Agile Processes

Software processes that are:

- Incremental (small software releases with rapid cycles)
- Cooperative (customer and developer working together with close communication)
- Straightforward (method is easy to learn and modify)
- Adaptive (able to make last moment changes)

Objectives

- To introduce the notion of an agile process
- To describe a number of different agile processes
- To pinpoint the advantages and disadvantages of agile processes

Manifesto

- Values individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over negotiation
- Responding to change over following a plan

Agility

- Readiness for motion
- Nimbleness
- Activity
- Dexterity in motion

Principles of agility

- Customer satisfaction at early stage, continuous delivery of software
- Accommodate changing requirements
- Deliver working software frequently
- Business people and developers work together daily
- Build projects around motivated individuals

Principles of agility

- Face-to-face conversation as most effective communication
- Working software – primary measure of progress
- Sustainable development
- Continuous attention to technical excellence and good design practices
- Self-organizing teams
- Proactive process improvement

Human factors

- Competence
- Common focus
- Collaboration
- Decision-making ability
- Fuzzy problem-solving activity
- Mutual trust and respect
- Self-organization

Comparison topics

- Process
- Roles and responsibilities
- Practices
- Adoption and experiences
- Scope of use

Agile methods

- Extreme programming or XP (Beck 1999)
- Scrum (Schwaber and Beedle 1995, 2002)
- Crystal family (Cockburn 2002)
- Feature Driven Development (Palmer and Felsing 2002)
- Rational Unified Process (Kruchten 1996, 2000)
- Dynamic Systems Development Method (Stapleton 1997)
- Adaptive Software Development (Highsmith 2000)
- Open Source Software Development (O'Reilly 1999)

Extreme Programming (XP)

- The most widely used agile process, originally proposed by Kent Beck
- XP Planning
  - Begins with the creation of "user stories"
  - Agile team assesses each story and assigns a cost
  - Stories are grouped to for a deliverable increment
  - A commitment is made on delivery date
  - After the first increment "project velocity" is used to help define subsequent delivery dates for other increments

Extreme Programming (XP)

- XP Design
  - Follows the "keep as simple as possible" principle
  - Encourage the use of CRC cards
  - For difficult design problems, suggests the creation of "spike solutions"—a design prototype
  - Encourages "refactoring"—an iterative refinement of the internal program design
- XP Coding
  - Recommends the construction of a unit test *before* coding commences
  - Encourages "pair programming"
- XP Testing
  - All unit tests are executed daily
  - "Acceptance tests" are defined by the customer and executed to assess customer visible functionality

Adaptive Software Development

- Originally proposed by Jim Highsmith
- ASD — distinguishing features
  - Mission-driven planning
  - Component-based focus
  - Uses "time-boxing" (See Chapter 24)
  - Explicit consideration of risks
  - Emphasizes collaboration for requirements gathering
  - Emphasizes "learning" throughout the process

Dynamic Systems Development Method

- Promoted by the DSDM Consortium
- DSDM—distinguishing features
    - Similar in most respects to XP and/or ASD
    - Nine guiding principles
        - Active user involvement is imperative.
        - DSDM teams must be empowered to make decisions.
        - The focus is on frequent delivery of products.
        - Fitness for business purpose is the essential criterion for acceptance of deliverables.
        - Iterative and incremental development is necessary to converge on an accurate business solution.
        - All changes during development are reversible.
        - Requirements are baselined at a high level
        - Testing is integrated throughout the life-cycle.

Scrum

- Originally proposed by Schwaber and Beedle
- Scrum—distinguishing features
  - Development work is partitioned into “packets”
  - Testing and documentation are on-going as the product is constructed
  - Work occurs in “sprints” and is derived from a “backlog” of existing requirements
  - Meetings are very short and sometimes conducted without chairs
  - “demos” are delivered to the customer with the time-box allocated

Crystal

- Proposed by Cockburn and Highsmith
- Crystal—distinguishing features
  - Actually a family of process models that allow "maneuverability" based on problem characteristics
  - Face-to-face communication is emphasized
  - Suggests the use of "reflection workshops" to review the work habits of the team

Feature Driven Development

- Originally proposed by Peter Coad et al
- FDD—distinguishing features
  - Emphasis is on defining "features"
    - a *feature* "is a client-valued function that can be implemented in two weeks or less."
  - Uses a feature template
    - <action> the <result> <by | for | of | to> a(n) <object>
  - A features list is created and "plan by feature" is conducted
  - Design and construction merge in FDD

Agile Modeling

- Originally proposed by Scott Ambler
- Suggests a set of agile modeling principles
  - Model with a purpose
  - Use multiple models
  - Travel light
  - Content is more important than representation
  - Know the models and the tools you use to create them
  - Adapt locally

Key points

- Agile processes geared towards organizational and human aspects of software processes
- No details about ensuring artifact consistency and correctness
- No elaboration or guidelines for the methods of transforming or producing output artifacts based on the input artifacts
- Is it appropriate for mission-critical software development ?