Agile Development

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 et al

Background: critics on traditional process

- *Prescriptive process models forget the frailties* of the people who build computer software
- Process models can deal with people's common weaknesses with *discipline* or *tolerance* and that *most prescriptive process models choose discipline*
- Because consistency in action is a human weakness, high disciple methodologies are fragile
  - Argued by Alistair Cockburn '02

Background: motivation for agile development

- Agile methods were developed in an effort to overcome perceived and actual weakness in conventional software engineering
- You must be *agile* enough to response to a fluid business environment where many realistic challenges exist
  - Difficult/impossible to predict how software evolves over time
  - Market conditions change rapidly, end-user needs evolve, new competitive threats arise without warning …
  - In many situations, you won't be able to define requirements fully before the project begins
- Fluidity implies change, and change is expensive
- Agile development can *reduce the costs of change* through the software process

Background: caveats for agile development

- Agile development can provide important benefits, but it is not applicable to all projects, all products, all people, and all situations
- It is also not antithetical to solid software engineering practice and can be applied as an overriding philosophy for all software work
- Agile development does not mean no documents are created
  - It means only creating documents that will be referred to later in the development process

What is “Agility”?

- Effective (rapid and adaptive) response to change
- Effective communication among all stake holders
- Drawing the customer onto the team
- Organizing a team so that it is in control of the work performed.

Yielding …

- Rapid, incremental delivery of software

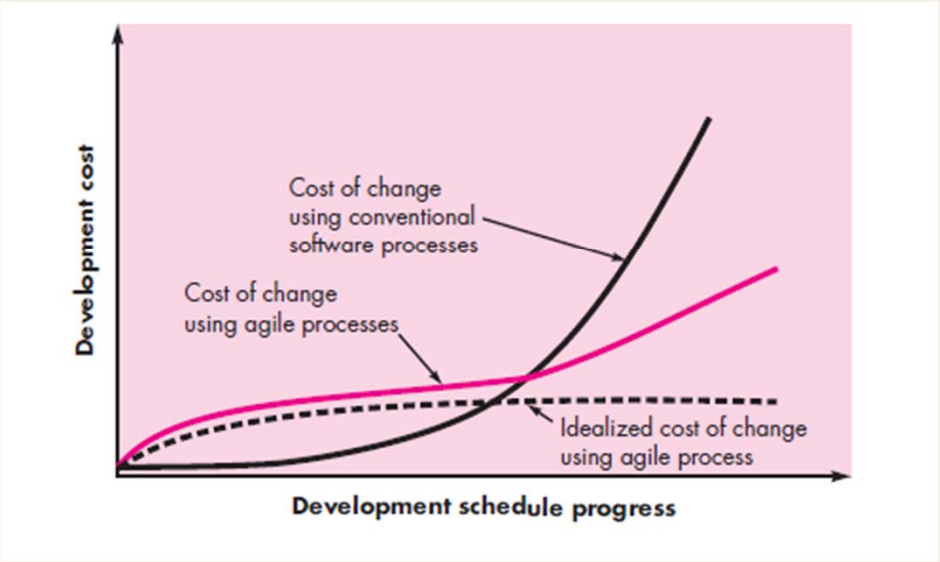
The Cost of Change

- Conventional wisdom
  - The cost of change increases nonlinearly as a project progresses.
    - Early in a project: If there are any changes, the costs of doing this work are minimal.
    - In the middle of validation testing: Costs escalate quickly
- Agile
  - A well-designed agile process may "flatten" the cost of change curve
    - software is released in increments
    - By coupling incremental delivery with other agile practices such as continuous unit testing and pair programming, the change can be better controlled and the cost of making a change is attenuated

It is relatively easy to accommodate a change when a team is gathering requirements early in a project;

Later, the change requires a modification to the architectural design, construction of new components, changes to other existing components, new testing and so on. Costs thus escalate quickly, and the time and cost required to ensure that the change is made without unintended side effects is nontrivial.

Cost of Change



Agile Process

- Key assumptions to address (mainly about unpredictability)
  - difficult to predict in advance which software requirements will persist and which will change
  - difficult to predict how much design is necessary before construction is used to prove the design
  - Analysis, design, construction, and testing are not as predictable (from a planning point of view) as we might like
- How to address
  - Unpredictability: Adaptability
  - Continual adaptation without forward progress accomplishes little: Adopt incrementally
  - How to accomplish incremental adaptation: an agile team requires customer feedback

Agile Process

- Features/characteristics in main
  - Is driven by customer descriptions of what is required (scenarios)
  - Recognizes that plans are short-lived
  - Develops software iteratively with a heavy emphasis on construction activities
  - Delivers multiple 'software increments'
  - Adapts as changes occur

12 Agility Principles

- 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.
- 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 Agility Principles (Cont)

- 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.
- 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.

Human Factors

- The process molds to the needs of the people and the team
- Key traits must exist among the agile team member
  - Competence.
  - Common focus.
  - Collaboration.
  - Decision-making ability.
  - Fuzzy problem-solving ability.
  - Mutual trust and respect.
  - Self-organization

Extreme Programming (XP)

- The most widely used agile process models
- Uses an object-oriented approach as its preferred development paradigm
- Encompasses a set of rules and practices that occur within the context of four framework activities: Planning, design, coding, and testing
- Core values
  - Communication
  - Simplicity
  - Feedback
  - Courage (Discipline)
  - Respect

Five values: communication, simplicity, feedback, courage, and respect.

Communication: close yet informal collaboration between customers and developers.

Simplicity: design only for immediate needs, but not future needs. Create a simple design that can be easily implemented.

Feedback: software itself (make use of unit test as primary testing tactic, developed along the way), customer, other team members.

Discipline: not over reaching.

Respect among members.

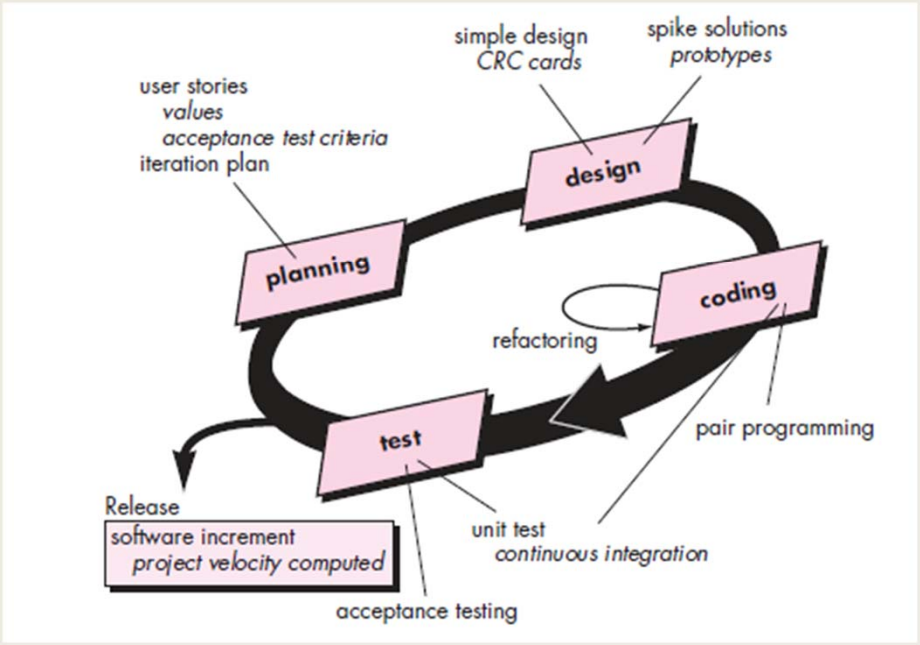
Extreme Programming (XP)

- Planning
  - Begins with the creation of "user stories"
  - customer can add stories, change the value of an existing story, split stories, or eliminate them
- Design
  - For difficult design problems, suggests the creation of "spike solutions"—a design prototype
  - Encourages "refactoring"—an iterative refinement of the internal program design

Extreme Programming (XP)

- Coding
  - Recommends the construction of a unit test for a store *before* coding commences
  - Encourages "pair programming"
- Testing
  - All unit tests are executed daily
  - "Acceptance tests" are defined by the customer and executed to assess customer visible functionality

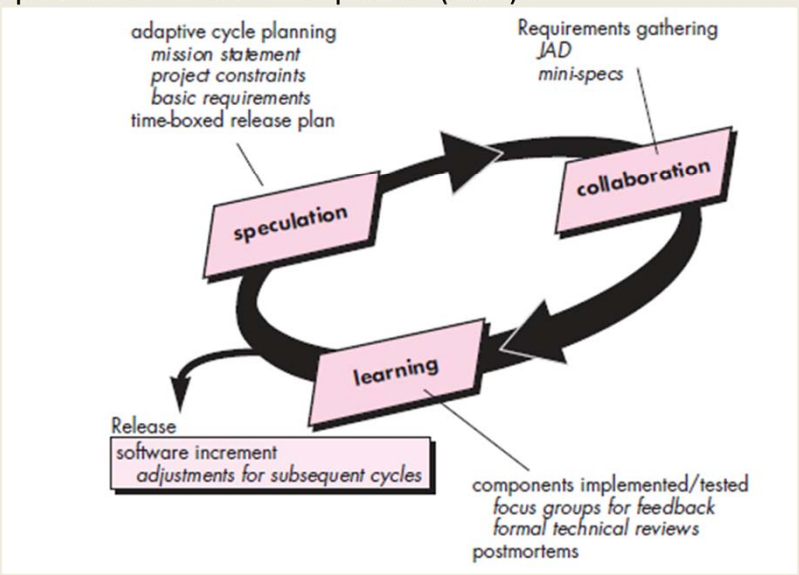
Extreme Programming (XP)



The XP process model diagram.

Other Agile Processes

- Adaptive Software Development (ASD)



Talk about the three phases: speculation, collaboration, learning

Other Agile Processes

- Scrum

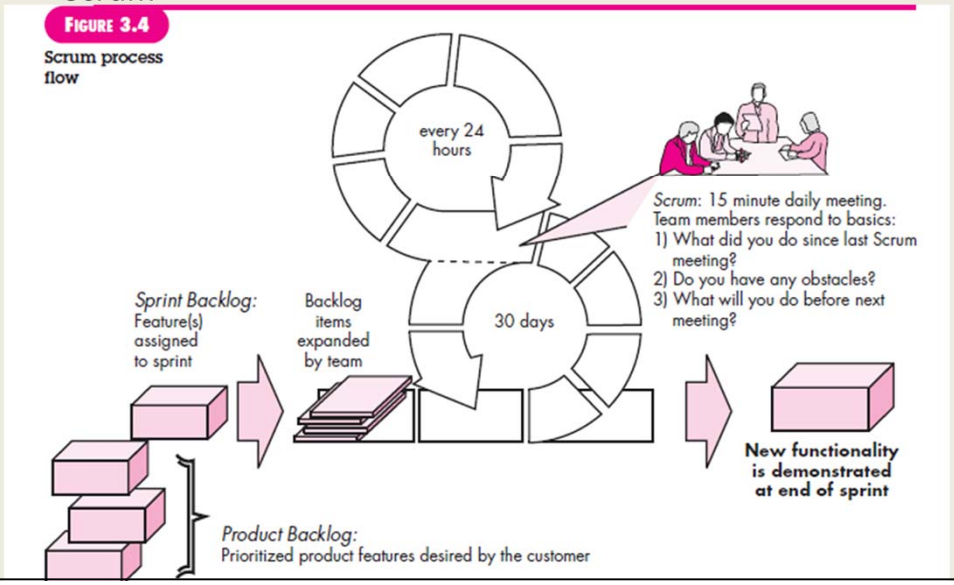


FIGURE 3.4
Scrum process flow

Go through the flow, covering four major activities in this process flow: Backlog, Sprints, Scrum meeting, Demos

Other Agile Processes

- More
  - Dynamic Systems Development Method (DSDM)
  - Crystal
  - Feature Driven Development (FDD)
  - Lean Software Development (LSD)
  - Agile Modeling (AM)
  - Agile Unified Process (AUP)

Summary

- Agile development paradigm
  - Background / Motivation
- Dealing with the Cost of Change
- Agile process
  - Assumptions
  - Features
  - Principles
  - Human factors
- The XP agile process
  - Core values
  - Four framework activities
- Other agile processes