Agile Development Methods: Philosophy and Practice

CPSC 315 – Programming Studio

Fall 2011

History of Agile Methods

- Particularly in 1990s, some developers reacted against traditional “heavyweight” software development processes.
- New methods were being developed and tested,
  - e.g. extreme programming, SCRUM, Feature-driven development
  - Generally termed “light” processes
- “Representatives” from several of these methods got together in Utah in 2001
  - Settled on term “Agile” as a way to describe these methods
  - Called themselves the “Agile Alliance”
  - Developed a “manifesto” and a statement of “principles”
  - Focuses on common themes in these alternative methodologies

Manifesto for Agile Software Development

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

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.

Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler
James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick
Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas

12 Principles behind the Agile Manifesto

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

12 Principles behind the Agile Manifesto

4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

12 Principles behind the Agile Manifesto

7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.

12 Principles behind the Agile Manifesto

10. Simplicity--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs
    emerge from self-organizing teams.
12. At regular intervals, the team reflects on how

    to become more effective, then tunes and adjusts
    its behavior accordingly.

Individuals and Interactions over Processes and Tools

- People make biggest impact on success
  - Process and environment help, but will not create success
- Strong individuals not enough without good team interaction.
  - Individuals may be stronger based on their ability to work on a team
- Tools can help, but bigger and better tools can hinder more than help
  - Simpler tools can be better

Working Software over Comprehensive Documentation

- Documentation important, but too much is worse than too little
  - Long time to produce, keep in sync with code
  - Keep documents short and salient
- Focus effort on producing code, not descriptions of it
  - Code should document itself (this does not eliminate the need for commenting!!!)
  - Knowledge of code kept within the team
- *Produce no document unless its need is immediate and significant.*

Customer Collaboration over Contract Negotiation

- Not reasonable to specify what's needed and then have no more contact until final product delivered
- Get regular customer feedback
- Use contracts to specify customer interaction rather than requirements, schedule, and cost

Responding to Change over Following a Plan

- Environment, requirements, and estimates of work required will change over course of large project.
- Planning out a whole project doesn't hold up
  - Changes in shape, not just in time
- Keep planning realistic
  - Know tasks for next couple of weeks
  - Rough idea of requirements to work on next few months
  - Vague sense of what needs to be done over year

Extreme Programming (XP)

- One of the most well-known agile methods
- Developed in 1990s
  - Kent Beck, 1996
  - Chrysler Comprehensive Compensation Project
  - Published book in 1999

Extreme Programming Practices

1. On-Site Customer
   - Customer is actively involved with development process
   - Customer gives “User Stories”
     - Short, informal “stories” describing features
     - Keep on “story cards”
2. Planning Game
   - Developers and customers together plan project
   - Developers give cost estimates to “stories” and a budget of how much they can accomplish
     - Can use abstract accounting mechanism
     - Later compare to actual cost, to improve estimates over time
   - Customer prioritizes stories to fit within budget

Extreme Programming Practices

3. Metaphor
   - Come up with metaphor that describes how the whole project will fit together
   - The picture in a jigsaw puzzle
   - Provides framework for discussing project in team
   - Tools and materials often provide good metaphors
4. Small Releases
   - Time between releases drastically reduced
     - A few weeks/months
   - Multiple iterations
   - Can have intermediate iterations between bigger “releases”

Extreme Programming Practices

5. Testing
   - Test-first programming
   - Unit testing frequently by developers
   - Acceptance tests defined by customers
6. Simple Design
   - Principles discussed in earlier lectures
   - Design should be quick, not elaborate
   - Pick the simplest thing that could possibly work
   - Resist adding stuff not ready yet

Extreme Programming Practices

7. Refactoring
   - Code gets worse with feature adds, bug fixes
   - Rewrite small sections of code *regularly*
   - Rerun all unit tests to know nothing broken
     - Means you should have designed comprehensive tests
8. Pair Programming
   - Discussed later
9. Collective Ownership
   - Anyone can edit anything
   - Errors are the fault of the whole team

Extreme Programming Practices

10. Continuous Integration
    - Commit changes frequently (several times a day)
    - Verify against entire test suite!
11. Coding Standards
    - Enables effective teamwork
12. Sustainable Pace
    - No overtime
    - Only exceptions in final week
    - Good estimation skills for budgeting will help ensure reasonable times
    - Time less likely to be wasted in pairs, bullpen rooms
    - Plan time each day for administrative work (<1 hour), breaks

SCRUM

- Idea first appeared in a business journal in 1986 (applied to product development management).
- Used in software development and presented in 1995 paper.
- Term is based on rugby term
- Small cross-functional teams

SCRUM Practices

- Product and release *backlog*
  - A list of the features to be implemented in the project (subdivided to next release), ordered by priority
  - Can adjust over time as needed, based on feedback
  - A product manager is responsible for maintaining

SCRUM Practices

- *Burn-down* chart
  - Make best estimate of time to complete what is currently in the backlog
  - Plot the time on a chart
  - By studying chart, understand how team functions
  - Ensure burndown to 0 at completion date
    - By adjusting what's in the backlog
    - By adjusting the completion date

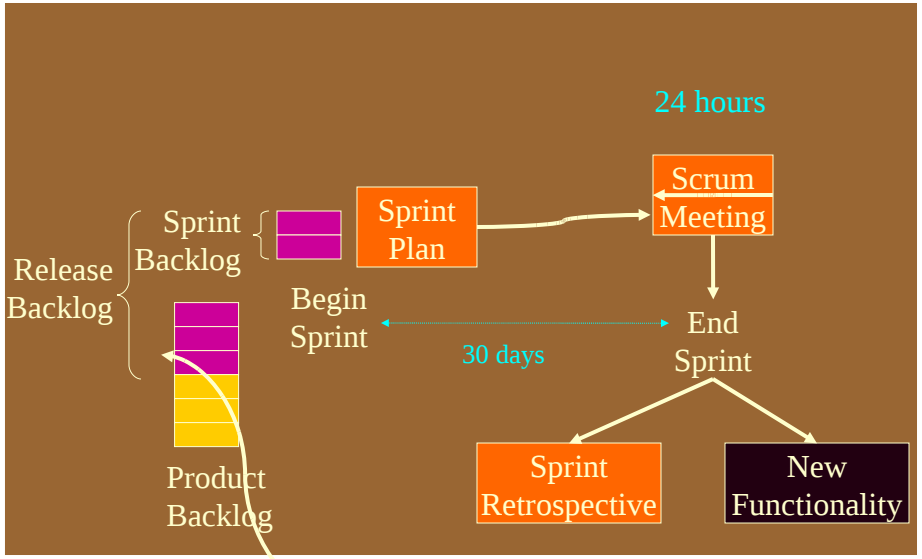
SCRUM Practices

- The *sprint*
  - The sprint is a ~1 month period after which some product is delivered
  - Features are assigned from the product backlog to a sprint backlog
    - Features divided into smaller tasks for sprint backlog
    - Feature list is fixed for sprint
  - Planning meeting
    - Tasks can be assigned to team members
    - Team members have individual estimates of time taken per item
  - During sprint, work through features, and keep a burn-down chart for the sprint
  - New functionality is produced by the end of the sprint
  - After sprint, a review meeting is held to evaluate the sprint

SCRUM Practices

- Scrum meeting
  - 15 minute *daily* meeting
  - All team members show up
  - Quickly mention what they did since last Scrum, any obstacles encountered, and what they will do next
  - Some team member volunteers or is appointed to be the *Scrum Master* - in charge of Scrum meeting, and responsible for seeing that issues raised get addressed
  - Customers, management encouraged to observe

SCRUM Practices



Other Agile Methods

- Crystal
- Feature-driven development (FDD)
- Adaptive software development (ASD)
- Dynamic System Development Method (DSDM)

Drawbacks and Challenges of Agile Methods

- Undefined goals
  - Feature creep
  - Neverending project with overruns
- Need clear, single, invested customer
- All-or-nothing adoption of techniques
  - Some parts work only if lots of other aspects used
- Team size limited (smaller teams)

Resources Used

- Agile methods: www.AgileAlliance.org
- Agile methods (especially XP): *Agile Software Development*, by Robert C. Martin, Prentice-Hall 2003.
- SCRUM: www.controlchaos.org
- XP: *eXtreme Programming in Action Practical Experiences from Real World Projects*, by Martin Lippert, Stefan Roock, Henning Wolf, John Wiley and Sons, 2002.