Strategies for Streamlining Enterprise Architecture in the Age of Agile

John Mallinger & Paul Newell
22nd Annual NDIA SE Symposium
10/24/2019

Approved for Public Release
Right Sizing Architecture for Agile

- The Case for Architecture on Agile Projects
- Acquisition Strategies - Contracting for Agile Success
- Aggressive Tailoring - Driving Architecture Value
- Changing at the Speed of Agile - Planning for Evolution
The agile challenge...

- The Agile Manifesto challenges software design and architecture to add value
  - Prioritizes collaboration and responding to change
  - Deemphasizes documentation and processes

- Architecture activities need to add clear and recognizable return on effort invested
  - Team strategy may leverage emergent design

- Agile development strives for a balanced approach to documentation
  - Avoid creating documentation shelfware
The Case for Defining Architecture

- Eliminating all design and architecture efforts drives substantial risk into agile development
  - Emergent designs fail to meet undefined quality attributes
  - Poorly aligned teams build incompatible interfaces
  - Incomplete system decomposition fails to satisfy user needs

- Defined architecture drives greater efficiency in agile
  - Architecture supports reuse, commonality, and adoption of common patterns
  - Coordinated approach helps agile scale to larger projects and teams
Agile Done Wrong

- **Big Design Up Front** - developing 600+ requirement System Spec
  - Agile principles undermined by defining large fixed and immutable MVP

- **Applying Mil-Std-1521B to Agile Acquisitions**
  - Artificially forces Big Design Up Front
  - Drives accelerated design to satisfy waterfall events
  -Limits tailoring of design and architecture artifacts

- **Using out-of-the-box Earned Value to manage Agile projects**
  - Constrains ability to re-plan and react to changes and discovers during sprints
Agile in the Enterprise

- Layers of Architecture
  - Big Design Up Front
  - Iteration-Driven Architecture
  - Architecture at the Sprint Level
- Hybrid Approach managing backlog
  - Small Design Up Front
  - Enterprise Architecture within Iterations
  - Component & Service Architecture within sprints
- Make architecture inclusive and drive participation from all stakeholders
  - Agile seeks inputs and buy-in from everyone
  - Team needs to own and leverage architecture to drive value
Agile Acquisition Strategies

Software is Never Done

Defense Innovation Board Report: Speed and cycle time are the most important metrics for software.

Faster reduces risk because it demands focus on the critical functionality rather than over-specification or bloated requirements.

Ten Most Important Things to Do (D3): Shift from the use of rigid lists of requirements … to a list of desired features and required interfaces / characteristics to avoid requirements creep, overly ambitious requirements, and program delays.

DIB SWAP Study

TechFAR Handbook

TechFAR Handbook:
• Use Product Vision Statement to scope high-level objectives for Agile Software acquisition
• Structure requirements as a Statement of Objectives
• Product Owner defines / revises high level requirements as part of scrum-based agile process

Digital Services Playbook

Digital Services Playbook:
Well-defined contract can facilitate good development practices like:
• Conducting research and prototyping phase
• Refining product requirements as service is built
• Contract gives government delivery team enough flexibility to adjust feature prioritization and delivery schedule as project evolves

Approved for Public Release
Driving Architecture Value

- Extreme tailoring to maximize design & architecture value
  - Focus on high value critical design tasks
  - Tailor out low value or redundant tasks and artifacts

- How extreme?
  - Everything is on the table
  - No artifact or task is sacred
Agile Architecture Objectives

- Capture Up Front Design and High Level Approach
- Support tactical decisions within Sprints
- Architecture Change Management and Responding to Update Requests
Quantifying Architecture Value

- Use an objective framework to assess value of architecture tasks

- Options
  - Agile-based planning
    - Points based voting or priority poker to rank
  - Risk Driven
    - Design and Architecture mitigate risk of leveraging emergent and undefined development strategies
  - Weighted Shortest Job First (WSJF) strategy
    - Don Reinertsen - The Principles of Product Development Flow
    - Quantify cost of delay from missing design or architecture tasks
Risk-Based Prioritization

Risk assessment and mitigation strategy drives:

- Prioritize architecture tasks to maximize risk mitigation
- Target moderate or high design risks
- Can approach with generic architecture tasks or specific design challenges

<table>
<thead>
<tr>
<th>#</th>
<th>Architecture Task</th>
<th>Risk</th>
<th>Initial Score</th>
<th>Mitigated Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>Develop CONOPS</td>
<td></td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>R2</td>
<td>Define External Interfaces</td>
<td>Late definition drives incompatibility</td>
<td>15</td>
<td>6</td>
</tr>
<tr>
<td>R3</td>
<td>Create User Storyboards</td>
<td>Displays do not meet user needs</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>R4</td>
<td>Identify &amp; Map Quality Attributes</td>
<td>Unable to meet reliability, maintainability, or scalability</td>
<td>20</td>
<td>5</td>
</tr>
</tbody>
</table>
Weighted Shortest Job First

- Method for prioritizing work based on likely impact
- Team assesses Cost of Delay and Job Size for each task
  - Cost of Delay encompasses value of artifact in driving design or implementation and alignment
  - Job size is relative complexity of a task compared to other tasks

<table>
<thead>
<tr>
<th>#</th>
<th>Architecture Task</th>
<th>Cost of Delay (CoD)</th>
<th>Job Size / Duration</th>
<th>WSJF</th>
<th>Dependencies?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Develop CONOPS</td>
<td>8</td>
<td>5</td>
<td>1.6</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Define External Interfaces</td>
<td>12</td>
<td>12</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Create User Storyboards</td>
<td>12</td>
<td>8</td>
<td>1.5</td>
<td>Task #1</td>
</tr>
<tr>
<td>4</td>
<td>Identify &amp; Map Quality Attributes</td>
<td>20</td>
<td>1</td>
<td>20</td>
<td></td>
</tr>
</tbody>
</table>
Implementing Agile Architecture

- Results of architecture value assessment drive implementation

- Small Design Up Front
  - Limited big picture design to communicate high level objectives
  - Define and select quality attributes to emphasize in architecture

- Plan for Change and Evolution of Design
  - Agile sprints and iterations will not follow prescribed order
  - Consider use of TOGAF Transition Architecture to track evolution


Approved for Public Release
Change at the speed of agile

- Agile Success starts with the contract
- Limit big design up-front
- Minimize architecture artifacts to those absolutely necessary
- Embed most design and architecture within iterations or springs
- Get entire team engaged in architecture