Continuous Architecture and Emergent Design

Scott W. Ambler
Senior Consulting Partner

scott@scottambler.com

@scottwambler
Agenda

1. Myths and misunderstandings
2. Ideas from Disciplined Agile Delivery (DAD)
3. Getting started
4. Emergence
5. Stuff happens
6. Parting thoughts
Myths and Misunderstandings
Common Myths Surrounding Agile Architecture

1. Agilists don’t “do architecture”
2. Agilists start coding right away
3. Agilists don’t follow enterprise architecture strategies
4. Agilists don’t model
5. Agilists don’t document
Realities of Agile Architecture

1. Architecture is so important agilists address it throughout the entire lifecycle
2. Disciplined agilists invest in some up front envisioning
3. Discipline agilists work closely with enterprise professionals
4. Disciplined agilists explicitly model in a lightweight manner throughout the lifecycle
5. Disciplined agilists write deliverable documentation continuously
Disciplined Agile
DAD Role: Architecture Owner

- Guides the creation and evolution of the solution’s architecture
- Mentors and coaches team members in architecture practices and issues
- Understands the architectural direction and standards of your organization and ensures that the team adheres to them
- Ensures the system will be easy to support by encouraging appropriate design and refactoring
- Ensures that the system is integrated and tested frequently
- Has the final decision regarding technical decisions, but doesn’t dictate them
- Leads the initial architecture envisioning effort
DAD Teams Are Enterprise Aware

- DAD teams strive to leverage and enhance the existing organizational eco system wherever possible

- Implications for architecture and design:
  - Work with enterprise architects
  - Follow existing roadmap(s) where appropriate
  - Leverage existing assets
  - Enhance existing assets

Community Awareness
“How can I give back to my community?”

Enterprise Awareness
“How can I help my organization?”

Team Awareness
“How can I help the team?”

Individual Awareness
“How can I be the best me?”

Copyright 2013 Disciplined Agile Consortium
Scrum Construction Lifecycle

- **Highest-Priority Requirements**
  - Sprint Backlog
  - Sprint Tasks

- **Sprint (2-4 weeks)**
  - Daily Work
  - Daily Scrum Meeting: Share status and identify potential issues

- **Sprint Review**
  - Demo system to stakeholders and gain funding for next sprint
  - Sprint Retrospective: Learn from your experiences

- **Planning session to select requirements for current Sprint and to identify work tasks**

- **Funding & Feedback**

Original Diagram Copyright Mike Cohn
A Scrum Delivery Lifecycle

- Initial Architectural Vision
- Initial Requirements and Release Plan
- Product Backlog

- Highest-Priority Requirements
- Sprint Backlog
- Sprint Tasks

- Daily Work
- Daily Scrum Meeting: Share status and identify potential issues

- Sprint (2-4 weeks)
- Sprint Review: Demo software to stakeholders and gain funding for next sprint
- Sprint Retrospective: Learn from your experiences

- Funding & Feedback
- Planning session to select requirements for current Sprint and identify work tasks

- Working Software
- Release software into production

© Scott Ambler + Associates
Governed Delivery Lifecycle

Inception
- One or more short iterations
- Stakeholder vision
- Proven architecture

Construction
- Many short iterations producing a potentially consumable solution each iteration
- Project viability (several)
- Sufficient functionality

Transition
- One or more short iterations
- Production ready
- Delighted stakeholders
Disciplined Agile Delivery: Basic Lifecycle

[Diagram showing the lifecycle of Disciplined Agile Delivery, including Inception, Construction, and Transition stages with corresponding activities and timeline]

© Scott Ambler + Associates
DAD is Goal-Driven, Not Prescriptive

<table>
<thead>
<tr>
<th>Goals for the Inception Phase</th>
<th>Goals for Construction Phase Iterations</th>
<th>Goals for the Transition Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Form initial team</td>
<td>- Produce a potentially consumable solution</td>
<td>Ensure the solution is consumable</td>
</tr>
<tr>
<td>- Develop common project vision</td>
<td>- Address changing stakeholder needs</td>
<td>- Deploy the solution</td>
</tr>
<tr>
<td>- Align with enterprise direction</td>
<td>- Move closer to deployable release</td>
<td></td>
</tr>
<tr>
<td>- Explore initial scope</td>
<td>- Improve quality</td>
<td></td>
</tr>
<tr>
<td>- Identify initial technical strategy</td>
<td>- Prove architecture early</td>
<td></td>
</tr>
<tr>
<td>- Develop initial release plan</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Form work environment</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Secure funding</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Identify risks</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Ongoing Goals**

- Fulfill the project mission
- Grow team members
- Address risk

- Improve team process and environment
- Leverage and enhance existing infrastructure
- Coordinate activities
Disciplined Agilists Take a Goal Driven Approach

- **Goal**: Explore the Initial Scope
- **Issue**: Form the Initial Team
- **Issue**: Address Changing Stakeholder Needs
- **Issue**: Source
  - Team size
  - Team structure
  - Team members
  - Geographic distribution
  - Supporting the team
  - Availability
- **Option**: Co-located
  - Partially dispersed
  - Fully dispersed
  - Distributed subteams

**Default Option**

- **Advantages**
- **Disadvantages**
- **Considerations**
Initial Architecture
Goal: Identify Initial Technical Strategy

- High-level overview
  - Detailed interface
  - Detailed end-to-end
  - None

- Technology
  - Business Architecture
  - User interface (UI)

- View Types

- Informal modeling sessions
  - Formal modeling sessions
  - Single candidate architecture
  - Multiple candidate architectures

- Modeling Strategy

- Delivery Strategy
  - Extend existing solutions
    - Configure a COTS package
    - Extend a COTS package
    - Build from scratch
Goal: Align With Enterprise Direction

- **High level**
  - Detailed
  - Enforced
  - None

- **Minimal**
  - Comprehensive
  - None

- **Collaborative**
  - Continuous
  - Gated
  - Formal
  - None

- **Managed – Collaborative**
  - Managed – Formal
  - Ad hoc
  - None

- **Agile**
  - Traditional
  - None
Survey Says: Initial Architecture Efforts

• 25% use enterprise or industry architecture models as references
• 77% performed high-level initial architecture envisioning
• 28% performed detailed initial architecture modeling
• 92% performed some form of up-front architecture modeling
• 97% performed some initial modeling OR worked from an existing legacy architecture

Source: SA+A 2013 Agile Project Initiation Survey
Ambysoft.com/surveys/
Non-functional requirements are critical drivers of your architecture
You’ll Need Several Views

Business process diagram
Data flow diagram (DFD)
Domain/conceptual model
UML component diagram

Business Architecture

Technical

User Interface

General

User interface (UI) flow diagram
UI prototype (high fidelity)

Architectural stack diagram
Deployment diagram
Network diagram
Threat model
UML component diagram

Some Architecture Frameworks:
Zachman
TOGAF
DoDAF
4+1
Emergence
Continuous Architecture Practices

Architectural vision guides development efforts

Initial architectural envisioning

Architecture owner facilitates architectural decisions throughout Construction

Architecture spikes to explore a technical issue

Architecture handbook and models updated as required

Reduce risk early by proving the architecture works
Emergent Design Practices

Discuss **design implications** during iteration planning/modeling.

Test-Driven Design (TDD) throughout Construction.

Look-ahead **modeling** for upcoming complex work items.

Consider **design issues** of incoming requests from production.

© Scott Ambler + Associates
Agile Model Driven Development (AMDD): Project Level

- Identify the high-level scope
- Identify initial “requirements stack”
- Identify an architectural vision

- Modeling is part of iteration planning effort
- Need to model enough to give good estimates
- Need to plan the work for the iteration

- Work through specific issues on a JIT manner
  - Stakeholders actively participate
  - Requirements evolve throughout project
  - Model just enough for now, you can always come back later

- Develop working software via a test-first approach
  - Details captured in the form of executable specifications
Goal: Prove Architecture Early

- Validation
  - End-to-end working skeleton
    - Architectural spikes
      - Solution bake-off
      - Pilot testing
    - Stakeholder demos
      - Informal reviews
      - Formal reviews
  - Informal reviews
    - Formal reviews

- Verification

© Scott Ambler + Associates
Goal: Produce a Potentially Consumable Solution

1. Development Strategy
   - Test-first programming (TFD)
   - Test-after programming
   - Testless programming
   - Active stakeholder participation
   - High-level requirements specification
   - Detailed requirements specification
   - Acceptance test-driven development (ATDD)
   - Just-in-time (JIT) model storming
   - Look-ahead modeling
   - Test-driven development (TDD)
   - JIT model storming
   - Look-ahead modeling
   - Architecture spike
   - Consumability design
   - Information design
   - Other “ility” design
   - Detailed design specification
   - Model-driven development (MDD)

2. Needs Exploration
   - Coordination meetings
   - Iteration planning
   - Just-in-time (JIT) planning
   - Visualize plan
   - Release planning
   - Look-ahead planning

3. Solution Exploration
   - Iteration demos
   - All-hands demos
   - Consumability design
   - Consumability testing
   - Regular deployment

4. Planning
   - Consumability Assurance

© Scott Ambler + Associates
Goal: Leverage and Enhance the Existing Infrastructure

- Guidelines
  - Adopt industry guidance
  - Evolve enterprise guidance
  - Develop new guidance

- Systems
  - Use existing assets
  - Adopt external assets
  - Refactor existing assets
  - Develop reusable assets

- Data
  - Database consolidation
  - Database refactoring
  - Use existing data sources

- Process
  - Tailor existing process
  - Adopt external process assets
  - Share process learnings
  - Evolve process assets
  - Adopt existing templates
Stuff Happens...
What if…

You don’t think all the issues through from the very beginning?

Another team needs to update the solution years from now?

The team gets hit by a truck?

There is a major requirements change that forces you to rethink the architecture in the middle of construction?
Parting Thoughts
General Solution Architecture Principles

Architecture owners should understand why

Look beyond technology

Adopt a multi-view approach

Architecture is driven by requirements

Architectures should be loosely coupled

Architectural elements should be highly cohesive
Principles of Agile Solution Architecture

Think about the future, but wait to act

Learn early, learn fast

Simplicity first

Build quality in

Model in an agile manner

Be enterprise aware
Thank You!

scott [at] scottambler.com
@scottwambler

AgileModeling.com
AgileData.org
Ambysoft.com
DisciplinedAgileConsortium.org
DisciplinedAgileDelivery.com
ScottAmbler.com

© Scott Ambler + Associates
Recommended Resources
Backup Slides
Disciplined Agile Delivery (DAD)

Disciplined Agile Delivery (DAD) is a process decision framework

The key characteristics of DAD:
- People-first
- Goal-driven
- Hybrid agile
- Learning-oriented
- Full delivery lifecycle
- Solution focused
- Risk-value lifecycle
- Enterprise aware
Goal Diagram Notation

Goal - Issue - Option

* Indicates a preference for the options towards the top

Default Option