Two-Pizza Team Heartburn Relief: Solutions for Team Dependencies

Overview and Context

- Why Small Teams?
- Dependency Issues
- Trends Increasing Dependencies
- Solutions
- Next Steps

Resources

Simulation

Case Study

Recommendations
Resources

Quantifying Dependency Problems
Visualizing Dependency problems
Product vs. Project Development

Why Small Teams?
Why Small Teams?

Physical Work vs Knowledge Work

<table>
<thead>
<tr>
<th></th>
<th>Physical Work</th>
<th>Knowledge Work</th>
</tr>
</thead>
<tbody>
<tr>
<td>Examples:</td>
<td>Picking apples, moving dirt, welding pipeline</td>
<td>Developing software, designing new products, filmmaking, educational design</td>
</tr>
<tr>
<td>Nature</td>
<td>Tangible, Visible, Nons (physical)</td>
<td>Intangible, Invisible, Bins (informational)</td>
</tr>
<tr>
<td>Ability to Scale</td>
<td>Easy</td>
<td>Difficult</td>
</tr>
<tr>
<td>Adding more people:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time</td>
<td>Work completes earlier</td>
<td>Work completes later</td>
</tr>
<tr>
<td>Costs</td>
<td>Costs increase linear</td>
<td>Costs increase non-linear</td>
</tr>
<tr>
<td>Economics of Scale</td>
<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?

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

---

© 2018 Leading Answers Inc.
Why Small Teams?

Number of communication channels = \( N(N-1)/2 \)

- 2 people, 1 comm. line
- 3 people, 3 comm. lines
- 4 people, 6 comm. lines

Why Small Teams?

Team Communication Channels

- Team of 8 people, 28 comm. lines
- Team of 16 people, 120 comm. lines

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

Dependency Simulation:

**Exercise 1**: As a team of one, time yourself answering 10 questions (No dependencies)
Answer = 450

**Exercise 2**: As a team of one, time yourself answering 10 questions (With dependencies – make sure each table has teams 1-4)

Team 1 = 60  Team 2 = 90  Team 3 = 70  Team 4 = 90

Single Team Workflow

Week 7

<table>
<thead>
<tr>
<th>BACKLOG</th>
<th>READY FOR DEV</th>
<th>IN DEVELOPMENT</th>
<th>TESTING</th>
<th>DONE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Task Duration ≈ Sprint Length = 1-2 weeks
Project Duration ≈ Backlog Size / Average throughpout * Sprint Length
≈ 13 / 2 * 1 = 7 weeks
Agenda

1. Single Team Flow ✓

2. Multi-Team Dependency and Handoff Issues

3. Solving Dependency Issues

Multiple Team Workflow

Week 13

TEAM 1

TEAM 2

TEAM 3

Task Duration = # Teams * Avg. Sprint Length * 1.5
3 * 2.67 * 1.5 > 13 weeks

Vendor 1 - UX

Vendor 2 - App

Project Team

Operations Team

Microservices Team

Environments Team
Dependencies:

Every dependency doubles chance of being late

4 people arriving at restaurant for group seating
Case Study:

Vendor 1
- eCommerce
- Account
- Mobile
Tech. Foundation
- Microservices
- Sustainment
- Ops
Program 1
- App.
- Support
- Authent.
Program 2
- Sustainment
- Microservices
- Ops

Six
Team Location
- Calgary
- Denver
- Toronto / India

Wastes
- Stocks
- Overproduction
- Waiting
- Non-Utilized Talent
- Transportation
- Inventory Excess
- Motion
- Extra Processing

Trend 1: Switch to Cloud – Environment Teams

Often a new group administers cloud environments
Trend 2: Switch to Microservices – Microservice Teams

Monolith → Microservices

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.
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 handoff waste
   • Knowledge is retained and developed
   • Investment focused on long term employees
   • Vendor costs minimized

Make Visible: Show Dependencies and Queues
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><img src="image1" alt="Dependencies" /></td>
<td><img src="image2" alt="Dependencies" /></td>
<td><img src="image3" alt="Dependencies" /></td>
<td><img src="image4" alt="Dependencies" /></td>
</tr>
<tr>
<td>No Dependencies</td>
<td><img src="image5" alt="No Dependencies" /></td>
<td><img src="image6" alt="No Dependencies" /></td>
<td><img src="image7" alt="No Dependencies" /></td>
<td><img src="image8" alt="No Dependencies" /></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
Expedite Blocks:

Level 0: SM as Team Impediment Remover (Daily at Stand-Up)

Level 1: Management Team escalation beyond teams (Daily Optional at 12:00)

Level 2: Director escalation beyond department (When needed)

Key:
- Team member
- Scrum Master
- Manager
- Director

Merge Teams:

Switch from Projects to Products

1. Organizational Changes
2. Business Engagement
3. Budgeting and Planning
Current: Separation by Function

<table>
<thead>
<tr>
<th>Sponsors</th>
<th>Vendors</th>
<th>Project Teams</th>
<th>Supporting Teams</th>
<th>Maintenance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business</td>
<td>Vendor 1</td>
<td>eCommerce</td>
<td>Micro-services</td>
<td></td>
</tr>
<tr>
<td>Vendor 2</td>
<td>Website</td>
<td>App</td>
<td>OPS</td>
<td></td>
</tr>
<tr>
<td>Vendor 3</td>
<td>Single Sign On</td>
<td>eCare</td>
<td>Env.</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Sponsors</th>
<th>Vendors</th>
<th>Product Teams</th>
<th>Supporting Teams</th>
<th>Maintenance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business</td>
<td>Vendor 1</td>
<td>Website</td>
<td>Sustainment</td>
<td></td>
</tr>
<tr>
<td>Vendor 2</td>
<td>Apps</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor 3</td>
<td>Product 3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Next Steps:

- Overview and Context
- Why Small Teams?
- Dependency Issues
- Trends Increasing Dependencies
- Solutions:
### Sample: An Iterative and Incremental Approach to Transitioning to Agile Product Development

<table>
<thead>
<tr>
<th>Step 1 - Foundation</th>
<th>Step 2 - Adoption</th>
<th>Step 3 - Development</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>People</strong>&lt;br&gt;Organize teams around products&lt;br&gt;Co-locate Calgary staff</td>
<td><strong>Process</strong>&lt;br&gt;Define products&lt;br&gt;Establish foundational metrics (roles, behaviors)&lt;br&gt;Robust DoD</td>
<td><strong>Tools</strong>&lt;br&gt;Set up Jira and Confluence&lt;br&gt;Create product and team boards&lt;br&gt;Agree reporting structures and processes</td>
</tr>
<tr>
<td><strong>Support</strong>&lt;br&gt;Vendor staff onsite as much as possible</td>
<td><strong>Support</strong>&lt;br&gt;Define Products&lt;br&gt;Set up Jira and Confluence&lt;br&gt;Create product and team boards&lt;br&gt;Agree reporting structures and processes</td>
<td><strong>Support</strong>&lt;br&gt;Support teams towards self-organizing and self-managing</td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong>&lt;br&gt;Teams self organizing and self managing</td>
<td><strong>Acceptance Criteria</strong>&lt;br&gt;Establish foundational metrics (roles, behaviors)&lt;br&gt;Robust DoD</td>
<td><strong>Acceptance Criteria</strong>&lt;br&gt;Team self-organizing and self managing</td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong>&lt;br&gt;Metrics based on number of deliveries, experiments, &amp; engagement levels</td>
<td><strong>Acceptance Criteria</strong>&lt;br&gt;Teams running frequent experiments and adapting process&lt;br&gt;Product roadmaps in place</td>
<td><strong>Acceptance Criteria</strong>&lt;br&gt;Teams running frequent experiments and adapting process&lt;br&gt;Product roadmaps in place&lt;br&gt;Productive retrospectives occurring</td>
</tr>
<tr>
<td><strong>Acceptance Criteria</strong>&lt;br&gt;Teams running frequent experiments and adapting process&lt;br&gt;Product roadmaps in place</td>
<td><strong>Acceptance Criteria</strong>&lt;br&gt;Teams running frequent experiments and adapting process&lt;br&gt;Product roadmaps in place&lt;br&gt;Productive retrospectives occurring</td>
<td><strong>Acceptance Criteria</strong>&lt;br&gt;Teams running frequent experiments and adapting process&lt;br&gt;Product roadmaps in place&lt;br&gt;Productive retrospectives occurring</td>
</tr>
</tbody>
</table>

### Summary:

- 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

© 2018 Leading Answers Inc.