

Balancing the Tension between Lean and Agile

James O. Coplien

Gertrud&Cope

Scrum Training Institute

cope@gertrudandcope.com

What is Agile?

- ③ We all know the Manifesto
- ③ It comes down to two things:
 - ③ Self-organization
 - ③ Feedback
- ▶ An exercise exemplifying Agile

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.

- Kent Beck
- Mike Beedle
- Arie van Bennekun
- Alistair Cockburn
- Ward Cunningham
- Martin Fowler
- James Grenning
- Jim Highsmith
- Andrew Hunt
- Ron Jeffries
- Jon Kern
- Brian Marick
- Robert Cecil Martin
- Steve Mellor
- Ken Schwaber
- Jeff Sutherland
- Dave Thomas

© 2001, the above authors
this declaration may be freely copied in any form,
but only in its entirety through this notice.

What is Lean?

- ① Lean is a more complex system aimed at adding value for the end user
- ① To do that,
 - ② We eliminate waste
 - ② We eliminate inconsistency
 - ② We smooth out flow
- ① We are constantly improving: Kaizen
- ▶ An exercise to illustrate Lean

Just a thought...

④ Lean:

- ④ Eliminate Waste
- ④ Eliminate inconsistency
- ④ Smooth out flow

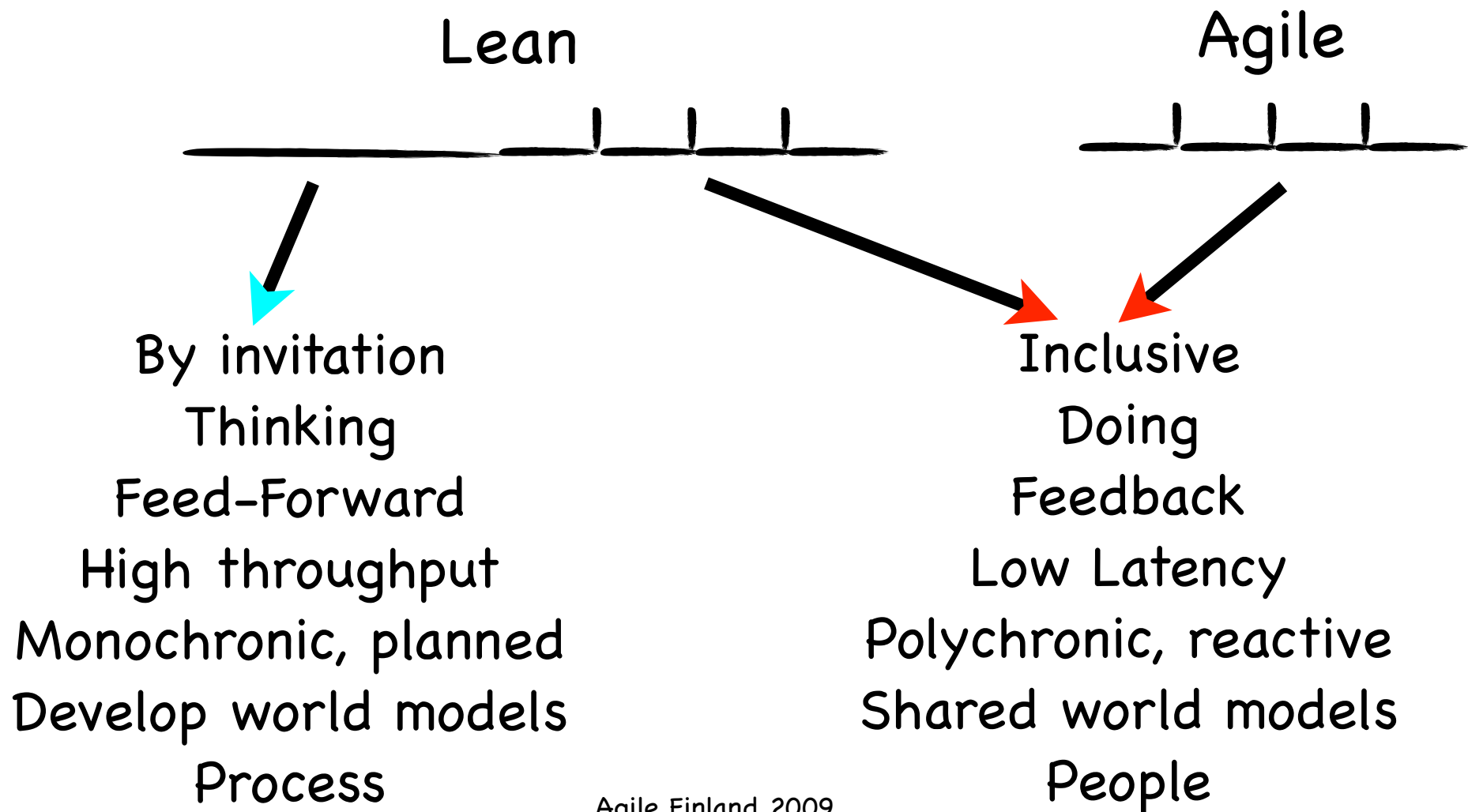
④ Agile:

- ④ Favor product over documentation
- ④ Communication
- ④ Sustainable pace

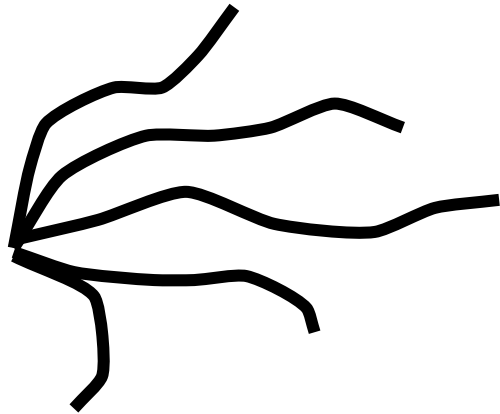
Reflection

- ④ What can you learn from the Agile exercise that would help software development?
- ④ What about the same for the Lean exercise?
- ④ What is common between these two exercises?
- ④ What is different?
- ④ Can you see any conflicts between them?

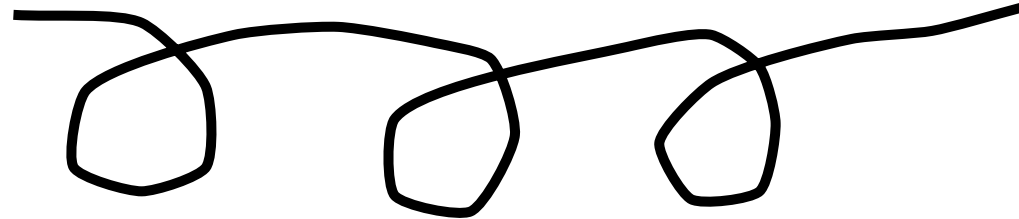
Lean and Agile Characteristics



Set-based Design: Complementary to Feedback



Ten
Alternatives



Refine the
one chosen

+ Rework in analysis adds value; rework in
manufacturing is cost

Complex versus Complicated

🌐 Agile: Dealing with complex systems: autopoietic systems, self-organization; wholes greater than the sum of their parts

🌐 Lean: Dealing with complicated systems. Building a car is complicated but not complex; the whole is the sum of its parts

Standards?

④ Agile: Inspect and adapt: anyone can do it, you don't need to ask permission, you are empowered, and you achieve continuous improvement

④ Lean: if you have a problem, spend up-front time seeking standards (Toyota Way, principle 6: Standardized Tasks are the Foundation for Continuous Improvement and Employee Empowerment)

Doing or Thinking?

- Agile: embrace change (but more on this later)
- Lean: Long deliberation and thought with rapid deployment only after a decision is made (The Toyota Way, Principle 13: Make decisions slowly by consensus, thoroughly considering all options)

Specialization

- ④ XP: No code ownership, no specialization. Scrum: cross-functional team
- ④ Lean: spend years carefully grooming each individual to develop a depth of knowledge (from Toyota Way, Principle 10)

Rework

- ⦿ Agile: Refactoring compensates for architectural short-sightedness, maintenance, and emergent requirements (as well as keeping the code clean)
- ⦿ Lean: Rework in design adds value, while rework in production is waste (Ballard: Negative Iteration, Lean Institute)

Last Responsible Moment

- ④ Agile: early decisions are likely to be wrong and cause rework, so defer to the last responsible moment
- ④ Lean: letting a decision go beyond the point where it affects other decisions causes rework, so bring decisions forward to a point where their results don't propagate

Summary

- ③ Complex vs Complicated
- ③ Ad-hoc vs Standards
- ③ Doing vs Thinking
- ③ Generalization vs Specialization
- ③ Rework vs Prototyping
- ③ Late versus Early Decisions

Focus

🕒 Agile: Doing

- 🕒 Where is the thinking in Scrum?
- 🕒 Scrum allows thinking but neither requires it nor provides techniques for it
- 🕒 Agile is about doing just enough to get just enough quality

🕒 Lean: Doing and thinking

- 🕒 Lean requires thinking and even planning
- 🕒 Lean is about doing the least you can do to achieve excellence

Repercussions

- ⦿ Agile fears future change; Lean embraces it
 - ⦿ Enlist a fire brigade or build a brick city?
- ⦿ Agile shuns architecture; Lean considers it essential
- ⦿ Agile avoids committing to agreed standards; Lean is about commitment
- ⦿ Agile is about individuals; Lean about teams
- ⦿ Agile diffuses responsibility; Lean encourages it
- ⦿ Agile puts value in production rework (refactor; do-inspect-adapt); Lean avoids rework (design, inspect-plan-do)

Reflection

- ④ We can mix Lean and Agile
- ④ Reflect on a mixture of practices, ideas, and philosophies that combine Lean and Agile thinking

Scrum

- ④ Comes from a Lean legacy, but has many trappings we associate with Agile
- ④ Common problems in Scrum implementations:
 - ④ People don't do the up-front analysis, lean architecture
 - ④ People don't take the value chain out to the end user
 - ④ People think locally rather than in terms of their networks

Does the Nokia Test test Lean or Agile?

- ③ Iterations
- ③ Expanding scope of Done to deployment
- ③ Up-front specifications w/User Stories
- ③ Product owner who plans
- ③ Up-front Product Backlog
- ③ Up-front estimates
- ③ Business-oriented burndown chart
- ③ Team disruption

Most Nokia Test points are Lean – not Agile

- ④ Iterations (Agile)
- ④ Expanding scope of Done to deployment (Lean)
- ④ Up-front specifications w/User Stories (Lean)
- ④ Product owner who plans (Lean)
- ④ Up-front Product Backlog (Lean: DSM)
- ④ Up-front estimates (Lean planning)
- ④ Business-oriented burndown chart (Lean)
- ④ Team disruption (Agile)

Summary: The Lean / Agile Compromise

- ④ Do up-front work – but minimize artefact inventory and be postured for change
 - ④ Prototypes and enabling specifications
 - ④ Architecture
- ④ Make short-term sacrifices for long-term ROI
- ④ Pay strong attention to common strengths, such as short feedback loops
- ④ Use common sense!

1. Iterations

- No iterations - 0
- Iterations > 6 - 1
- Variable length < 6 weeks - 2
- Fixed iteration length 6 weeks - 3
- Fix iteration length 5 weeks - 4
- Fixed iteration 4 weeks or less - 10

2. Testing

- No dedicated QA - 0
- Unit tested - 1
- Feature tested - 5
- Feature tested as soon as completed - 7
- Software passes acceptance testing - 8
- Software is deployed - 10

3. Agile Specification

- No requirements - 0
- Big requirements document - 1
- Poor user stories - 4
- Good requirements - 5
- Good user stories - 7
- Software is deployed - 10
- Just enough, just in time specifications - 8
- Good user stories tied to specifications as needed - 10

4. Product Owner

- No Product Owner - 0
- Ignorant product owner - 1
- Meddling product owner - 2
- Detached product owner - 2
- Product owner with clear product backlog estimated by team before Sprint Planning meeting (READY) - 5
- Product owner with release road map based on team velocity - 8
- Product owner who motivates the team - 10

5. Product Backlog

- No Product Backlog - 0
- Multiple product backlogs - 1
- Single product backlog - 3
- Product backlog clearly specified and ordered for ROI before sprint planning (READY) - 5
- Product owner with release plan based on Product Backlog - 7
- Product Owner can measure ROI based on revenue, cost per story point, or other metrics - 10

6. Estimates

- Product Backlog not estimated - 0
- Estimates not produced by team - 1
- Estimates not produced by planning poker - 5
- Estimates produced by planning poker by team - 8
- Estimate error < 10% - 10

7. Burndown Chart

- No burndown chart - 0
- Burndown chart not updated by team - 1
- Partial task burndown - 2
- Using Track Done - 4
- Using Track Story done - 5
- Add 3 points if the team knows velocity
- Add 2 points if Product Owner release plan is based on known velocity

8. Team Disruption

- Manager / Project Leader disrupts team - 0
- Product Owner disrupts team - 1
- Managers, Project Leaders or Team Leaders assign tasks - 3
- Have Project Leader and Scrum roles - 5
- No one disrupts team, only Scrum roles - 10