Agile Teams: Self-Organizing, Collocated, and Distributed

Jutta Eckstein
je@it-communication.com
http://it-communication.com

Agenda

- Building Teams
- Team Maturity Stages & Collaboration
- Multi-Projecting and Part-Time Members
- Large and/or Global Self-Organizing Teams
Building Teams

Self-Organizing Team

- Cross-functional
- Organizes itself and its work
- Integration instead of separation
  - Competency, responsibility and task inseparable
  - Thinking and acting
  - Leads to more flexibility toward customer requests
Self-Responsible Feature Teams

- **Comprehends all necessary roles**
  - Domain expert, tester, ...

- **Comprehends all required know-how**
  - UI, database, ...
  - Or gains the required know-how

- **Ensures to complete stories (or features) in an iteration**
  - According the definition of done

Knowledge Transfer

- **Agile principles and practices ensure knowledge transfer**
  - Joint estimation and iteration planning
  - Collective ownership of delivery and process
  - Daily Scrum
  - Pair programming

- **Higher competency inside a group**
  - Team members learn from one another
  - Absences are easier to compensate
  - Continuous optimization
Ensuring the Business Value

- **Customer / product owner**
  - Decides on highest business value
  - Steers the iteration
  - Provides feedback on delivery
  - Obtains feedback from the teams

- **Represents customer perspective**

---

Coach (aka Scrum Master)

- **Responsible for:**
  - Coaching team in self-organization and creation of valuable products
  - Cooperation with roles and functions interfacing the team
  - Removing barriers

- **Ensures:**
  - Process is understood
  - Organization of team meetings
  - The organization supports and understands the process
Team Maturity Stages & Collaboration

Team Maturity Stages

- Tuckman's Team Development Model
  - Forming → Storming
  - Norming → Performing
  - Adjourning or Transforming

© Jana Friedrich, use with permission
Agile Chartering

- **Initial understanding and agreement on:**
  - What & How
- **Alignment of whole team**
  - Business people and developers
- **Note from Diana Larsen & Ainsley Nies:**
  - Chartering is more important than charter!

Retrospectives

- **Continuous learning**
  - Recognize and extract best practices
  - Reflection on and optimization of the process
  - Prepare for next iteration/release/project
- **As a team get more effective and efficient**
  - Reinforce your joint history
  - Make your shared values explicit
  - Define your common rules
Personal Development

- Regard failures as learning possibilities
  - Not as malfunction
- Learning is required by everyone at any time
- Sustainable Improvement
  - Continuous reflection and improvement
  - Everyone has to act self-responsibly

Multi-Projecting and Part-Time Members
Multi Projecting

- Multi projecting delays all projects
  - Later ROI for all projects
- Fully staff each project
  - Work on one project after the other
Part-Time Members

- Ensure every individual works for exactly one project
- Ensure team coherence
  - Team has joint goal
- Multi-tasking reduces productivity
  - It is a sign for lack of decision
- Everyone can keep the focus during the iteration

Large and/or Global Self-Organizing Teams
Building Teams

- **Avoid the typical structure**
  - According activities and know-how
    - Analysis in Germany, UI in India, middleware in Ireland...

- **Instead structure along features**
  - For ensuring the business value and the customer's advantage
  - Features/Stories shouldn't be split across teams

Collocated vs. Dispersed Feature Teams

- **Distributed but collocated subteams**
  - But: required know-how is often not collocated
  - Cross-team communication is harder

- **Dispersed subteams**
  - Cross-team communication is enabled by collocation
    - Eases conceptual integrity
  - Inner team communication is enforced by common goal
Supporting Whole Teams

- Every feature team needs the product owner’s support
  - Educate product owner(s) by shadowing

- Team of product owners with one lead product owner
  - Major contact to the real customer
  - Ensures big picture
  - Final decision on priorities

Feature Focus and Conceptual Integrity

- Feature focus might prevent conceptual integrity

- Yet:
  - “Simplicity comes from conceptual integrity” (Parnas)

- And:
  - Required support depends on system’s complexity
Supporting Stable Architecture

- **Chief architect is premise for conceptual integrity**
  - Chief architect pulls the strings
  - Communicates the vision
  - Servant with courage and experience

- **Community of Practice**
  - Role of architect in each feature team
  - Architects meets for synchronization and decision making
Supporting Unstable Architecture

- **Technical service team provide architecture as a service**
  - A good architecture evolves

- **Feature teams provide a product owner**
  - Formulates and prioritizes the requirements
  - Steers the iterations of the technical service team

Supporting Adaptive Architecture

- **Technical Consulting Team**
  - 1-x architects support several feature teams
  - Architects will work with feature teams on demand
Different Models for Architectural Support

- **Technical Service Team**: Unstable, Complex
- **Technical Consulting Team**: Adapting
- **CoP / Chief Architect**: Stable, Uncomplex

<table>
<thead>
<tr>
<th>Uncertainty</th>
<th>Changes</th>
<th>CoP / Chief Architect</th>
<th>Technical Consulting Team</th>
<th>Technical Service Team</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>Few</td>
<td>Stable, Uncomplex</td>
<td>Adapting</td>
<td>Unstable, Complex</td>
</tr>
<tr>
<td>High</td>
<td>Many</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Complexity Decreases over Time

- **Technical Service Team**: Unstable, Complex
- **Technical Consulting Team**: Adapting
- **CoP / Chief Architect**: Stable, Uncomplex

<table>
<thead>
<tr>
<th>Uncertainty</th>
<th>Changes</th>
<th>CoP / Chief Architect</th>
<th>Technical Consulting Team</th>
<th>Technical Service Team</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>Few</td>
<td>Stable, Uncomplex</td>
<td>Adapting</td>
<td>Unstable, Complex</td>
</tr>
<tr>
<td>High</td>
<td>Many</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Summary

- Cross-functional teams
- Congruency of competency, responsibility & task
- Feature team(s) & product owner(s) ensure value delivery
- Distributed or dispersed feature teams
- Technical teams & roles are service providers

Many Thanks!

Jutta Eckstein
je@it-communication.com
www.it-communication.com

JEstkstein.com/retrospectives
www.distributed-teams.com
www.agile-practice.com
leanpub.com/divingforhiddentreasures