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
Agile/Lean Myths

KISS
Simple is Best

Incorporate feedback
Daily Review
Run a Sprint

Sprint 1
Sprint 2
Sprint 3

Deploy to Stakeholders

Functional Acceptance Testing

Identify and Control Quality Issues

Develop and Test

www.agilemyths.com

© 2019 Joseph Yoder & The Refactory
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
Before First Iteration
Have a sustainable architecture
Start the project fast, being agile

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
Climbing on the Shoulders of Giants to start the architecture

Find Where It Hurts to focus on relevant issues

Find Where It Hurts
Climbing on the Shoulders of Giants

Find Where It Hurts

to focus on relevant issues

to be prepared for handling the most critical issues

Plan for Responsible Moments

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

Copyright 2020 Joseph Yoder & The Refactory, Inc.
## Qualify the Roadmap

![Roadmap Diagram](image)

### Development
- **MOBILE WEB v1**
- **MOBILE WEB v2**
- **RICH MOBILE WEB APPS**
- **PC PLATFORM v1**
- **PC PLATFORM v2**
- **ONGOING RELEASSE**
- **MOBILE RESEARCH**
- **ANDROID v1**
- **iOS v1**
- **RESPONSIVE DESIGN**

### Enterprise Architecture
- **PERSISTENCE FRAMEWORK**
- **MOBILE GENERIC SERVICES**
- **SYBASE TO ORACLE MIGRATION**
- **LOAD BALANCING**
- **MOBILE SECURITY**
- **PLATFORM STABILITY**
- **CLOUD RESEARCH**
- **MICROSERVICES**
- **NO SQL / BIG DATA v1**
- **NO SQL / BIG DATA v2**

### Delivery
- **DELEIVERY**
- **BUDGET**
- **RESOURCE**
- **ARCHITECTURE**
- **DEPENDENCIES**
- **RISKS**
- **ISSUES**
- **ON RADAR**

### Notes
- **Climbing on the Shoulders of Giants**
- **Find Where It Hurts**
- **Plan for Responsible Moments**
- **Tracer Bullets**

---

**Copyright 2020 Joseph Yoder & The Refactory, Inc.**
Tracer Bullets

Climbing on the Shoulders of Giants

to define how tests should be created

to start the architecture

Test Architecture

Find Where It Hurts

to focus on relevant issues
to define low level architecture

can guide the choice

can validate each other

Tracer Bullets

Plan for Responsible Moments

to be prepared for handling the most critical issues
Test Architecture

Climbing on the Shoulders of Giants
- to start the architecture
- to define how tests should be created

Find Where It Hurts
- to focus on relevant issues
- to define low level architecture
- can guide the choice
- can validate each other

Tracer Bullets
- to be prepared for handling the most critical issues

Plan for Responsible Moments

Note these can be done throughout the project...
During the Project

Climbing on the Shoulders of Giants
- to define how tests should be created

Find Where It Hurts
- to focus on relevant issues
- can guide the choice

Tracer Bullets
- to define low level architecture
- can validate each other

Plan for Responsible Moments
- to be prepared for handling the most critical issues
- to plan tasks related to the architecture

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

Climbing on the Shoulders of Giants
- Find Where It Hurts
- Plan for Responsible Moments
- Test Architecture
- Tracer Bullets
- Architectural Spike

Architecture in the Backlog
- Can guide the choice
- Can include
- Can add tasks
- Can validate each other

To start the architecture
To define how tests should be created
To focus on relevant issues
To define low level architecture
To be prepared for handling the most critical issues

Copyright 2020 Joseph Yoder & The Refactory, Inc.
Architectural Spike

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

• Can be a Sprint!!!
Climbing on the Shoulders of Giants

To start the architecture

Find Where It Hurts

To focus on relevant issues

Plan for Responsible Moments

To be prepared for handling the most critical issues

Tracer Bullets

To define low level architecture

can validate each other

Architectural Spike

can include

Architecture in the Backlog

can include

can add tasks

Test Architecture

To define how tests should be created

to be aware of problems in the architecture

Technical Debt Management

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

Graziela Simone Tonin
Alfredo Goldman
System “Qualities”

- Performance
- Security
- Scalability
- Availability
- Reliability
- Maintainability
- Evolvability
- 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...

- Little or no docs, Testing, QA
- eXplore
- Takeoff
- eXtract
- Maturity

requires formal docs, QA, clean architecture, good testing

3X
HOW CAN WE IMPROVE WHAT WE CANNOT SEE?

Visibility is Key “Radiators”
Without valuable and timely information, the organization dies.

Therefore:

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

“"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

Automate As You Go

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
**Climbing on the Shoulders of Giants**

- To start the architecture
- To define how tests should be created

**Find Where It Hurts**

- To focus on relevant issues
- To guide the choice
- To define low level architecture

**Plan for Responsible Moments**

- To be prepared for handling the most critical issues

**Tracer Bullets**

- Can validate the choice
- Can validate each other

**Architectural Spike**

- Can include
- Can be part of

**Technical Debt Management**

- Can use

**Continuous Inspection**

- Can use

**Architecture in the Backlog**

- Can add tasks

---

**Have a sustainable architecture**

- Start the project fast
Periodically Re-Evaluate Architecture Risks

Delivery and Feedback

Iteration Planning

Continuous Improvement

Architecture Quality

Implementation

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

Pause Points Help

© 2020 Joseph Yoder & The Refactory
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*

---

© 2020 Joseph Yoder & The Refactory
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

- Climbing on the Shoulders of Giants
  - How can you quickly define the basic application architecture and the main component types that will satisfy these 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.

- Tracer Bullet
  - 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 reference for upcoming functionality. Use this implementation to face technical challenges that were planned to be tackled before the project iterations.

- 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 its appropriate address these technical challenges.
Patterns for Evolving Agile Architecture

Continuous Inspection

CODE SMELL DETECTION METRICS (TEST COVERAGE, CYCLOMATIC COMPLEXITY, TECHNICAL DEBT, SIZES, …)
APPLICATION SECURITY CHECKS ARCHITECTURAL CONFORMANCE

AUTOMATE WHERE YOU CAN!!!
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*, Rebecca Wirfs-Brock, Adamar Agius†
*The Refactory, Inc.
†Wirfs-Brock Associates, Inc.
†FPLP

Abstract. An organization transition from waterfall to agile processes. Quality assurance (QA) activities and roles used 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 developers and QA staff are working concurrently on software. Agile teams embrace a “whole team” approach. Even though special skills may be required to perform certain development and QA functions, each member 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
...PATTERNS FOR TRANSITIONING FROM TRADITIONAL TO AGILE QA AND AGILE ARCHITECTURE

Copyright 2020 Joseph Yoder & The Refactory, Inc.
Tech Debt as I see it relate to Kent Beck’s 3x Model

- Risk Appetite: Worth taking risks, incurring tech debt, but...
  - Tech Debt is not really "debt" yet...

- Experiment dies, tech debt is written off
  - Experiment lives, tech debt becomes a liability

Explore, Expand, Extract

Tech Debt is the more there is to lose, the less appetite for risk...

- If experiment dies, then the debt is written off. No liability.
- If experiment lives & matures, debt becomes a liability...
  - Pay it back where interest is highest.

More Architecture, QA, Testing & Docs

https://twitter.com/AntonyMarcano
Agile/Lean

Ideas

Build

Data

Code

Learn

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

Technical Skills
It is a Journey

Commitment
Follow-through
Deliberate practices
Slack Time to Improve
Paying attention
Continuous Learning

Today
• Anger not
• Worry not
• Grateful be
• Diligent, in your endeavours, be
• To others, kindness show

What do you Value?
@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”