Being Agile About Architecture

Joseph W. Yoder
The Refactory

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

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 Myths

KISS
Simple is Best

Values

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
Before First Iteration
Have a sustainable architecture
Start the project fast, being agile

Climbing on the Shoulders of Giants to start the architecture
Climbing on the Shoulders of Giants

Build on a Reference Architecture

IoT Reference Architecture

Microsoft, Azure IoT Reference Architecture, https://aka.ms/iotrefarchitecture — 2018
Find Where It Hurts
to start the architecture

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
Qualify the Roadmap

<table>
<thead>
<tr>
<th>2017</th>
<th>2018</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jan</td>
<td>Feb</td>
</tr>
<tr>
<td>MOBILE WEB v1</td>
<td>MOBILE WEB v2</td>
</tr>
<tr>
<td>PC PLATFORM v1</td>
<td>PC PLATFORM v2</td>
</tr>
<tr>
<td>MOBILE RESEARCH</td>
<td>ANDROID v1</td>
</tr>
<tr>
<td>DELAYS expected to</td>
<td></td>
</tr>
<tr>
<td>Budget will need bolstering in Q2 2017</td>
<td></td>
</tr>
<tr>
<td>All resources on track.</td>
<td></td>
</tr>
<tr>
<td>Persistence Framework under enabling cloud</td>
<td></td>
</tr>
<tr>
<td>Migration issues all in place and on track</td>
<td></td>
</tr>
<tr>
<td>RISKS ON RADAR</td>
<td></td>
</tr>
<tr>
<td>COMPETITOR E Corp – new product</td>
<td></td>
</tr>
<tr>
<td>ARCHITECTURE</td>
<td>Performance under evaluation</td>
</tr>
<tr>
<td>Tweaks needed for velocity and execution</td>
<td></td>
</tr>
<tr>
<td>ISSUES</td>
<td></td>
</tr>
<tr>
<td>DELIVERY</td>
<td>BUDGET</td>
</tr>
<tr>
<td>Delivery expected to Version 1</td>
<td></td>
</tr>
<tr>
<td>Budget will need bolstering in Q2 2017</td>
<td></td>
</tr>
<tr>
<td>All resources on track</td>
<td></td>
</tr>
<tr>
<td>Persistence Framework under enabling cloud</td>
<td></td>
</tr>
<tr>
<td>Migration issues all in place and on track</td>
<td></td>
</tr>
<tr>
<td>TBD</td>
<td></td>
</tr>
<tr>
<td>LOW</td>
<td></td>
</tr>
<tr>
<td>NORMAL</td>
<td></td>
</tr>
<tr>
<td>HIGH</td>
<td></td>
</tr>
<tr>
<td>TBD</td>
<td></td>
</tr>
</tbody>
</table>

Climbing on the Shoulders of Giants

Find Where It Hurts

to start the architecture

to focus on relevant issues

can guide the choice

to be prepared for handling the most critical issues

Tracer Bullets

to define low level architecture

Plan for Responsible Moments
Tracer Bullets

Climbing on the Shoulders of Giants

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
can guide the choice
can validate each other

Find Where It Hurts

Tracer Bullets

Plan for Responsible Moments

Test Architecture

Architectural Requirements

not enough for the requirements

Simple

Feature

controller

web pages

Copyright 2021 Joseph Yoder & The Refactory, Inc.
Test Architecture

Climbing on the Shoulders of Giants

Find Where It Hurts

to focus on relevant issues
to define low level architecture

to be prepared for handling the most critical issues

can guide the choice

can validate each other

Tracer Bullets

Plan for Responsible Moments

to start the architecture

to define how tests should be created

Test Architecture

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

- Climb on the Shoulders of Giants: to start the architecture
- Find Where It Hurts: to focus on relevant issues
- Tracer Bullets: to define low level architecture
- Plan for Responsible Moments: to be prepared for handling the most critical issues
- Test Architecture: to define how tests should be created
- Architecture in the Backlog: to plan tasks related to the architecture

Find Where It Hurts can guide the choice, Tracer Bullets can validate each other.
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**

- 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

Test Architecture to define how tests should be created
to focus on relevant issues

Find Where It Hurts to define low level architecture
can guide the choice
to be prepared for handling the most critical issues
can validate each other

Plan for Responsible Moments to plan tasks related to the architecture
can include
can add tasks

Architecture in the Backlog can include
can add tasks

Architectural Spike can add tasks
to be aware of problems in the architecture
can include
can add tasks

Tracer Bullets

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  
- eXtract  
- Maturity  
- requires formal docs, QA, clean architecture, good testing  
- Takeoff  
- 3X  
- more docs, QA, Architecture, …  
- eXpand  
- Ferment
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**.

[picture of a Scrum board with text: "A Scrum Book: The Spirit of the Game"]


---

**Make things visible!**
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???”
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
Climbing on the Shoulders of Giants

Find Where It Hurts

to focus on relevent issues
to define low level architecture
to be prepared for handling the most critical issues

to include

can validate each other

can be part of

can add tasks

can use

to detect problems in the code and in the architecture

to be aware of problems in the architecture

to define how tests should be created

to start the architecture

Test Architecture

Plan for Responsible Moments

to plan tasks related to the architecture

can include

can be part of

can add tasks

can use

to detect problems in the code and in the architecture

to be aware of problems in the architecture

to define how tests should be created

to start the architecture

Tracer Bullets

Architectural Spike

Architecture in the Backlog

Technical Debt Management

Continuous Inspection

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

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

1. **Climbing on the Shoulders of Giants**
   - How can you quickly define the basic application architecture and the main component types that will satisfy this requirement?
   - Use an existing reference compatible with the application platform and suitable to fit needs as a starting point.

2. **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.

3. **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 use stories and implement them as reference for upcoming functionality. Use this implementation to face technical challenges that were planned to be fixated before the project iteration.

4. **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’s appropriate address these technical challenges.

Asian PLoP 2015
Patterns for Evolving Agile Architecture

Plan for Responsible Moments

- How can you identify the technical debt problems and develop a plan to address them?
- Communicate the technical plan for loan and when to handle each of the technical challenges and include the team in the process.
- Specify how important technical elements should be aligned and when it is appropriate to address technical challenges.

Continuous Inspection

- How can the team address the technical problems that arise in the architecture?
- Outline inspection as a process to identify where the architecture that may require a bug in the architecture or an architecture that needs to be improved.

Technical Debt Management

- How can the team manage the technical debt problems and develop a plan to address them?
- Identify and manage the technical debt problems that arise in the architecture and develop a plan to address them as they arise in the architecture.

Sustaining Your Architecture

- How can you detect and improve the architecture and core problems as soon as they arise?
- Implement automated processes for the continuous inspection of the architecture and core problems as soon as they arise.

Continuous Inspection

- How can you detect and improve the architecture and core problems as soon as they arise?
- Implement automated processes for the continuous inspection of the architecture and core problems as soon as they arise.

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

Patterns to Develop and Evolve Architecture in an Agile Environment

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 Wiebe-Brock2, Alastair Agius
1The Refactory, Inc.
2Wals-Brock Associates, Inc.

Abstract. An organizational 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 the development and product teams have the means to achieve quality objectives.

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 3 patterns for transitioning from a traditional QA practice to a new 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 2021 Joseph Yoder & The Refactory, Inc.
Tech Debt as I see it relates to Kent Beck’s 3x Model

@AntonyMarcano

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

Explore
Risk appetite is still high. Tech debt is still worth taking.

Expand
Risk appetite is still high. Tech debt is still worth taking.

Extract
Risk appetite is low. Tech debt becomes a liability.

Lifecycles of a Product/Feature’s Reach/Return

- 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

Kent Beck’s 3x Model

https://twitter.com/AntonyMarcano

Being Agile

Agile Mindset
Agile/Lean

Ideas → Build → Code

Learn → Measure → 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
It is a Journey

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

What do you Value?

Yoda's Principles Licensed under CC: https://www.flickr.com/photos/rlajki857774/106289335/
@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”