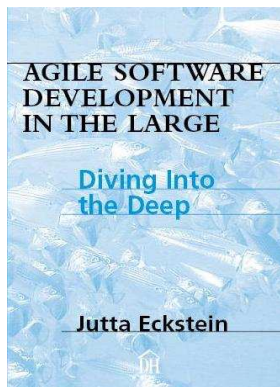


Agile Software Development in the Large

Jutta Eckstein
Agile Portugal 2010



Agility is Hot

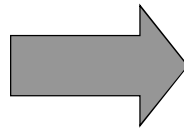
- Agile Project Management
- Agile Testing
- Agile Architecture
- Agile Offshoring
- Agile Investing
- Agile PLM
- Agile Web Development
- Agile Game Development
- Agile SOA
- Agile AJAX
- Agile Web Solutions
- Agile MDA / MDD

...

Agility is Old Stuff

- E.g. Scrum and XP have been published 1995

XP
SCRUM
DSDM
Crystal Methodologies
FDD
Adaptive Software Development
...



Agile Manifesto
Feb. 2001
agilemanifesto.org

„Large“

- Large in...
 - Scope
 - Time
 - People
 - Money
 - Risks
- We concentrate on „Large Teams“
- 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

josuttis | eckstein

5

©2009-2010 by IT-communication.com

No Method out of the Box

■ But: 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>

josuttis | eckstein

6

©2009-2010 by IT-communication.com

Agile Principles

■ Value system is based on the following 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

■ Balance risk reduction with feature accomplishment

- Two-week iterations have been proven
- About three iterations make a releases
- Same heartbeat for all teams

■ Time-boxed development:

- Knowledge about velocity
- Improvement of overall project plan

Delivery of Valuable Software

- **Focus on completing valuable features**
 - Often unusual for large teams
 - 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

Therefore: Customer surrogate

- Product Owner
- One product owner might not be enough
 - Team of product owners with one chief product owner
 - One product owner might steer 2-3 feature teams

Agility means Welcoming Change

- **Success is defined by the customer, not by anybody else.**
- **Change in features and in priorities**
 - Clear sign that the customer gets a better understanding of the system.
 - Most often this can be addressed with iterations
- **But:**
 - A change means:
 - Time delay
 - Elimination of another feature

Promote Sustainable Development

- **Determine each team's velocity**
 - Realistic planning
 - Note: velocities are not comparable
- **Reviews and code inspection**
 - External
 - Internal
 - Review team
 - Continuously: pair programming

Trust Motivated Individuals

Trust is based on:

- Communication
- Transparency
- Honesty
- Touch

Trust regards all:

- Developers
- Customers
- Managers

Agile Development inside the Organization

■ Agile development is a trouble detector

- Acknowledgement that bad news are also good news
- Support for appropriate actions

■ Cross-functional integration of departments

- Projects as customers

■ Close customer relationship

- Ensures feedback and thus the project success

Face-to-Face Conversation

- **Face-to-face communication is always preferred**
- **Daily Synchronization is a must**
 - to have a common understanding
 - to deal with roles
 - to deal with problems
 - to get feedback
- **Think about:**
 - Daily Scrum of Scrums
 - People exchange
 - Communication facilitator

Simplicity is Essential

- **Conceptual Integrity**
 - „*Simplicity comes from conceptual integrity*“ (Parnas)
 - A system architect is the main step towards conceptual integrity
- **Architectural Lead pulls the strings**
 - Communicates the vision
 - 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**
 - Provide the safety net for refactoring
 - Automated and synchronized with development
- **Refactoring happens continuously**
 - Shared responsibility
 - Every big refactoring can be accomplished by several small ones

Technical Service Team

- **Don't try to finalize the architecture *before* growing the team**
 - Allow the architecture to evolve
- **Feature teams provide a product owner**
 - Formulate and prioritizes the requirements
 - Steers the iterations of the technical service team
- **Architecture is a service**

Working Software requires Integration

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

Regular Team Reflection and Adjustment

- **Iteration Review in feature teams**
 - Present deliveries
 - Acceptance / Rejection
- **Retrospective**
 - Reflect on:
 - Things that worked well,
 - need to be improved,
 - are still difficult
 - Define actions for top 3
 - Staged retrospective

Self-Organizing Teams

- **People and teams differ**
- **Let teams decide**
 - Each team defines its own process
 - Retrospectives help to shape the process
- **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”

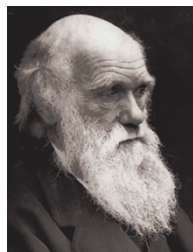
Servants for Large Teams

- **Scrum Master / Coach**
 - Process view
- **Product Owner**
 - Business view
- **Especially for large projects:**
 - **Project Manager**
 - Organizational view
 - **Architect**
 - Technical view

Summary

- Feature teams and product owner(s) ensure the business value
- Short iterations ensure frequent feedback for risk reduction
- Technical teams as service providers
- Communication is even more important

150 Years Ago and Still Up-To-Date



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

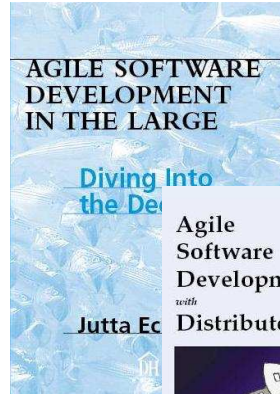
[Charles Darwin, The Origin of Species, 1859]

Many Thanks!

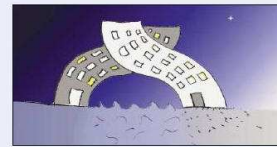
Contact information:

Jutta Eckstein

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



Agile
Software
Development
with
Distributed Teams



Staying Agile in a Global World

by Jutta Eckstein

