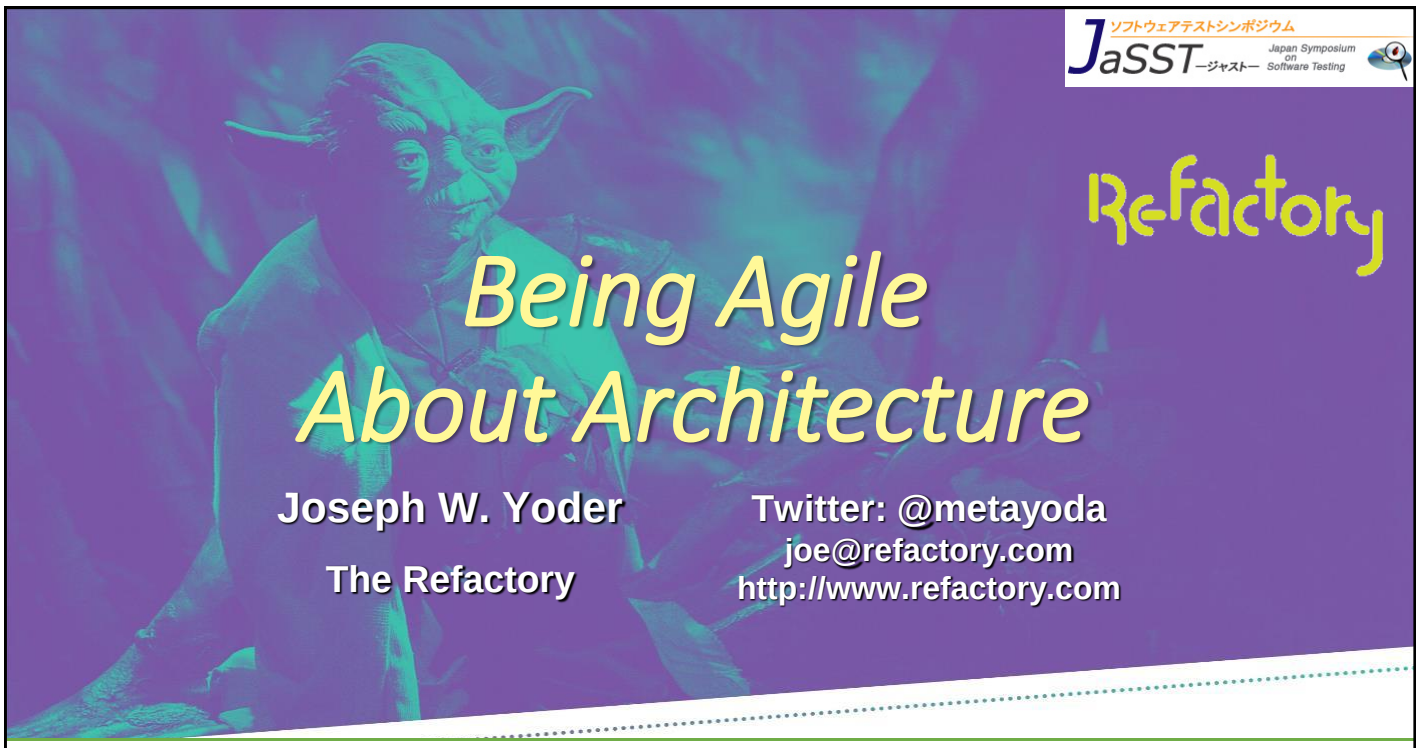
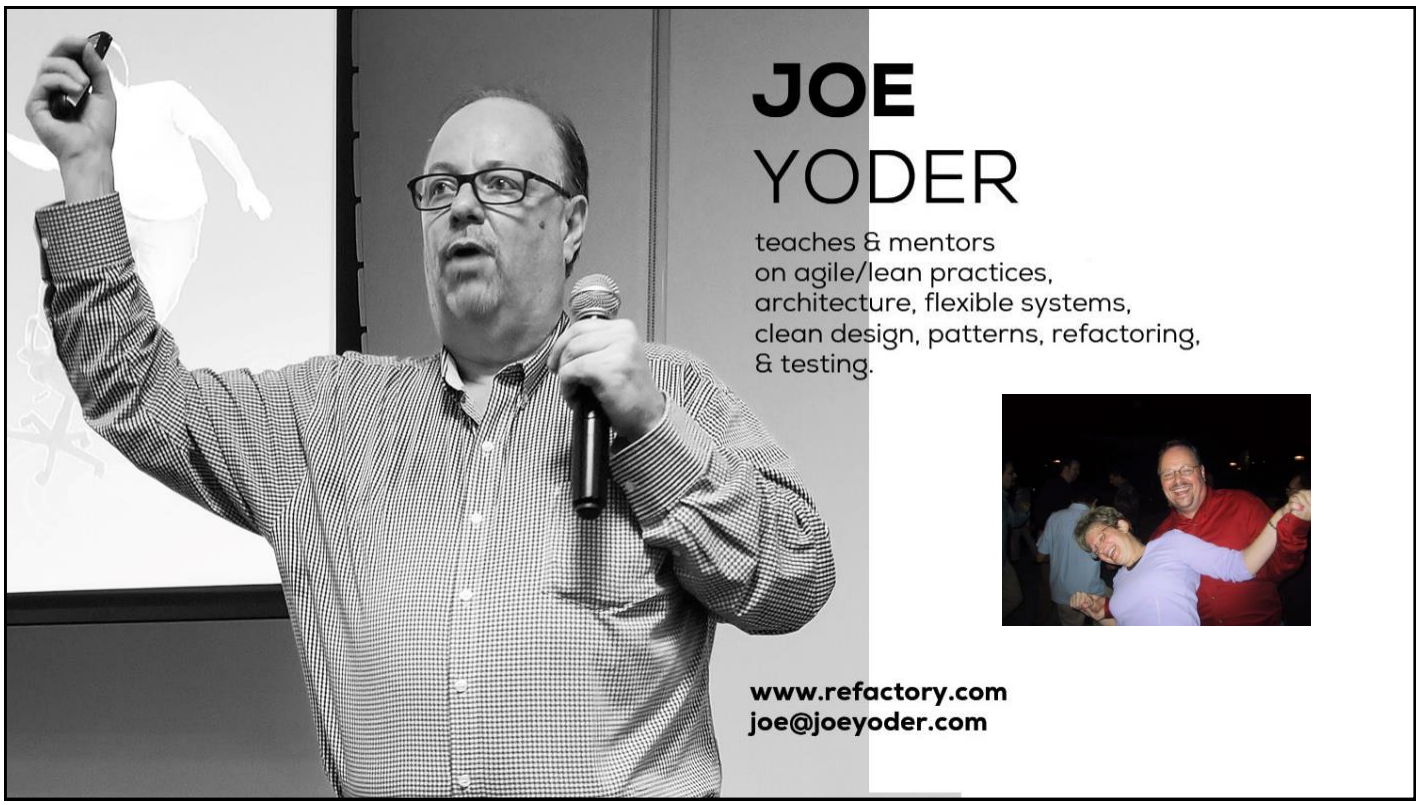


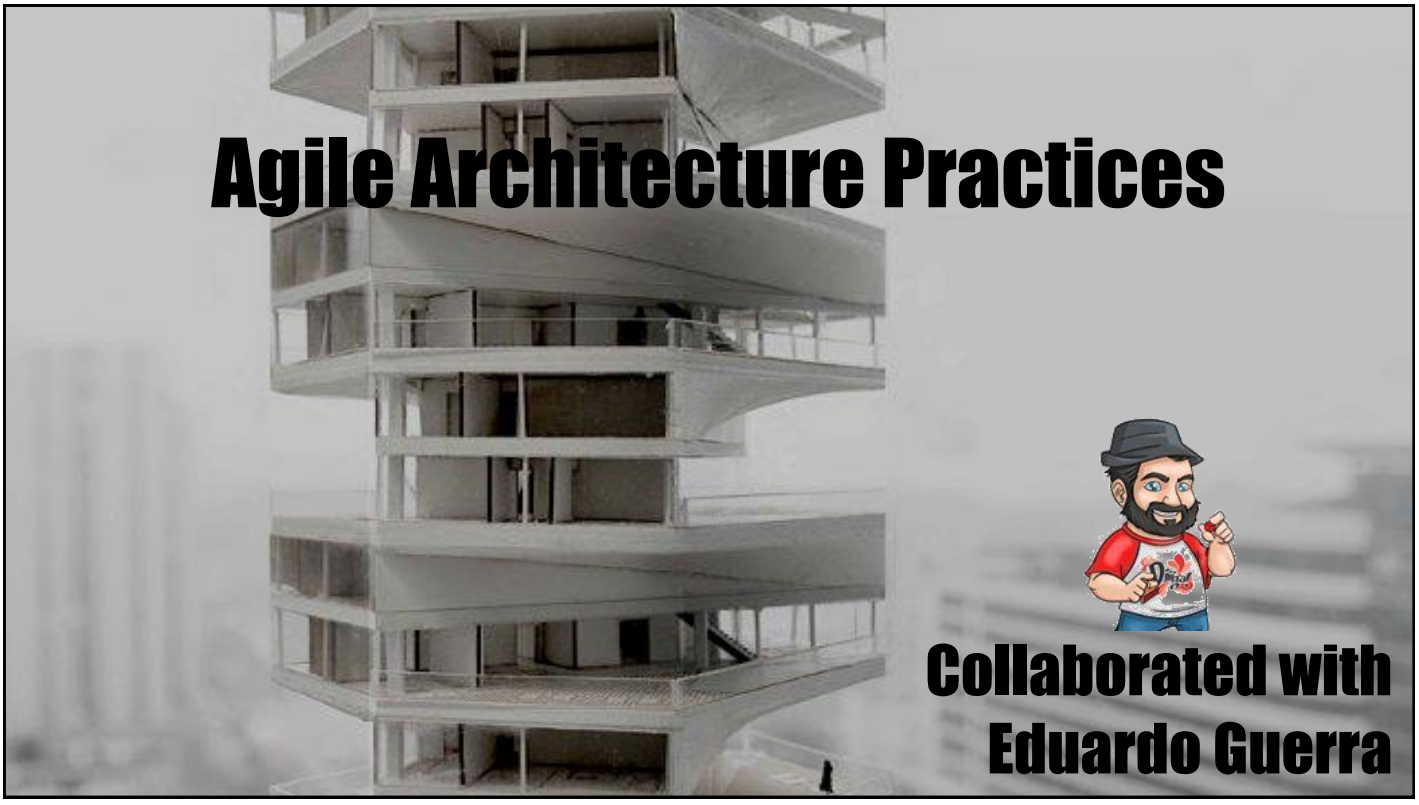


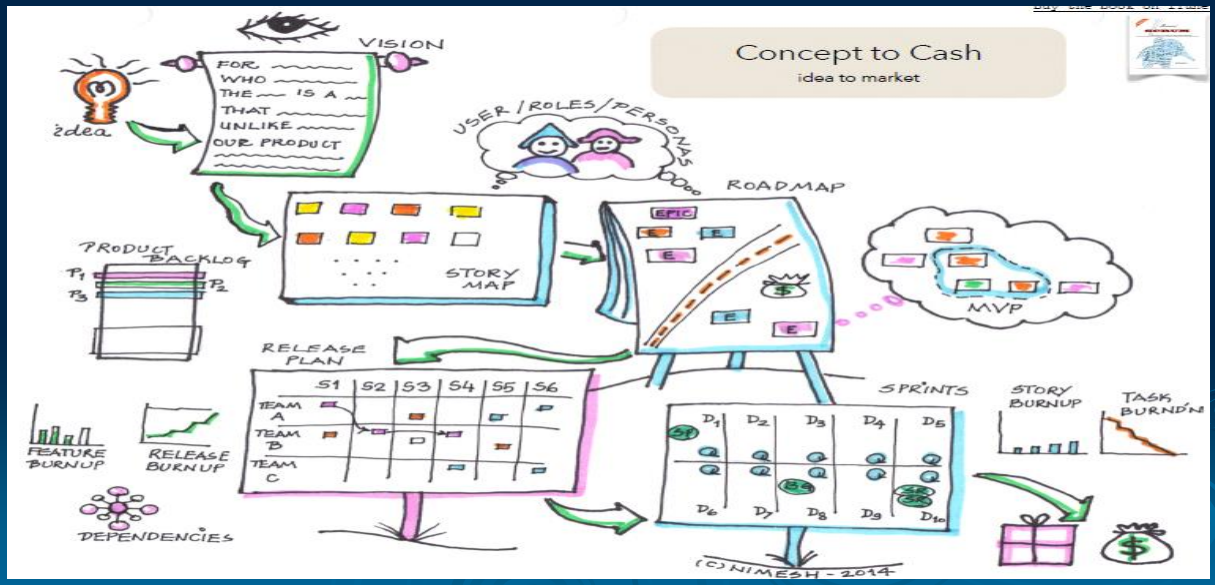
# Disclaimer

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.

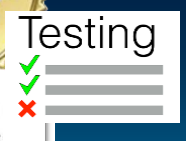


# Agile/Lean Processes



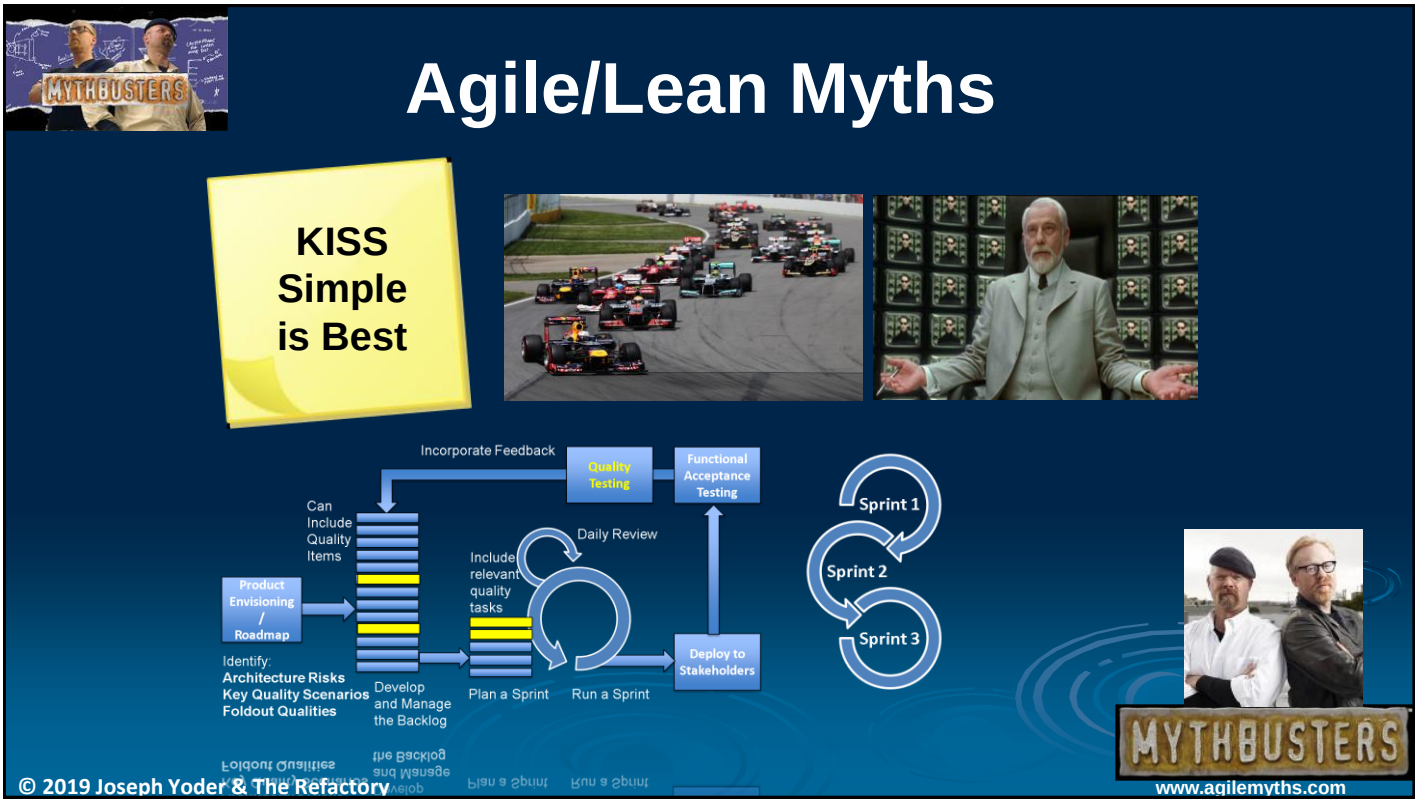

# Agile Principles and Practices





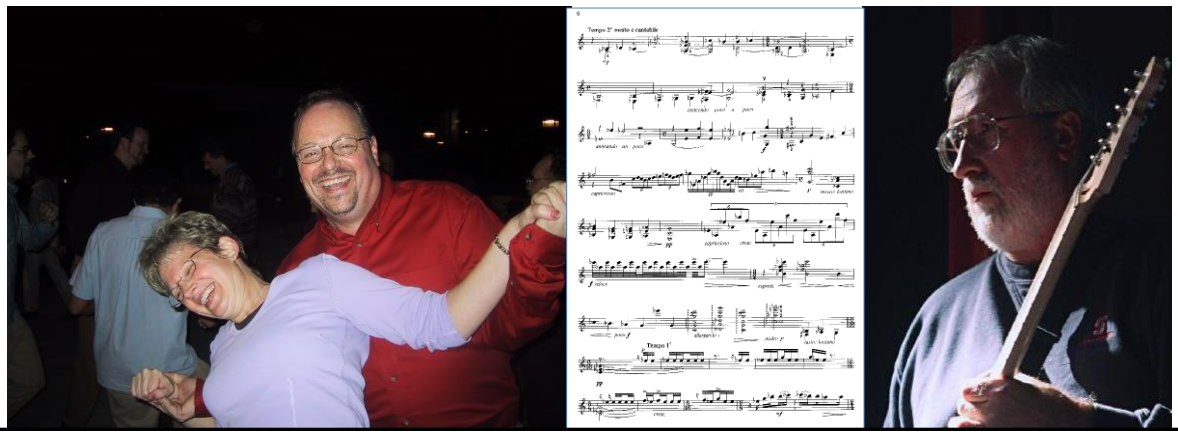





MYTHBUSTERS
Agile/Lean Myths
KISS
Simple
is Best
Incorporate Feedback
Quality Testing
Functional Acceptance Testing
Can Include Quality Items
Daily Review
Include relevant quality tasks
Product Envisioning / Roadmap
Identify:
Architecture Risks
Key Quality Scenarios
Foldout Qualities
Develop and Manage the Backlog
Plan a Sprint
Run a Sprint
Deploy to Stakeholders
Sprint 1
Sprint 2
Sprint 3
MYTHBUSTERS
www.agilemyths.com
© 2019 Joseph Yoder & The Refactory


values

# 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**"

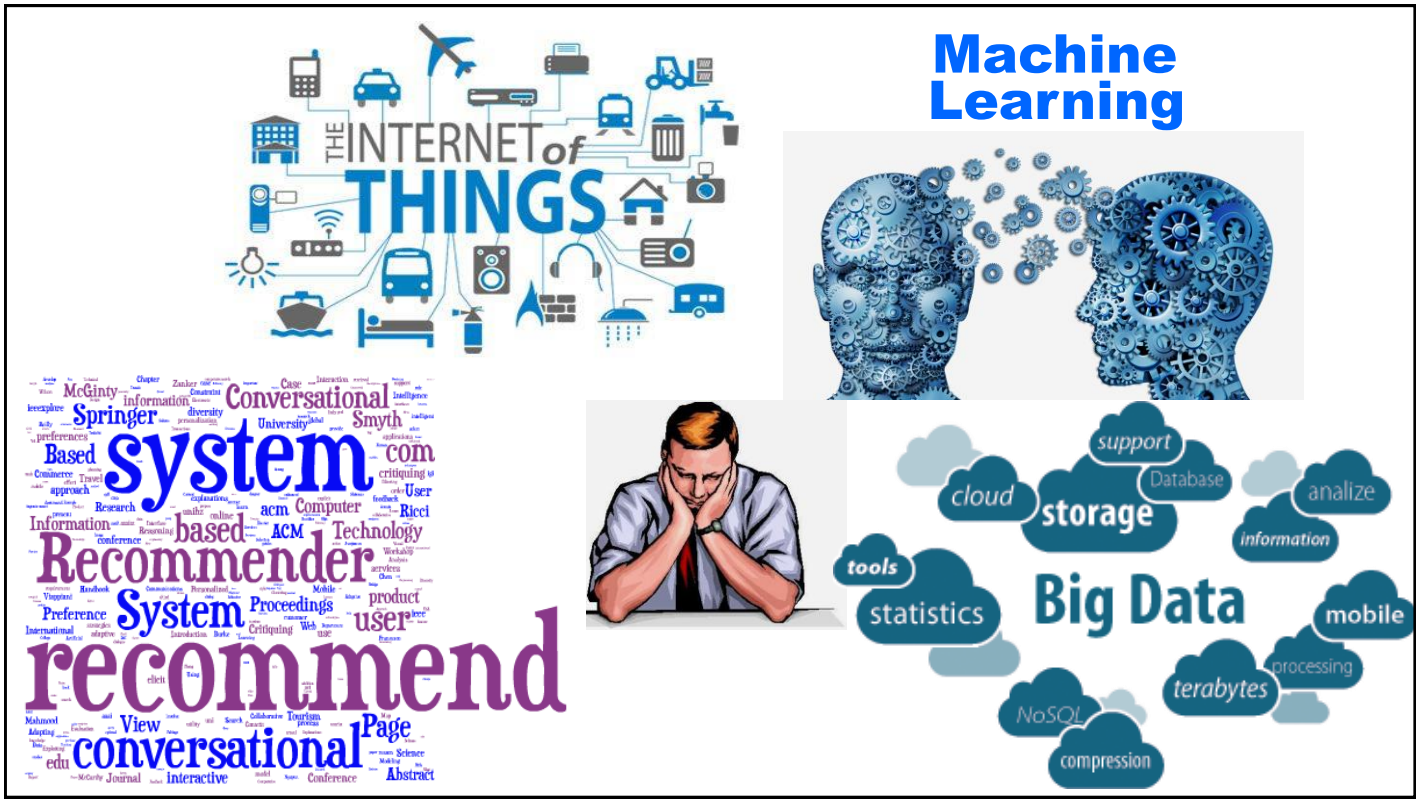
**Proven Solutions** to **Repeating Problems**

**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

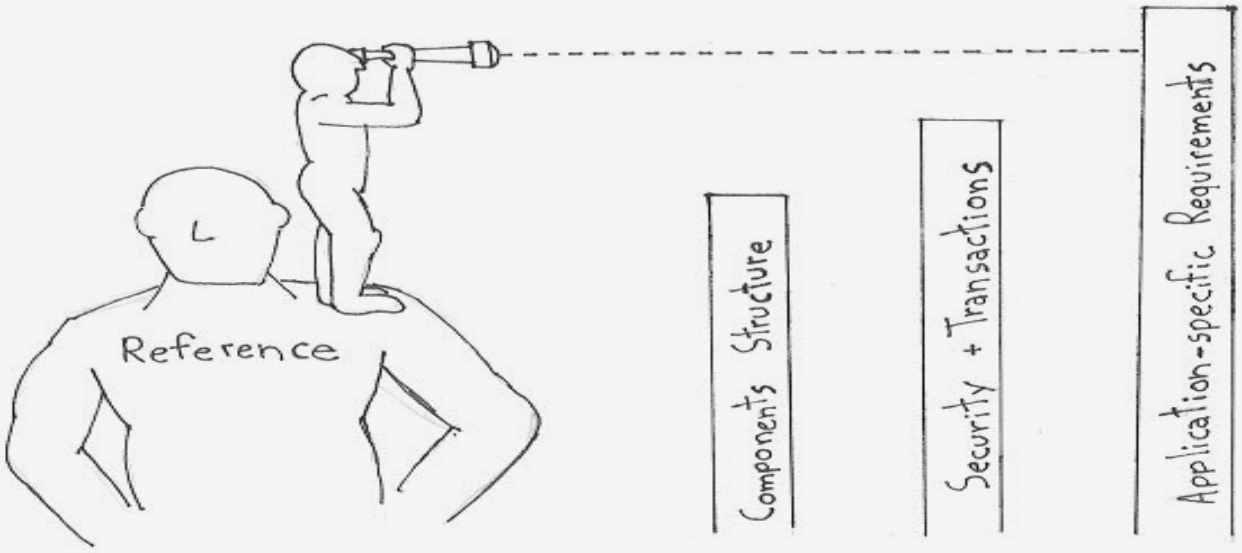



## Have a sustainable architecture
## Start the project fast, being agile

to start the architecture

**Climbing on the Shoulders of Giants**

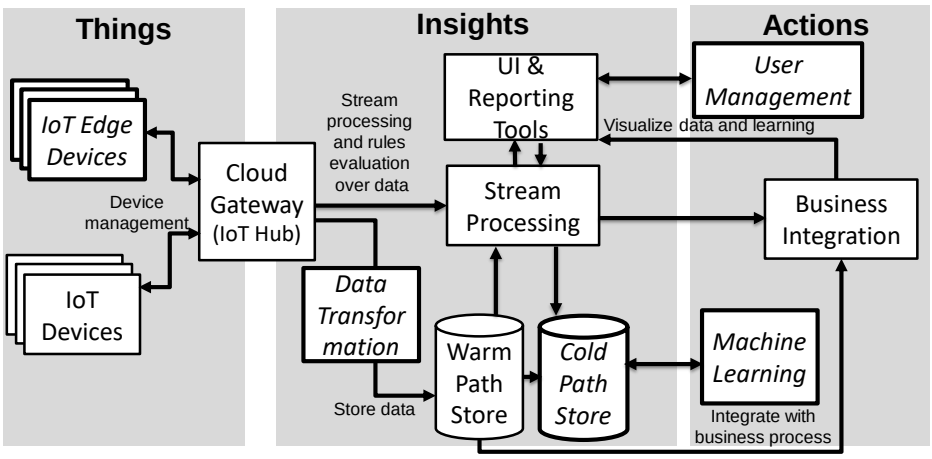
# Climbing on the Shoulders of Giants



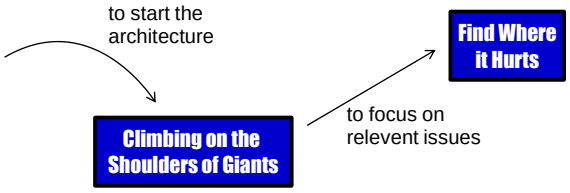

*Build on a Reference Architecture*

# IoT Reference Architecture

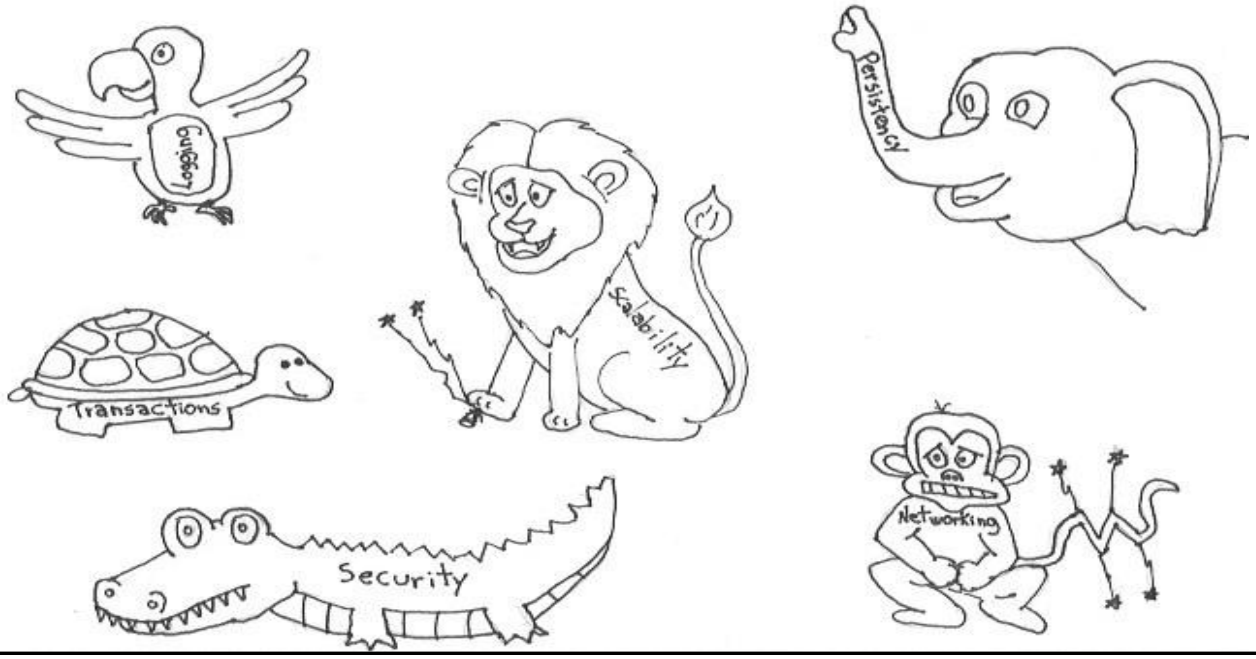
**[Azure IoT]**

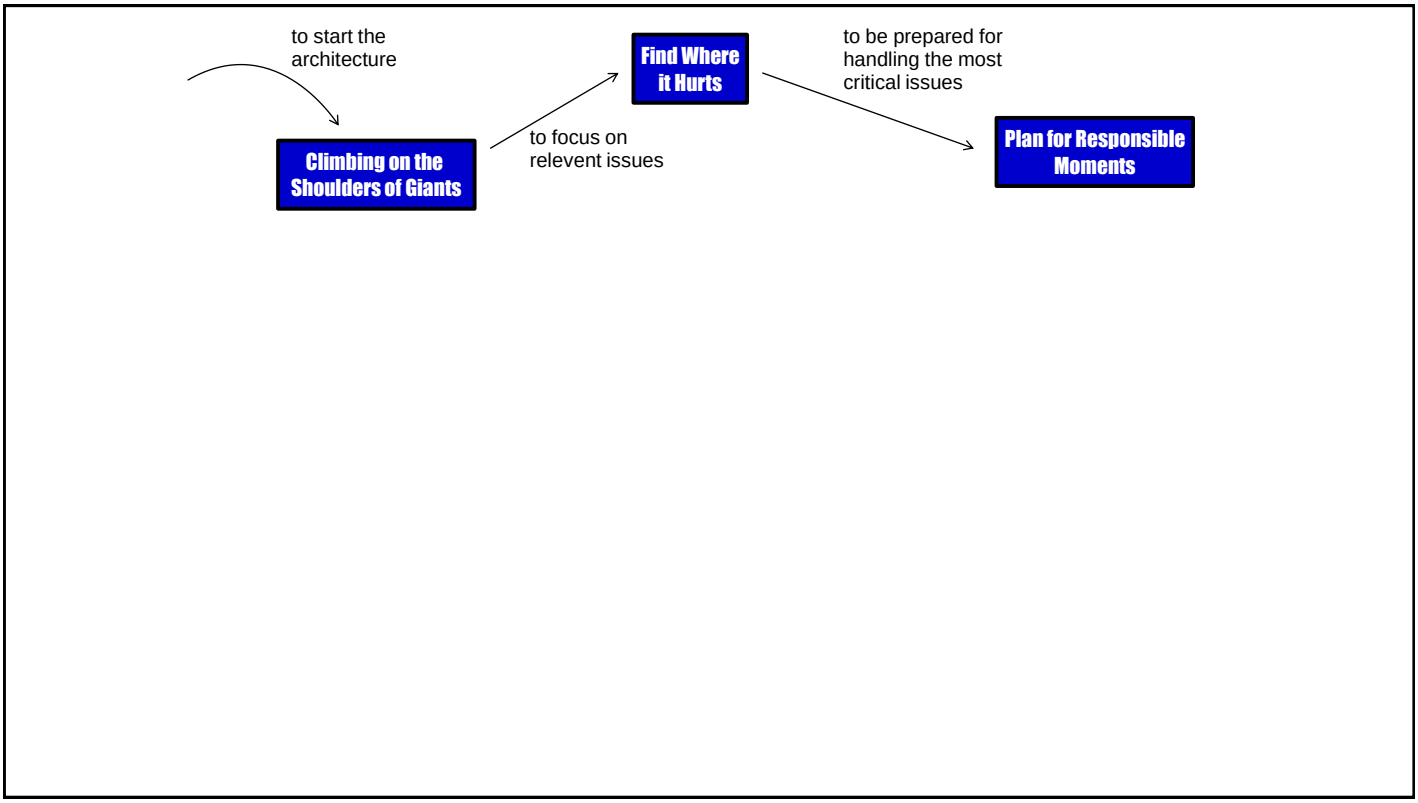


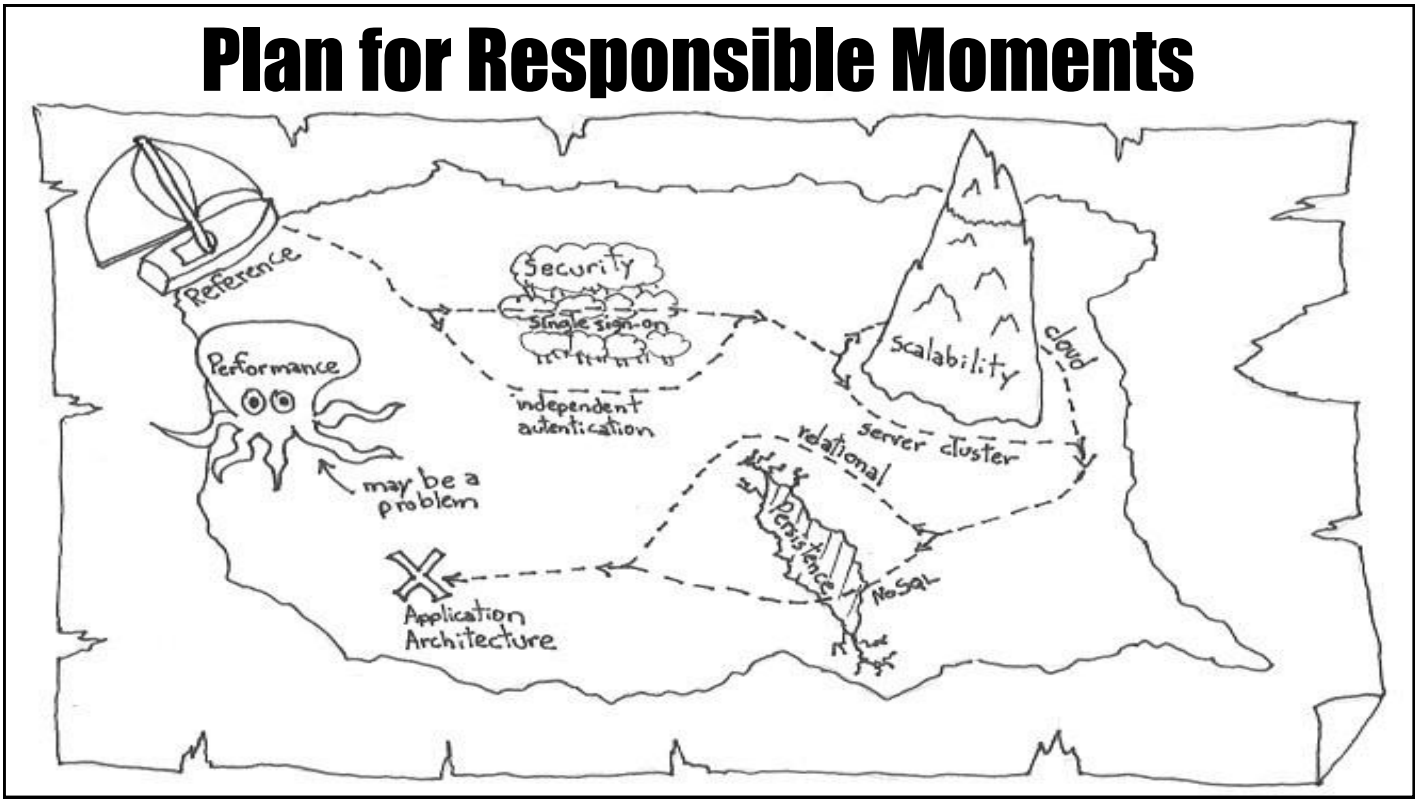

**Microsoft, Azure IoT Reference Architecture, https://aka.ms/iotrefarchitecture — 2018**




# Find Where It Hurts





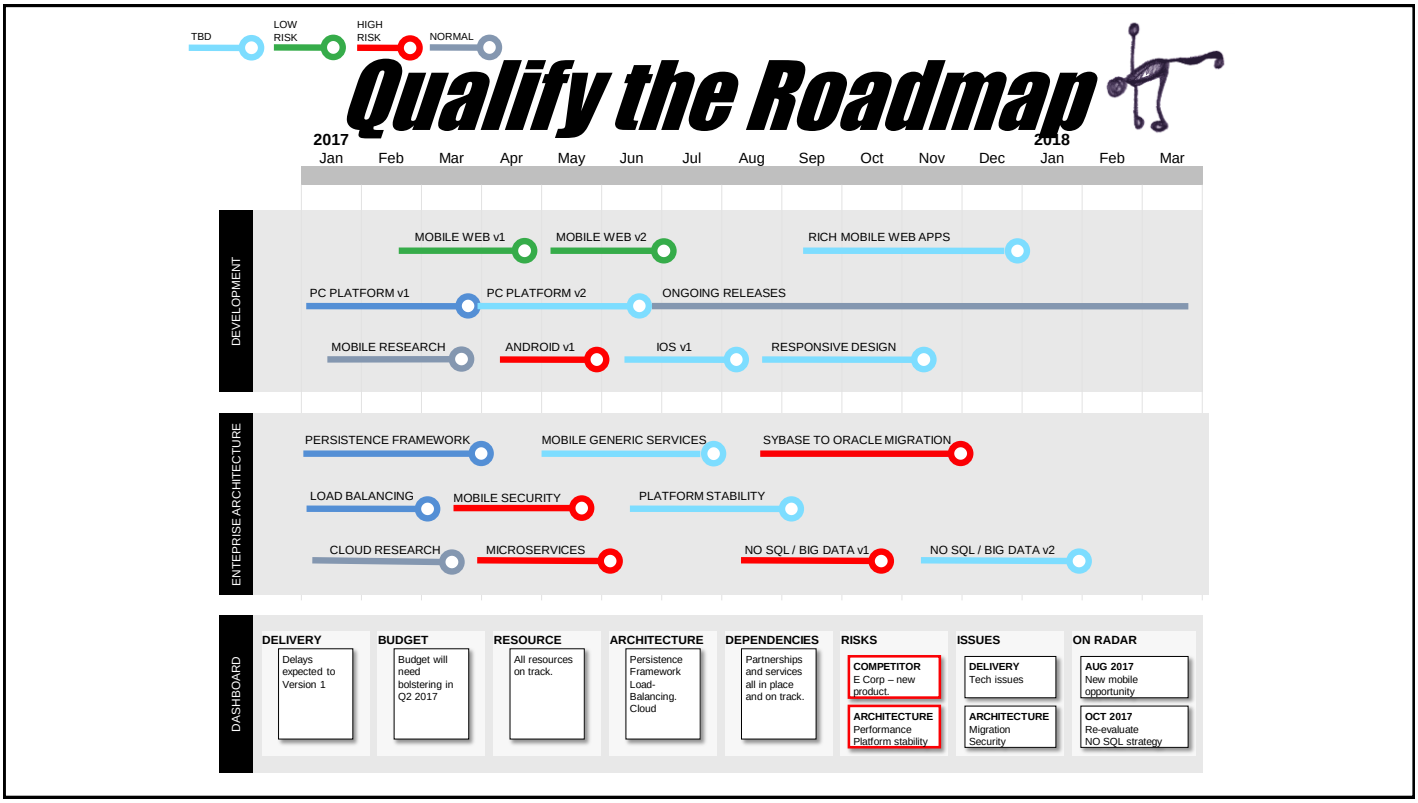
Some decisions and actions are too important to leave until The Last Responsible Moment

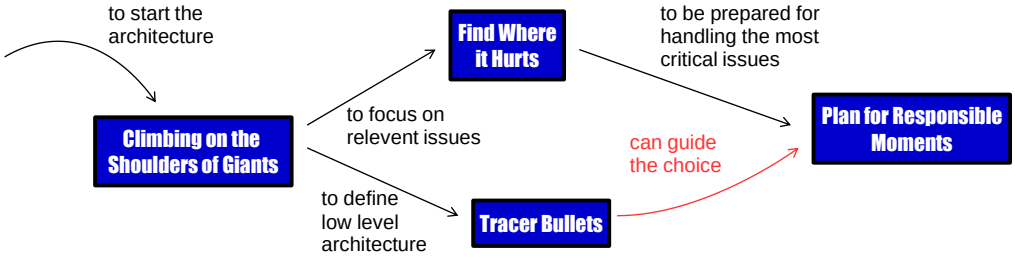
so

# Choose the Most Responsible Moment

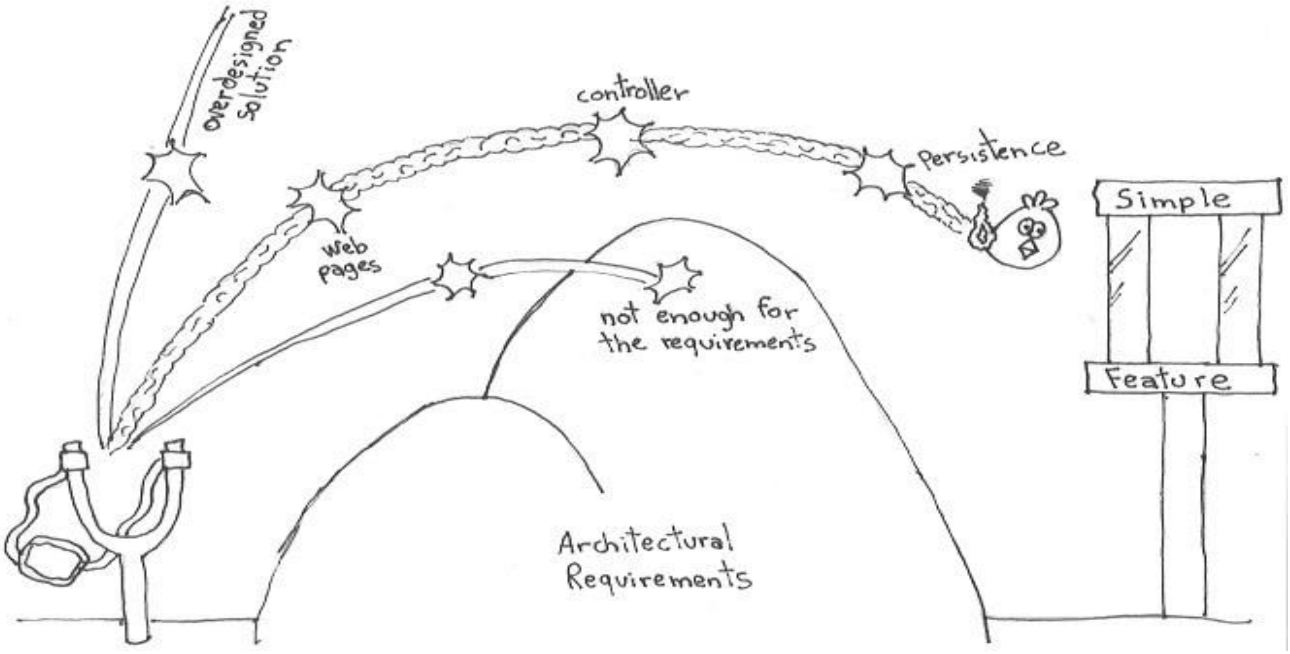
# Qualify the Roadmap with Architecture Decisions

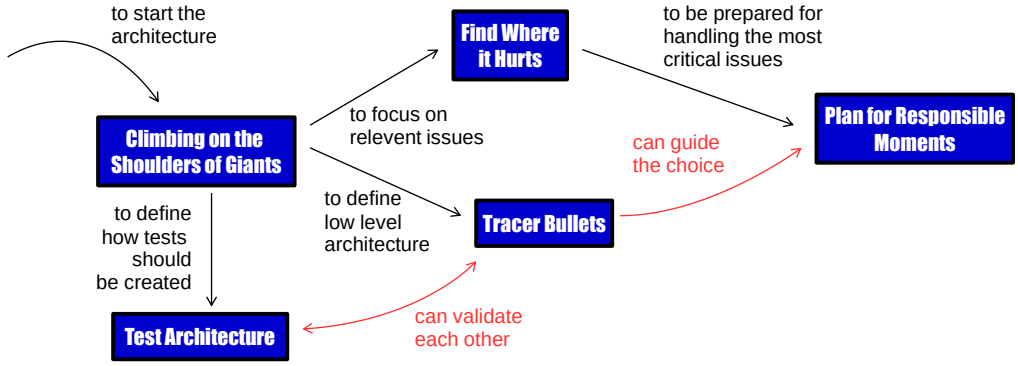
*"All you need is the plan, the roadmap, and the courage to press on to your destination"*
— Earl Nightingale



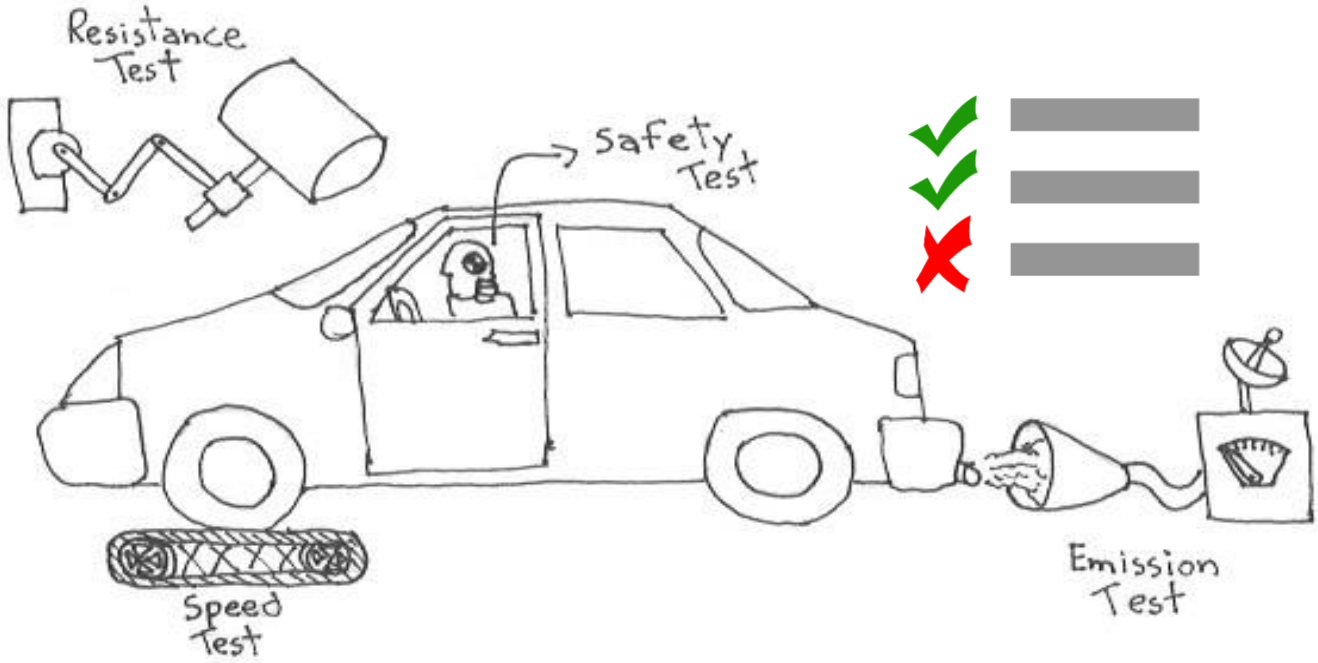



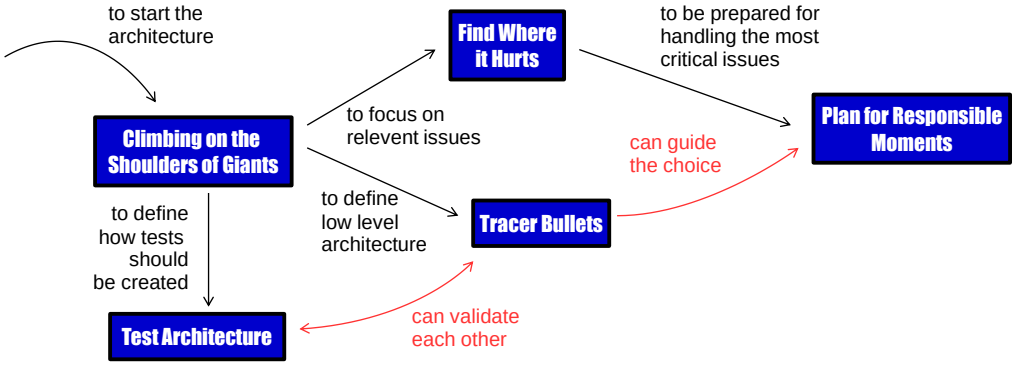
# Tracer Bullets



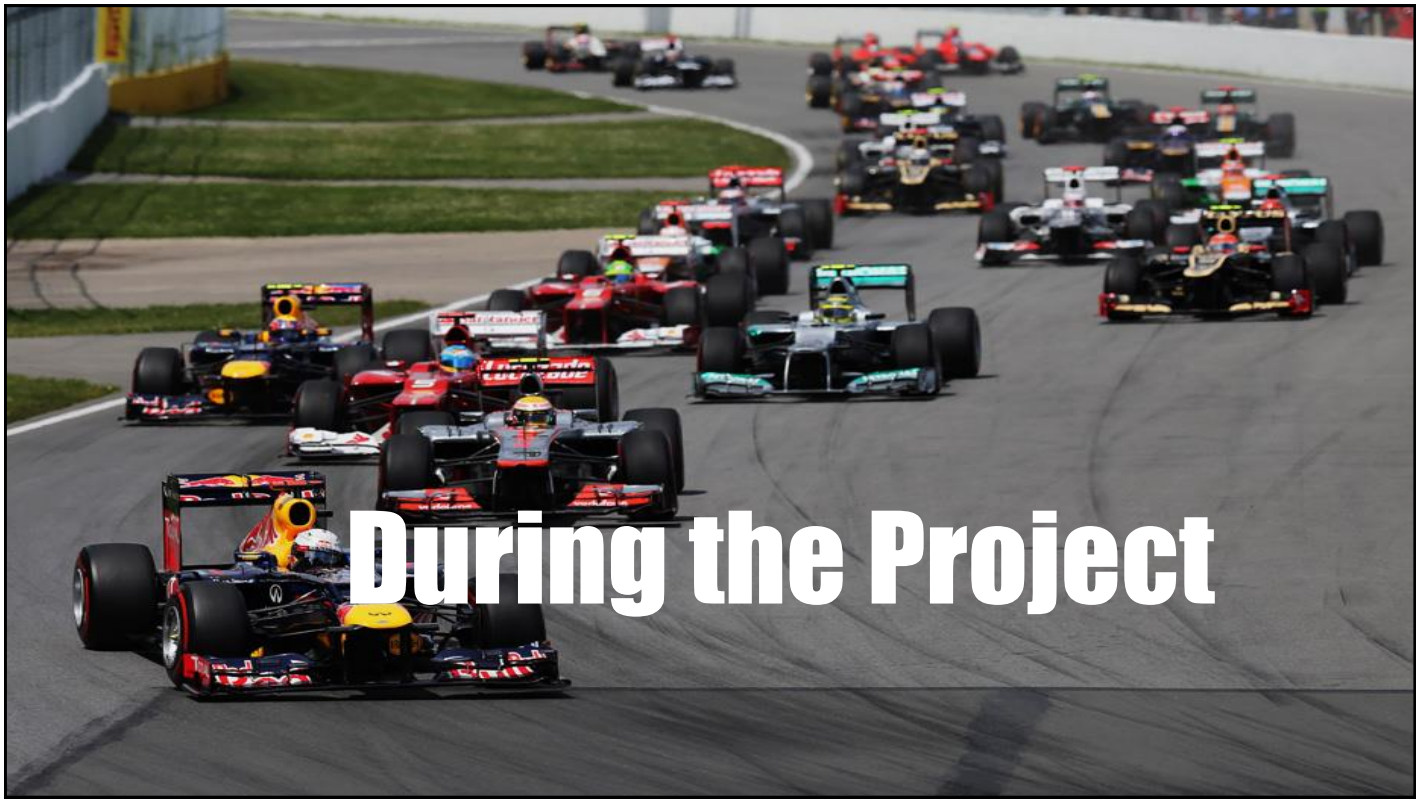



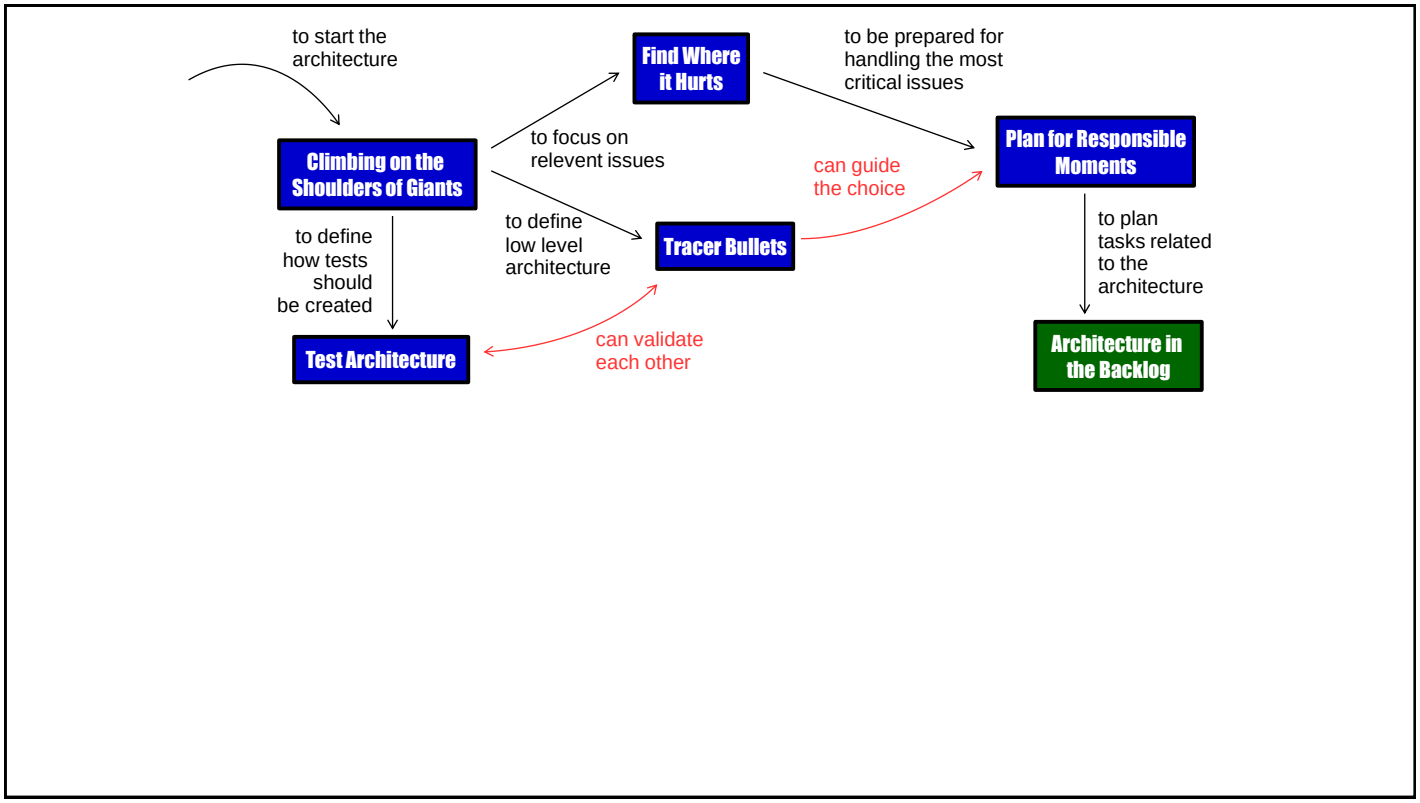
# Test Architecture



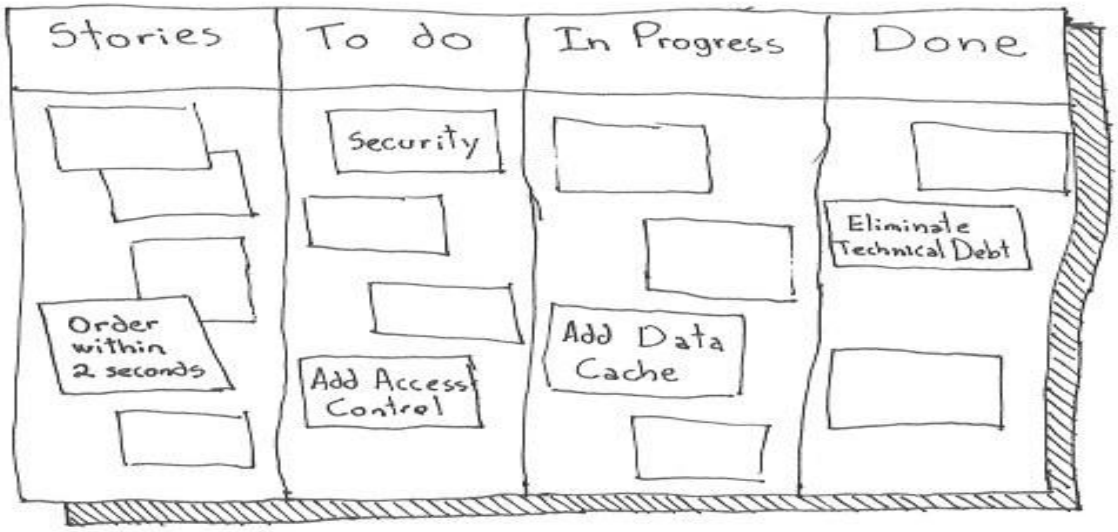




**Note these can be done throughout the project...**

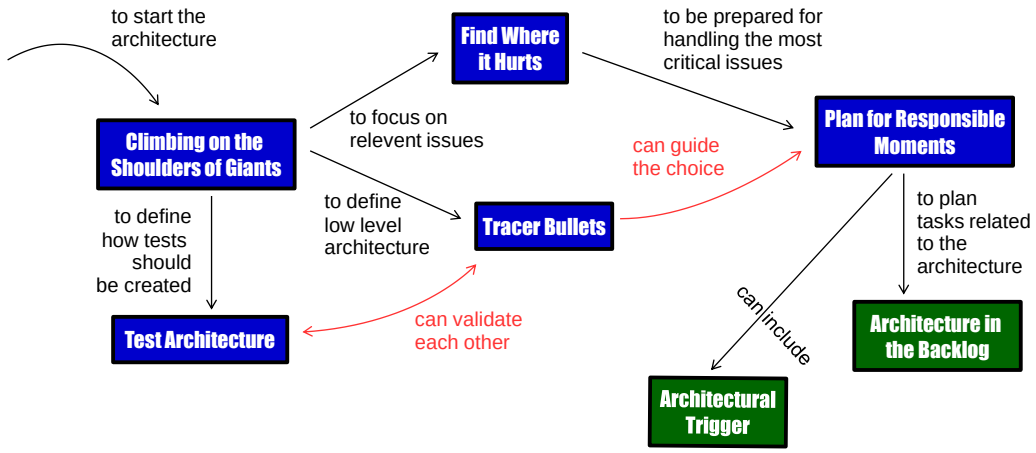


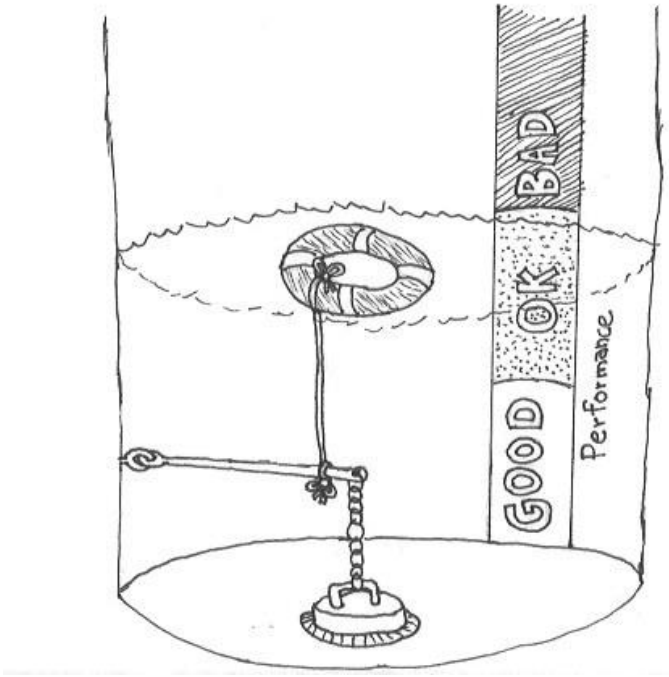
# Architecture in the Backlog



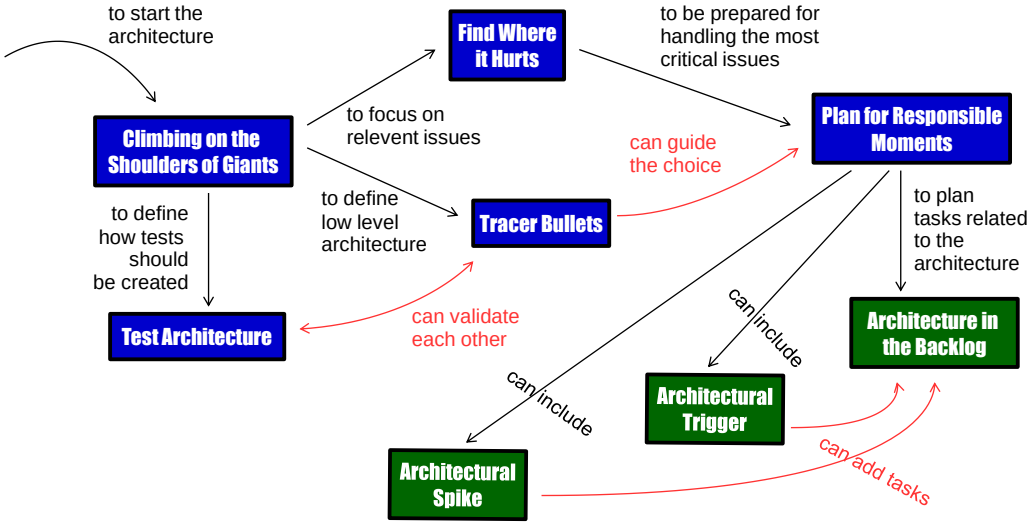

**You can add backlog items for technical debt and quality-related architecture work… "yes, you can"**



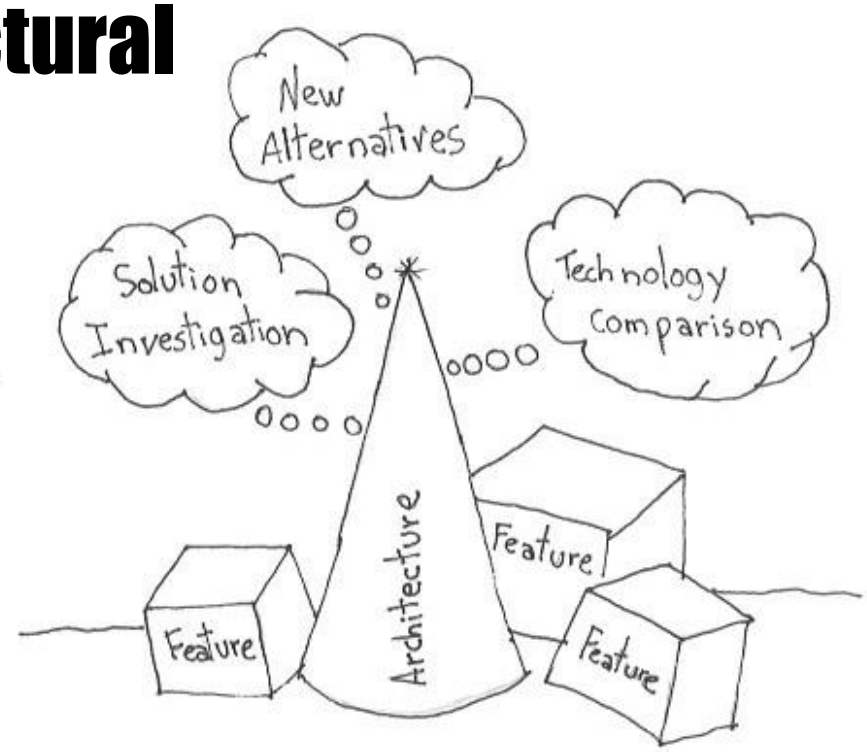
# Architectural Trigger




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



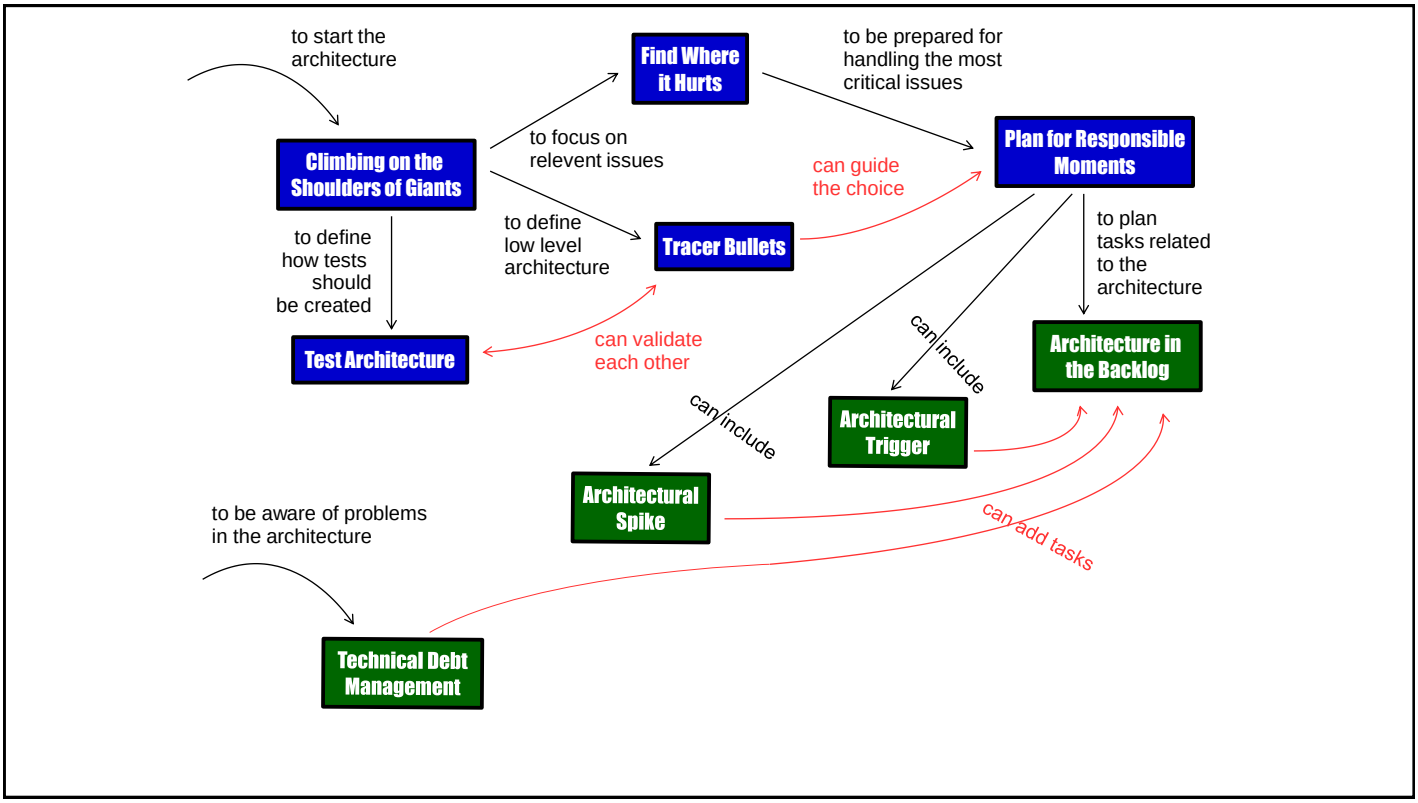
# Architectural Spike



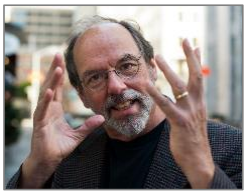

## Architecture Spikes & Explorations

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

- Can be a Sprint!!!



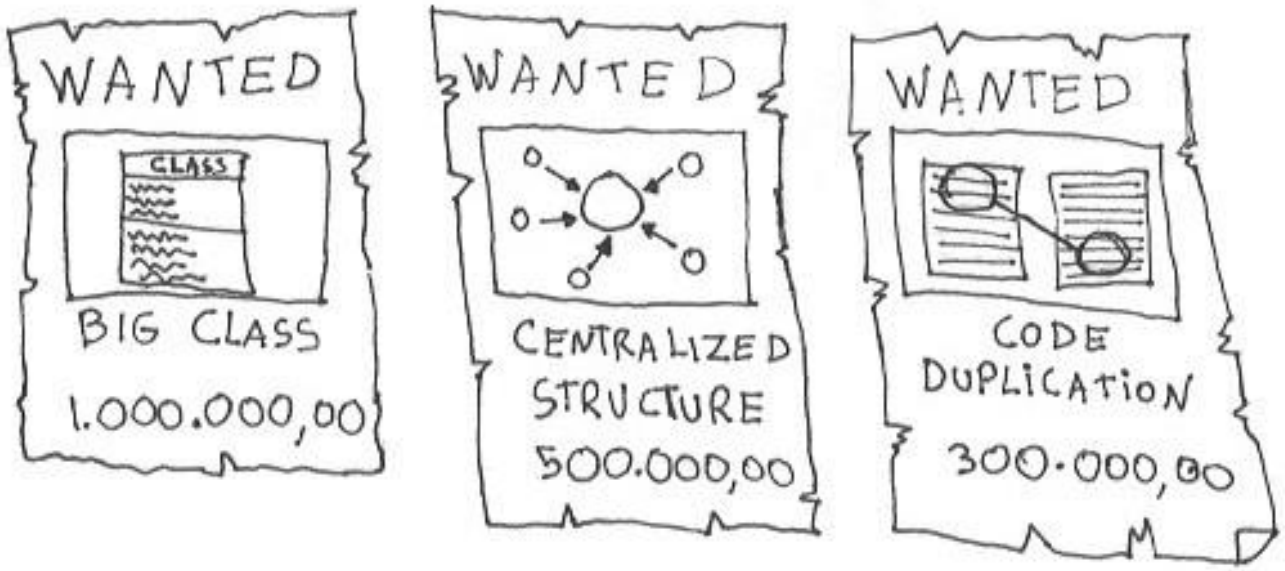

Some dirt becomes very hard to clean if you do not clean it right away!



**Architecture Debt, Quality Debt, Test Debt, …**

# Technical Debt?

# Technical Debt Management

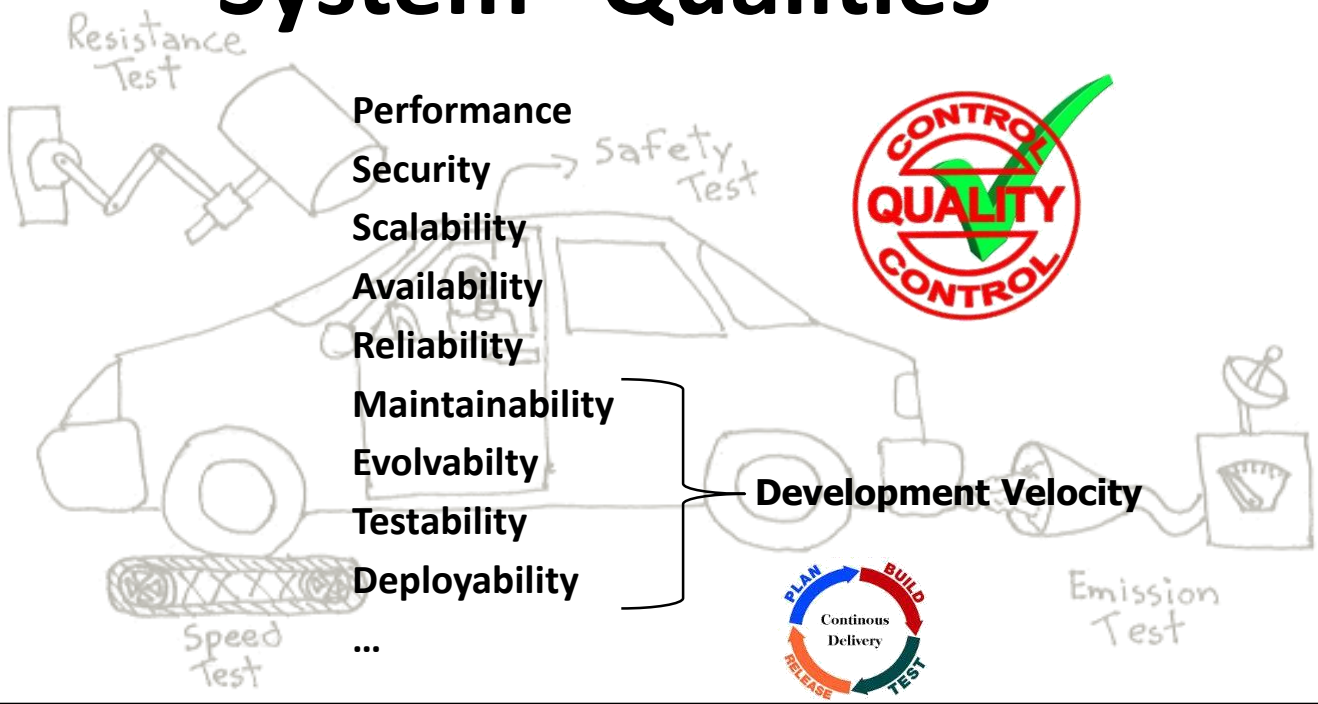



# Technical Debt Management




Graziela Simone Tonin
Alfredo Goldman

# System “Qualities”




**Performance**
**Security**
**Scalability**
**Availability**
**Reliability**
**Maintainability**
**Evolvabilty**
**Testability**
**Deployability**
**…**

→ **Development Velocity**

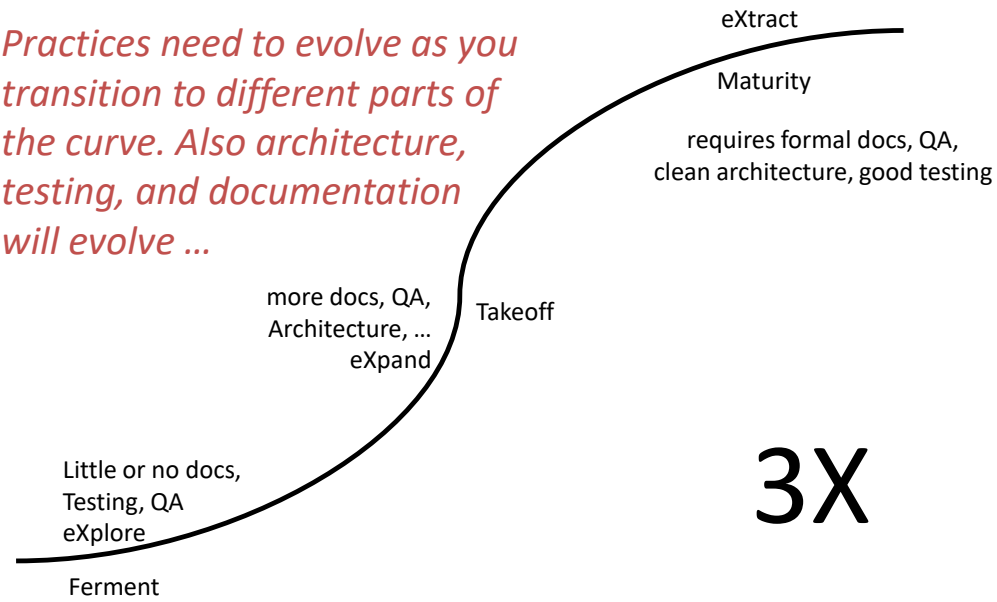




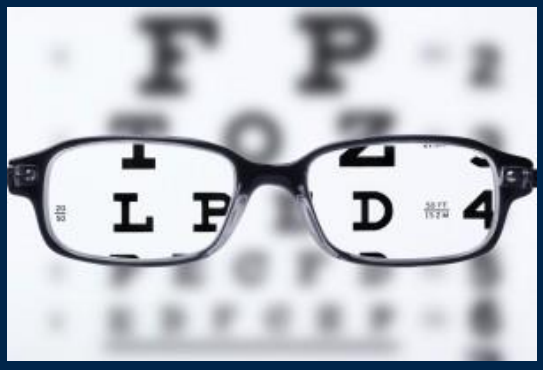


# The Business ‘S’ Curve

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




3X

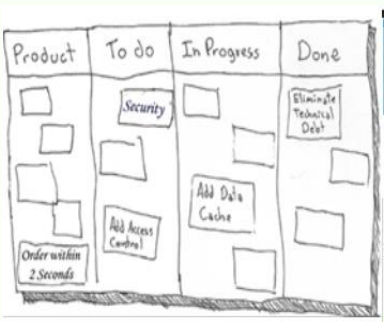


**HOW CAN WE IMPROVE WHAT WE CANNOT SEE?**



# Visibility is Key

## “Radiators”

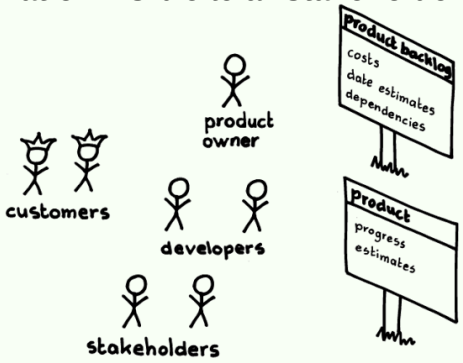




## ¶56 Information Radiator

Without valuable and timely information, the organization dies.

*Therefore:*

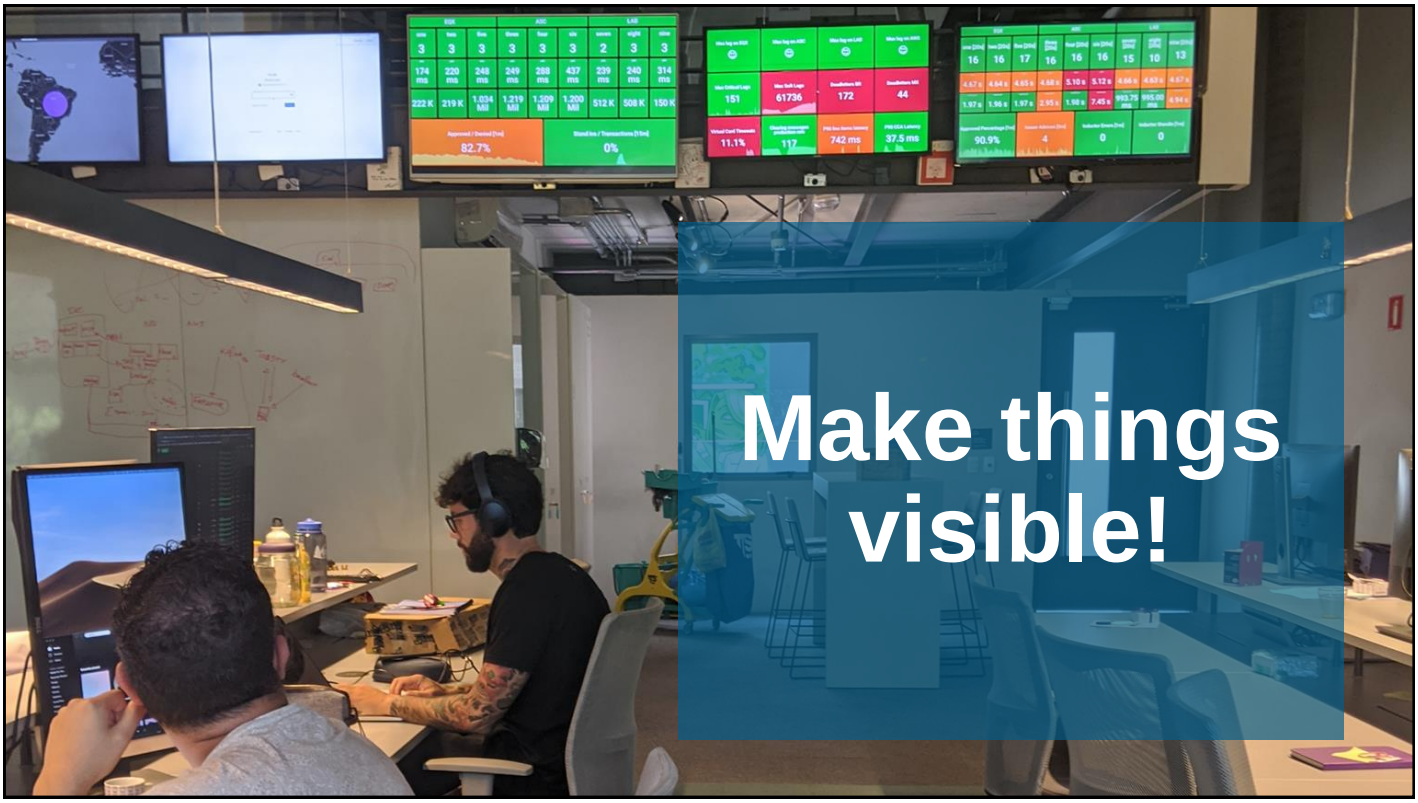
Collaboratively maintain **physical artifacts** that keep **information visible** to all **stakeholders.**

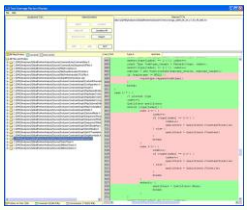


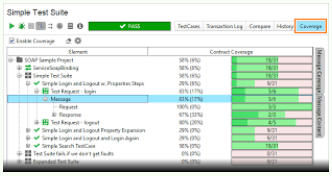


https://pragprog.com/book/jcscrum/a-scrum-boo










# Know your audience!

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



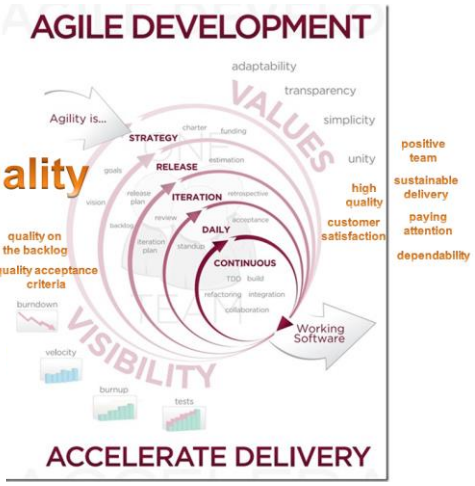
Make Visible

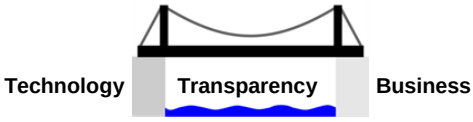
# Transparency

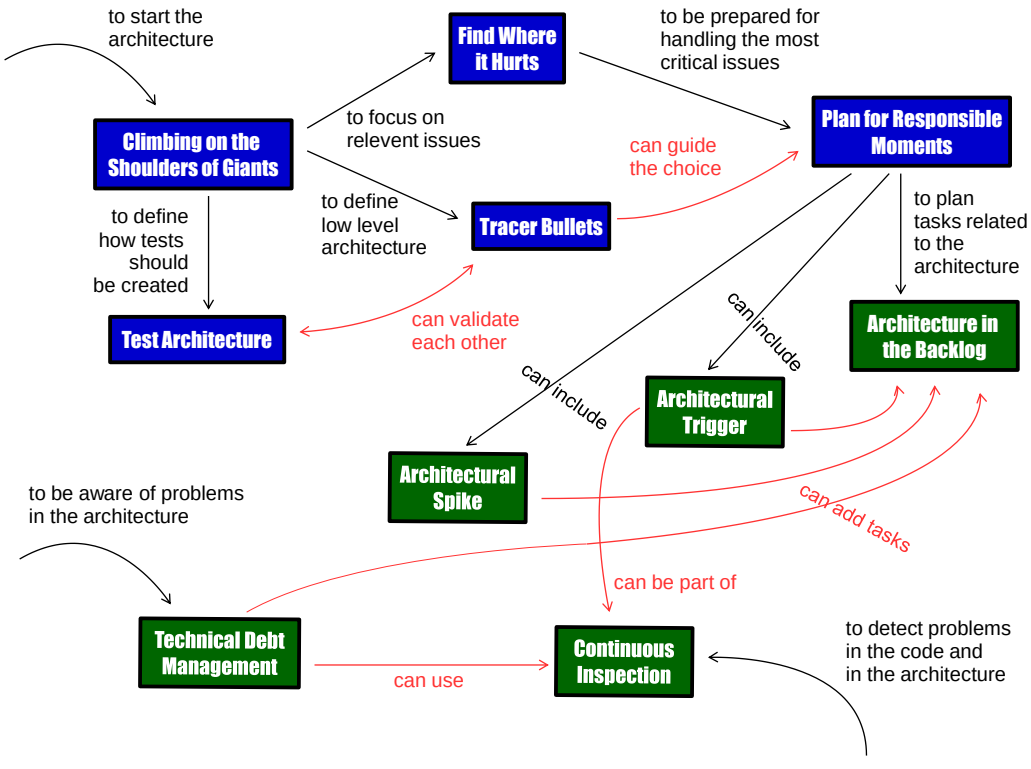
Agile Lean Core values:

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

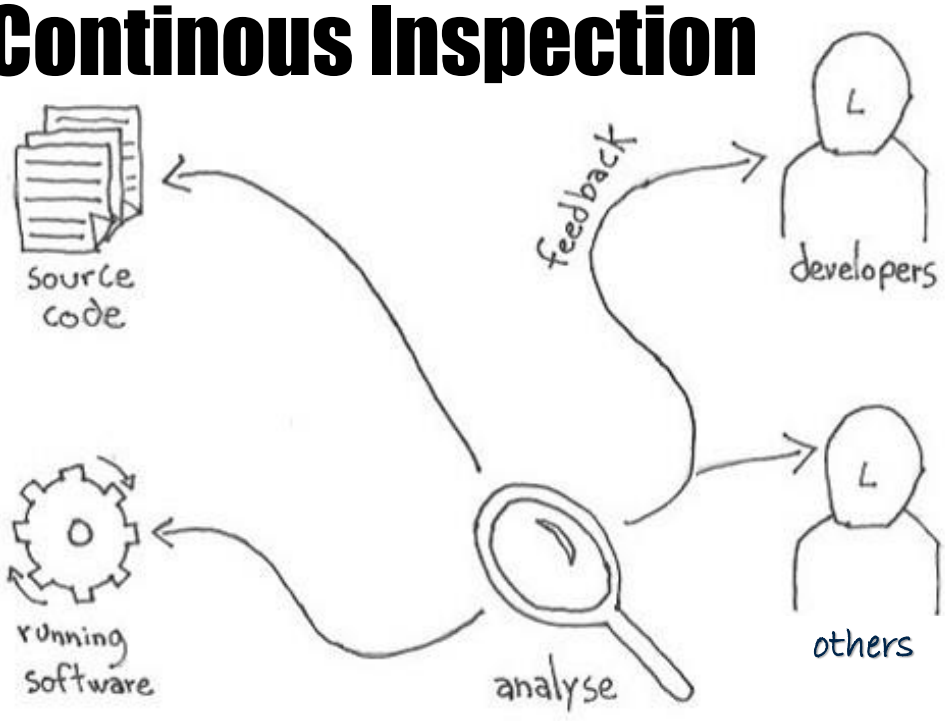
Tech Backlog ties into business values
"Part of the Business Backlog???"



# Continous Inspection



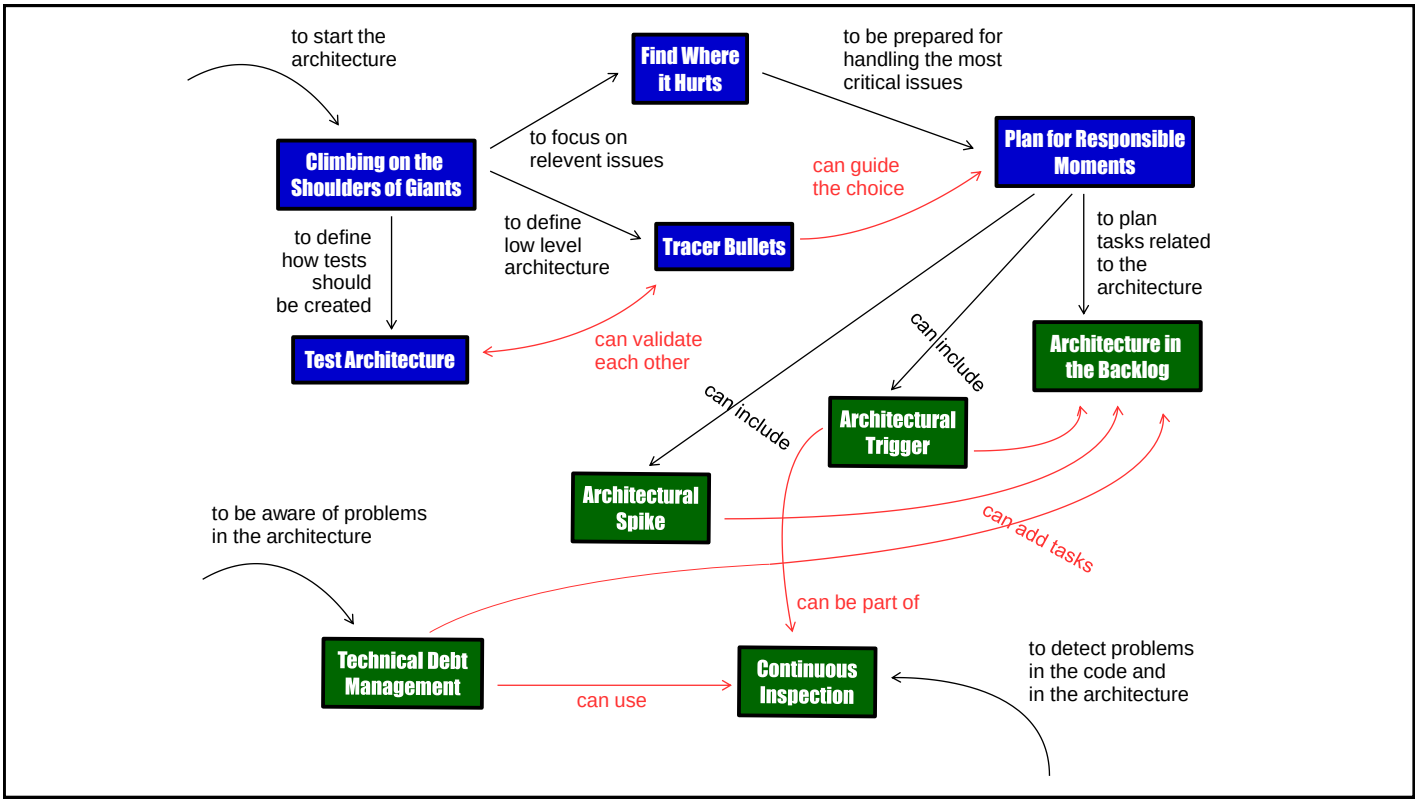

# Automate As You Go

*"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

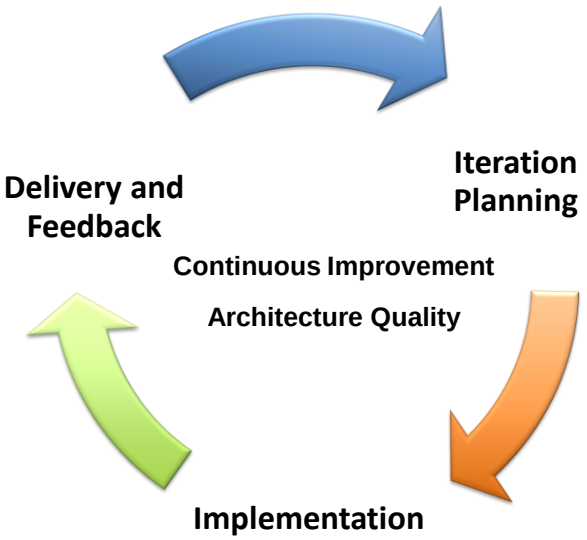
Notes/checklists
- Less frequent but important
- Harder to automate → inconsistent → requires thought




# Have a sustainable architecture

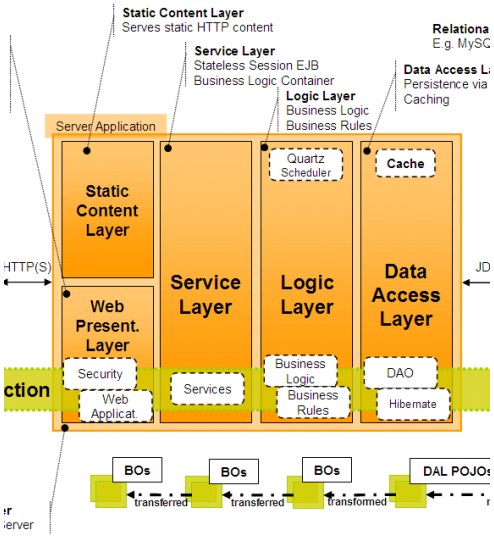
# Start the project fast

# Periodically Re-Evaluate Architecture Risks



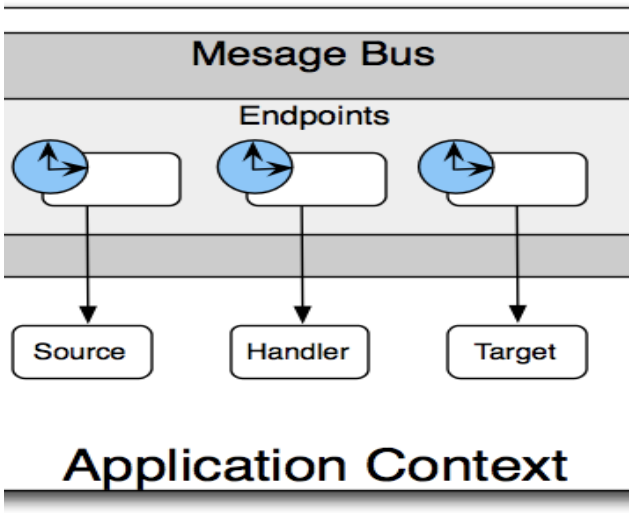

# Incrementally Test Key Components' Performance

- 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

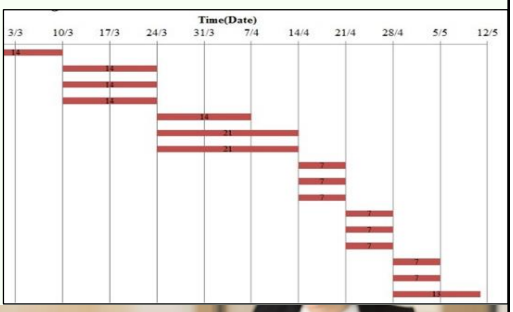
# Slack Time

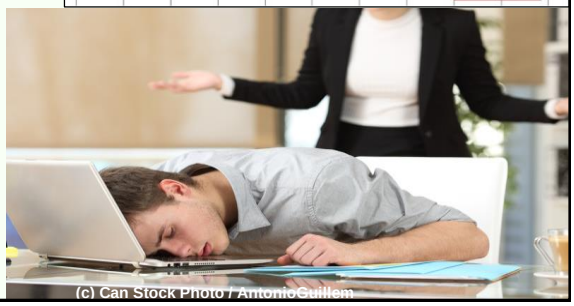
***Need Slack time to improve and to ensure quality***

**Ways to get slack time…**

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

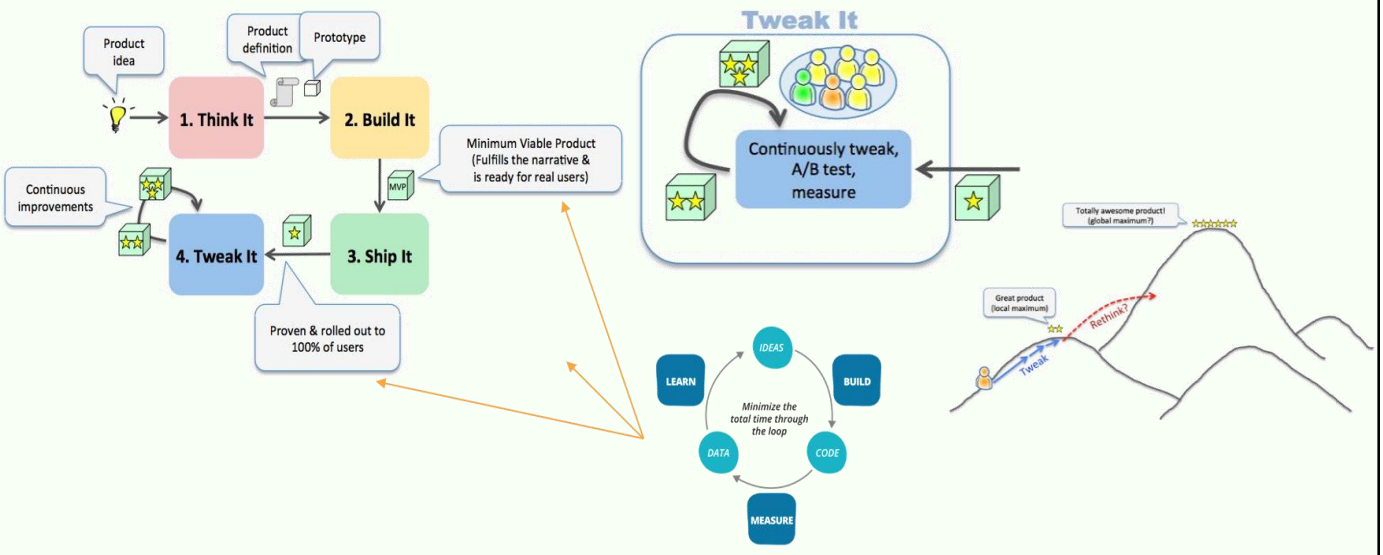
Try little experiments…



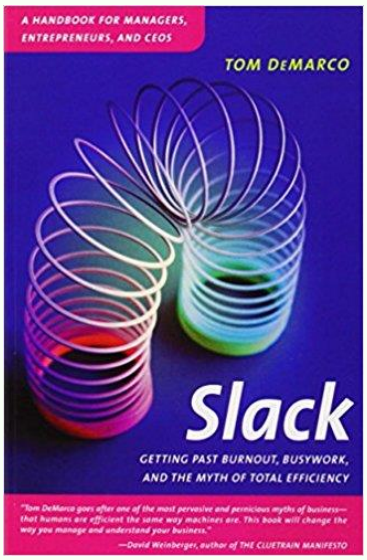



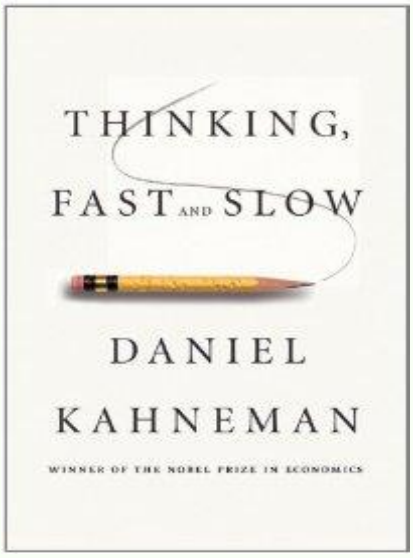



# Spotify: Innovation



# Slack Time





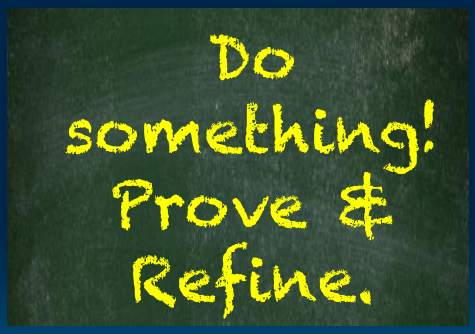

**Velocity ≠ Productivity**

**Velocity ≠ Value**

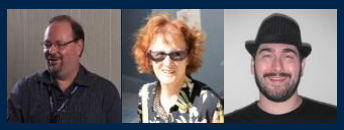
**Going Fast isn't always good!!!**

# 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

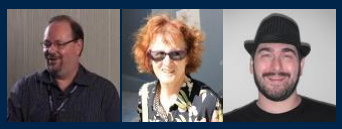



# Patterns for Evolving Agile Architecture

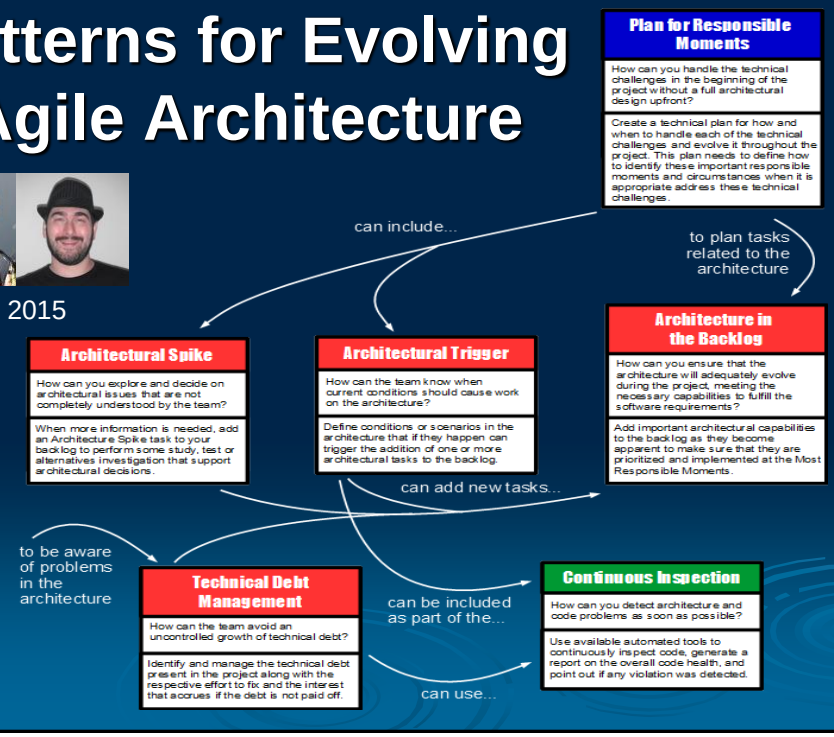


Asian PLoP 2015

# Patterns for Evolving Agile Architecture

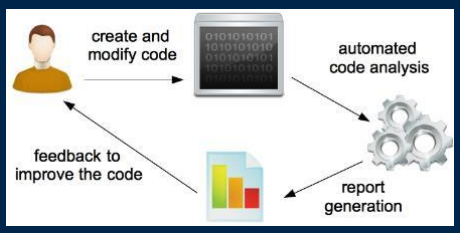


USA PLoP 2015



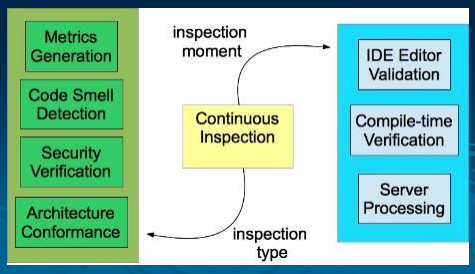

# Continuous Inspection

Asian PLoP 2014 Paper



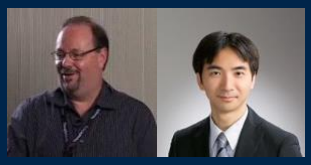

- *Code Smell Detection*
- *Metrics (Test Coverage, Cyclomatic Complexity, Technical Debt, Sizes, …)*
- *Application Security Checks*
- *Architectural Conformance*

*Automate where you can!!!*

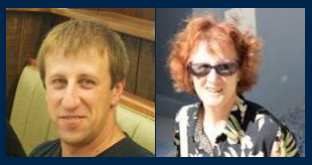



Sustaining Your Architecture

# 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. 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.*


Categories 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

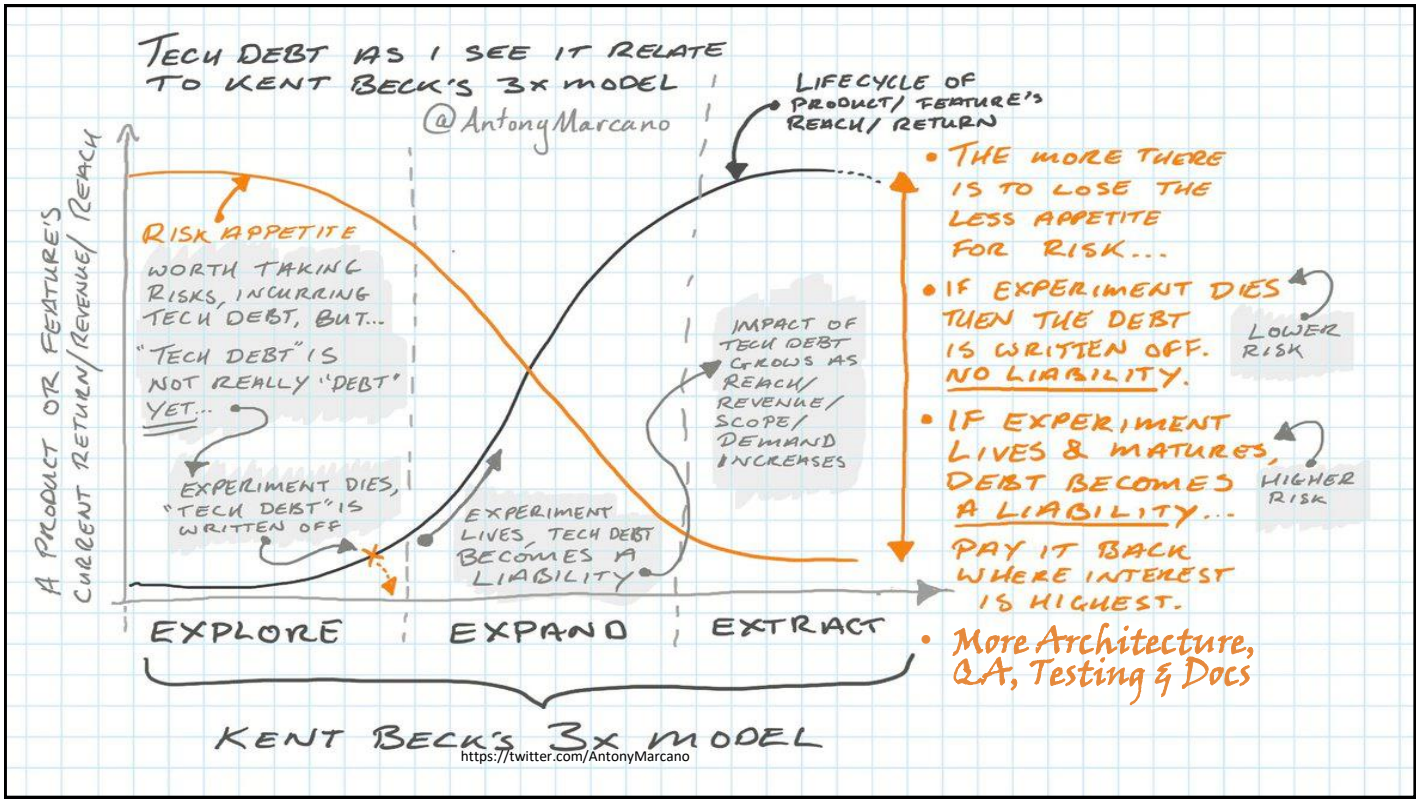
Continuous Inspection: A Pattern for Keeping your Code Healthy and Aligned to the Architecture, AsianPLoP 2014

Patterns for Initial Architecture Design on Agile Projects AsianPLoP 2015

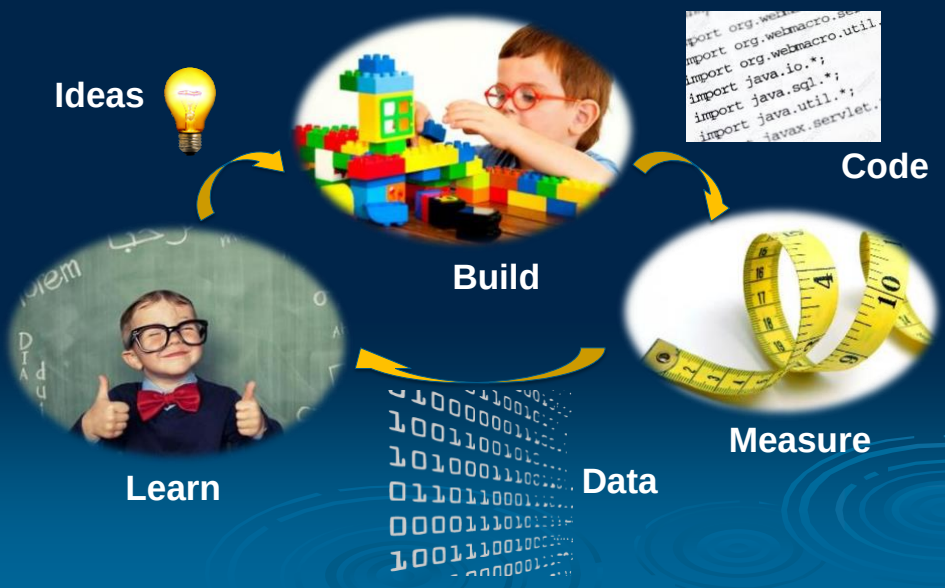
Patterns to Develop and Evolve Architecture in an Agile Project, PLoP 2016

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

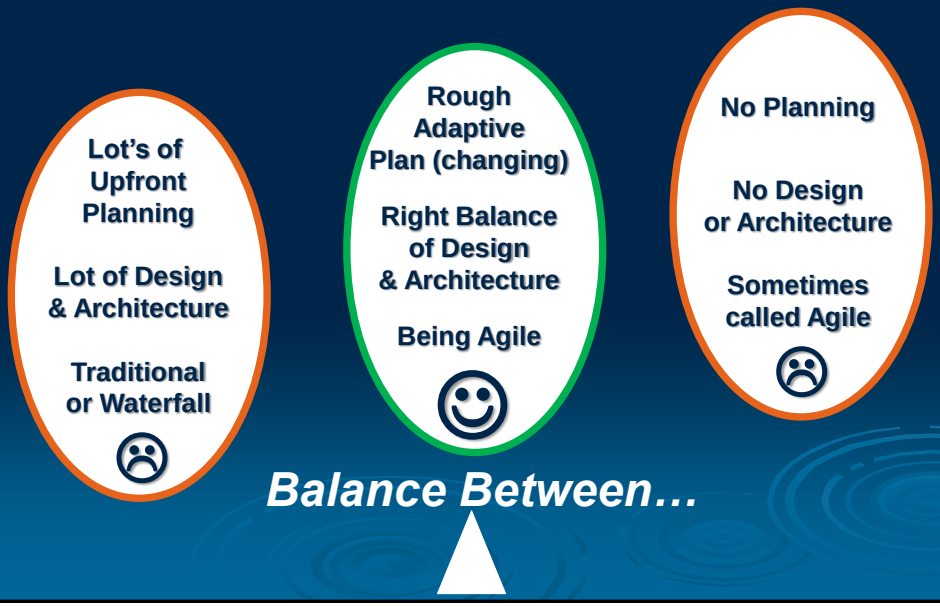
Copies available off our websites

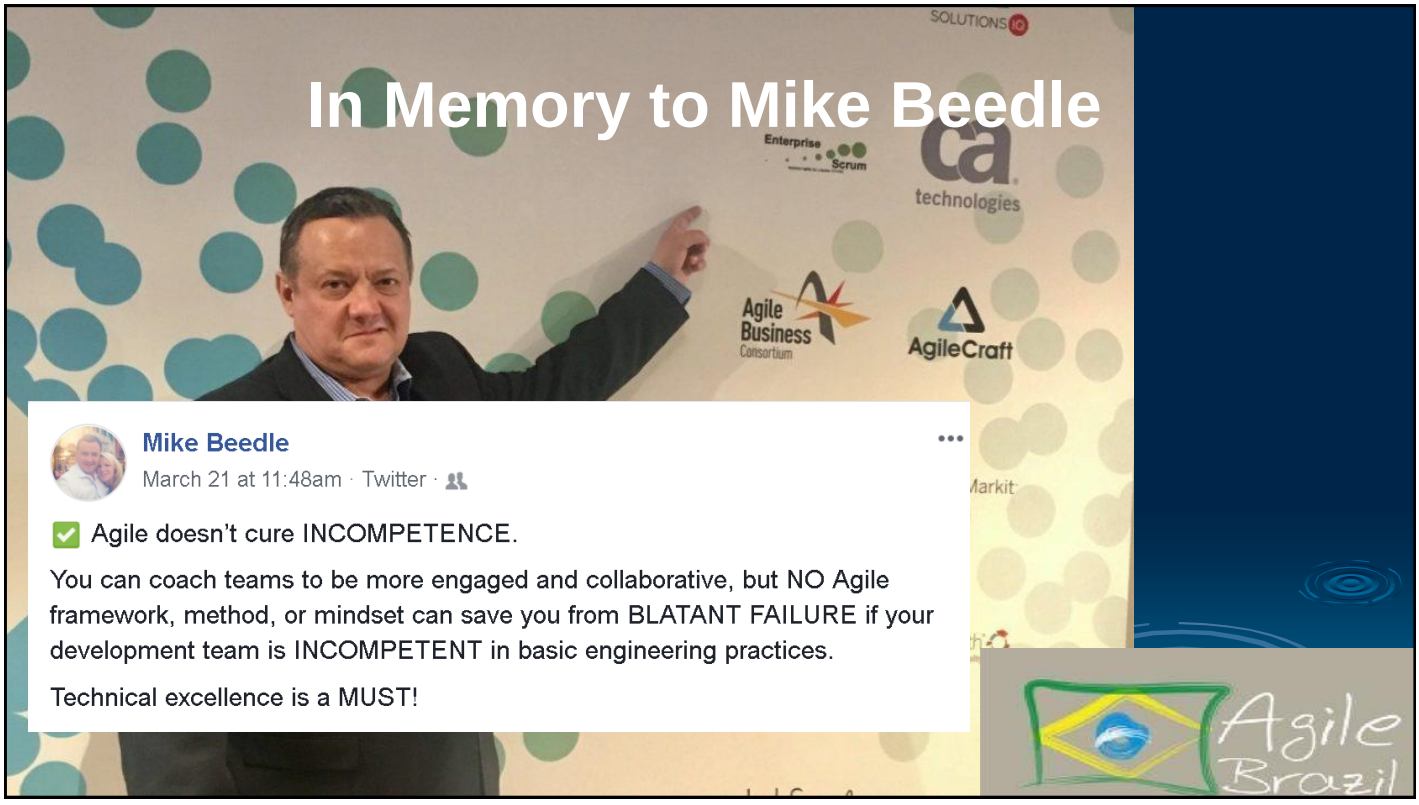



# Agile/Lean

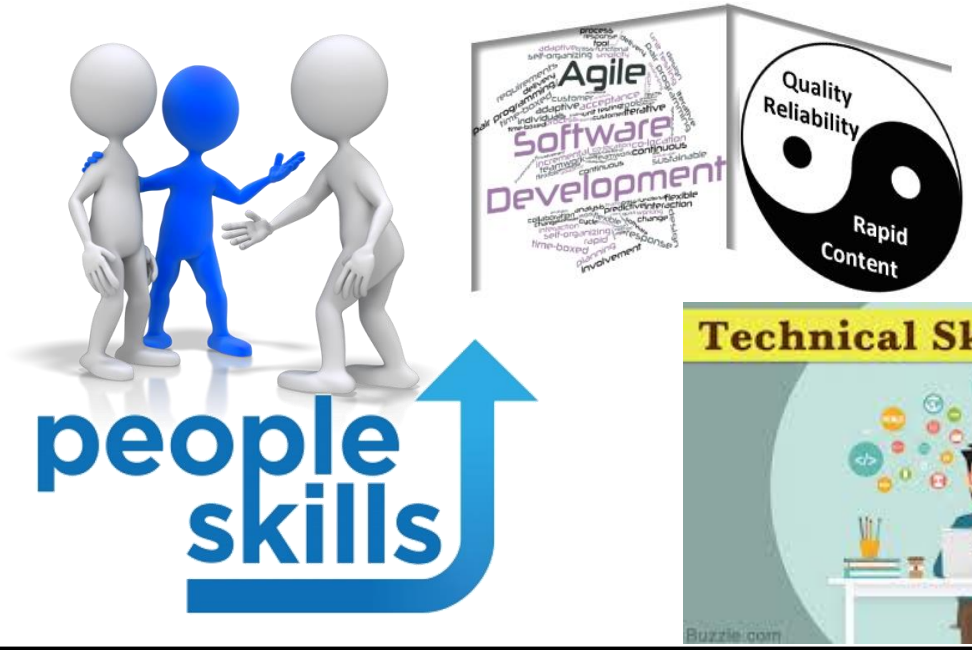



# Being Pragmatic






# 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!



yodamann

joe@refactory.com

@metayoda

**“You can’t fix what you can’t see”**
**“If you think good architecture is expensive, try bad architecture"**