Scaling Scrum

Colin Bird & Rachel Davies
Scrum Gathering London 2007
Scrum on a Slide

Preparation
- Business case & funding
- Contractual agreement
- Vision
- Initial product backlog
- Initial release plan
- Stakeholder buy-in
- Assemble team

Process
- Sprint planning meeting
- Daily cycle
  - Daily scrum
  - Daily work
- Update product backlog
- Product increment
- Product review
- Sprint retrospective

Scrum artefacts
- Product backlog
- Impediment list
- Product backlog burndown
- Product backlog delta report
- Sprint backlog
- Sprint backlog burndown

Scrum roles
- Product owner
- Scrum master
- Users
- Team members
- Stakeholders
Does Scrum Scale?

Ok, so Scrum is great for a small team but what happens when you have to work on a big project?

- Large projects have a high complexity and greater risk of failure
- Scrum is empirical so seems a logical choice
- It is possible
- Although it’s hard work!
What does “Big” mean?

- A Scrum team is 7 +/- 2 people
  - Above this number the team doesn’t function efficiently

- What else?
  1. Many teams
  2. Geographically dispersed
  3. Agile adoption at an organisational level
  ➢ Often all of the above!
Raise your hand if...

- You tried Scrum on a project like this:
  - 1-5 teams
  - 5-15 teams
  - Multi-site teams
  - Entire IT department
  - Entire company

Any others?
Scrum works because ..

SCRUM uses these tools:
- Focus on goal
- Visible progress and results
- Removing interruptions
- Daily review of issues
- Accepted responsibility
- Simple rules

But some may disappear in scaling
What does Scrum literature say?

- The beauty of Scrum is that it’s simple
- It’s a toolkit for an “Inspect & Adapt” approach
- Scrum does not contain optional elements
- We learn about scaling thru case studies
Scrum practice:

- Subdivide into Scrum teams and hold “scrum of scrums” meeting
- These meetings allow clusters of teams to discuss their work, focusing especially on areas of overlap and integration.
- Attended by a representative from each team
- Not the 3 question format
- More focus on issues
Challenges to Scaling Scrum
The Agile Manifesto

Scale means that it is even more important to understand the fundamental principles behind the Agile Manifesto and apply them to the specific challenges.

While there is value in the items on the right, we value the items on the left more.

<table>
<thead>
<tr>
<th>Individuals &amp; interactions</th>
<th>over</th>
<th>Processes &amp; Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Working Software</td>
<td>over</td>
<td>Comprehensive Documentation</td>
</tr>
<tr>
<td>Customer Collaboration</td>
<td>over</td>
<td>Contract Negotiation</td>
</tr>
<tr>
<td>Responding to Change</td>
<td>over</td>
<td>Following a Plan</td>
</tr>
</tbody>
</table>

http://www.agilemanifesto.org
What impact does scale have on the Agile principles?

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
• The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

• Working software is the primary measure of progress.

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

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

• Simplicity--the art of maximizing the amount of work not done--is essential.

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

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

http://www.agilemanifesto.org/principles.html
Scrum Roles Challenged

• **Product Owner**
  • Workload
  • Being at all the meetings

• **Scrum Master**
  • More inter-team issues
  • Dependencies

• **Scrum Team**
  • Less empowered
  • Large or distributed
Scrum Practices Challenged

- **Sprint Planning**
  - Harder to establish goal as need to consider other teams
- **Daily Scrum**
  - Distributed team members
- **Product Increment**
  - Dependencies on other teams
- **Sprint backlog**
  - Emphasis on electronic communications for remote team members and other teams
- **Product backlog**
  - More money at stake so more political wrangling
- **Sprint Review**
  - Whole product or team output?
- **Sprint Retrospective**
  - Joining forces to resolve common problems

Any others?
Other Scaling Challenges

• Organisational
  • Departmental structure and line management
  • Cultural impedance
  • Reward/Motivation systems
  • Political – loss of power/control
  • Existing project/programme management and measurement

• Logistical
  • Working environment
  • Technical infrastructure
Notes

- Vision - Communicate Widely and Often
- Models for inter team synchronisation
- Train the teams and management
- Coach the teams and management
- Don’t be too prescriptive and complete with Scaling model – let the team find their own solutions
- Ensure leadership is within the teams not above it
- Start with a consolidated view of the whole backlog – A Programme Backlog
Approaches to the Challenges

Team Structures
Start Small

Start with a “Beachhead” team and then build from there.

Initial Team

Add New Team members

Time
Existing Teams Unite

- Extract core team
  Focusing on forward plan

- Integration team
  Keeps the whole working
Independent Projects

Independent Teams

- Multiple Customers
  - Requirements independently prioritised
- Teams Are Independent
  - Separate Product Backlogs
  - Separate Sprint Planning
  - Separate Sprint Backlogs

But what about reuse?
A Bigger Project  1 < Teams < 3

1st Stage Scale Model

- Single Customer
  - Single set of prioritised requirements
- Joint Sprint Planning & Reviews
- Separate Sprint Backlogs
- Dynamic team split from iteration to iteration
- Works well for a small number of teams
- Runs into issues above 3 teams
  - Sprint Planning & Reviews
  - Daily Scrums
  - Common code base and check-in/out
  - Code contention
2nd Stage Scale Model

• Streams are logical subsets of the Backlog(s) split by:
  • Business Service
    • E.g. CRM
  • Technical Service
    • E.g. Web Platform

Stream Backlog 1
- 1 Feature a
- 2 Feature b
- 3 Feature c
- 4 Feature d

Stream Backlog 2
- 1 Feature e
- 2 Feature f
- 3 Feature g
- 4 Feature h

Team 1
Team 2
Team 3
Team 4
Teams as “Customers”

- Customer facing teams can generate their own requirements for common “Platform” features
  - Teams 1 & 2 are “Customers” for Team 3
  - Team 3 has a Product Owner who prioritises the Technical backlog
  - Team 3 consults with Teams 1 & 2 while building their platform features
  - Team 3 Sprint Review to show output
  - Teams 1 & 2 can feedback
  - Teams 1 & 2 integrate platform features into their own delivery in subsequent iterations
  - Team 3 may work to a smaller iteration length, but still in phase with Teams 1 & 2
  - Team 3 may “Borrow” team members from Teams 1 & 2 – agreed in Sprint Planning
Combining the Models

Programme or Portfolio
Product Backlog

Programme Backlog
- 1 Feature a
- 2 Feature b
- 3 Feature c
- 4 Feature d
- 5 Feature e
- 6 Feature f
- 7 Feature g

Stream
Product Backlogs

Enterprise Arch Stream
- 1 Feature aa
- 2 Feature dd

Ecommerce Stream
- 1 Feature a
- 2 Feature d

Inventory Mgt Stream
- 1 Feature b
- 2 Feature f
- 3 Feature g

CRM Stream
- 1 Feature c
- 2 Feature e

Infrastructure Stream
- 1 Feature a
- 2 Feature b
- 3 Feature f

Fulfilment Teams
Sprint Backlogs

Enterprise Architecture
Ecomm 1
Ecomm 2

Pricing
Allocation & Movements
UI +

CRM
Infrastructure
And With Multiple Customers
Distributed Teams

• Co-location is ideal
  • Anything else is less than ideal!

• If the teams have to be distributed – minimise the communications gap
  • Arrange the teams to match geographical and architectural boundaries
  • Instant Messenger
  • VOIP enables low cost voice communication - Skype
  • Web Conferencing for team presentations – WiredRed, Live Meeting
  • Good quality phone conference facilities
  • Programme and team level wikis
  • Face to Face meetings/working – swap team members

• Maintain daily “Stand-up” meetings

• Ensure the teams have an integrated development infrastructure
  • Common source control system and structure

• Ensure regular integration of all teams output
Approaches to the Challenges

Inter-team Communication and Synchronisation
Inter-team Communication

How do Multiple Teams work together?

Informal team to team collaboration

Communities of common disciplines

E.g. Resolve integration issue

E.g. DBAs
Stagger Daily Stand-up Meetings

Any team members can attend any other team stand-up meetings.
Limit to Staggered Daily Stand-ups

Not practical to stagger all Daily Scrums with many teams
Parallel Daily Stand-up Meetings

Run parallel sets of Scrums grouping by Stream
### Scrum of Scrums

Classic model for scaling Scrum projects

- **Coordination across teams through the Scrum meetings**
  - Aggregated view of requirements
  - Programme view of impediments
- **Prone to Command & Control mentality**
  - Implied Hierarchy

<table>
<thead>
<tr>
<th>Scrum of Scrums of Scrums</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Scrum of Scrums</td>
<td></td>
</tr>
<tr>
<td>Teams</td>
<td></td>
</tr>
</tbody>
</table>
Align Iterations Across Teams

Enables Joint Planning

Team 1
Iteration 1

Team 2
Iteration 1

Team 3
Iteration 1

Smaller Iterations In Phase

Enables Joint Delivery

Team 1
Iteration 2

Team 2
Iteration 2

Team 3
Iteration 2

Team 3
Iteration 3

Team 3
Iteration 4
Approaches to the Challenges

Planning
Scrum Lifecycle

- Add a **Sprint Zero** to get Product Backlog, Release Plan and Infrastructure in place
- Follow this with Sprints resulting in tested and integrated product increment
- Apply final polish in **Consolidation** sprint
Levels of Planning

Release Planning & Programme Backlog

Sprint Goals & Sprint Planning

Stream Planning

Team Sprint Planning

> Sprint Backlogs
Maintaining Levels in Plans

Planning precision

• Vision
• Roadmap with goals per release
• High-level view with Themes per sprint
• Product Backlog
• Sprint Backlogs per team
Product Owner?

- One person in this role for a large project may not scale!
- Consider creating a Product Owner team
  - One person to set priorities and mediate between stakeholders
  - One person per team to help communicate what’s required for each product backlog item
Common Language

- Need to create shared **domain model** and **glossary** to help teams maintain consistency
- Also need architecture guidelines
Visible Progress

- It may be useful to start using a planning tool to help create an overall backlog across teams (such as VersionOne, Rally, ScrumWorks, Scrum for Team System, etc)
- You can still use spreadsheets or wikis but then you need dedicated admin time to maintain the current picture
Integration?

• If you divide the teams into component teams or feature teams working in parallel using Scrum then you need to put some thought into Integration.
• Form a virtual team to tackle integration issues - establish another S2 meeting for this.
• You may need a Release Sprint but watch out you don’t slip into Scrumfall.
Testing

- You are likely to need an additional team of testers to support integration, UAT, regression, stress testing of the whole integrated system rather than the output of individual teams
- This is not a substitute for embedded testers working within each Scrum team
Product Showcase

• Each team will hold a Sprint Demo of the features developed but it’s important not to lose sight of the whole
• Organise **Product Showcase** meetings open to all teams at regular intervals within the project
Scaled Infrastructure Requirements

- Integrated Source Control
  - Consistent source structure
  - Share common code
- Consistent engineering and quality practices
- Lots of environments
  - Sandboxed team development environments
  - Sandboxed team testing environments
  - Integration environments
- Automation
  - Builds – Team and Integration
  - Deployments
  - Smoke testing
  - Regression testing
- Regular performance testing
- Backlog(s) visibility across teams
  - Dependency mapping
  - Reuse opportunities – components and frameworks
- Team metrics
- Programme metrics
Scaled Infrastructure Support

Build & Deployment Smoke Tests, Regression and Load & Performance Testing

Testers
Team Test Controller
Team Test Agent(s)

Developers
Team Foundation Server
Team Build Server

Centralised Source Control is the basis for build and deployment to all environments
Striking a balance

More team members
- longer meetings
- less focus

Contention for single PO
- higher workload
- less available

Cross-functional team
- less time with skill group

More teams
- more time to co-ordinate between teams
- working at cross-purposes

PO team
- more churn on decisions

Specialist teams
- miss the big picture
 Scaling Summary

• Keep to Agile Principles
• Build up rather than “Big Bang”
• Prepare for scale but don’t be prescriptive
  • Communicate widely
  • Provide the infrastructure, tools and logistical support
  • Mitigate corporate Programme reporting requirements
• Let the teams evolve their own inter-team communication and working practices
  • But ensure common engineering practices and supportive infrastructure
• Be prepared for mistakes and unforeseen issues
  • But ensure the lessons are learned and applied
Questions?

colin.bird@conchango.com
http://blogs.conchango.com
www.scrum-master.com
www.scrumforteamsystem.com

rachel@agilexp.com
http://www.agilexp.com/
http://twelve71.typepad.com/rachel