Agile at Scale

Eric Cattoir
Client Technical Professional, IBM Software Group - Rational
eric_cattoir@be.ibm.com

Innovate 2013
Application Development and Systems Engineering

October 10 | Brussels

© 2013 IBM Corporation
Please note the following

IBM’s statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM’s sole discretion.

Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user’s job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.
Agenda

- Agile Development
- Typical Agile Methods
- Extended Agile Frameworks
  - Disciplined Agile Development
  - Agile at Scale
  - Scaling Agile Framework
- Supporting Agile at Scale with Rational Tools
- Conclusion
Agenda

- Agile Development
- Typical Agile Methods
- Extended Agile Frameworks
  - Disciplined Agile Development
  - Agile at Scale
  - Scaling Agile Framework
- Supporting Agile at Scale with Rational Tools
- Conclusion
What Agile is NOT!!

WE NEED THREE MORE PROGRAMMERS.

USE AGILE PROGRAMMING METHODS.

AGILE PROGRAMMING DOESN'T JUST MEAN DOING MORE WORK WITH FEWER PEOPLE.

FIND ME SOME WORDS THAT DO MEAN THAT AND ASK AGAIN.

WE'RE GOING TO TRY SOMETHING CALLED AGILE PROGRAMMING.

THAT MEANS NO MORE PLANNING AND NO MORE DOCUMENTATION. JUST START WRITING CODE AND COMPLAINING.

I'M GLAD IT HAS A NAME. THAT WAS YOUR TRAINING.
What is Agile Software Development?

Principles driving Agile adoption

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

manifesto for agile software development

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 Bennekum  
Alistair Cockburn  
Ward Cunningham  
Martin Fowler  
James Grenning  
Jim Highsmith  
Andrew Hunt  
Ron Jeffries  
Jon Kern  
Brian Marick  
Robert C. Martin  
Steve Mellor  
Ken Schwaber  
Jeff Sutherland  
Dave Thomas

www.agilemanifesto.org

Innovate2013

Application Development and Systems Engineering
Why are Organizations moving to Agile?

Project success rates are up

- Iterative: 71%
- Agile: 70%
- Traditional: 66%
- Ad-Hoc: 62%

More effective at delivering higher quality software

- Quality: Agile 4.9, Iterative 2.3, Traditional 2.7
- Functionality: Agile 6.0, Iterative 1.8, Traditional 2.7
- Money: Agile 3.9, Iterative 3.0, Traditional 0.2, Ad-Hoc 0.8
- Time: Agile 4.4, Iterative 4.0, Traditional 0.8, Ad-Hoc 0.8

**Bottom Line:** Agile teams produce higher quality work, are quicker to deliver, are more likely to deliver the right functionality, and more likely to provide greater ROI than traditional teams

Agenda

- Agile Development
- **Typical Agile Methods**
- Extended Agile Frameworks
  - Disciplined Agile Development
  - Agile at Scale
  - Scaling Agile Framework
- Supporting Agile at Scale with Rational Tools
- Conclusion
Agile Methodologies

- ARE approaches used by software teams to coordinate their activities and how they work together (e.g. software processes)
- Share common principles, see Agile Manifesto, but use different practices
- Stress continuous customer feedback used to refine and deliver software
- Typically use the iterative and incremental software development practice

Some Common Agile Methodologies

- Lean Development (LD)
- Scrum
- eXtreme Programming (XP)
- Evolutionary Project Management (Evo)
- Test Driven Development
- User Story Driven Development
- Crystal
- RUP - Rational Unified Process
- ASD - Adaptive Software Development
- DSDM - Dynamic System Development Method
- FDD - Feature Driven Development
Agile Practices

**Practices:**
- Product Backlog
- Value-Driven Life Cycle
- Self Organization
- Release Planning
- Sprint Planning
- Daily Scrum Meeting
- Sprint Demo
- Retrospectives

**Roles:**
- Scrum Master
- Product Owner, Team Member

---

**Scrum Lifecycle**

- Daily Scrum Meeting
- Sprint Planning
- Release Planning
- Self Organization
- Value-Driven Life Cycle
- Product Backlog
- Potentially Shipable Product Increment

**XP Practices**

- Whole Team
- Coding Standard
- Test-Driven Development
- Software Ownership
- Planning Game
- Refactoring
- Simple Design
- Metaphor
- Small Releases
- Sustainable Pace
- Continuous Integration
- Team Development
- Planning Game
- Customer Tests
- Pair Programming
Agenda

- Agile Development
- Typical Agile Methods
- **Extended Agile Frameworks**
  - Disciplined Agile Development
  - Agile at Scale
  - Scaling Agile Framework
- Supporting Agile at Scale with Rational Tools
- Conclusion
Problems of Scaling

- **SoS – Scrum Of Scrums**
  - Becomes more difficult after 6 or so Teams
  - Planning & Ceremonial Events conflict

- **Doesn’t really address a Portfolio & Program View**
  - Still thinks of smaller “projects”
  - Planning Roadmap horizons are still short

- **Fails to recognize that Waterfall still exists**

- **Governance & Authority start to fail**
  - No Clear Content Authority once you scale to a Program or Portfolio level
  - Who resolves priorities across dozens of teams?
  - Who then drives releases?

- **Reporting & Metrics aren’t sufficient across large numbers of teams or programs**

- **Traditional sources of information (Scrum/Agile Alliance) aren’t mature to help this**
  - Note: In Jan ’2013 Ken Schwaber introduced CIF –Continuous Improvement Framework
What Should a Scaled Framework Address?

- **Multiple Agile Teams**
  - Should be able to handle dozens of teams (Scrum to break around 7)
  - Incorporation of XP Engineering practices
- **Waterfall Teams**
  - They still exist. Not everything can be Agile
- **Program Level planning and views**
- **Governance and shared resources**
  (like Enterprise/System Architects, UX, etc.)
- **Specialized teams for Release planning, system integration**
- **Clear content authority**
- **Portfolio Management and the management of WIP**
The Disciplined Agile Lifecycle: An extension of Scrum
Concept: The Agile 3C rhythm

The coordinate-collaborate-conclude rhythm occurs at several scales on a DAD project:

### Release rhythm

<table>
<thead>
<tr>
<th>Inception</th>
<th>Construction</th>
<th>Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coordinate</td>
<td>Collaborate</td>
<td>Conclude</td>
</tr>
</tbody>
</table>

### Iteration rhythm

<table>
<thead>
<tr>
<th>Iteration Planning</th>
<th>Development</th>
<th>Stabilize</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coordinate</td>
<td>Collaborate</td>
<td>Conclude</td>
</tr>
</tbody>
</table>

### Daily rhythm

<table>
<thead>
<tr>
<th>Coordination Meeting</th>
<th>Daily work</th>
<th>Stabilize</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coordinate</td>
<td>Collaborate</td>
<td>Conclude</td>
</tr>
</tbody>
</table>
The Inception Phase

- Initiate team
- Schedule stakeholders for envisioning sessions

Coordinate

Up to a few hours

Collaborate

Ideally: Up to a few weeks
Average: 4 weeks
Worst case: Several months

Conclude

Up to a few hours

Light-weight milestone review
Communicate vision to stakeholders

Stakeholder consensus
Typical Construction Iteration

- Iteration planning
- Iteration modeling

“Standard” practices:
- Visualize work
- Daily coordination meeting
- Refactoring
- Developer regression testing
- Model storming
- Continuous integration (CI)
- Sustainable pace
- Prioritized requirements
- Architecture spike
- Configuration management
- Burn-down chart
- Automated metrics

“Advanced” practices:
- Test-driven development (TDD)
- Acceptance TDD (ATDD)
- Continuous deployment (CD)
- Model a bit ahead
- Parallel independent testing
- Continuous documentation
- Non-solo development
- Iteration preplanning

- Iteration demo
- Retrospective
- Release planning (update)
- Consider sufficient functionality

Coordinate

Collaborate

Conclude

Typical: Two to four weeks
Average: Two weeks
Worst case: Six weeks

2 hours for each week of the iteration length

One hour per week of iteration length

Iteration start

Potentially consumable solution
Typical day during construction

- Daily coordination meeting
- Update task board
- Update iteration burndown

**Coordinate**

Up to 15 minutes

**Collaborate**

Typical: 5-6 hours

- Address blocking issues
- Create tests
- Develop code
- Integrate
- Fix problems
- Model storm
- Deploy to test/demo environment

**Conclude**

Ideally: Not a concern

**Stabilize build**

End of day

Start of day

Innovate2013
Application Development and Systems Engineering
The Transition phase

- Phase planning
  - Transition planning
  - End-of-lifecycle testing and fixing
  - Pilot/beta the solution
  - Finalize documentation
  - Communicate deployment
  - Train/educate stakeholders

Coordinate

Ideally: Nothing
Typical: One hour per week of collaborate time
Sufficient Functionality

Collaborate

Ideally: Nothing
Average: 4 weeks
Worst case: Several months

Conclude

Ideally: Less than an hour
Worst case: Several months
Production Ready

Innovate2013
Application Development and Systems Engineering
Agile Scaling Factors

- Team size: Under 10 developers ↔ 1000's of developers
- Compliance requirement: Low risk ↔ Critical, audited

- Geographical distribution: Co-located ↔ Global
- Domain Complexity: Straight-forward ↔ Intricate, emerging

- Enterprise discipline: Project focus ↔ Enterprise focus
- Organization distribution (outsourcing, partnerships): Collaborative ↔ Contractual

- Organizational complexity: Flexible ↔ Rigid
- Technical complexity: Homogenous ↔ Heterogeneous, legacy

* Slide Courtesy of IBM
Disciplined Agile Delivery (DAD) is the Foundation for Agile at Scale

<table>
<thead>
<tr>
<th>Compliance</th>
<th>Domain Complexity</th>
<th>Technical Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geographic Distribution</td>
<td>Team Size</td>
<td>Organizational Distribution</td>
</tr>
</tbody>
</table>

Outside In Dev. | SAFe | And more…
XP | Agile Modeling |
Scrum | Kanban | Lean

DAD leverages proven strategies from several sources providing a decision framework to guide your adoption and tailoring in a context-driven manner.
The Scaled Agile Framework is a proven, publicly-facing framework for applying Lean and Agile practices at enterprise scale.

- Synchronizes alignment, collaboration and delivery
- Well defined in books and now on the web
- Scales successfully to large numbers of practitioners and teams

Core values:
1. Code Quality
2. Program Execution
3. Alignment
4. Transparency

http://ScaledAgileFramework.com
Roots of the Scaled Agile Framework
Scaled Agile Framework – Big Picture
Agile Teams

- Empowered, self-organizing, self-managing cross-functional teams
- Valuable, fully-tested software increments every two weeks
- Scrum project management practices and XP-inspired technical practices
- Teams operate under program vision, system, architecture and user experience guidance
- Value description via User Stories
Scale to Program Level

- Self-organizing, self-managing team-of-agile-teams
- Continuous value delivery
- 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 description via Features and Benefits
Scale to Portfolio

- Centralized strategy, decentralized execution
- Investment themes provide operating budgets for trains
- Kanban systems provide portfolio visibility and WIP limits
- Objective metrics support governance and kaizen
- Value description via **Business** and **Architectural Epics**
Agenda

- Agile Development
- Typical Agile Methods
- Extended Agile Frameworks
  - Disciplined Agile Development
  - Agile at Scale
  - Scaling Agile Framework
- **Supporting Agile at Scale with Rational Tools**
- Conclusion
Rational tooling to support Agile

Tool integration supports **end-to-end traceability.**

### Requirements Management
- User Stories and AC linked to support coverage and impact analysis
- Rqts. Comm. / sign-off
- Custom artifacts and templates implemented to support Agile Process

### Work Items & Collaboration
- Agile backlogs
- Tasks to track work effort
- Agile dashboards for executive reporting
- Code management via BuildForge

### Quality Management
- Test Plans
- Test Cases
- Evaluating Rational Tester for regression automation

### Rational tooling
- Rational Team Concert
- Jazz Platform
- Rational Rqts. Composer
- Rational Quality Manager
Process Templates – Out of the Box

There are a number of process templates that come out of the box with RTC v4.x:

• Cloudburst Sample Process
• Formal Project Management
• OpenUP Process
• Scrum
• Simple Team Process
• Unconfigured Process

Other process templates:
• Disciplined Agile Delivery
• SAFe Portfolio
• SAFe Program
DAD Timeline

**Timelines**
The project timeline defines a start and end date along with an iteration breakdown. Additional timelines can be defined to track secondary activities.

**Tip:** Iterations can be manually reordered using drag and drop.

**Defined Timelines**
- Main Development
  - Release x.y
    - Inception Iteration [n]
    - Construction Iteration [n]
    - Transition Iteration [n]
DAD Work Item Types

- Name: defect
- ID: defect
- Category: com.ibm.team.workitem.workItemType
- Icon:
- Aliases: bug

Work Item Editor
- None (Default: com.ibm.team.workitem.editor.default)

Inline Work Item Editor
- com.ibm.team.workitem.web.inline.default

Lightweight Work Item Creation Dialog
- com.ibm.team.workitem.lightweight.editor.section

Plan Editor Preview
- com.ibm.team.apt.planPreview.default
SAFe Template for RTC

- RTC already has Scrum & Kanban process templates but it didn’t have one for SAFe
- New workitem types needed to be created:
  - Feature
  - Theme
  - Risk & Risk Action
- New Roles and Security
  - Product Manager
  - Release Train Engineer (Conductor)
- Specialized Virtual Teams
  - System Team
  - Release Management
  - Architecture & UX
SAFe Portfolio (RTC Process Template)

**Roles**
- Portfolio Manager
- Business Owner
- Enterprise Architect
- Business Analyst
- Strategic Planner
- System Architect
- Stakeholder
- Portfolio Administrator

**Teams**
- SAFe Portfolio
  - Architecture Kanban Team
  - Business Kanban Team

**Work Item Types**
- Task
- Feature
- Story
- Epic
- Spike
- NFR
- Impediment

**Timeline**
- Portfolio [Project Timeline]
  - Portfolio Kanban
    - The Funnel
    - Backlog
    - Analysis
    - Implementation

**Categories**
<table>
<thead>
<tr>
<th>Categories</th>
<th>Associated Project/Team Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unassigned &lt;Root Category&gt;</td>
<td>SAFe Portfolio [Project Area]</td>
</tr>
<tr>
<td>SAFe Portfolio</td>
<td>SAFe Portfolio [Project Area] [Inherited]</td>
</tr>
<tr>
<td>Architecture</td>
<td>Architecture Kanban Team</td>
</tr>
<tr>
<td>Business</td>
<td>Business Kanban Team</td>
</tr>
</tbody>
</table>
Solution outline for Outsourced Software Delivery Governance

*Built on a core solution set; augmented with IBM Best Practices*

- **PORTFOLIO STRATEGY and MANAGEMENT**
  Develop outsourcing strategy and consolidate organizational demand to prioritize capability needs

- **REQUIREMENTS MANAGEMENT**
  Translate Business Needs into Actionable SW Requirements, driving SOW agreements

- **QUALITY MANAGEMENT**
  Achieve “quality by design” with integrated user acceptance testing, facilitating customer sign-off

- **COLLABORATION, PLANNING & CHANGE MANAGEMENT**
  Collaborate across geographically distributed software development suppliers using common planning and scheduling

- **PERFORMANCE REPORTING and ANALYSIS**
  Automated, transparent collection of role-based dashboards to drive SLA and Supplier performance reporting and analysis across the SW Supply Chain
Agenda

- Agile Development
- Typical Agile Methods
- Extended Agile Frameworks
  - Disciplined Agile Development
  - Agile at Scale
  - Scaling Agile Framework
- Supporting Agile at Scale with Rational Tools
- Conclusion
Questions

Innovate 2013
Application Development and Systems Engineering
## Acknowledgements and disclaimers

**Availability**: References in this presentation to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates.

The workshops, sessions and materials have been prepared by IBM or the session speakers and reflect their own views. They are provided for informational purposes only, and are neither intended to, nor shall have the effect of being, legal or other guidance or advice to any participant. While efforts were made to verify the completeness and accuracy of the information contained in this presentation, it is provided AS-IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this presentation or any other materials. Nothing contained in this presentation is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth or other results.

© Copyright IBM Corporation 2013. All rights reserved.
- **U.S. Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.**

IBM, the IBM logo, ibm.com, Rational, the Rational logo, Telelogic, the Telelogic logo, Green Hat, the Green Hat logo, and other IBM products and services are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml)

If you have mentioned trademarks that are not from IBM, please update and add the following lines:

[Insert any special third-party trademark names/attributions here]

Other company, product, or service names may be trademarks or service marks of others.
Thank You