Principles of Lean-Agile Leadership:

*Foundation of the Scaled Agile Framework*

Dean Leffingwell
On Managing Knowledge Workers

“Workers are knowledge workers if they know more about the work they perform than their bosses.”

Workers themselves are best placed to make decisions about how to perform their work

To effectively lead, the workers must be heard and respected

Knowledge workers have to manage themselves. They have to have autonomy

Continuing innovation has to be part of their work, the task, and the responsibility of knowledge workers

—Peter Drucker
SAFe Foundation: Leadership

Goal: Value

- Respect for People
- Product Development Flow
- Kaizen

Foundation: Leadership

Lean-Agile Leaders

- Take responsibility for Lean-Agile success
- Understand and teach Lean-Agile behaviors
- Are trained in practices and tools of continuous improvement
- Teach problem solving and corrective action
- See with their own eyes. “No useful improvement was ever invented at a desk.”
- Develop people. People develop solutions.
Lean-Agile Leadership Values

“Lean-Agile Leaders are lifelong learners who help teams build better software systems through understanding and exhibiting the values, principles and practices of Lean, systems thinking, and Agile development.”

—Scaled Agile Framework 3.0
Principles of Lean-Agile Leadership

1. Take a Systems View
2. Embrace the Agile Manifesto
3. Implement Product Development Flow
4. Unlock the Intrinsic Motivation of Knowledge Workers
1. Take a Systems View

- Understand the economics, the full value chain, and the Cost of Delay
- Optimize the whole, not the parts, of the organization
- Optimize the whole, not the parts, of the software system
- Own the system; take responsibility for systemic change
- Align everyone to a common mission
- Implement Lean-Agile Budgeting
If You Can’t Change the System, Who Can?

- Everybody is already doing his best
- It’s the system, not the workers, that ultimately determine quality and productivity
- Only management can change the system
- It is not enough that management commit themselves to quality and productivity

They must know what it is they must do.

Simple Truths About Systems

- Optimizing a component does not optimize the system
- The value of a system passes through its interconnections
- Complex systems development requires disciplined, systematic, systems thinking
- A system can evolve no faster than its slowest integration point
Example: DevOps

What happens if we don’t take a systems view with respect to deploying software?

Development

• Create Change
• Add or Modify Features

Operations

• Create Stability
• Create or enhance services
Embrace the Agile Manifesto

- Support the Agile Manifesto values and principles; Deliver more frequently
- Know how to implement XP, Scrum, Kanban, SAFe and evolving practices
- Foster excellence in software engineering, craftsmanship, and collaborative system design
- Exhibit Kaizen mind and a bias for action
- Empower high-performing, cross-functional teams
Agile Values – The Agile Manifesto

“We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more.”

—agilemanifesto.org
Agile Manifesto Principles

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

3. Working software is the primary measure of progress.

4. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

5. Business people and developers must work together daily throughout the project.

6. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
Agile Manifesto Principles (continued…)

7. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

9. Continuous attention to technical excellence and good design enhances agility.

10. Simplicity—the art of maximizing the amount of work not done—is essential.

11. The best architectures, requirements, and designs emerge from self-organizing teams.

12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

—agilemanifesto.org/principles.html
Question

Are we still uncovering, or are we done?
Exercise: Scaling Agile Principles

- With the person next to you, discuss one principle of the manifesto
- Do you think it?
  a) Scales as-is?
  b) Requires rethinking in the larger enterprise?

Timebox: 5 minutes
Some Thoughts on Agile Methods

- **Scrum**
  Works great. Less filling. Ubiquitous software project management. Clear roles. **Scrumptious.** Let’s Sprint.

- **Extreme Programming**
  Really great code from really great coders. **Extremely** useful. Let’s **Program** with it.

- **Kanban**
  Clear thinking on flow, demand management and limiting Work in Process. Let’s **limit WIP, manage demand and flow**.
And Some Thoughts on Scaling

- **But If** these innovative methods don’t have the native constructs to address the view beyond the team – the *systems view* –

  shouldn’t we do something about that?

- **And**, on behalf of millions of practitioners, working on really big systems in really big companies, and struggling badly with existing approaches

  don’t we have an obligation to try?
Nothing Beats an Agile Team

- Empowered, self-organizing, self-managing, cross-functional teams
- Valuable, fully-tested system increments every two weeks
- Scrum project management practices; XP-inspired technical practices; Lean thinking; flow centric
- Vision, system, architecture and user experience guidance
- Value delivery via User Stories
Scale to the Program

- Self-organizing, self-managing team-of-agile-teams
- Continuous value delivery. Full system demo every two weeks. Release on demand.
- Aligned to a common mission via a single backlog

- Common sprint lengths and estimating
- Face-to-face planning cadence for collaboration, alignment, synchronization, and assessment
- Value Delivery via Features and Benefits
Scale to the Portfolio

- Lean approaches to Strategy and Investment Funding, Program Management, and Governance
- Lean-Agile Budgeting
- Centralized strategy, decentralized execution
- Kanban systems provide portfolio visibility and WIP limits
- Objective metrics support governance and kaizen
- Value delivery via Business and Architectural Epics
Suggested Reading

To understand and implement Lean-Agile at scale:


Check out the framework@*Scaledagileframework.com*
Implement Product Development Flow

- Reduce queues and backlogs
- Visualize work; expose WIP and bottlenecks
- Reduce batch size
- Accelerate feedback
- Manage and exploit variability with cadence and synchronization
- Limit WIP
- Limit demand to capacity
Long Queues – All Bad

Email from a client service organization: “Thank you for contacting us. We are experiencing increased volumes and apologize in advance for the delay. Our goal is to contact you within ... ”

Understand Little’s Law:

Avg wait time = avg queue length divided by the avg processing rate

Faster processing time decreases wait

Lesson: It’s far easier to control wait times by controlling queue lengths

Fig. Source: Reinertsen, Don. Principles of Product Development Flow.
Some Queues are Really, Really Bad
Team Visibility and WIP Constraints

One team’s Big Visible Information Radiator

<table>
<thead>
<tr>
<th>Wednesday</th>
<th>Thursday</th>
<th>Friday</th>
<th>Monday</th>
<th>Tuesday</th>
<th>Wednesday</th>
<th>Thursday</th>
<th>Friday</th>
<th>Monday</th>
<th>Tuesday</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not Started</td>
<td>Development</td>
<td>Test</td>
<td>Accepted</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Question: How is this team doing? How do you know that?
Exercise – WIP Constraints

Consider the BVIR on the prior page, then discuss:

1. What would be the effect of a three-story WIP constraint on Development and Test?

2. Consider the scenario: You are a coder. You just finished story 6. What would you do if:
   a. There is no WIP constraint
   b. The three-story WIP constraint is in place

3. Which scenario has the highest throughput?

Timebox: 5 minutes
Even More Important: The Big Stuff

<table>
<thead>
<tr>
<th>Epics</th>
<th>Funnel</th>
<th>Review (2)</th>
<th>Analysis (2)</th>
<th>Portfolio Backlog (4)</th>
<th>Implementing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Epic 50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Epic 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Epic 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Epic 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Epic 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Epic 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Epic 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Epic 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Epic 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Question:**

1) What happens if you have excess portfolio level WIP in the System
2) What can you do about it?
Read *Principles of Product Development Flow*:

- Don’t worry too much about understanding it
- Read it again
- Understand it this time
- Apply it
Read *The Lean Machine*:

- Learn how big (and fast!) systems are built using Lean
- Understand how Lean supports innovation and empowerment
- Look beyond software for clues
Unlock the Intrinsic Motivation of Knowledge Workers

- Build a learning organization and emphasize lifelong learning
- Create an environment of mutual influence
- Foster decentralized decision-making
- Provide vision, with minimum specific work requirements
- Eliminate demotivating policies, procedures, MBOs*
  Revamp personnel evaluation

* Management by Objectives
Create an Environment of Mutual Influence

Create a safe environment for learning, growth and mutual influence

Encourage direct reports:

- To disagree where appropriate
- To advocate for the positions they believe in
- To push for their own needs
- To enter into joint problem solving
- To negotiate, compromise, agree, commit

Garth Andrews holding the coveted Horse’s Rear award
Drive: The Puzzling Puzzles of Harry Harlow

The 1949 Experiment

- Eight rhesus monkeys for a two week experiment on motivation and learning
- Puzzles were placed in their cages

Results

- Unprompted by any external motivation, the monkeys solved the puzzles on their own
- This was an interesting and little understood phenomenon
- As a motivator, raisins were added as rewards
- Result: the monkeys made more errors and solved the problems less frequently

“It appears that the performance of the task provides its own intrinsic reward … this drive … may be as basic as the others … “

What Motivates Us

Drive: The Surprising Truth About What Motivates Us

Source: Drive: The Surprising Truth About What Motivates Us by Daniel H. Pink. 2011

youtube.com/watch?v=u6XAPnuFjJc&feature=youtu.be
What is Our Larger Purpose?

Better software makes the world a better place!
Conclusion

The foundation of Lean is LEADERSHIP

The foundation of SAFe is YOU

Take the next STEPS ...