Software Engineering

Chapter 4 Agile Development

Moon kun Lee
Division of Electronics & Information Engineering
Chonbuk National University

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 (Agile Alliance) , 2001.

What is “Agility”? Changes!

- 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

12 Principles for Agile Process

1. Highest priority: satisfaction of the customer through early & continuous delivery of valuable SW.
2. Welcome changing requirements, even late in development.
3. Deliver working SW frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people & developers must work together daily throughout the project.
5. Build project around motivated people.
6. The most efficient & effective methods of conveying information to and within a development team is face-to-face conversation.
7. Working SW is the primary measure of progress.
8. Agile process promote sustainable development.
9. Continuous attention to technical excellence and good design enhance.
10. Simplicity-the art of maximizing the amount of work not done-is essential.
11. The best architecture, requirements, and design 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.

Assumptions

- It is difficult to predict in advance which SW requirements will persist and which will change.
- For many types of SW, design & construction are interleaved.
- Analysis, design, construction, and testing are not as predictable (from a planning point of view) as we might like.

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

Human Factors

- Competence
- Common focus
- Collaboration
- Decision-making ability
- Fuzzy problem-solving ability
- Mutual trust & 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" is used to help define subsequent delivery dates for other increments

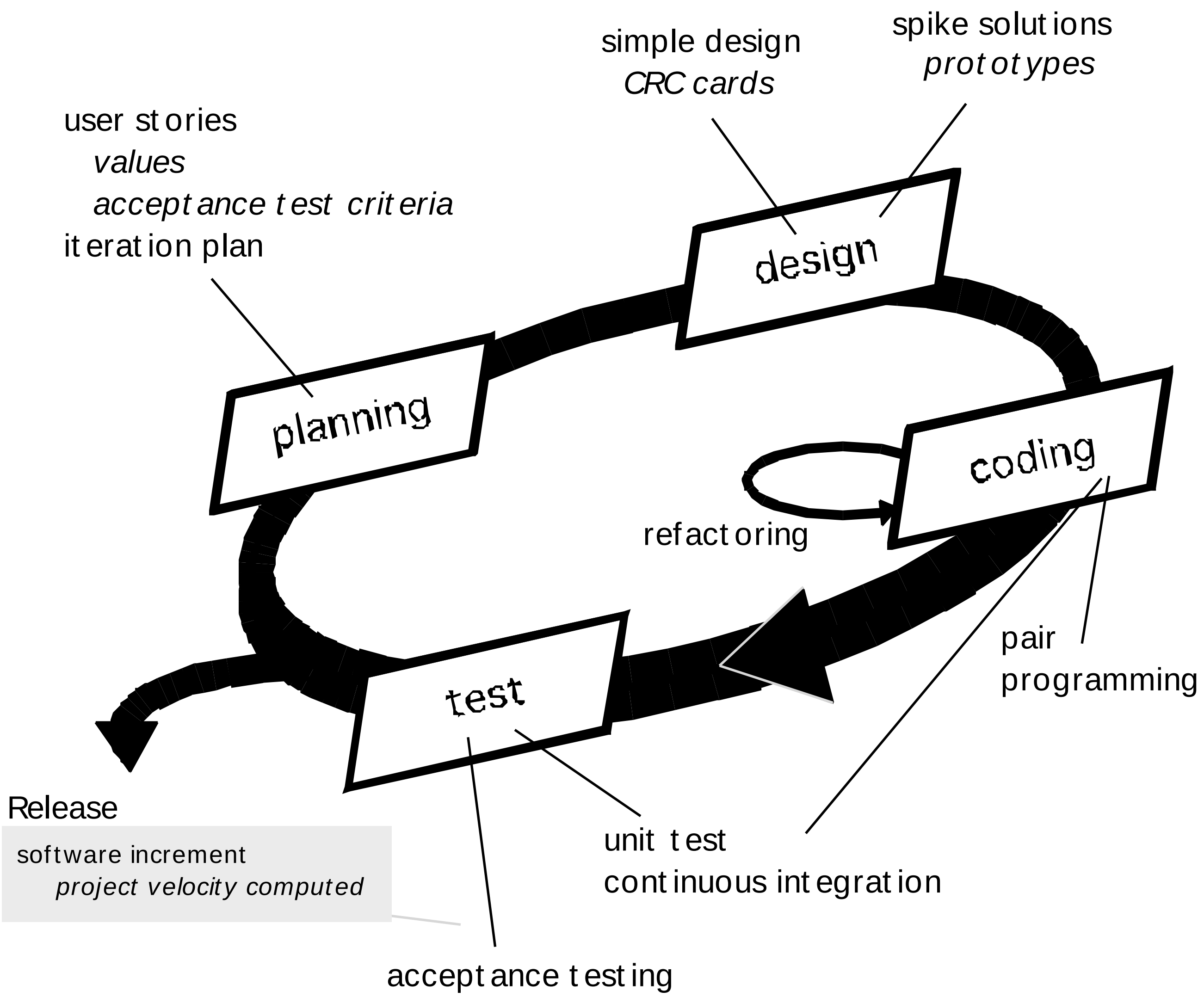
    - User story: story that describes required features & functionalities to be built.
    - Project velocity: the number of stories implemented during the first release.
    - Cost: period for an increment to be completed in weeks.

Extreme Programming (XP)

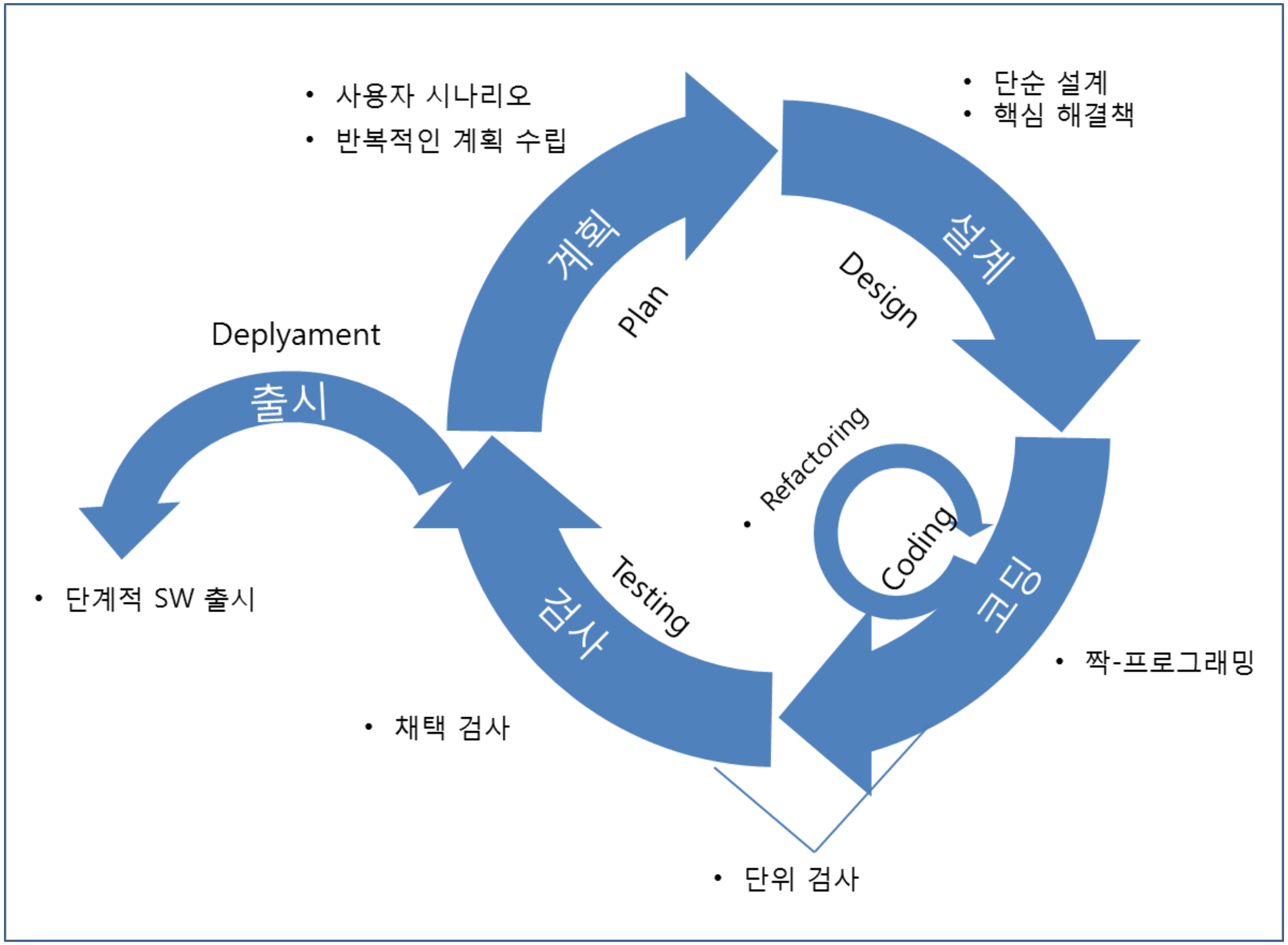
- XP Design
  - Follows the KIS (keep it simple) principle
  - Encourage the use of CRC cards (see Chapter 8)
  - For difficult design problems, suggests the creation of "spike solutions"—a design prototype
  - Encourages "refactoring"—an iterative refinement of the internal program design
- XP Coding
  - Recommends the construction of a unit test for a story *before* coding commences
  - Encourages "pair programming"
  - Continuous integration
- XP Testing
  - All unit tests are executed daily
  - "Acceptance tests" are defined by the customer and executed to assess customer visible functionality

Class Responsibility Collaboration

Extreme Programming (XP)



Extreme Programming (XP) Model



Adaptive Software Development (ASD)

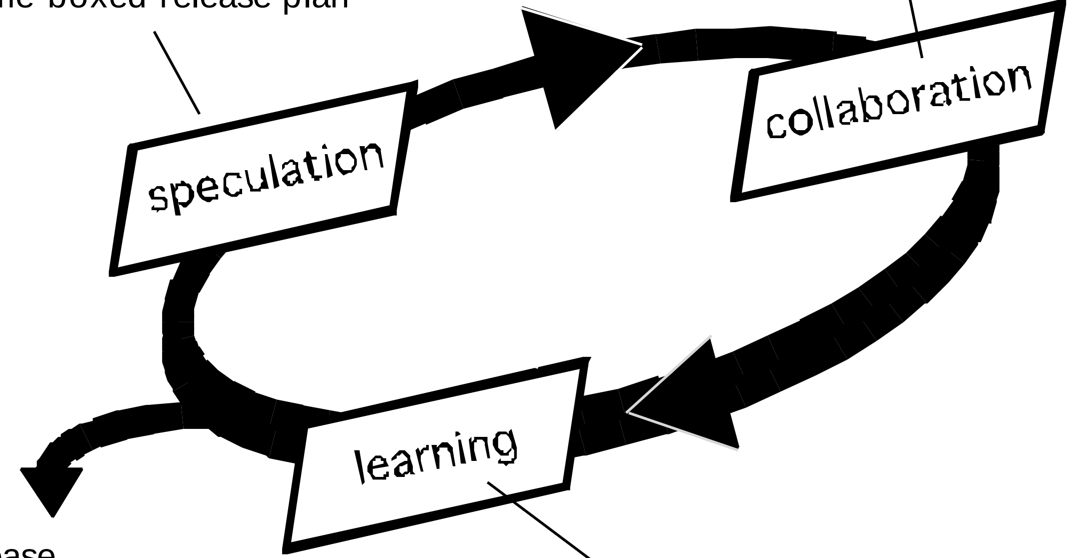
- Originally proposed by Jim Highsmith
  - A technique for building complex SW & systems
  - Philosophy: human collaboration & team self-organization
- ASD — distinguishing features
  - Mission-driven planning
  - Component-based focus
  - Uses "time-boxing" (See Chapter 24)
  - Explicit consideration of risks
  - Emphasizes collaboration for requirements gathering
  - Emphasizes "learning" throughout the process

Adaptive Software Development

Joint
Application
Design

adaptive cycle planning
uses mission statement
project constraints
basic requirements
time-boxed release plan

Requirements gathering
JAD
mini-specs

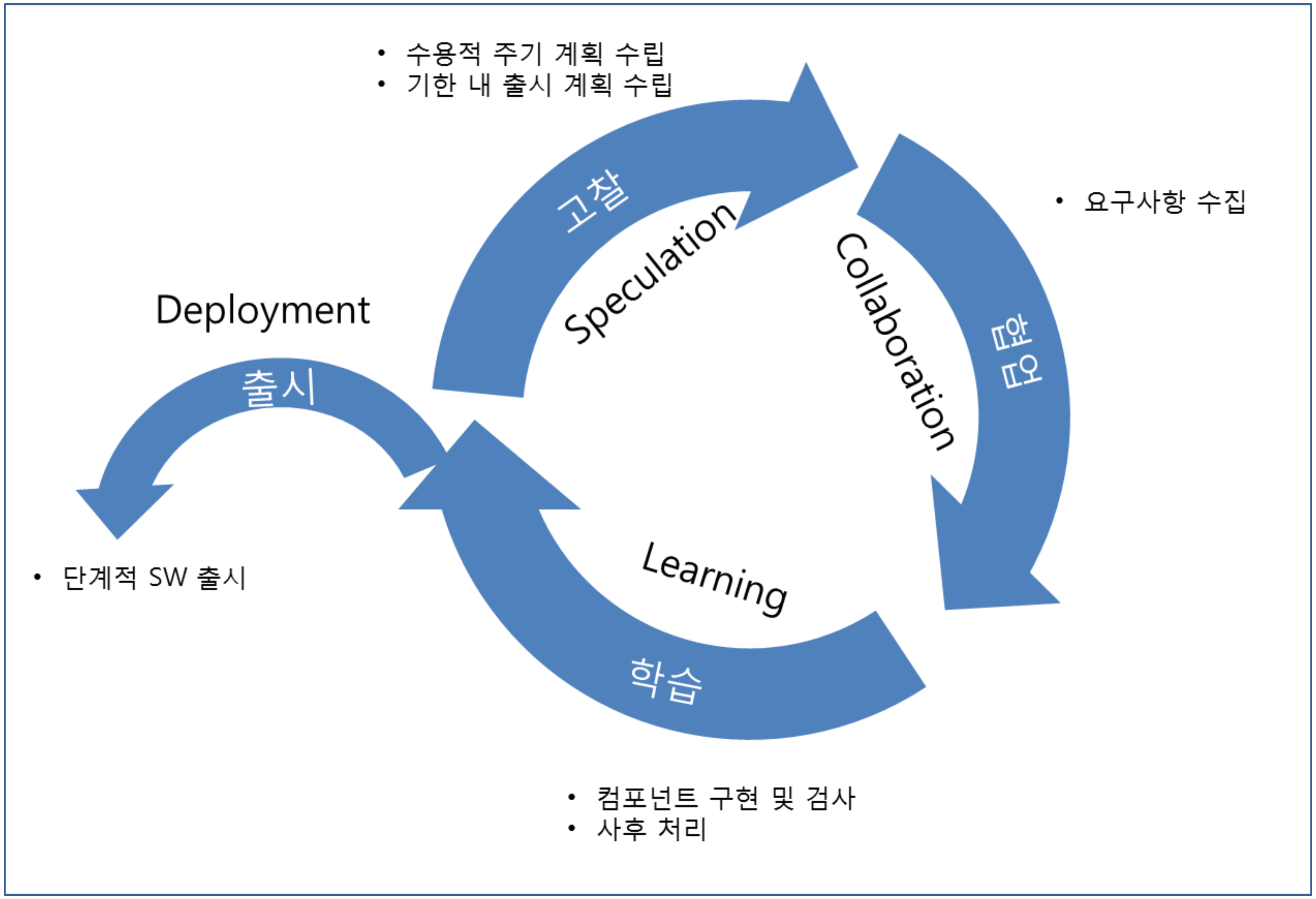


Release

software increment
adjustments for subsequent cycles

components implemented/tested
focus groups for feedback
formal technical reviews
postmortems

Adaptive Software Development (ASD) Model



Dynamic Systems Development Method(DSSM)

- Promoted by the DSDM Consortium (www.dsdm.org)
- DSDM—distinguishing features
  - Similar in most respects to XP and/or ASD
  - Tight time constraints
  - 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.

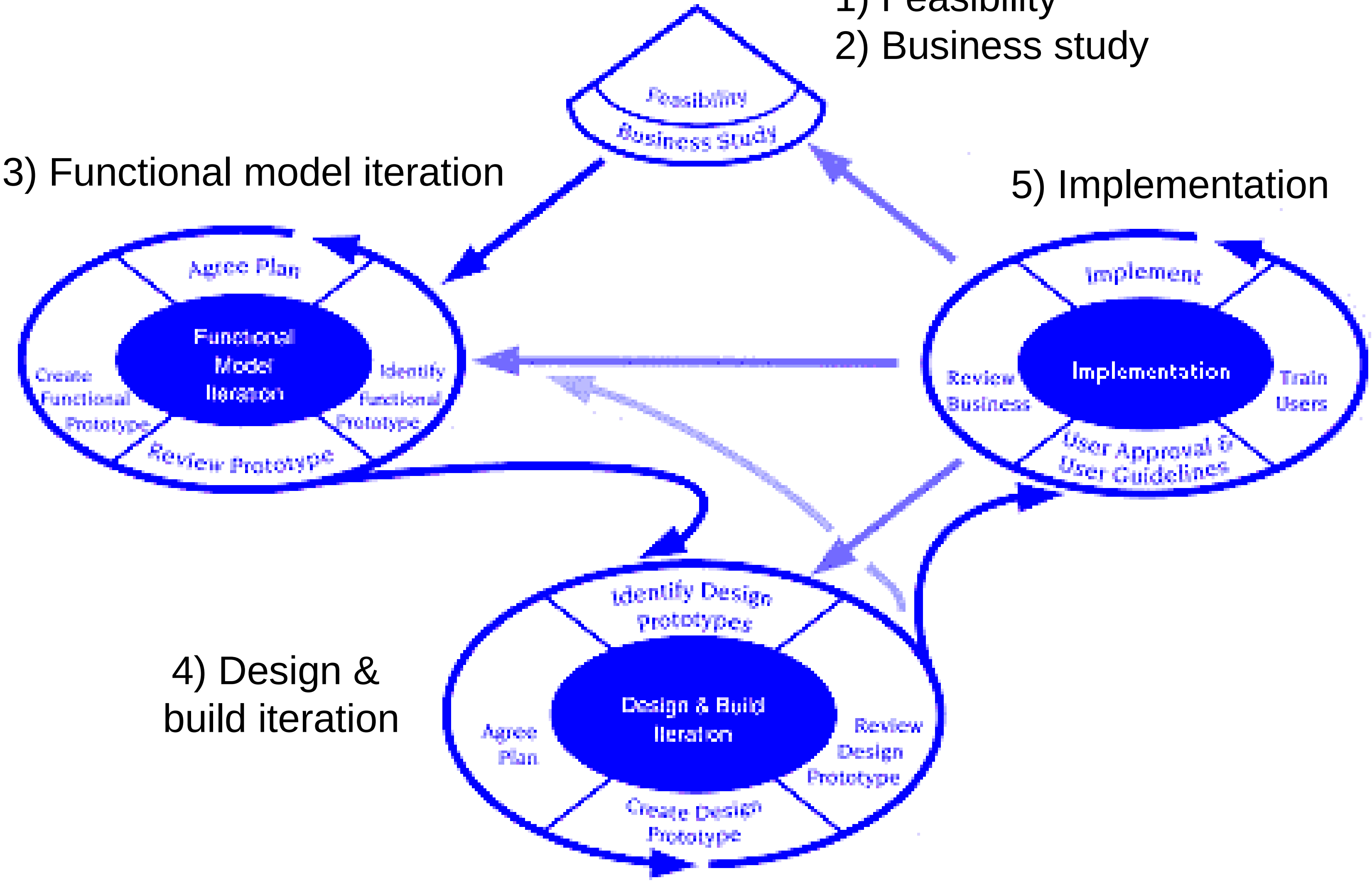
Dynamic Systems Development Method

1) Feasibility
2) Business study

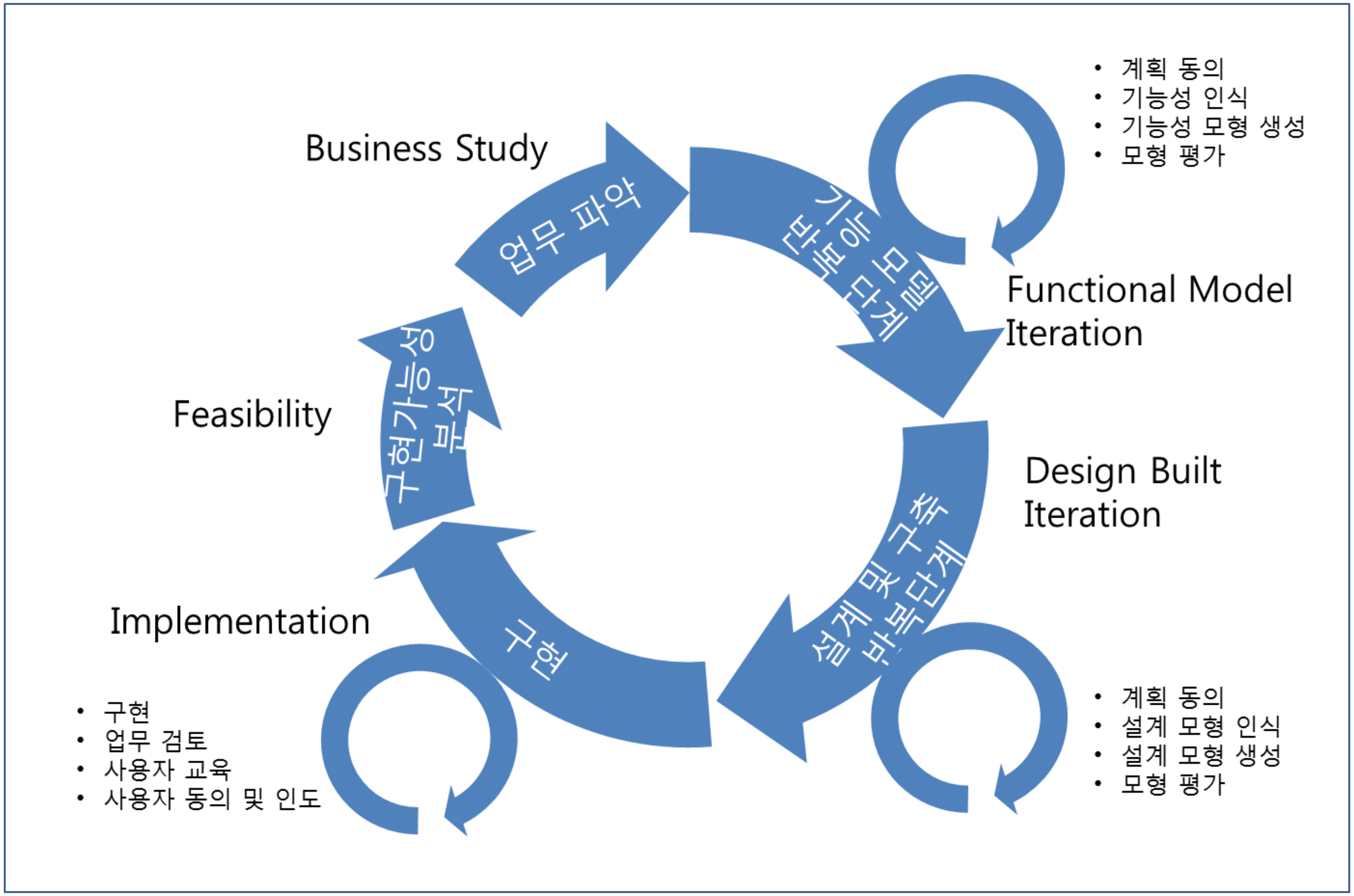
3) Functional model iteration

5) Implementation

4) Design &
build iteration



Dynamic Systems Development Method (DSDM)



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

Scrum

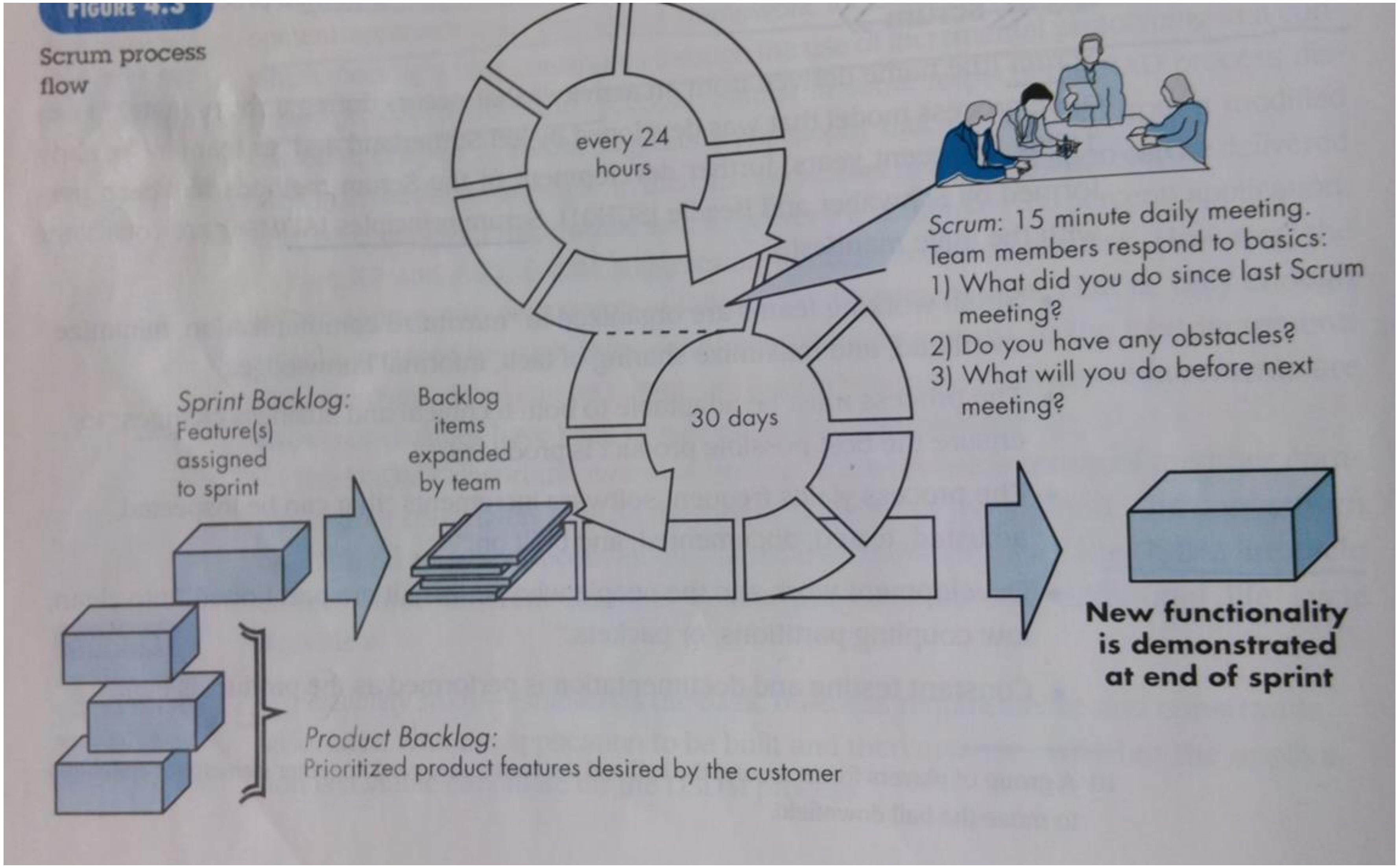
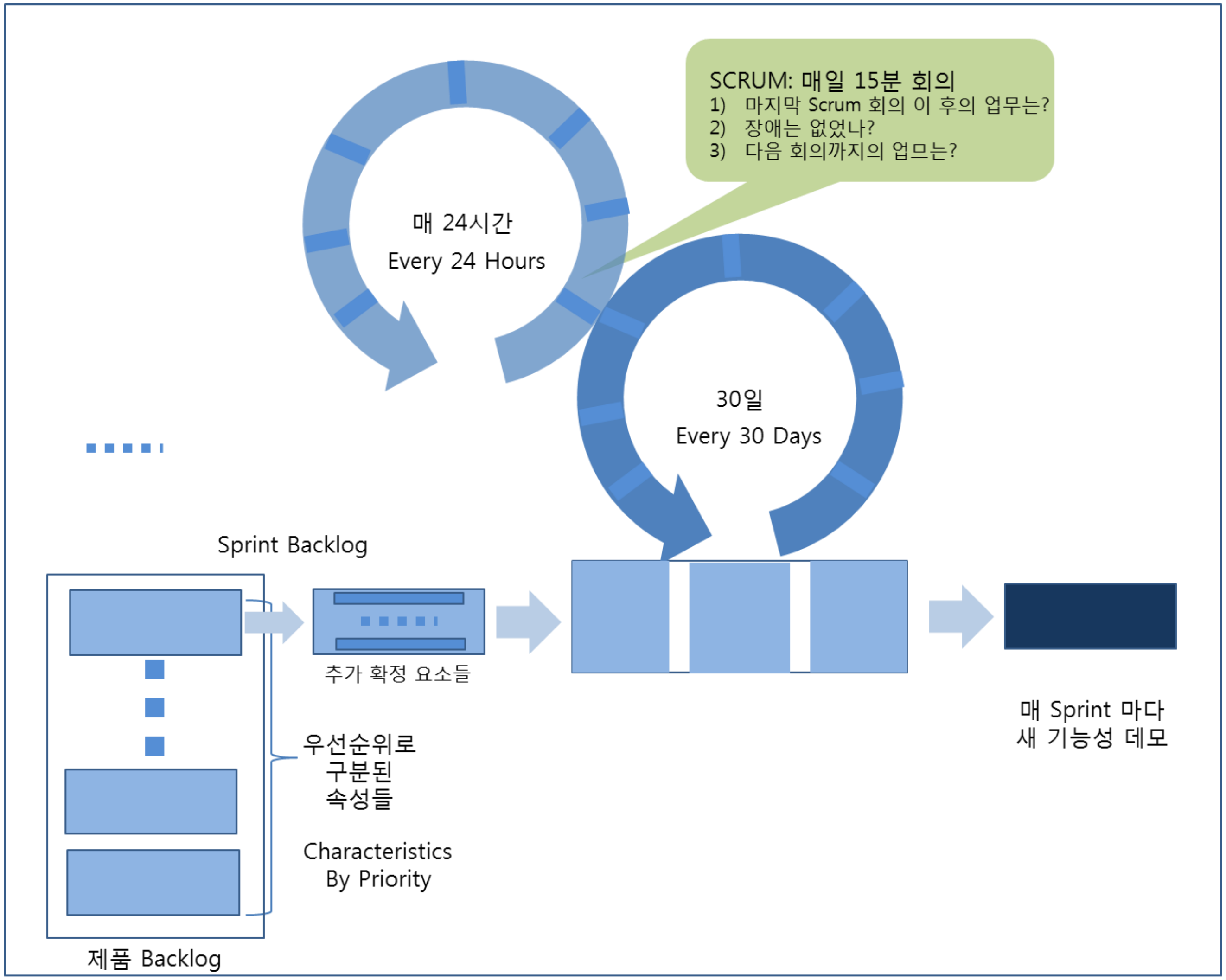


FIGURE 4.3
Scrum process flow
every 24 hours
30 days
Scrum: 15 minute daily meeting.
Team members respond to basics:
1) What did you do since last Scrum meeting?
2) Do you have any obstacles?
3) What will you do before next meeting?
Sprint Backlog: Feature(s) assigned to sprint
Backlog items expanded by team
New functionality is demonstrated at end of sprint
Product Backlog: Prioritized product features desired by the customer

SCRUM Model


SCRUM: 매일 15분 회의
1) 마지막 Scrum 회의 이 후의 업무는?
2) 장애는 없었나?
3) 다음 회의까지의 업므는?
매 24시간
Every 24 Hours
30일
Every 30 Days
Sprint Backlog
추가 확정 요소들
우선순위로 구분된 속성들
Characteristics By Priority
제품 Backlog
매 Sprint 마다 새 기능성 데모

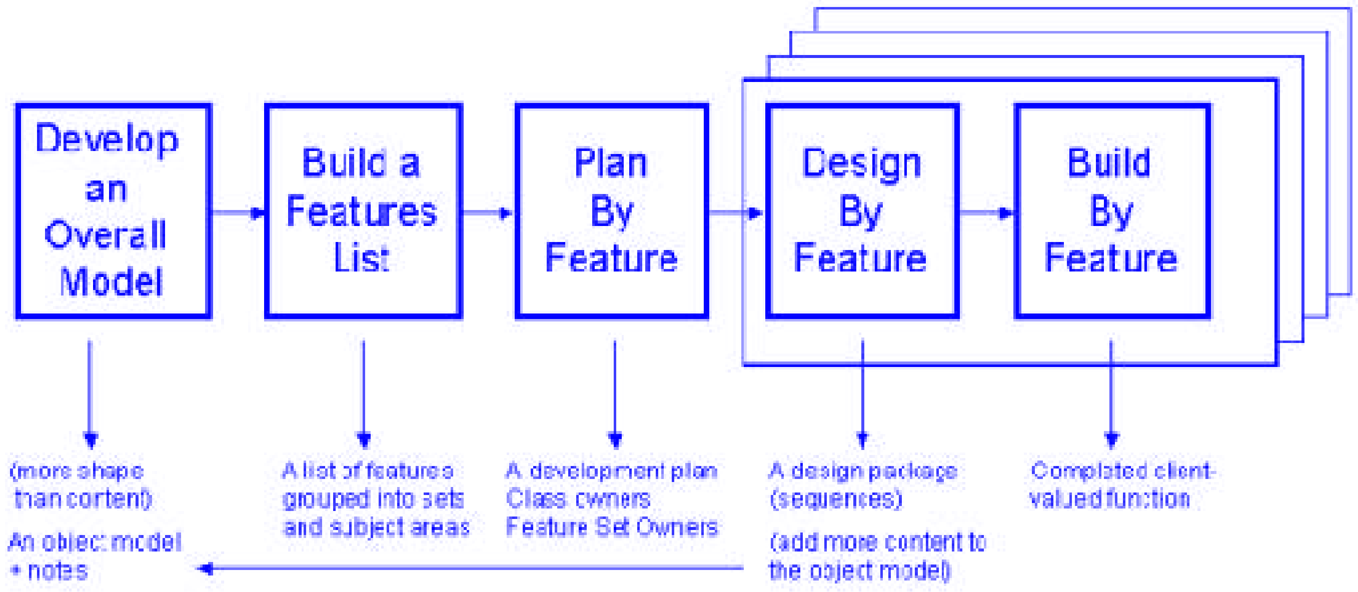
Crystal

- Proposed by Cockburn and Highsmith
- Crystal—distinguishing features
  - Actually a family of process models that allow "maneuverability" based on problem characteristics
    - A resource-limited, collaborative game of invention and communication, with a primary goal of delivering useful, working SW and a secondary goal of setting up for the next game.
  - Face-to-face communication is emphasized
  - Suggests the use of "reflection workshops" to review the work habits of the team

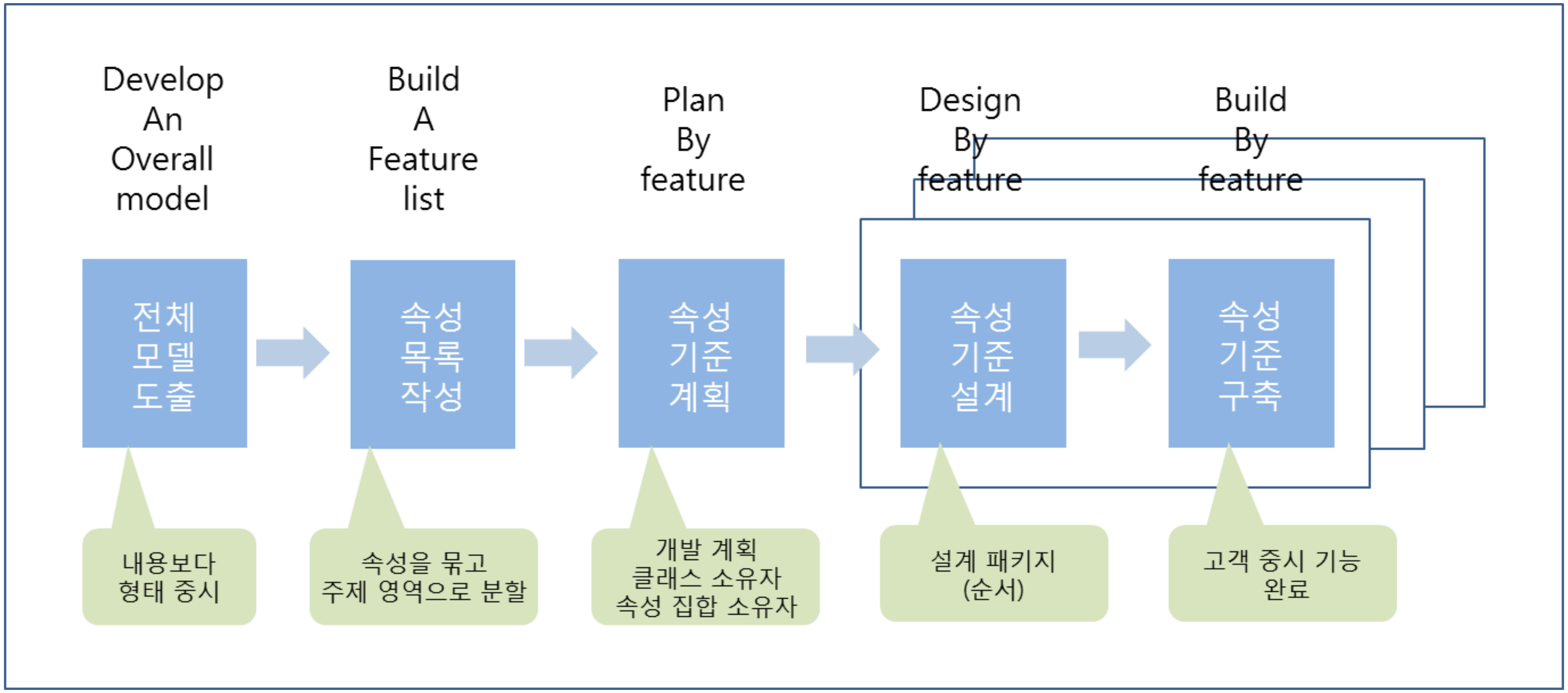
Feature Driven Development

- 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

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

Feature Driven Development (FAA) Model



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