Overview and Context

Resources

Why Small Teams?

Physical Work vs Knowledge Work

<table>
<thead>
<tr>
<th>Physical Work</th>
<th>Knowledge Work</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td>Example</td>
</tr>
<tr>
<td>Picking apples</td>
<td>Developing software</td>
</tr>
<tr>
<td>Shoveling</td>
<td>Engaging deep problems</td>
</tr>
<tr>
<td>Nature</td>
<td>Nature</td>
</tr>
<tr>
<td>Tangible, Visible, Atomic</td>
<td>Intangible, Invisible, Bits</td>
</tr>
<tr>
<td>Ability to scale</td>
<td>Ability to scale</td>
</tr>
<tr>
<td>Easy</td>
<td>Difficult</td>
</tr>
<tr>
<td>Adding more people</td>
<td>Adding more people</td>
</tr>
<tr>
<td>Time</td>
<td>Time</td>
</tr>
<tr>
<td>Work completes faster</td>
<td>Work completes later</td>
</tr>
<tr>
<td>Projects</td>
<td>Projects</td>
</tr>
<tr>
<td>Economics of scale</td>
<td>Economics of scale</td>
</tr>
<tr>
<td>Positive</td>
<td>Negative</td>
</tr>
</tbody>
</table>

Physical Work has economies of scale
Knowledge Work has diseconomies of scale
Why Small Teams?

1. Diseconomy of scale
   - Adding more people does not make it go faster
   - More people adds communications overhead
   - More people just makes it cost more

2. Knowledge work based on solving problems and sharing information

3. Knowledge work is driven by Communication

Why Small Teams?

Number of communication channels = \(\frac{N(N-1)}{2}\)

- 2 people, 1 comm. line
- 3 people, 3 comm. lines
- 4 people, 6 comm. lines
- 8 people, 28 comm. lines
- 16 people, 120 comm. lines

Why Small Teams?

- Face-to-face communication:
  - Fast
  - Cheap
  - Allows for Q&A
  - Conveys body language and emotions
  - Does not scale well

Why Small Teams?

Two Pizza Teams
5-9 people – Miller’s Law

Local Optimization

Goldilocks Team Size
- Not too big – Smooth communications
- Not too small – Few handoffs

Optimized for team performance
**Single Team Lifecycle**

- **Product Backlog**
- **Sprint Backlog**
- **Sprint Stories**
- **Sprint**
- **Plan & Develop**
- **Learn & Evaluate**
- **New Functionality**

**Single Team Workflow**

<table>
<thead>
<tr>
<th>Week</th>
<th>Backlog</th>
<th>Ready for Dev.</th>
<th>In Development</th>
<th>In Testing</th>
<th>Done</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Agenda**

1. Single Team Flow ✓
2. Multi-Team Dependency and Handoff Issues
3. Solving Dependency Issues

**Multiple Team Workflow (With Dependencies)**

**Handoffs – 8 SW Wastes - D.O.W.N.T.I.M.E.**

- **Defects**
  - Correction and rework
- **Overproduction**
  - Unnecessary features or process
- **Waiting**
  - Delays for approvals and people availability
- **Non-Utilized Talent**
  - Under used skills and knowledge
- **Transportation**
  - Handoffs between teams lose information
- **Inventory Excess**
  - Partially done work, no ROI returned yet
- **Motion**
  - Task switching, catch-up
- **Extra-Processing**
  - Relearning, overengineering
Agile Team Structures

Agile promotes moving from Silos to Holistic Teams of Generalizing Specialists

**Holistic Teams**: All skills necessary to develop, test, deploy and get work accepted

**Generalizing Specialists**: Possessing both a core competence and ability in supporting roles. Sometimes called "T" shape people.

Agile Teams Using Cloud and Microservices

The Delivery Team: A team that has everything and everyone they need to deliver a working increment of tested, documented, deployable solutions.

Sponsors

The Team

Facilitator

Product Owner(s)

Cloud Support

Agile Team Structures

Solving Dependency Problems

1) **Visibility**: Make dependencies, blockers and queues visible

2) **Metrics**: Measure # of dependencies, durations of blocked items, lengths of queues

3) **Expedite**: Assign people to unblock stuck work, focus on items with most dependencies

4) **Merge Teams**: Consider fewer, larger teams to improve workflow

5) **Move from Projects to Products**: Restructure for long term value delivery

   - Major reduction in handoffs
   - Knowledge is retained and developed
   - Investment focused on long term employees
   - Vendor costs minimized

Make Visible: Show Dependencies and Queues

<table>
<thead>
<tr>
<th>On Deck</th>
<th>Ready for Dev.</th>
<th>In Dev.</th>
<th>Dev. Complete</th>
<th>Ready for Test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Make Visible: Show Dependencies and Queues**

<table>
<thead>
<tr>
<th></th>
<th>Ready for Dev</th>
<th>In Dev</th>
<th>Dev Complete</th>
<th>Ready for Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>With Dependencies</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No Dependencies</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Metrics:**

Tracked and Reported:
- Measure # of dependencies
- Durations of blocked items
- Lengths of queues

**Expedite Blocks:**

Assign people to unblock stuck work, focus on items with most (critical) dependencies

- Need sufficient authority and connections to resolve problems
- Have a clear escalation path

**Merge Teams:**

Switch from Projects to Products

1. Organizational Changes
2. Business Engagement
3. Budgeting and Planning
### An Iterative and Incremental Approach to Transitioning to Agile Product Development

**Current: Separation by Function**

- **Sponsors**
  - Business
  - Vendor 1
  - Vendor 2
  - Vendor 3
- **Vendors**
  - eCommerce
  - Website
  - Single Sign On
  - eCare
- **Project Teams**
  - Marketing Cloud
  - Microservices
- **Supporting Teams**
  - OPS
  - Emc.
- **Maintenance**

**Future: Align Around Products. Bring Business and Sustainment Work to Product Teams**

- **Sponsors**
  - Business
  - Vendor 1
  - Vendor 2
  - Vendor 3
- **Vendors**
  - Website
  - App
  - Product 3
- **Product Teams**
- **Supporting Teams**
  - OPS
  - Emc.

### Incremental Rollout – One Product First, then a couple more, then the remainder

- **Products 1, 2 & 3**
- **Products 4-end**

### Incremental Rollout – One Product First, then a couple more, then the remainder

### This experience did not work out. Yet if the problems sound familiar:

- Director let go
- Interim director, no agile experience
- Initiative halted
- Why Agile?
- Most projects outsourced
- My contract terminated

### Think global, not just local optimization

### Reduce Handoffs / Dependencies

### Visualize work and issues

### Sometimes bigger teams might be better

### Think products not projects
Next Steps, Questions

Get in touch:
- Email: Mike@LeadingAnswers.com
- Website: www.LeadingAnswers.com
- Twitter: @AgileMikeG
- LinkedIn: https://www.linkedin.com/in/mikegriffiths/

Visualizing Dependency Problems  Product vs. Project Development