Scaling Agile Software Development

Scott W. Ambler
Chief Methodologist/Agile
Agenda

- The Astronomy Analogy
- The Agile Scaling Model
- Some Industry Statistics
- Scaling Strategies
Around 300 B.C. ...
Around 1530…
Around 1700...
Around 1970
2001

80% Operations

20% Development
Around 2050???

- Systems which actually work within a common infrastructure
- Systemic reuse
- High quality across all architectural layers
- IT an integral part of the business
- Full system lifecycle
- Systemic process improvement
- Effective human resources for IT
- IT as a true profession
## Agile Scaling Model (ASM)

### Core Agile Development
Focus is on construction
Goal is to develop a high-quality system in an evolutionary, collaborative, and self-organizing manner
Value-driven lifecycle with regular production of working software

### Disciplined Agile Delivery
Extends agile development to address full system lifecycle
Risk and value-driven lifecycle
Self organization within an appropriate governance framework

### Agility@Scale
Addresses one or more scaling factors:
- Team size
- Geographical distribution
- Organizational distribution
- Regulatory compliance
- Technical complexity
- Organizational complexity
- Enterprise discipline

White paper coming soon on ibm.com
What is Agility@Scale?

Disciplined Agile Delivery

- **Team size**: Under 10 developers ↔ 1000’s of developers
- **Compliance requirement**: Low risk ↔ Critical, Audited
- **Geographical distribution**: Co-located ↔ Global
- **Organization distribution (outsourcing, partnerships)**: Collaborative ↔ Contractual
- **Enterprise discipline**: Project focus ↔ Enterprise focus
- **Technical complexity**: Homogenous ↔ Heterogeneous, Legacy
- **Organizational complexity**: Flexible ↔ Rigid
Largest team size attempted vs. successful

Source: Dr Dobb’s 2008 Agile Adoption Survey
Some observations

- Agile approaches require high levels of trust and communication
- Distribution reduces trust and makes communication more difficult
- Distribution and large team size often go hand-in-hand

Source: Dr Dobb’s 2008 Project Success Survey
Does your team have to comply to industry regulations?

- Yes: 33%
- No: 60%
- Don't Know: 7%

Source: Ambysoft Agile Practices 2009 Survey
Does your team follow a CMMI compliant agile process?

- Yes: 9%
- No: 78%
- Don't Know: 13%

Source: Ambysoft Agile Practices 2009 Survey
Agile and legacy systems

- Working with Legacy in Some Way: 78%
- Integrating with Legacy Systems: 57%
- Evolving Legacy Systems: 51%
- Working with Legacy Data: 45%

Source: Ambysoft Agile Project Initiation 2009 Survey
Scaling daily stand-up meetings

**Small, Co-located Team**
Scrum strategy is to ask 3 questions: What did you do yesterday?, What do you plan to do today?, and What is blocking you?

**Large Team**
Kanban strategy is to ask 1 question: What issues do you foresee?

**Distributed Team**
Meeting occurs electronically
Rational Team Concert (RTC) to share information
Change meeting times to reflect team distribution – spread the pain

**Regulatory Environment**
Take meeting attendance and record action items (if any)
Practice: Initial Requirements Envisioning

- Your goals are to:
  1. Identify and agree to the initial scope of your project
  2. Develop the initial stack of requirements
  3. Gather enough information to address initial scheduling and estimating concerns

- Critical models for business application development:
  - Some sort of usage model (use cases, user stories, …)
  - Conceptual/domain model
  - Some UI sketches

- Observation: Scope is not a requirements document, it is a continuous negotiation
Agile Testing at Scale

- You need to prove that the system works. Critical issues:
  - System integration testing
  - Installation testing
  - Stakeholder acceptance testing
  - Investigative testing
  - Non-functional testing

Effective agile teams push their working builds to an independent test team on a regular basis for investigative testing. Defects must be prioritized and put back on the team’s work stack. Defects == Requirements.

Scales TDD: TDD is a form of confirmatory testing. TDD is a great start, but it’s not the full testing picture.

Critical strategy for addressing non-functional requirements.
Lean Governance: Automate measure collection

Open Dashboard Items (36) Priority
- High
- Medium
- Low
- Unassigned

Open vs Closed Work Items
- Team Area: Jazz Development/Dashboard
- Category: Dashboards
- Interval: 0.0 RC3

Open Work Items by Type
- Team Area: Jazz Development/Dashboard
- Category: Dashboards
- Interval: 0.0 RC3

Open W
- Team Area: Category: Interval:

Open Dashboard Items (36) Priority
- High (8)
- Medium (20)
- Low (14)
- Unassigned (54)

Priority
- High
- Medium
- Low
- Unassigned

Count
- 8
- 20
- 14
- 54

Open Dashboard Items (36) Priority
- High
- Medium
- Low
- Unassigned

Count
- 8
- 20
- 14
- 54

Open Dashboard Items (36) Priority
- High
- Medium
- Low
- Unassigned

Count
- 8
- 20
- 14
- 54