CSE4006 Software Engineering

04. Agile Development

Scott Uk-Jin Lee

Department of Computer Science and Engineering
Hanyang University ERICA Campus

1st Semester 2015

Background of Agile Software Development

- Software development until late 90s
  - Conducted with the input of many people and sufficient fund over a long period of time
    = primarily targeted project of Software Engineering
- Recent Software Development
  - short development period, small cost investment, very complex and open
  - severe changes according to the social situation and the market fluctuations
  - requirements are diverse and changes moment to moment
- object-oriented as technical solution
  - needs suitable process for object-oriented development
    $\Rightarrow$ Agile development process

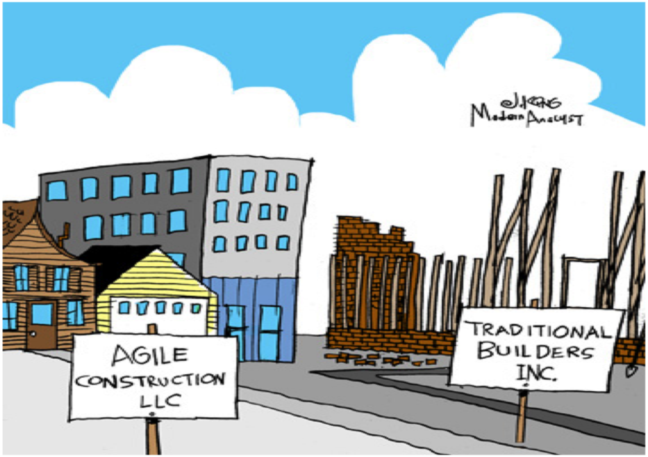
Agile vs. Traditional Methods

- Problems (Characteristics) of large-scale system development process
  - overhead: requirement of careful plan and quality assurance
  - end up spending more time on other works (documentation, meeting, design, etc) than program development
  - heavy methodology

- Agile method
  - focuses more on software itself than design and documentation
  - provides environment where frequent change of user's requirement can be reflected
  - fast feedback

e.g., extreme programming, SCRUM, Crystal, Adaptive software development, feature driven development

Software Development Life Cycle



- **Agile Methods**: Lack of coherent design
- **Traditional Methods**: Incomplete Project

The Manifesto for Agile Software Development

- Kent Beck et al (16 others) in 2001
- 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** vs. processes and tools
  - **Working software** vs. comprehensive documentation
  - **Customer collaboration** vs. contract negotiation
  - **Responding to change** vs. following a plan
- That is, while there is value in the items on the right, we value the items on the left more.

What is "Agility"?

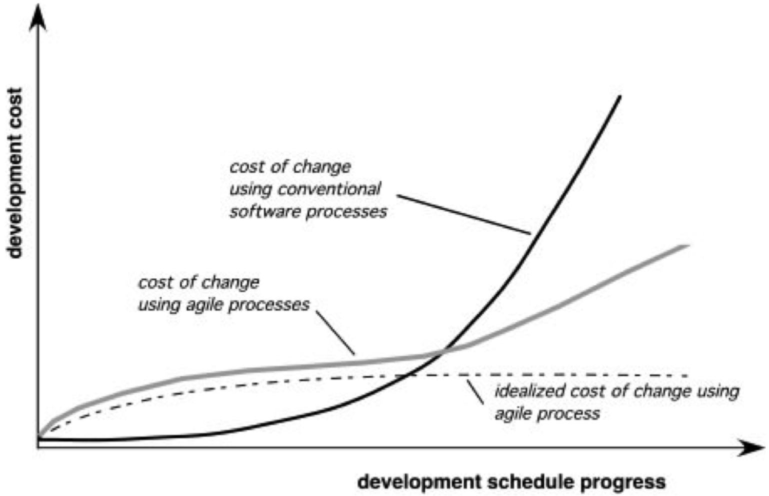
- Agile = able to move quickly and easily
- Effective (rapid and adaptive) response to change
- Effective communication among all stakeholders
- Drawing the customer onto the team
  - customer's role is very important in providing, giving priority to, evaluating requirements
- Organizing a team so that it is in control of the work performed

Yielding ...

- Rapid, incremental delivery of software

lab(se);

Agility and the Cost of Change



lab(se);

An Agile Process

- 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

lab(se);

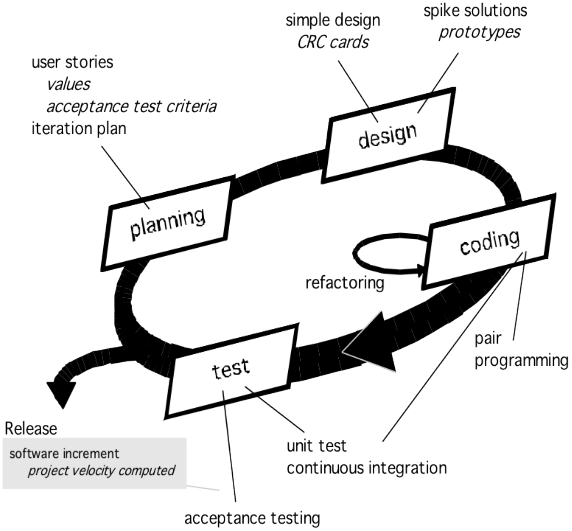
Agility Principles

1. To satisfy the customer through early and continuous delivery of valuable software
2. Welcome changing requirements, even late in development
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
5. Build projects around motivated individuals
6. The most efficient and effective communication is face–to–face conversation

Agility Principles

7. Working software is the primary measure of progress
8. Agile processes promote sustainable development
9. Continuous attention to technical excellence and good design
10. Simplicity 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

Extreme Programming (XP)



lab(se);

Extreme Programming (XP)

- The most widely used agile process, originally proposed by Kent Beck in 1999
- XP Planning
  - Begins with the creation of "user stories" that describe required features and functionality of software
  - Customer assigns a value to the story.
  - Agile team assesses each story and assigns a cost 시간 (in weeks)
  - 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

(project velocity: no. of user stories implemented during the first release)

Extreme Programming (XP)

- XP Design
  - Follows the KIS principle
  - Encourage the use of CRC (Class-Responsibility-Collaborator) 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 for a story before coding commences
  - Encourages "pair programming"

smoke 테스팅

- XP Testing
  - All unit tests are executed daily (whenever code is modified)
  - "Acceptance tests" are defined by the customer and executed to assess customer visible functionality

Extreme Programming (XP)

Pair Programming

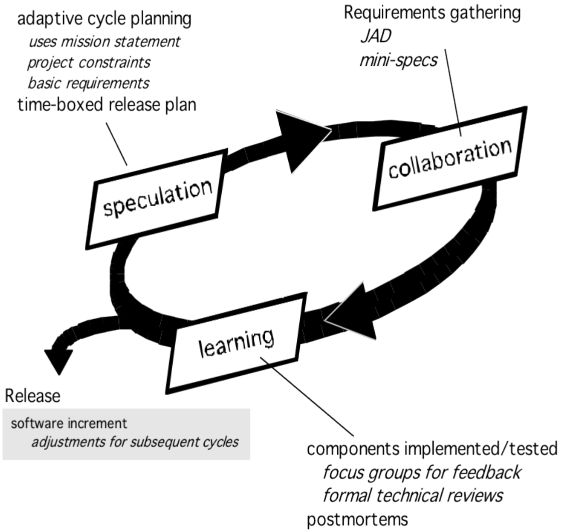
- two people pair up and develop using the same computer
- when one person code, the other can figure out how to test
- Pros
  - faithful to principles than when developing alone
  - can produce better design and code
  - work becomes more vibrant and prevents disruption
  - better problems-solving and robust design
  - mentoring, strong cohesive
- Cons
  - if there is a large gap in developers' ability, then it can be boring and be a burden
  - difficult to determine the exact productivity

Adaptive Software Development

- Originally proposed by Jim Highsmith
- Focus on human collaboration and team self-organization
- ASD — distinguishing features
  - Mission-driven planning
  - Component-based focus
  - Uses "time-boxing" ⇒ risk management
    - Time-box : a number of separate time periods (normally two to six weeks long)
    - Each part has its own deliverables, deadline and budget
  - Explicit consideration of risks
  - Emphasizes collaboration for requirements gathering
  - Emphasizes "learning" throughout the process

lab(se);

Adaptive Software Development



Dynamic Systems Development Method

- Promoted by the DSDM Consortium (www.dsdm.org)
- 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.

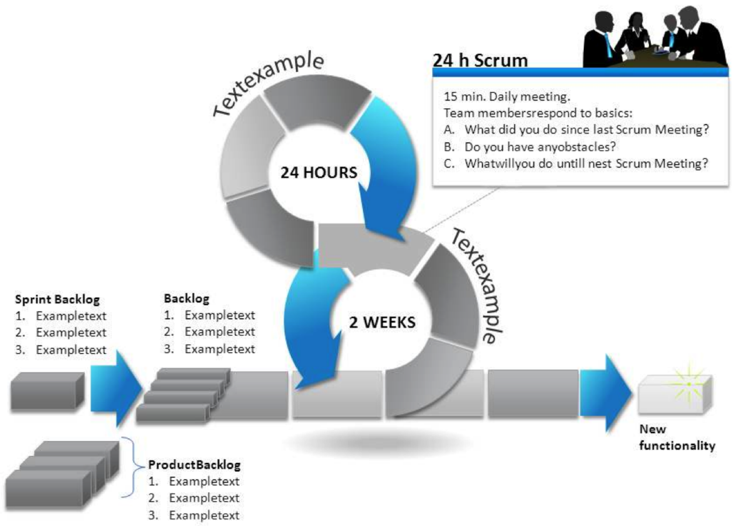
lab(se);

Scrum

- Originally proposed by Schwaber and Beedle
- Small working team ⇒ max. communication, min overhead, max. sharing of info.
- 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
    - Backlog: a prioritized list of project requirements or features
    - Sprint: work tasks within a process pattern
  - Meetings are very short and sometimes conducted without chairs
  - "Demos" are delivered to the customer with the time-box allocated

lab(se);

Scrum



15 min. Daily meeting.
Team membersrespond to basics:
A. What did you do since last Scrum Meeting?
B. Do you have anyobstacles?
C. Whatwillyou do untill nest Scrum Meeting?

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."
    - users can describe feature more easily
  - Uses a feature template
    - $< action > the < result >< by \mid for \mid of \mid to > a(n) < object >$

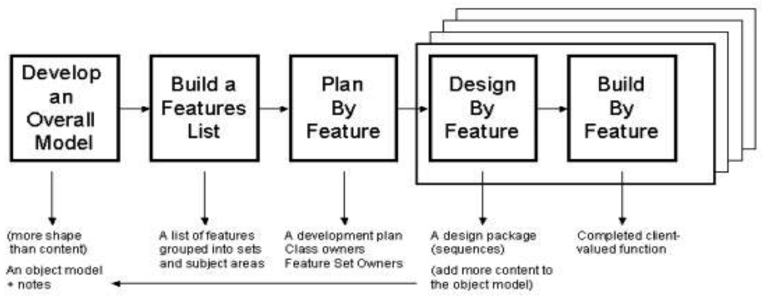
    e.g., add the product to a shopping cart

    e.g., store the shipping information for a customer
  - Feature set template
    - $< action >< -ing > a(n) < object >$

    e.g., making a product sale
  - A features list is created and "plan by feature" is conducted
  - Design and construction merge in FDD

lab(se);

Feature Driven Development



Develop an Overall Model
Build a Features List
Plan By Feature
Design By Feature
Build By Feature
(more shape than content)
An object model + notes
A list of features grouped into sets and subject areas
A development plan Class owners Feature Set Owners
A design package (sequences)
(add more content to the object model)
Completed client-valued function

- 6 milestones during the design and implementation
  - Design walkthrough, design, design inspection, code, code inspection, promote to build

Informal Reviews: Walkthroughs

"Walkthroughs"

- developer technique (usually informal)
- used by development teams to improve quality of product
- The purpose of walkthrough is to:
  - Find problems
  - Discuss alternative solutions
  - Focusing on demonstrating how work product meets all requirements
- Leader, recorder, author

Formal Reviews: Inspections

"(Fagan) Inspections"

- a process management tool (always formal)
- used to improve quality of the development process
- The objectives of the inspection process are to:
  - Find problems at the earliest possible point in the SW dev. process
  - Verify that the work product meets its requirement
  - Ensure that work product has been presented according to predefined standards
  - Provide data on product quality and process effectiveness
  - Inspection advantages are to build technical knowledge and skill among team members by reviewing the output of other people
  - Increase the effectiveness of software testing

Agile Modeling

- Originally proposed by Scott Ambler
- Suggests a set of agile modeling principles for building large, business critical systems
  - 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
  - Prototyping application