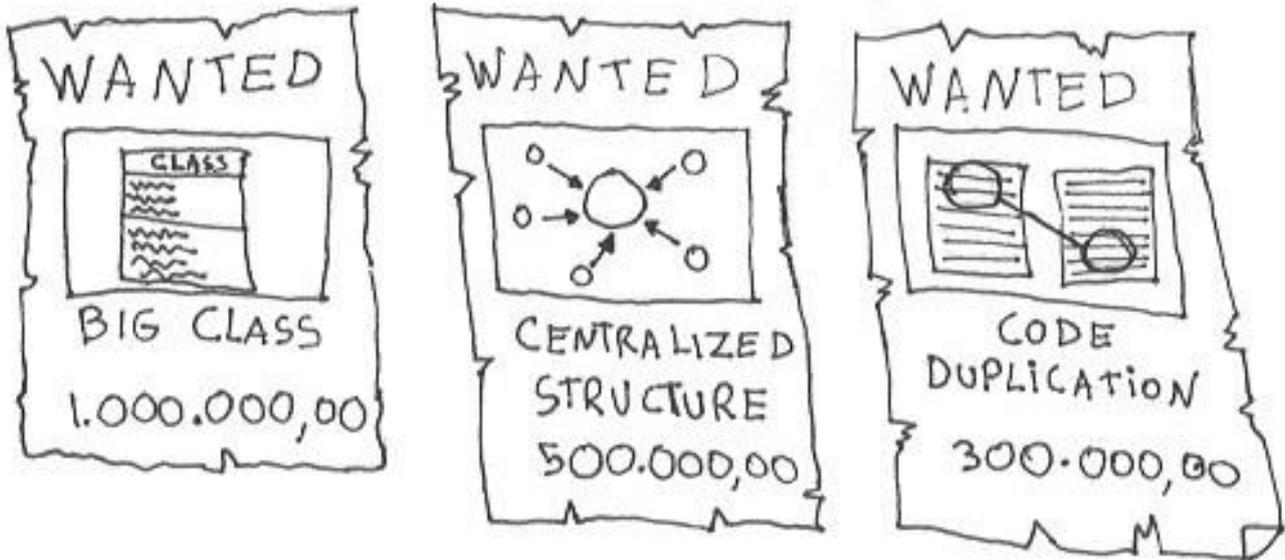


Some dirt becomes
 very hard to clean
 if you do not clean
 it right away!

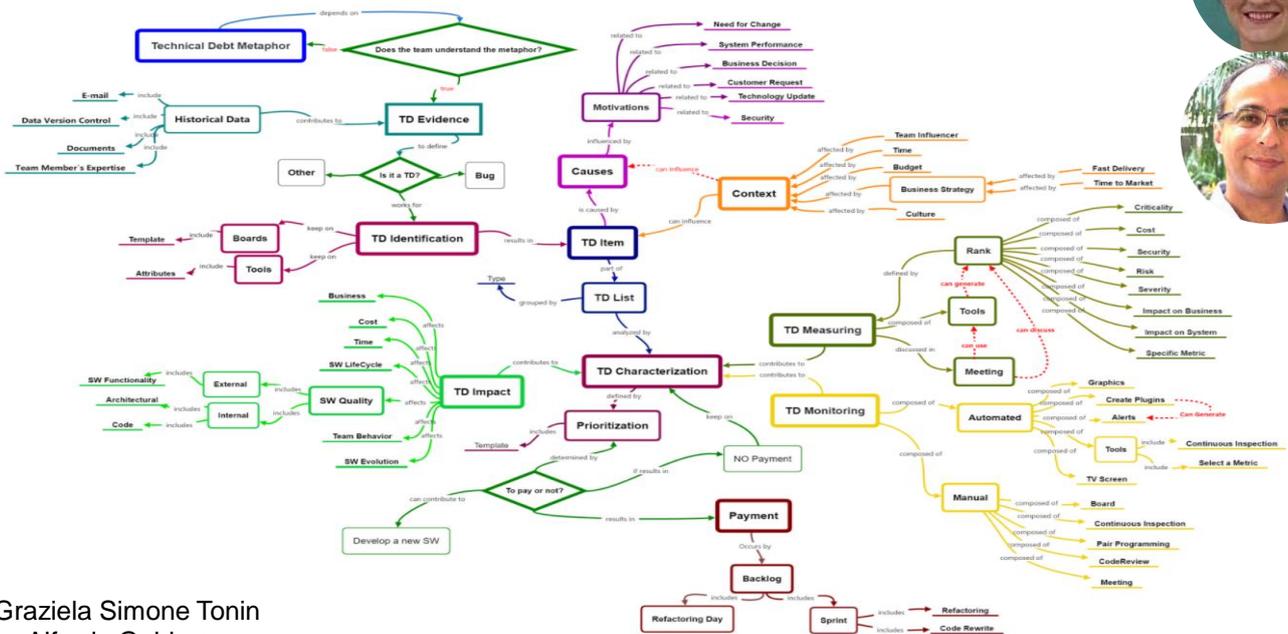
Architecture Debt,
 Quality Debt, Test Debt, ...

Technical Debt?

Technical Debt Management



Technical Debt Management



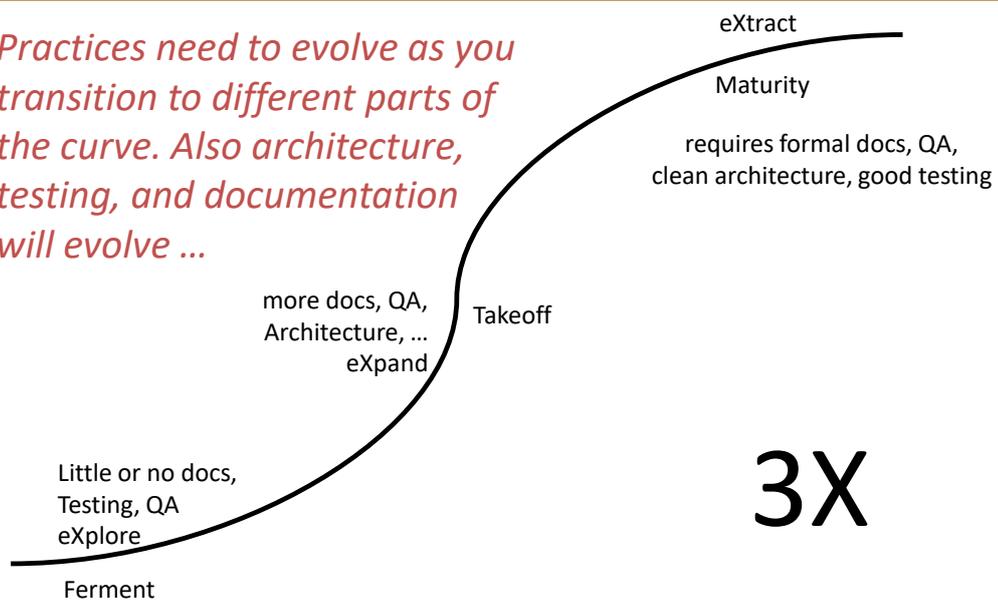
Graziela Simone Tonin
Alfredo Goldman

System "Qualities"



The Business 'S' Curve

Practices need to evolve as you transition to different parts of the curve. Also architecture, testing, and documentation will evolve ...

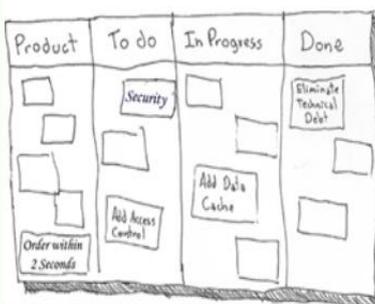




**HOW CAN WE IMPROVE
WHAT WE CANNOT SEE?**

© 2020 Joseph Yoder & The Refactory

Visibility is Key "Radiators"

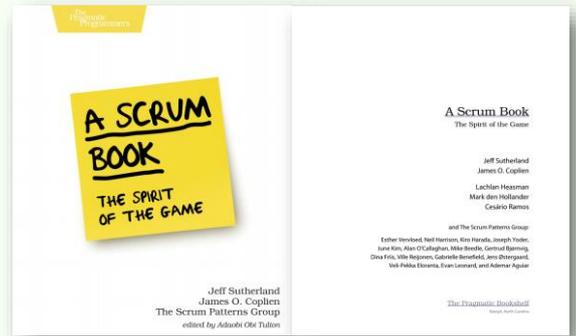
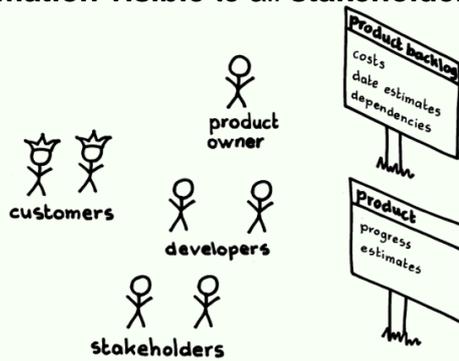


¶56 Information Radiator

Without valuable and timely information, the organization dies.

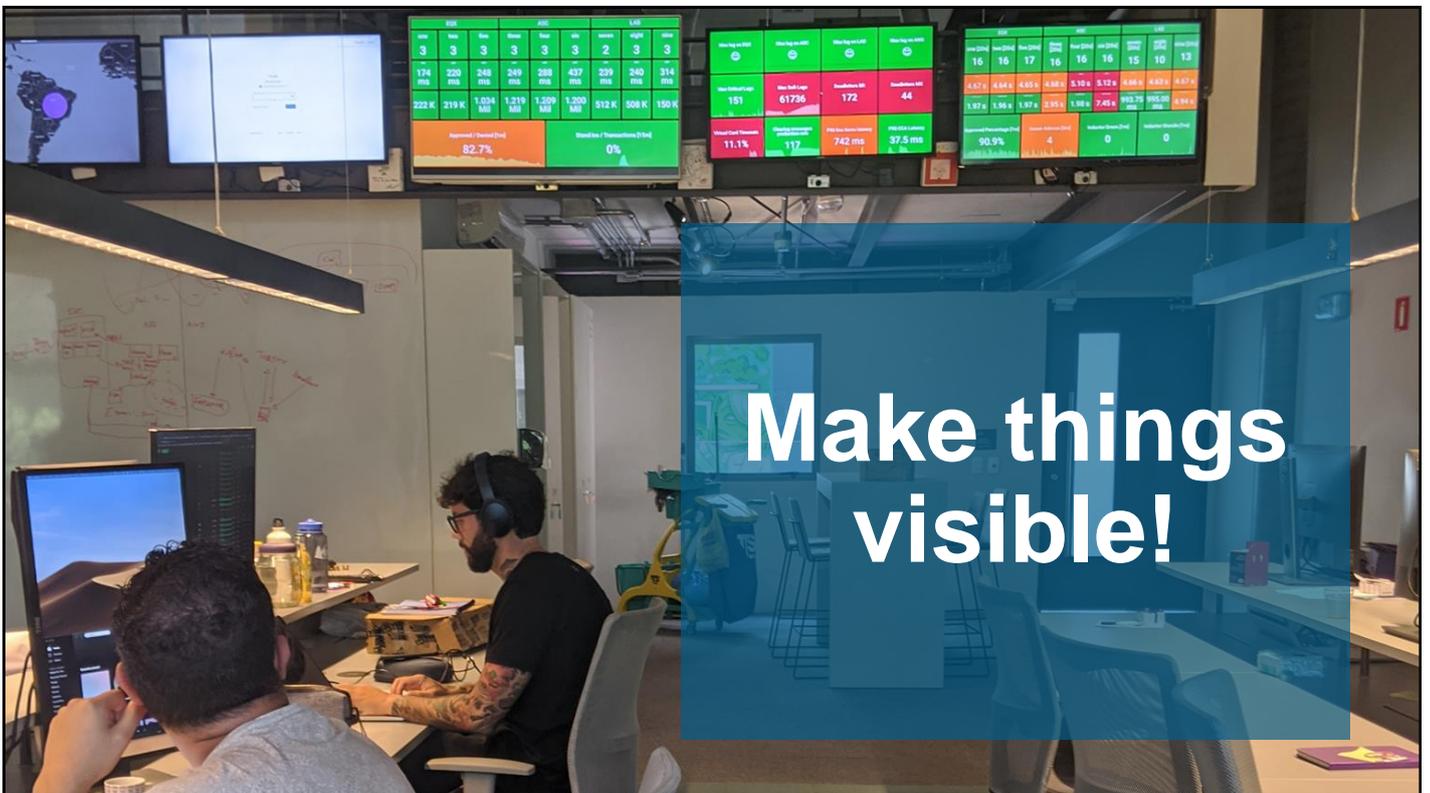
Therefore:

Collaboratively maintain **physical artifacts** that keep **information visible** to all **stakeholders**.



Copyright 2020 Joseph Yoder & The Refactory

<https://pragprog.com/book/jcscrum/a-scrum-booc>



Item	Contract Coverage
0001 Sample Project	95% (95%)
0002 Authentication	95% (95%)
0003 Sample Test Suite	95% (95%)
0004 Sample Login and Logout in Properties Page	95% (95%)
0005 Sample Register - Login	95% (95%)
0006 Sample Logout	95% (95%)
0007 Register	95% (95%)
0008 Register - Logout	95% (95%)
0009 Sample Login and Logout Property Registration	95% (95%)
0010 Sample Login and Logout and Logout again	95% (95%)
0011 Sample Search Test Case	95% (95%)
0012 Test Suite fails if we don't get results	95% (95%)
0013 Completed Test Suite	95% (95%)

Know your audience!

Define Metrics that add Value for the Team, Project and Business

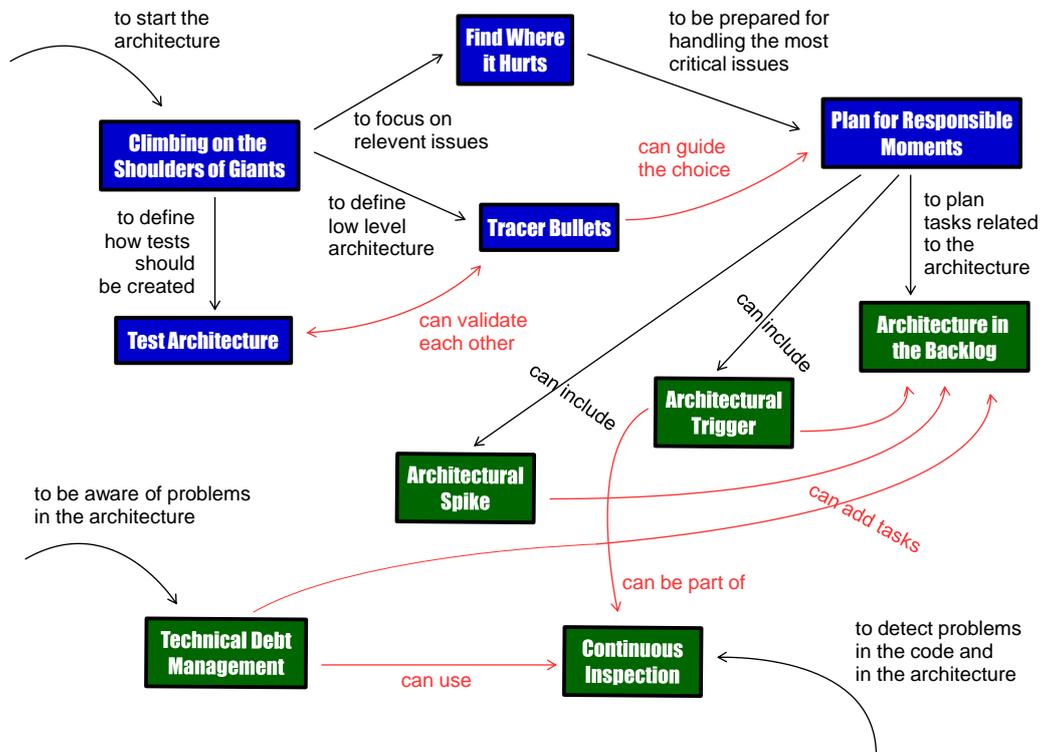
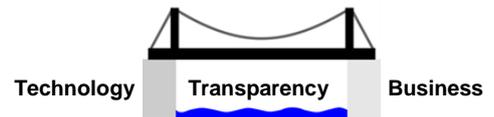
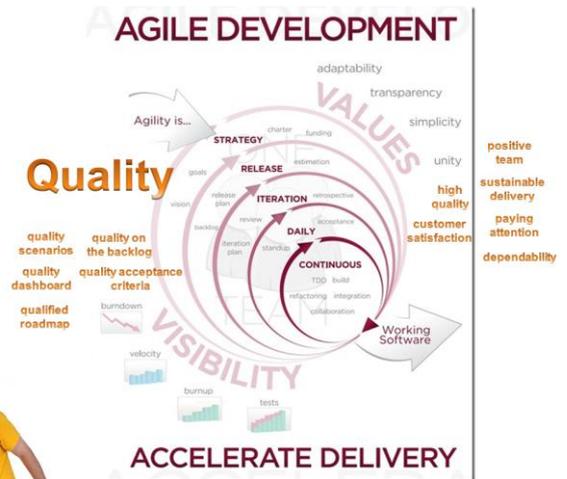
Make Visible

Transparency

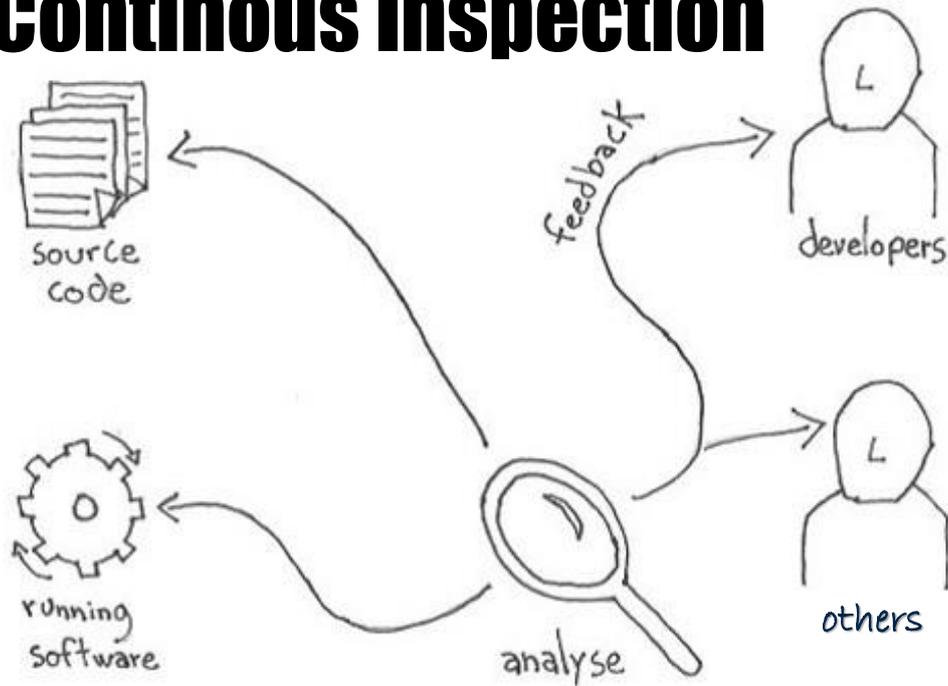
Agile Lean Core values:

- Learning
- **Visibility/Sharing**
- Quick Feedback
- **Communication**
- **Teamwork/Trust**
- **Continuous Improvement**
- Satisfying stakeholder needs

Tech Backlog ties into business values
 "Part of the Business Backlog???"



Continuous Inspection



Automate As You Go

“The first rule of any technology used in a business is that automation applied to an efficient operation will magnify the efficiency. The second is that automation applied to an inefficient operation will magnify the inefficiency” — Bill Gates

Things to Script or Automate

Repetitive Tasks
(builds, integration, tests, ...)

Involves Waiting 

Error Prone and/or Tedious

Setup of Environments ...

Quality Metrics
(performance, reliability, security, ...)

Code Smell Detection

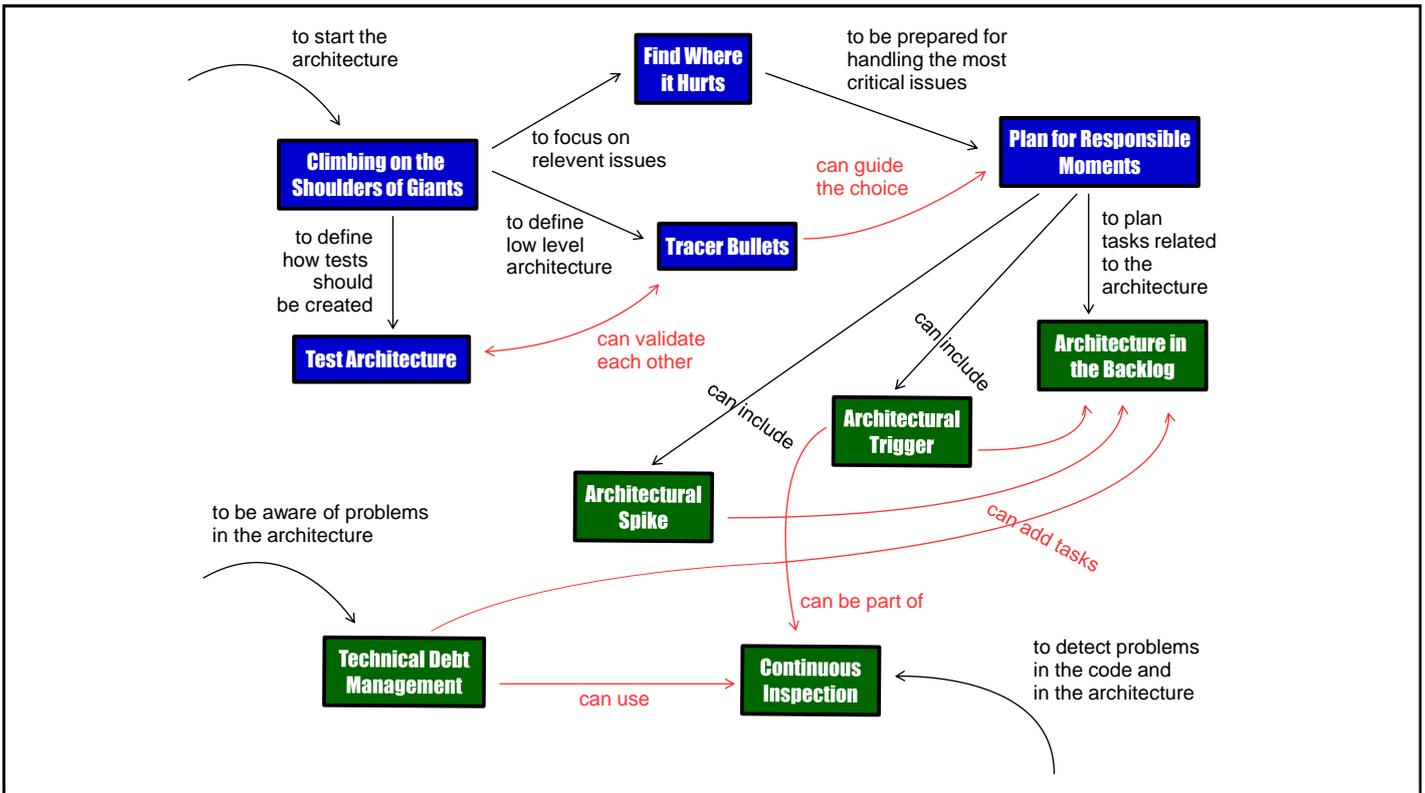
Architecture Conformance

Things that take a lot of time

Notes/checklists

- Less frequent, but important
- Harder to automate
 - inconsistent
 - requires thought



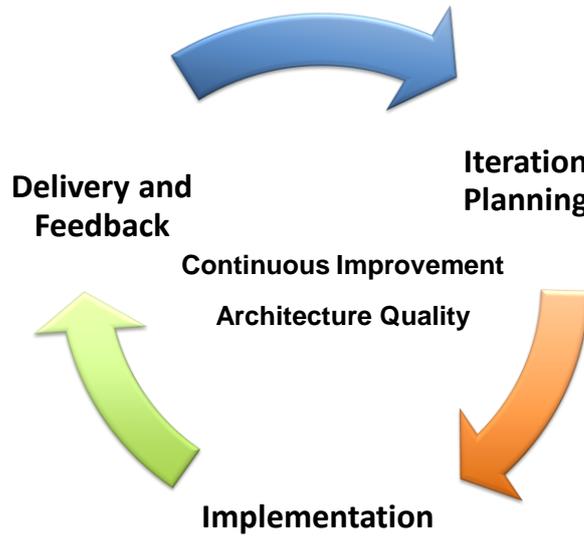


Have a sustainable architecture



Start the project fast

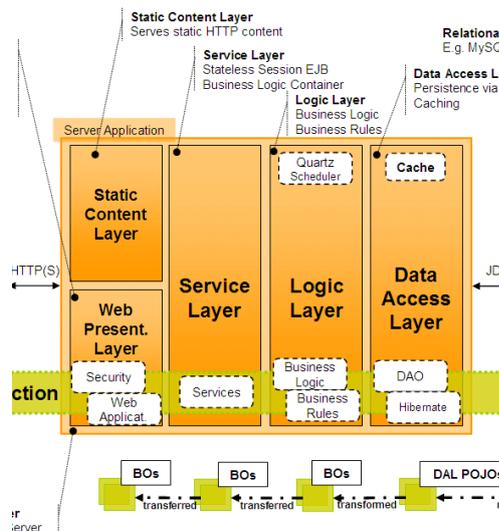
Periodically Re-Evaluate Architecture Risks



Incrementally Test Key Components' Performance



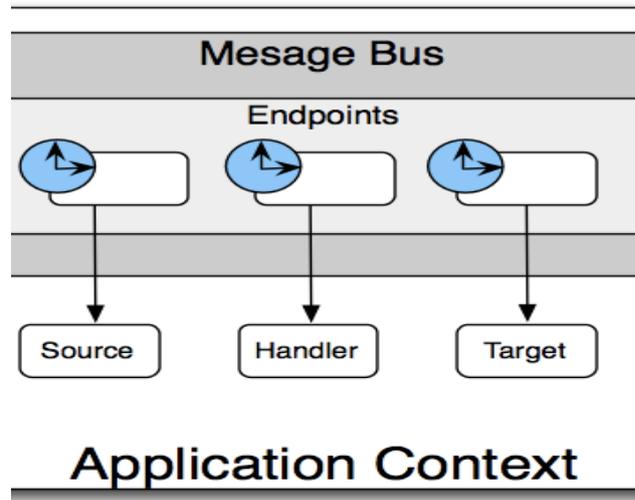
- Identify key pathways and critical components
- Test components as they arrive to access performance
- Use mocks, stubs, and auto-responders to simulate missing components



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

Pause Points Help

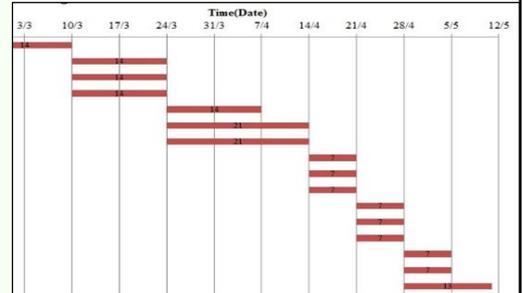


Slack Time

*Need Slack time to improve
and to ensure quality*

Ways to get slack time...

- Monitor and Make Visible
- Reduce Waste (Muda)
- Inject time into process
(retros, daily cleanup, ...)

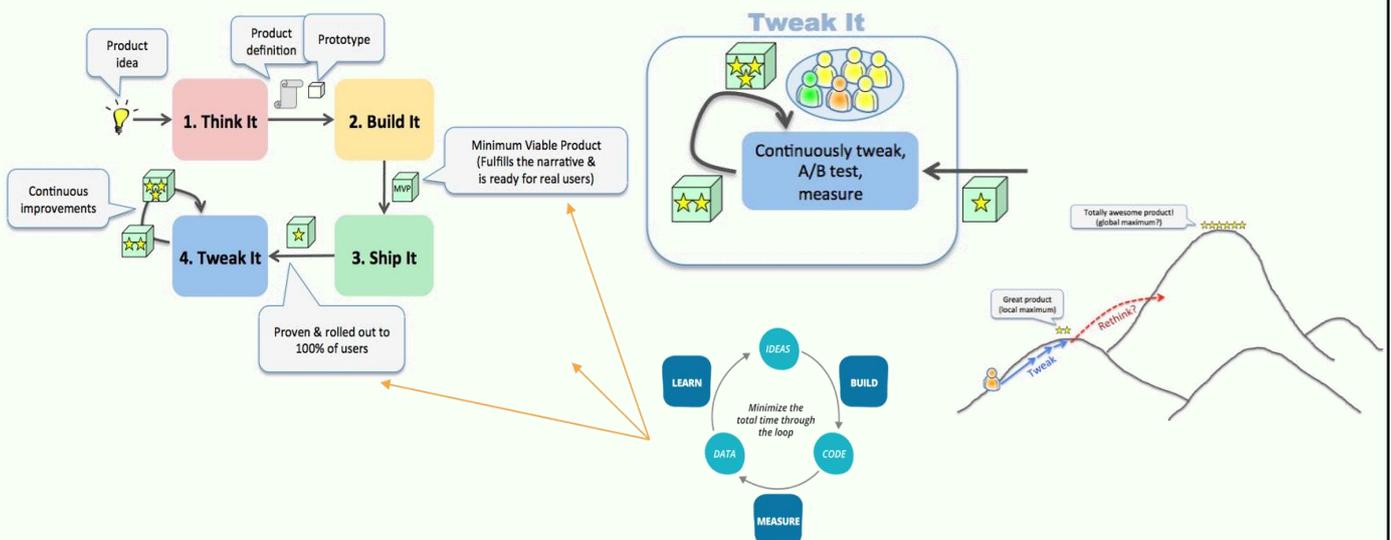


Try little experiments...

© 2020 Joseph Yoder & The Refactory

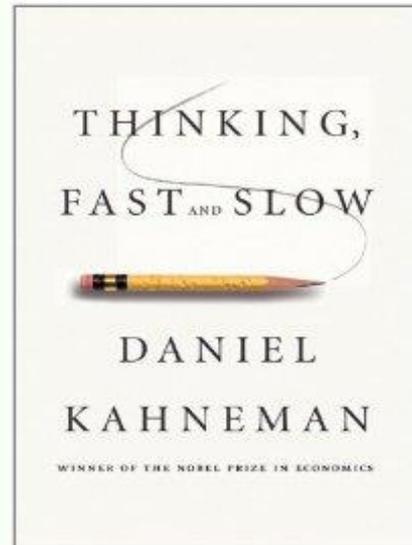
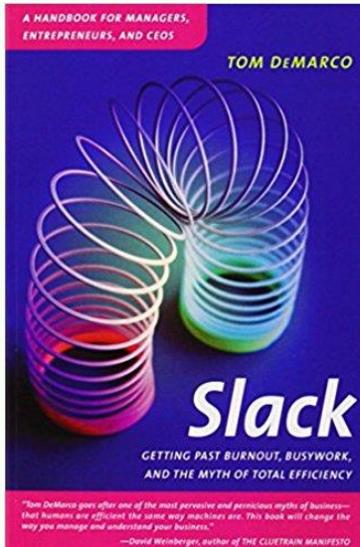
(c) Can Stock Photo / AntonioGuillem

Spotify: Innovation



© 2020 Joseph Yoder & The Refactory

Slack Time



© 2020 Joseph Yoder & The Refactory

Velocity ≠ Productivity

Velocity ≠ Value

Going Fast isn't always good!!!

Agile Values Can 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



Patterns for Evolving Agile Architecture

to start working with the architecture

Climbing on the Shoulders of Giants

How can you quickly define the basic application architecture and the main component types that will satisfy the requirements?

Use an existing reference compatible with the application platform and suitable to its needs as a starting point.

Find Where it Hurts

How can you identify relevant points where the architectural design should focus?

Early on, identify the challenging technical requirements that are important for the project, so they can be handled at the optimal time.



Asian PLoP 2015

to prepare to handle the most critical issues

Plan for Responsible Moments

How can you handle the technical challenges in the beginning of the project without a full architectural design upfront?

Create a technical plan for how and when to handle each of the technical challenges and evolve it throughout the project. This plan needs to define how to identify these important responsible moments and circumstances when it is appropriate to address these technical challenges.

can guide the choice of...

Tracer Bullets

How can you define low-level details about the architecture without spending a lot of time upfront on a detailed investigation?

Select the smallest set of architectural relevant user stories and implement them as references for upcoming functionality. Use this implementation to face technical challenges that were planned to be targeted before the project iterations.

to know how tests should be created

Test Architecture

How can you define how architectural components should be tested?

Define the test approach for each kind of component, considering its scope, technique, and kind of tests and tools that are going to be used.

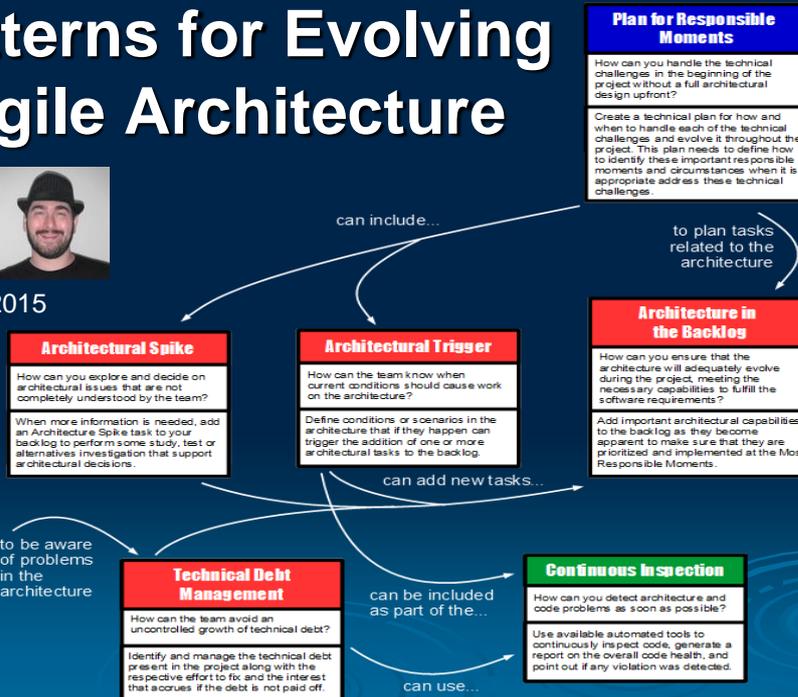
can validate each other

can include

Patterns for Evolving Agile Architecture



USA PLoP 2015

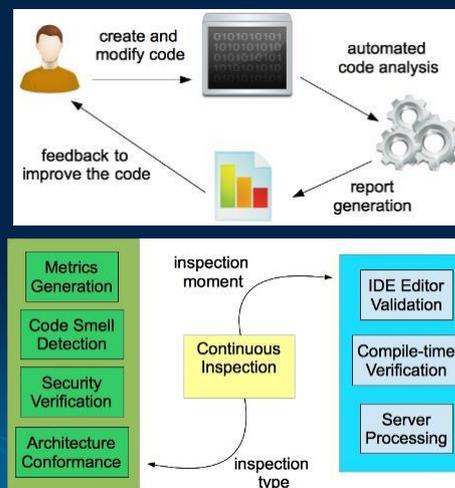


Continuous Inspection



Asian PLoP 2014 Paper

CODE SMELL DETECTION
METRICS (TEST COVERAGE, CYCLOMATIC COMPLEXITY, TECHNICAL DEBT, SIZES, ...)
APPLICATION SECURITY CHECKS
ARCHITECTURAL CONFORMANCE
AUTOMATE WHERE YOU CAN!!!



Sustaining Your Architecture

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 As You Go
 Quality Checklists

<https://bit.ly/2sDX6FS>

QA to AQ

Patterns about transitioning from Quality Assurance to Agile Quality

Joseph W. Yoder¹, Rebecca Wirfs-Brock², Ademar Aguiar³

¹The Refactory, Inc.,

²Wirfs-Brock Associates, Inc.

³FEUP

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.

Citation and Subject Descriptors

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

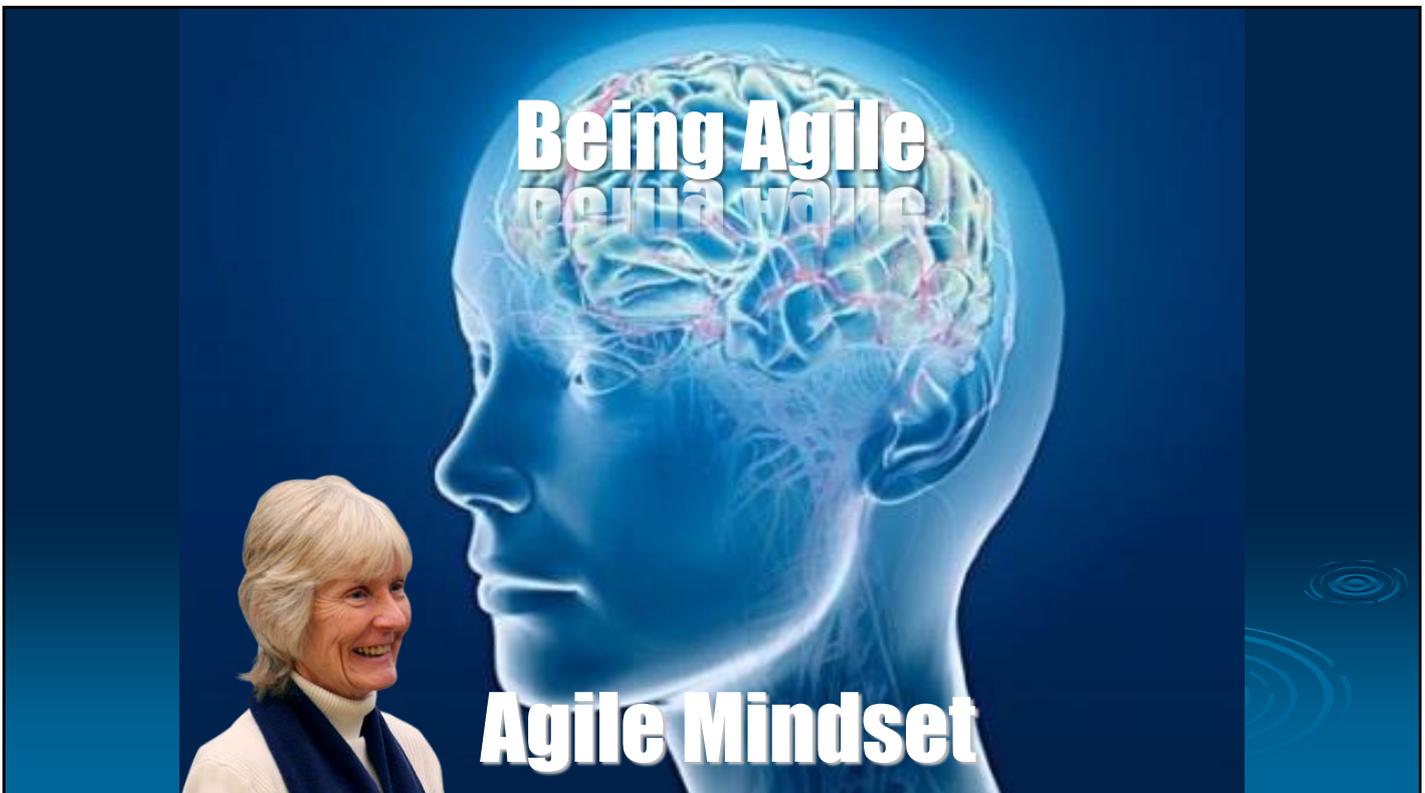
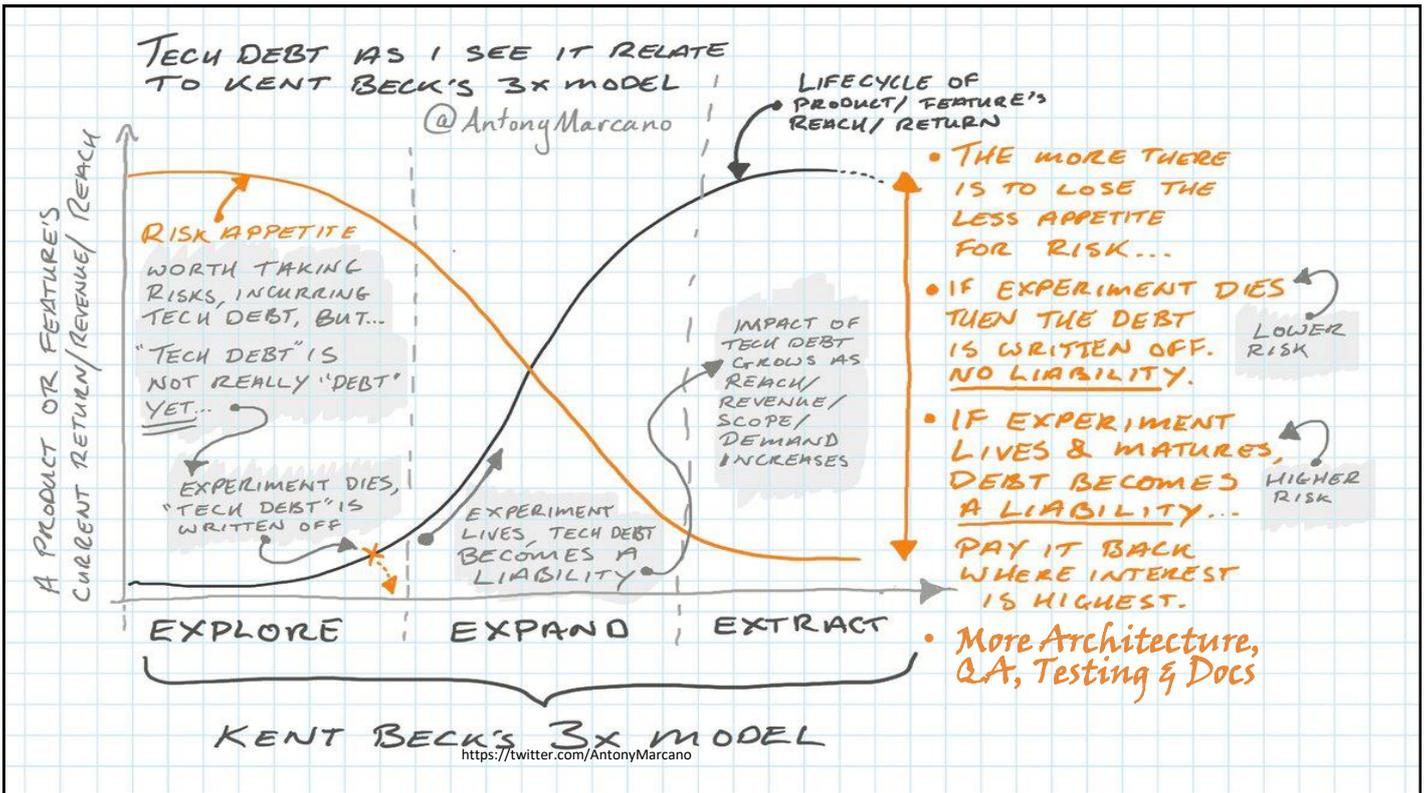
Continuous Inspection: A Pattern for Keeping your Code Healthy and Aligned to the Architecture, AsianPLOP 2014

Patterns for Initial Architecture Design on Agile Projects AsianPLOP 2015

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



Agile/Lean

Ideas



```
import org.wel...
import org.welmacro.util...
import java.io.*;
import java.sql.*;
import java.util.*;
import javax.servlet.*;
```

Code

Build



Measure



Learn



Data

Being Pragmatic

Lot's of
Upfront
Planning

Lot of Design
& Architecture

Traditional
or Waterfall



Rough
Adaptive
Plan (changing)

Right Balance
of Design
& Architecture

Being Agile



No Planning

No Design
or Architecture

Sometimes
called Agile



Balance Between...



In Memory to Mike Beedle



Mike Beedle

March 21 at 11:48am · Twitter · 👤

✅ Agile doesn't cure INCOMPETENCE.

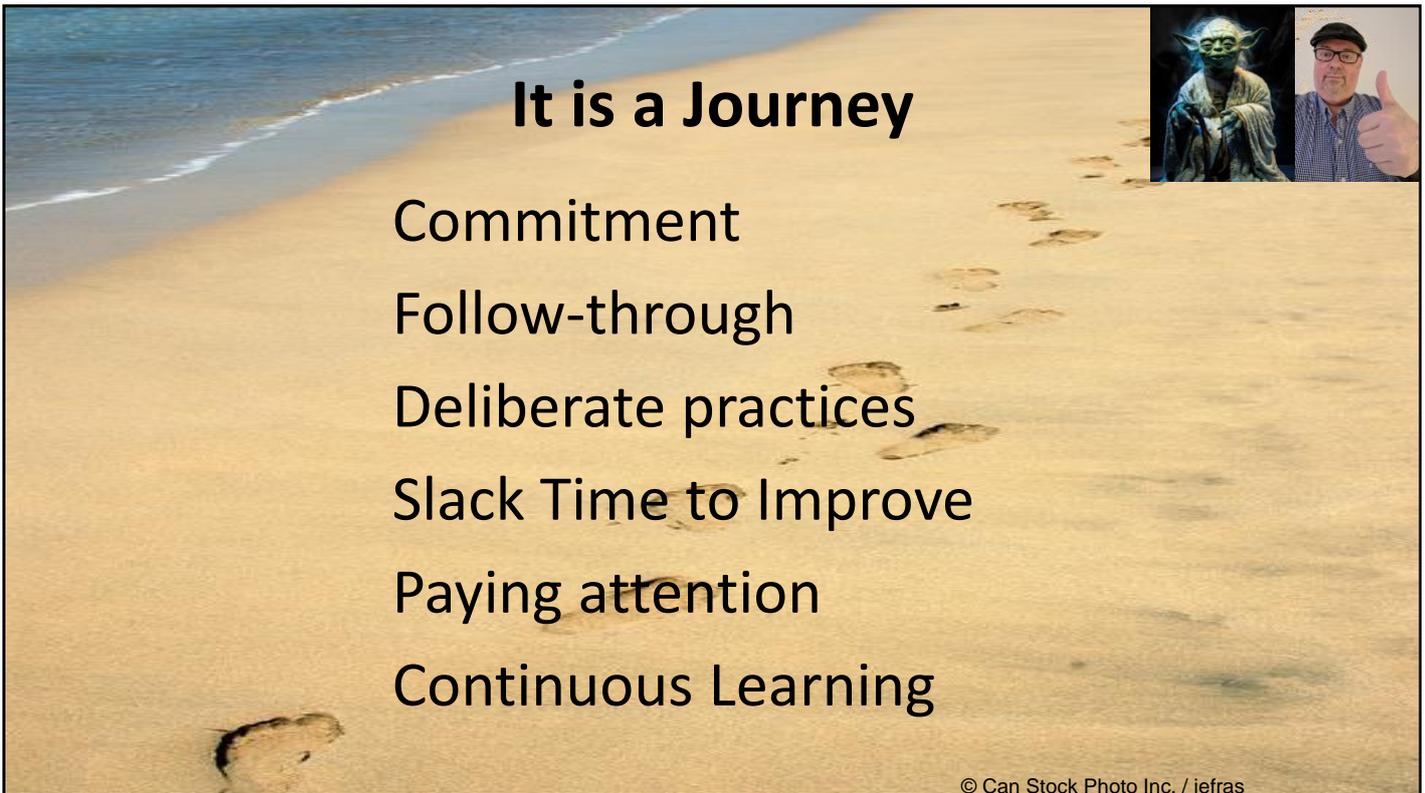
You can coach teams to be more engaged and collaborative, but NO Agile framework, method, or mindset can save you from BLATANT FAILURE if your development team is INCOMPETENT in basic engineering practices.

Technical excellence is a MUST!



Architecture and Agility





```
@Test
public void presentationEnd(){
    Presentation p = new Presentation();
    Audience a = new Audience();
    p.setAudience(a);
    p.perform();
    p.end();
    p.thanksEveryone();
    assertTrue(a.isApplauding());
}
```

Arigato!



“You can’t fix what you can’t see”

“If you think good architecture is expensive, try bad architecture”

© 2020 Joseph Yoder & The Refactory