Software Engineering: A Practitioner's Approach, 6/e

Chapter 4
Agile Development

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*** **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.

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 frequent, multiple ‘software increments’
- Adapts as changes occur

Agile Team Characteristics

- Competence
- Common focus
- Collaboration
- Decisiveness
- Fuzzy problem-solving ability
- Mutual trust and respect
- Self-organization

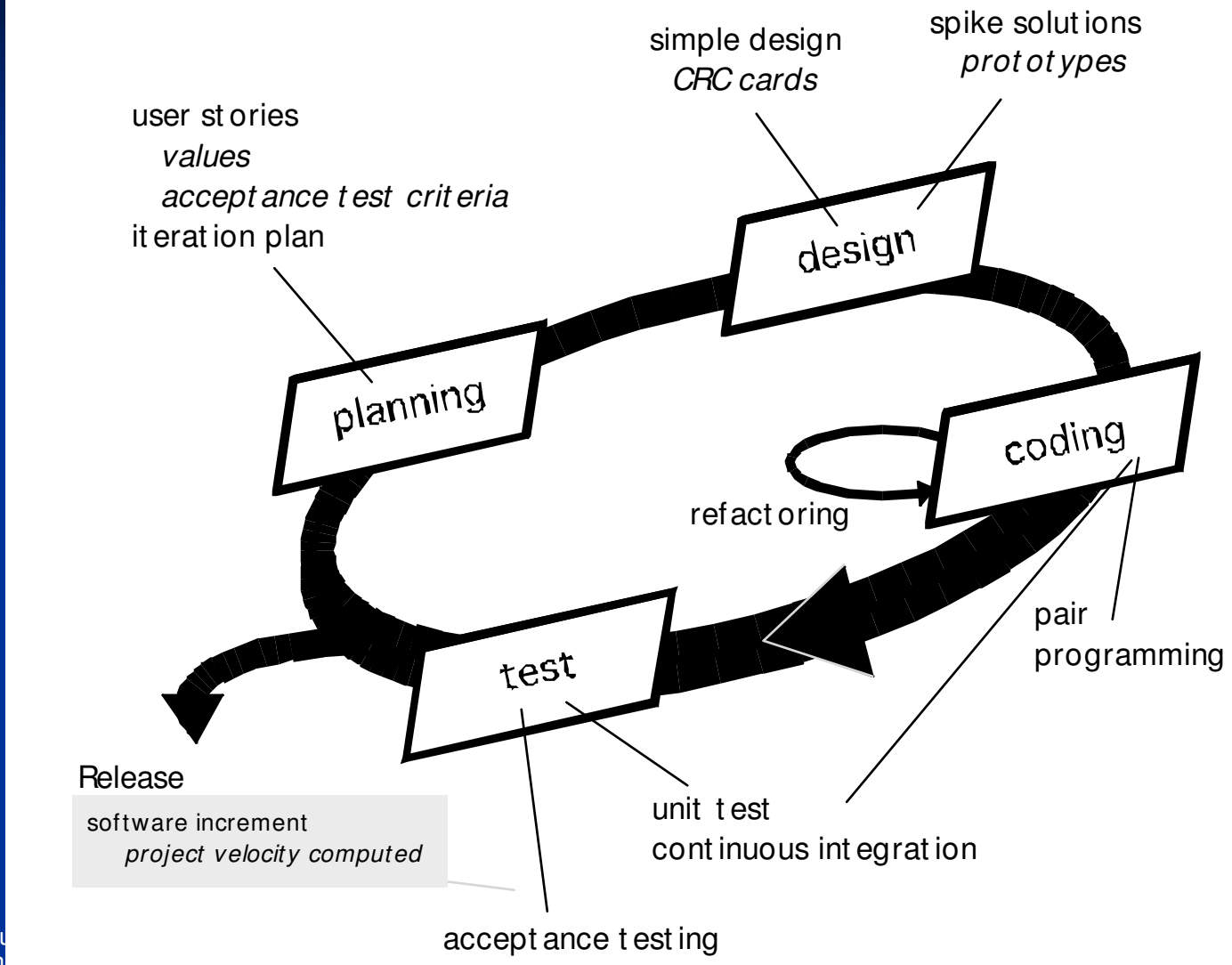
Extreme Programming (XP)

- 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"(I.e. No. of "stories" implemented during 1st release) is used to help define subsequent delivery dates for other increments

Extreme Programming (XP)

- XP Design
  - Follows the KIS principle
  - Encourage the use of CRC (Class-Responsibility Collaborator) cards (see Chapter 8)
  - For difficult design problems, suggests the creation of "spike solutions"—a design prototype (tool for lowering risks)
  - 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
  - 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

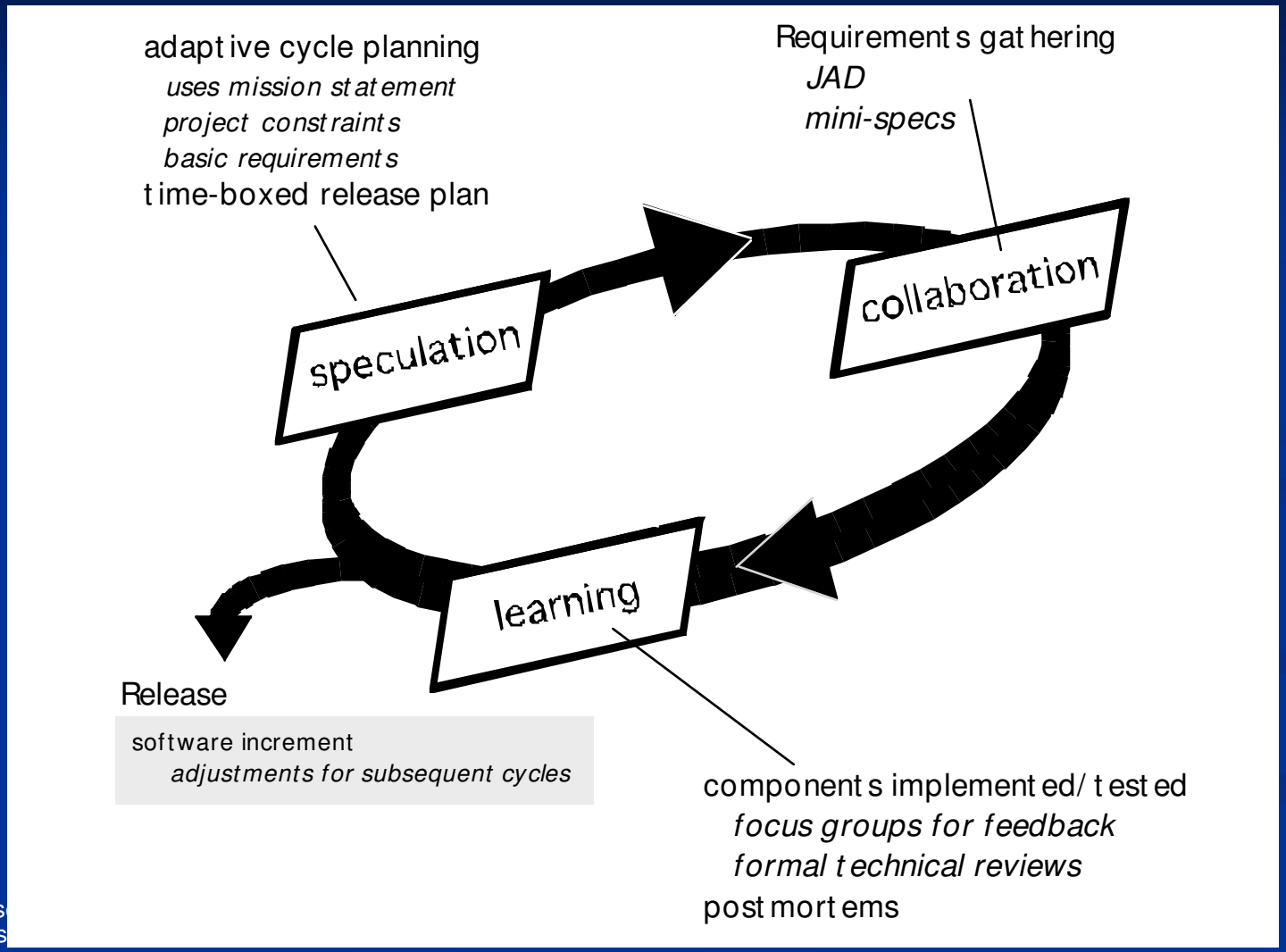
Extreme Programming (XP)



Adaptive Software Development

- Originally proposed by Jim Highsmith
- Useful to changing project and business conditions
- 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 (focus groups; technical reviews; postmortems)

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 (functional+information; application architecture; maintainability) are baselined at a high level.
    - Testing is integrated throughout the life-cycle.

Scrum

- Originally proposed by Schwaber and Beedle
- “Assumes up-front that chaos exists”
- Scrum — distinguishing features:
  - Small work teams (min. overhead; max. communication)
  - 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 (frequent S/W increments)
  - Meetings are very short and sometimes conducted without chairs
  - “demos” are delivered to the customer with the time-box allocated
  - Constant testing and documentation

Crystal

- Proposed by Cockburn and Highsmith
- Crystal — distinguishing features:
  - Actually a family of process models that allow "maneuverability" (resource-limited coop. game of invention and communication) 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>, viz:
      - Add the product to a shopping cart
      - Display the technical specs of a product
  - 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 for different aspects of the system
  - Travel light (keep only long-term models)
  - Content is more important than representation
  - Know the models and the tools you use to create them
  - Adapt locally (to the needs of the agile team)