Short introduction to Agile Methods

Ciprian Radu
Software Engineer, Ropardo SRL

Based on:
Agile Software Process Management
by
Jutta Eckstein (http://jeckstein.com)
University of Bolzano, Italy,
CASE Summer School, 2009

Outline

- Introduction to Agile Process Management
  - Values and Principles
- Planning
  - Features, Priorities, Estimates, Change
- Practices
  - Integration, Refactoring, Test Driven Development
- Communication and Iteration Review
  - Synchronization, Roles & Responsibilities, Reporting
- Retrospectives
  - Continuous Process Improvement

Introduction to Agile Process Management

- Agile methods are not something new!
  - Lean development
  - Patterns
  - Smalltalk (programming language)
    - Extreme Programming (XP) and Scrum were published in the early 90ies
  - Agile Manifesto (http://agilemanifesto.org) created in 2001
    - The Agile System of Values
    - The Agile Principles

The Agile System of Values

- The Agile Development Process is defined by the next system of values:
  - Individuals and interactions *over* processes and tools
  - Working software *over* comprehensive documentation
  - Customer collaboration *over* contract negotiation
  - Responding to change *over* following a plan
- Things in red are valued but, we value things in green more

The Agile Principles

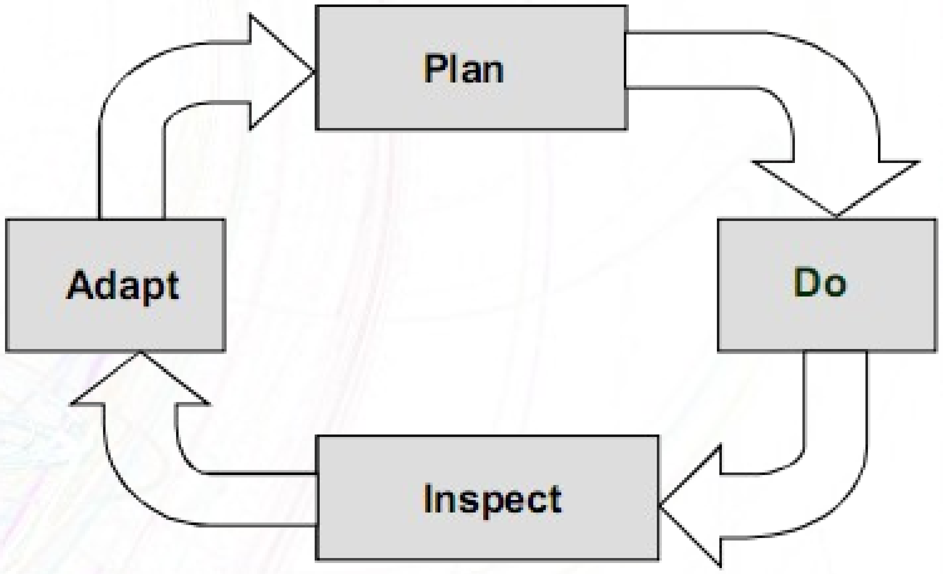
1. Early and continuous delivery of valuable software
2. Welcome changing requirements
3. Deliver working software frequently
4. Business people and developers work together
5. Trust motivated individuals
6. Face-to-face conversation
7. Working software is the primary measure of progress
8. Promote sustainable development
9. Technical excellence and good design
10. Simplicity is essential
11. Self-organizing teams
12. Team reflection and adjustment

3. Deliver working software frequently

- Use development cycles:
  - Normally iterations take 2 weeks
  - Typically releases are made each 3 months
    - internal and / or product releases
- Use **time-boxed development**:
  - Teams concentrate on results
  - Teams learn to estimate their speed
  - Allows early computation of remaining tasks

Agility in a nutshell

- Agile process management is a 4 step process:
  - Plan
  - Do (Develop)
  - Inspect (Test)
  - Adapt (Refactor)



- The key: feedback and focus on delivering business value
- Prioritizing, planning, and learning are ongoing activities

Planning – gathering features

- A feature (use case, use story, requirement) is a brief statement, from the user's perspective, about the functionality of the product
- Each feature has to be defined so that:
  - It provides business value for somebody (who can accept it)
  - It is measurable (by what criteria)
- Product backlog
- Product owner

Planning - prioritization

- The developers must make the product owner aware of technical issues but, the product owner will always have the final say
- The product owner can take estimation into account but, won‘t estimate
- 20% user stories will provide 80% of the value
- Keep the vision in focus
- Assess both cases of implementing or not a feature

Planning – estimation of features

- Do NOT estimate a feature by time
- Estimate by complexity, using relative estimates
- Estimation unit is really unimportant (can be: ideal time, feature points)
- Collaboratively estimate features
- Clarify the features for coming to an agreement
- Split features if necessary
- Repeat estimating until agreement is reached

Planning iterations and releases

- An iteration must last enough so that something measurable is accomplished, priorities don't change (usually ~2 weeks)
- Iteration planning is a responsibility for both developers and product owner
- Reach the production release via several internal releases
- Business determines
  - The time and the content of a release
  - Acceptable level of quality and cost of a release
- Development determines
  - How long it will take to build it
  - Creates preliminary estimates
  - Refines estimates as priority increases

Planning – changing requirements

- Agile Methods means welcoming changes
- From a team point of view
  - Each iteration is steered by high priority features
  - No difference if features are old, new or changed
- Sustainable development
  - Amount of work must match amount of time
- No changes during a iteration!

Practices – Integration

- Changes will be integrated as often as possible
  - Makes progress visible and measurable
- Conflicts are easier to solve, the more often integration happens
  - Verifies frequently if the system still runs
- Each integration results in a running system
  - Provides immediate feedback

Practices – Continuous integration

- Synchronous integration
  - Single integration at a time
- Asynchronous integration
  - Integration is time-controlled
- Integration happens after accomplishing a task
- All unit tests will pass 100% after integration
- As much automation as possible

Practices - Refactoring

- Refactoring = understandability + maintainability
- Refactoring happens continuously
  - uncompromising
  - disciplined in little steps
- Every big refactoring can be accomplished by several small ones
- Refactoring means changes but, the same functionality must be kept

Practices - Refactoring

- Shared responsibility
- No fear of „foreign“ code
- Refactoring helps understanding unknown code
- Unit Tests provide the safety net
- Adherence to the programming guidelines
- Code smells indicate the need for refactoring

Practices - TDD

- Programmers write *unit tests*
    - White box-tests isolate the units
    - Builds trust in the code
    - Provides immediate feedback
    - Are run permanently
- Customer/PO with Tester specify *acceptance tests*
    - Functional black box-tests for features
    - For ensuring the functionality and integration

Communication and Iteration Review

- Daily Scrum
  - Daily 10-15 minutes meeting
  - Same place and time every day
  - Status exchange for the team
    - What have I done since the last daily scrum?
    - What will I do until the next daily scrum?
    - What‘s in my way?
  - No management information meeting
- Scrum of Scrums
  - Across team boundaries

Pair programming

- Two heads, one keyboard
  - Knowledge transfer
    - Technical, domain, handling of tools
  - Higher quality
    - Continuous review
- the 2 individuals involved in pair programming have to be at a similar technical level
- a pair should exist half a day
- the 2 members of the pair switch roles from 10 to 10 minutes

Iteration review

- After each iteration, the progress is reviewed with:
  - Team (including Scrum master and Product owner)
  - Customer
  - Everybody interested in the project
- Product owner revisits goals of the past iteration
- Coach / Scrum master reviews
  - Features planned
  - Features completed
- Team presents accomplishments

Presentation

- Maintaining trust with customer by not hiding undone work
- Functionality is
  - Done-done
- Make achievements visible
  - Everybody sees the big picture
  - Pride in work
- Automated acceptance testing tools should support this

Evaluating consequences

- Unfinished features have to be re-added to the product backlog and prioritized
- Remove features from backlog that have been accomplished as well unplanned features
- Re-prioritize product backlog on new findings
- Troubleshooting
  - Release Iteration
  - Stop the project
  - Enlarge / change the team

Retrospectives – Continuous process improvement

- A retrospective is an opportunity for the participants to learn how to improve. The focus is on learning - not fault-finding.
- How does it work?
  - Continuous learning
    - Learn from failure, recognize and extract best practices
    - Prepare for next iteration/project
  - Course of action
    - What to prepare
    - What happens before, during and after

RO PARDO

Ground rules

- Regardless of what we discover, we must understand and truly believe that everyone did the best job he or she could, given:
  - what was known at the time,
  - his or her skills and abilities,
  - the resources available,
  - and the situation at hand.

Summary

- Agile Manifesto provides a value system and clear guidance through principles
- Planning of iterations and releases
- Practices
  - Continuous integration, TDD and refactoring
- Communication
  - Knowledge transfer via daily scrum and Pair Prog.
- Iteration Review
  - Feedback from the customer and continuous process improvements via retrospectives