Chapter 4
Agile Development

Moonzoo Kim

CS Division of EECS Dept.
KAIST
moonzoo@cs.kaist.ac.kr
http://pswlab.kaist.ac.kr/courses/cs350-07

Ex. UP Work Products

Inception phase

Vision document
Initial use-case model
Initial project glossary
Initial business case
Initial risk assessment.
Project plan,
  phases and iterations.
Business model,
  if necessary.
One or more prototypes

Elaboration phase

Use-case model
Supplementary requirements
  including non-functional
Analysis model
Software architecture
  Description.
Executable architectural
  prototype.
Preliminary design model
Revised risk list
Project plan including
  iteration plan
  adapted workflows
  milestones
  technical work products
Preliminary user manual

Construction phase

Design model
Software components
Integrated software
  increment
Test plan and procedure
Test cases
Support documentation
  user manuals
  installation manuals
  description of current
    increment

Transition phase

Delivered software increment
Beta test reports
General user feedback

The 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 v.s. processes and tools
- Working software v.s. comprehensive documentation
- Customer collaboration v.s. contract negotiation
- Responding to change v.s. 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

What is "Agility"?

- Effective (rapid and adaptive) response to change
- Effective communication among all stakeholders
- 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

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

12 Principles of Agile Process

- To satisfy the customer through early and continuous delivery of valuable SW
- Welcome changing requirements, even late in development
- Deliver working SW frequently (from a couple of weeks to a couple of months)
- Business people and developers must work together daily
- Build projects around motivated individuals
- The most effetive method of communication is face-to-face conversation
- Working SW is the primary measure of progress
- Agile processes promote sustainable development
- Continuous attention to technical excellence and good design
- Keep it simple (KIS)
- The best architectures, requirements, and designs emerge from self-organinzing teams
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior

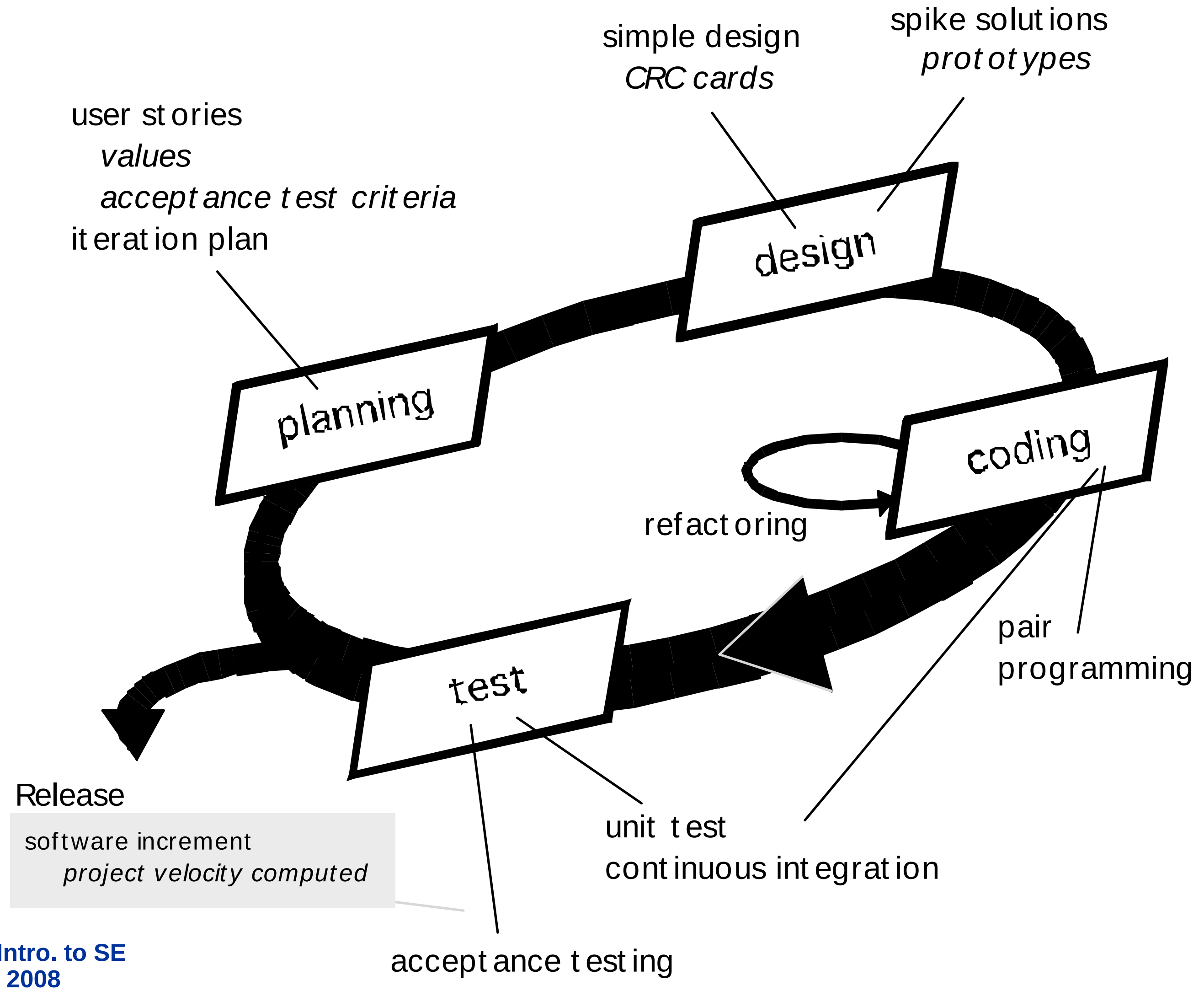
Extreme Programming (XP) (1/3)

- The most widely used agile process, originally proposed by Kent Beck
- XP Planning
  - Begins with the creation of “user stories” that describe required features and functionality for SW
  - Agile team assesses each story and assigns a cost (in weeks)
  - Stories are grouped to for a deliverable increment
  - A commitment is made on delivery date
  - After the first increment “project velocity” (# of user stories implemented during the first release) is used to help define subsequent delivery dates for other increments

Extreme Programming (XP) (2/3)

- XP Design
  - Follows the KIS principle
  - Encourage the use of CRC cards (see Chapter 8)
  - For difficult design problems, suggests the creation of a spike solution (a design prototype)
    - A spike solution is a very simple program to explore potential solutions.
  - Encourages “refactoring”—an iterative refinement of the internal program design
- XP Coding
  - Recommends the construction of a unit test for a store *before* coding commences
    - Test oriented implementation
  - 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

Extreme Programming (XP) (3/3)



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
  - Explicit consideration of risks
  - Emphasizes collaboration for requirements gathering
  - Emphasizes “learning” throughout the process

Adaptive Software Development

adaptive cycle planning

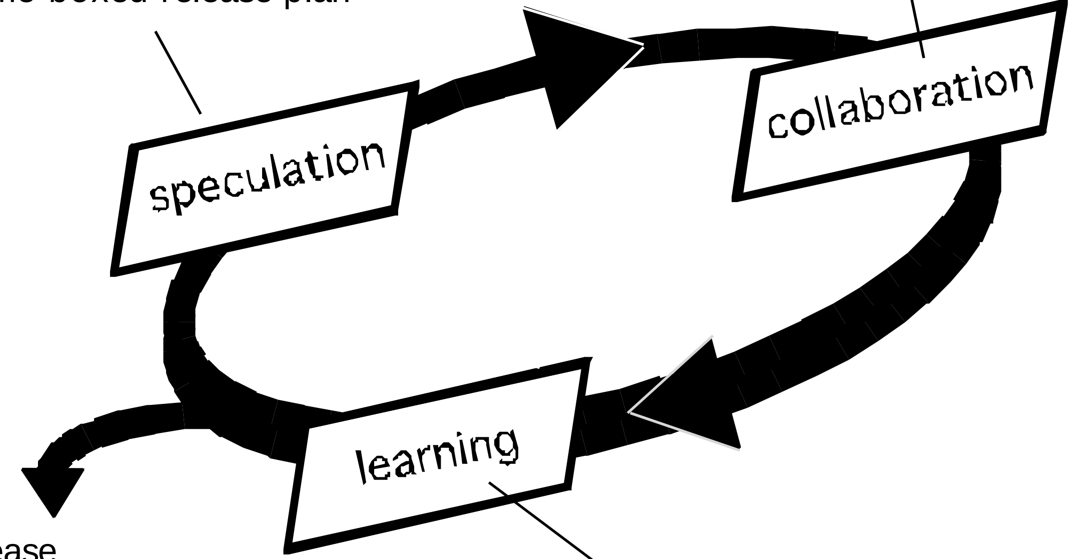
uses mission statement

project constraints

basic requirements

time-boxed release plan

Requirements gathering



Release

software increment

adjustments for subsequent cycles

components implemented/tested

focus groups for feedback

formal technical reviews

postmortems

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.

Scrum

- Originally proposed by Schwaber and Beedle
- Scrum—distinguishing features
  - 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 requiremetns or features
    - Sprint: work tasks within a process pattern
  - Meetings are very short and sometimes conducted without chairs
  - Demos are delivered to the customer within the time-box allocated

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 features more easily
  - Uses a feature template
    - <action> the <result> <by | for | of | to> a(n) <object>
      - Ex. Add the product to a shopping cart
      - Ex. Sotre the shipping-information for a customer
  - Feature set template
    - <action> <-ing> a(n) <object>
      - Ex. Making a product sale
  - 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 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