Agile Software Development in the Large

QCon San Francisco 2007
Jutta Eckstein

What Does "Large" Mean?

- **Large in ...**
  - scope
  - time
  - people
  - money
  - risks
- **We focus on „Large Teams"**
  - which implies everything else
- **Large is relative**
  - 1, 2, 10, 100, 2000 people

„Large“ and Agile Methods

- **XP**
  - typical team size < 12
- **Scrum**
  - Scrum of Scrums
- **Crystal**
  - different colors for different team sizes:
    - clear: for teams < 10
    - orange: for teams < 40
    - red, blue, ... (not defined yet)
- **FDD**
  - teams are assembled for designing a feature

The Agile Value System

Agile development is defined by the value system:

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.

Source: http://agilemanifesto.org

Agile Value System is based on Principles

- **Early and continuous delivery of valuable software**
- **Welcome changing requirements**
- **Deliver working software frequently**
- **Business people and developers work together**
- **Trust motivated individuals**
- **Face-to-face conversation**
- **Working software is the primary measure of progress**
- **Promote sustainable development**
- **Technical excellence and good design**
- **Simplicity is essential**
- **Self-organizing teams**
- **Team reflection and adjustment**

Deliver Working Software Frequently

- **No need to prolong development cycles**
  - To steer in the right direction you need frequent feedback
  - Short cycles to reduce all risks
- **Two-week iterations have been proven**
  - Balance risk reduction with feature accomplishment
  - Ensure delivery at the end of the iteration
- **Same heartbeat across all sites**
  - Holidays can require some adaptation

Delivery of Valuable Software

- **Plan for accomplishing a valuable feature**
  - Often unusual for large teams
    - They are used to develop and plan against components or activities.
  - Accomplishment means integration, test and documentation
    - Delivery
- **Structure the teams along features**
  - For ensuring the business value and the customer‘s advantage
  - Feature teams
    - Comprehend all necessary roles and all required know-how

Customer Involvement

Defined single customer is rare, more typical are:

- **Large invisible customer base**
  - Typical for standard software
- **Community of customers**
  - Often not homogenous, but competitive
  - No accepted representative

Therefore:

- **„Customer on-site office“**
  - Specifies and performs acceptance tests
- **Customer surrogate**
  - Product Owner

Welcome Change

Plan driven versus planning driven

- **There is no such thing as a *final* plan at the beginning of the project**
- **Feedback has a direct impact on the plan**

„A plan is nothing;
planning is everything.“

Dwight D. Eisenhower

Promote Sustainable Development

- **Amount of work must match amount of time**
  - Realistic planning
  - Consider vacation and real time
- **Reviews and code inspection**
  - External
  - Internal (by developers and scripts)
    - Review team
    - Continuously: pair programming

Trust Motivated Individuals

Trust is based on:

- Communication
- Transparency
- Honesty
- Empowerment for decisions
- **Touch**

Trust regards all:

- Developers
- Customers
- Managers

Face-to-Face Conversation

- **Face-to-face communication is always preferred**
- **Frequent Synchronization is a must**
  - to have a common understanding
  - to deal with roles
  - to deal with changes
  - to deal with problems
  - to get feedback
- **Think about:**
  - People exchange
  - Communication facilitator
  - Different communication channels

Simplicity is Essential

- **Conceptual Integrity**
  - „*Simplicity comes from conceptual integrity*“
    [David Parnas, XP2002]
  - A system architect is the main step towards conceptual integrity
- **Architectural Lead pulls the strings**
  - Communicates the vision
  - Single point for
    - Key ideas
    - Technical responsibility assignments
    - Technical decisions
  - A servant with courage and experience

Continuous Attention to Technical Excellence

- **A good (and simple) design is never easy**
  - And: Make it right the first time never works
  - The best architecture evolves
- **Tests ensure the existing functionality when enhancing and changing the code**
  - Provide the safety net for refactoring
  - Synchronized with development
  - Automatically repeatable
- **Don‘t try to finalize the architecture *before* growing the team**
  - Feature teams will formulate the requirements

Working Software requires Integration

- **Changes will be integrated as often as possible**
  - Makes progress visible and measurable
  - Conflicts are easier to solve, the more often integration happens
- **Each integration results in a running system**
  - Provides immediate feedback
- **Integration needs support**
  - Plan at least 10% of development effort for integration/build

Regular Team Reflection and Adjustment

- **Iteration Review**
  - Revisit goals for the past iteration
  - Present deliveries – working software
  - Acceptance / Rejection
- **Retrospective**
  - Reflect on:
    - Things that worked well
    - Things that need to be improved
    - Things that are still difficult
  - Define an action plan on the top 3
  - Staged retrospective
    - In subteams
    - With team representatives

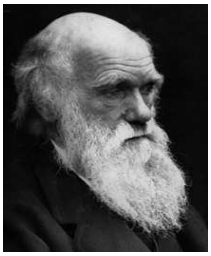
Self-Organizing Teams

- **People and teams are different**
- **Let teams decide**
  - Each team defines its own process
  - Retrospectives help to find the best process for this team
- **Don‘t over specify and overrule**
  - Use a starting line and adjust from there
    - E.g. the experience of the project members is a good start
- **Remember:**
  - “Individuals and interactions over processes and tools”

Lessons Learned

- **Agile Manifesto provides a guideline for scaling**
- **Feature teams and product owner(s) ensure the business value**
- **Risk reduction by short feedback cycles**

Charles Darwin:



„It is not the strongest
of the species that survive,
nor the most intelligent,
but the ones most responsive to change.“

Many Thanks!

Contact information:

Jutta Eckstein

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