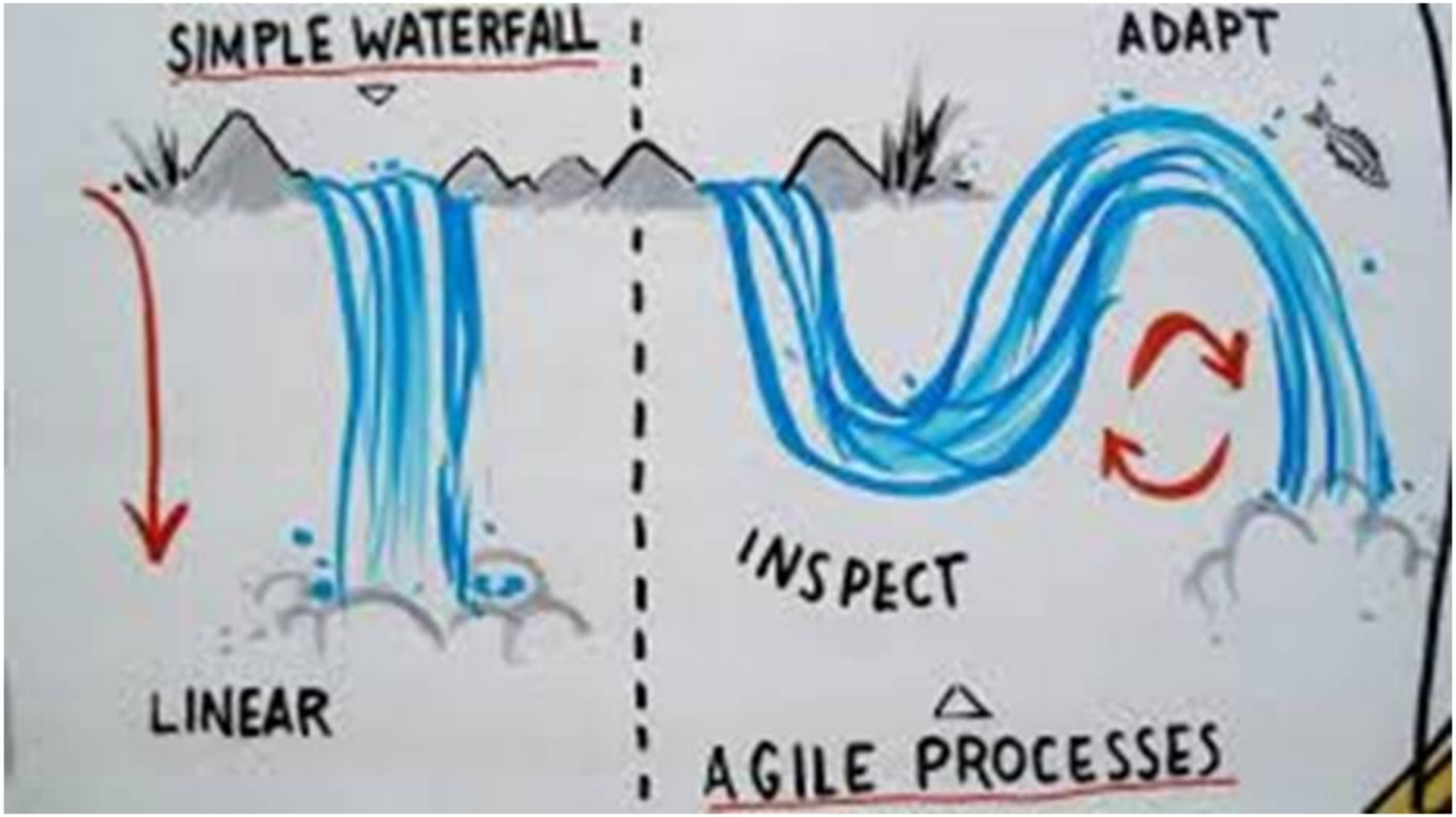
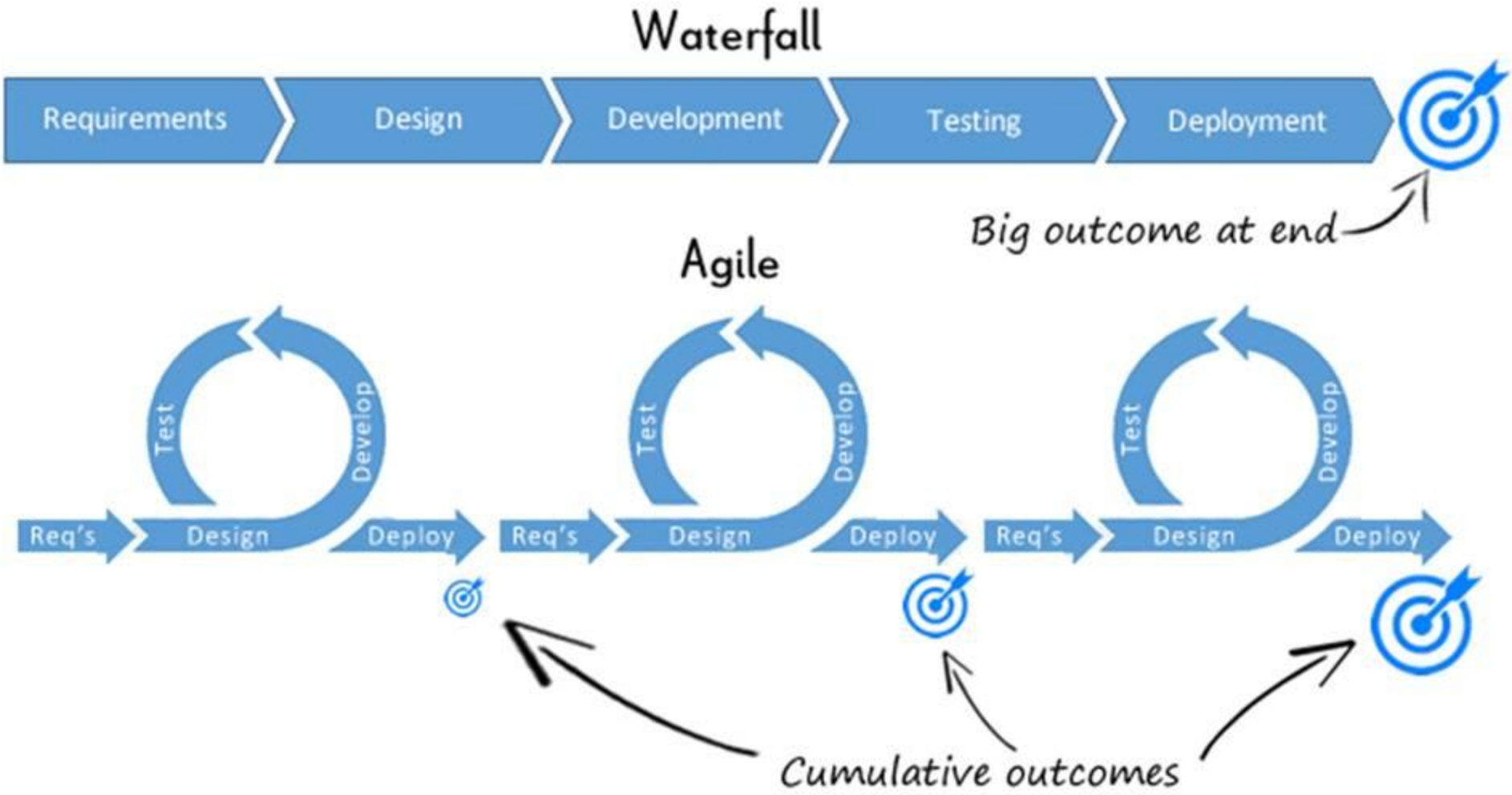
Unit V

Chapter 5 Agile Development

Chapter 6 Human Aspects of Software Engineering

Chapter 5 Agile Development


SIMPLE WATERFALL
ADAPT
INSPECT
LINEAR
AGILE PROCESSES


Waterfall
Requirements
Design
Development
Testing
Deployment
Big outcome at end
Agile
Test
Develop
Req's
Design
Deploy
Test
Develop
Req's
Design
Deploy
Test
Develop
Req's
Design
Deploy
Cumulative outcomes

The Manifesto for Agile Software Development

Uncovering better ways of developing software by doing it and helping others do it. Through this we 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

Kent Beck et al

Why and What Steps are "Agility" important?

- Why? The modern **business environment is fast-paced** and ever-changing. Represents an **alternative to conventional** software engineering for certain classes of software projects.

- What? Basic activities- communication, planning, modeling, construction and deployment. Agile pushes the team towards construction and delivery sooner.

- Real **important work product** is operational "software increment" that is delivered.

5.1 What is "Agility"? (Jacobson discussion)

- Effective (rapid and adaptive) response to change (team members, new technology, requirements).
- Effective communication among team members, technological and business people, software engineers and managers.
- Drawing customer into the team. Eliminate "us and them" attitude. Plan must be flexible.
- Organizing a team so that it is in control.
- Eliminate all but the most essential work products and keep them lean.
- Emphasize an incremental delivery strategy that gets working software to the customer as rapidly as feasible.

5.1 What is "Agility"? (Jacobson discussion)

Yielding ...

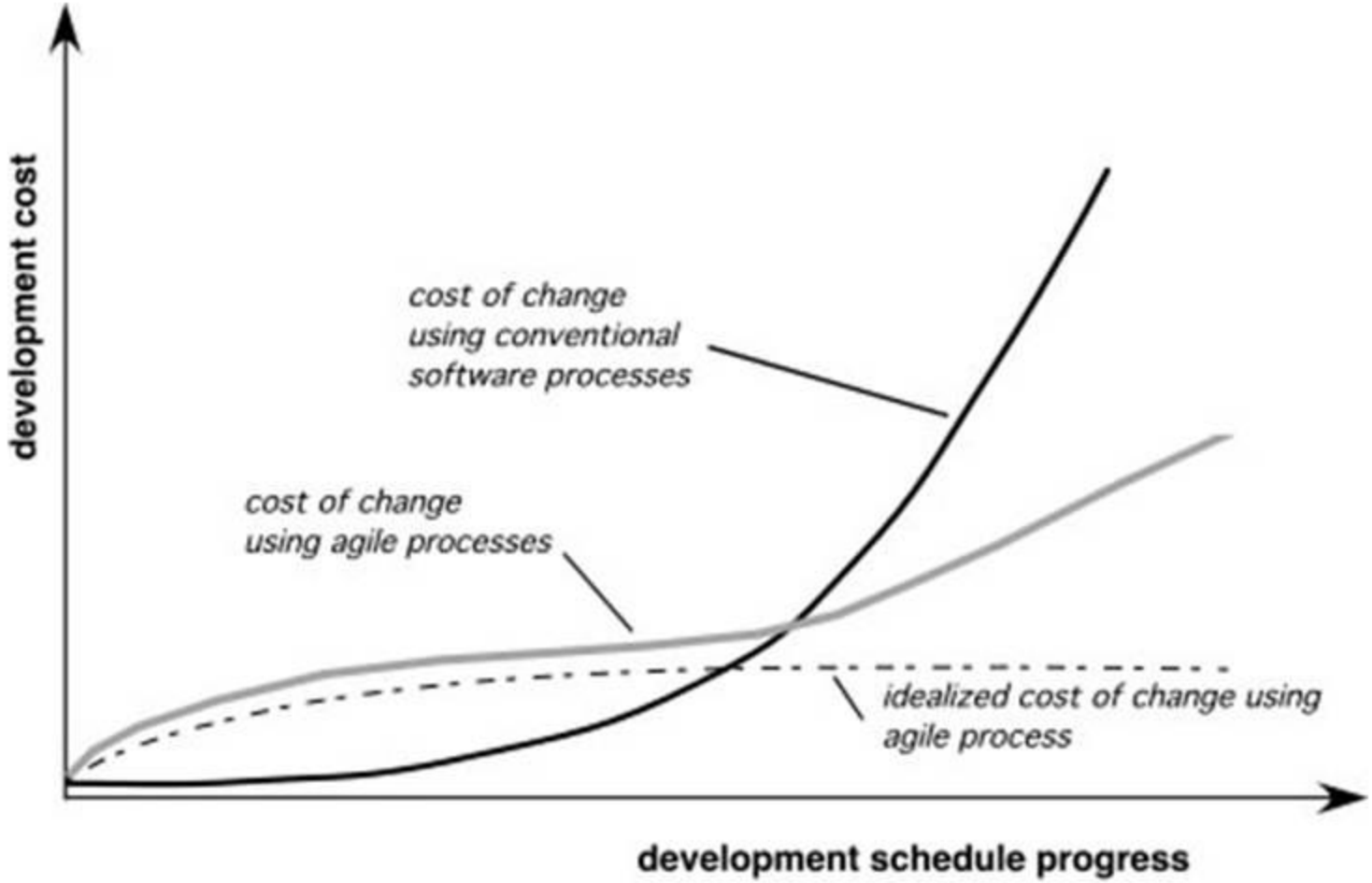
- Rapid, incremental delivery of software
- Delivery over analysis and design
- Active and continuous communication between developers and customers.

5.2 Agility and the Cost of Change

- Conventional wisdom: cost of change increases nonlinearly as a project progresses.

- Agile process may "flatten" the cost of change curve by coupling incremental delivery with agile practices such as continuous unit testing and pair programming.

5.2 Agility and the Cost of Change



5.3 What Is An Agile Process

- Driven by customer descriptions
- Some assumptions:
  - Recognizes plans are short-lived (requirements and customer priorities change)
  - Develops software iteratively with emphasis on construction activities
  - Analysis, design, construction and testing are not predictable.
- Thus has to Adapt as changes occur due to unpredictability.
- Delivers multiple 'software increments', deliver an operational prototype or portion of an OS to collect customer feedback for adaption.

5.3 What Is An Agile Process

Agility Principles - I

1. Highest priority is to satisfy the customer through early and continuous delivery.
2. Welcome changing requirements, even late in development.

3. Deliver working software frequently.
4. Business people and developers must work together daily throughout project.

5. Build projects around motivated individuals. Give them environment and support, and trust them to get the job done.
6. Efficient and effective method of conveying information to and within team is face–to–face conversation.

Agility Principles - II

7. Working software is the primary measure of progress.
8. It promotes sustainable development. Sponsors, developers, and users should be able to maintain constant pace indefinitely.

9. Continuous attention to technical excellence and good design.
10. Simplicity - art of maximizing the amount of work not done – is essential.

11. 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.

5.3 What Is An Agile Process

The politics of Agile Development

Traditional Vs Agile (difficult to achieve)

5.4 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

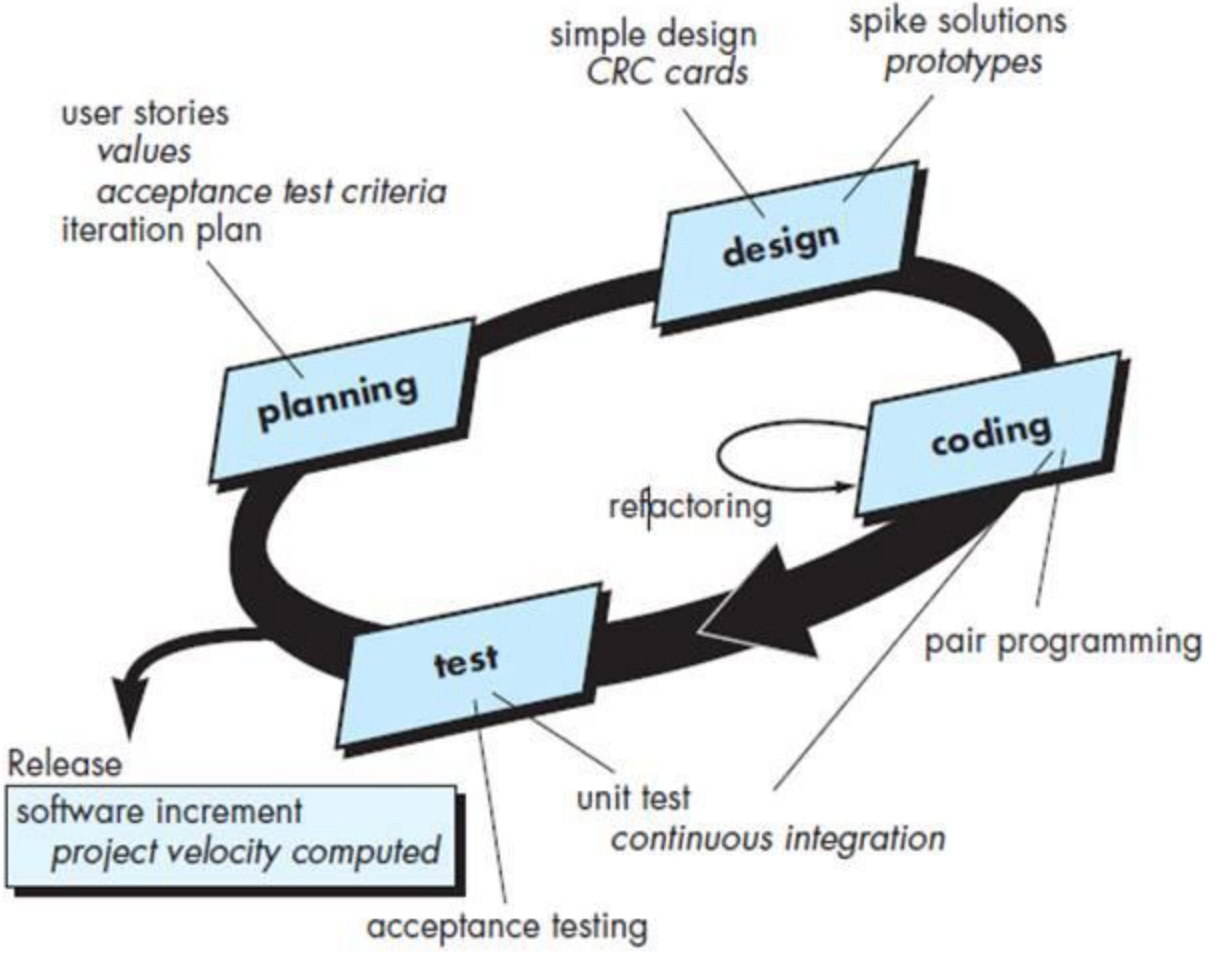
5.4 Extreme Programming (XP)

- **XP Design**
  - Follows the **KIS(Keep it simple) 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 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

5.4 Extreme Programming (XP)

FIGURE 5.2

The Extreme Programming process



Industrial XP (IXP)

- IXP has greater inclusion of management, expanded customer roles, and upgraded technical practices for significant projects within the large organizations

- IXP incorporates six new practices:
  - Readiness assessment
  - Project community
  - Project chartering
  - Test driven management
  - Retrospectives
  - Continuous learning

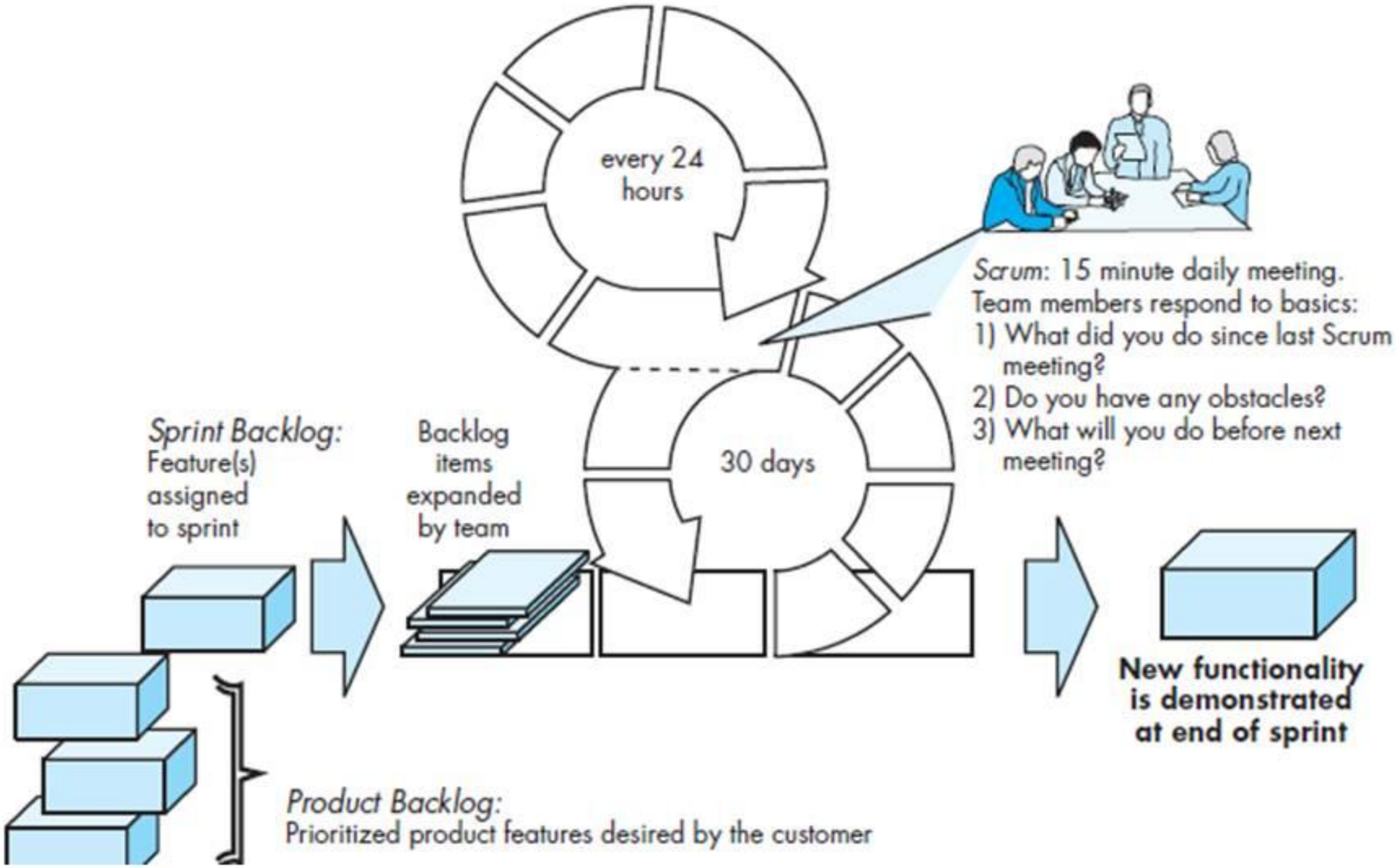
5.5.1 Scrum

- 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.

5.5.1 Scrum

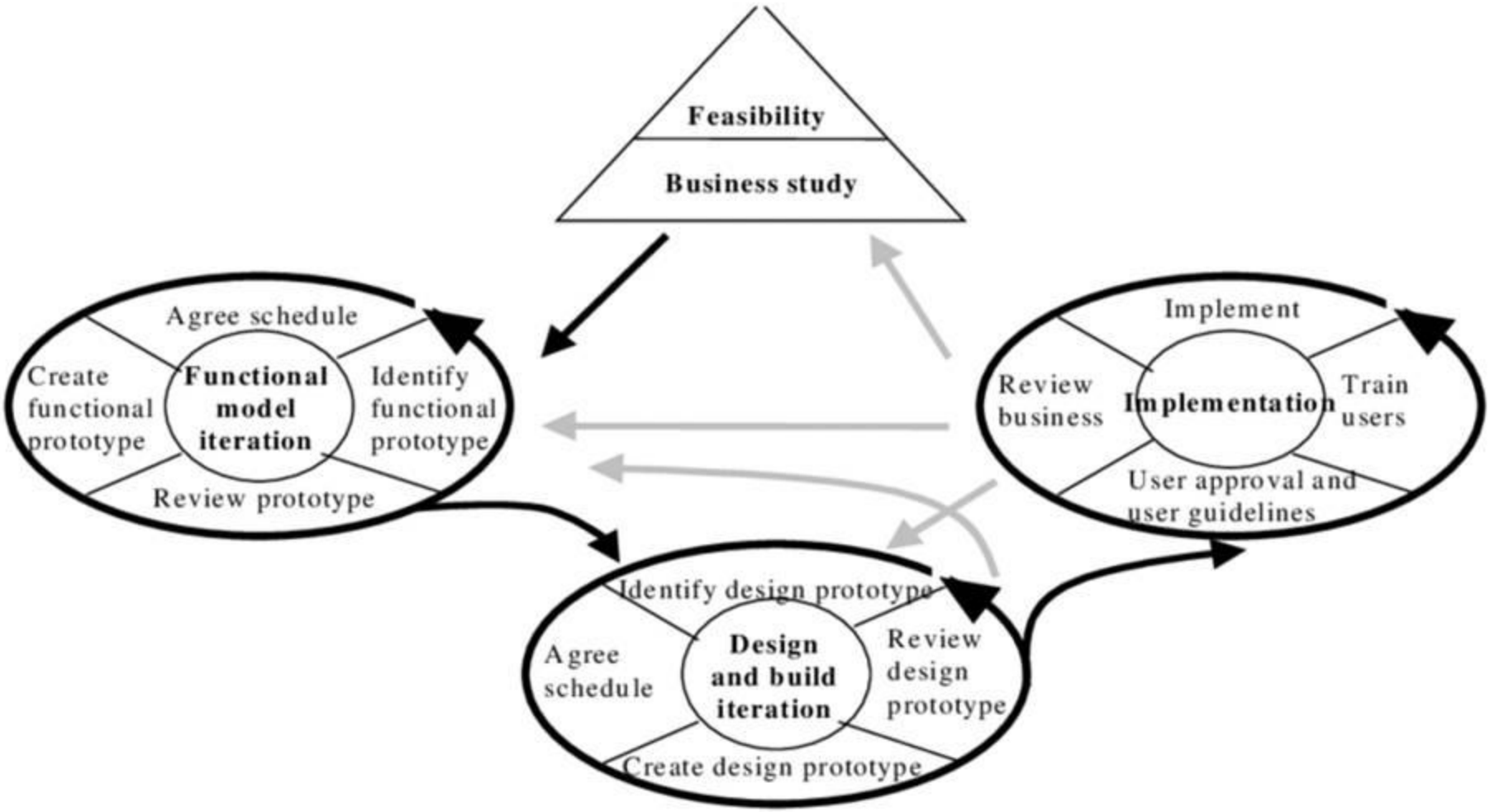
FIGURE 5.3 Scrum process flow



5.5.2 Dynamic Systems Development Method

- Promoted by the DSDM Consortium (www.dsdm.org)
- DSDM—distinguishing features
  - **Similar in most respects to XP**
  - **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.

5.5.2 Dynamic Systems Development Method



5.5.3 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

5.5.4 Agile Unified Process

- Each AUP iteration addresses these activities:
  - Modeling
  - Implementation
  - Testing
  - Deployment
  - Configuration and project management
  - Environment management

5.6 A Tool Set for the Agile Process

- Some proponents of the agile philosophy argue that automated software tools should be **viewed as a minor supplement** to the team's activities, and not at all pivotal to the success of the team.

- Others argue that **tools benefit the agile teams and permit the rapid flow** of understanding.

- Some tools are **technological, helping distributed teams simulate being physically present.**

- **Collaborative and communication** "tools" are generally low tech and incorporate any mechanism ("physical proximity, whiteboards, poster sheets, index cards, and sticky notes" or modern social networking techniques) that provides information and coordination among agile developers.

- **Active communication** is achieved via the team dynamics (e.g., pair programming), while **passive communication** is achieved by "information radiators" (e.g., a flat panel display that presents the overall status of different components of an increment).

5.6 A Tool Set for the Agile Process

Project management tools deemphasize the Gantt chart and replace it with
earned value charts or "graphs of tests created versus passed . . . other agile tools are using to optimize the environment in which the agile team works (e.g., more efficient meeting areas), improve the team culture by nurturing social interactions (e.g., collocated teams), physical devices (e.g., electronic whiteboards), and process enhancement (e.g., pair programming or time-boxing)".

Are any of these things really tools? They are, if they facilitate the work performed by an agile team member and enhance the quality of the end product

Human Factors

- Process **molds** to **needs of the people** and team.
- **Key traits** exist among the people:
  - **Competence.** (talent, skills, knowledge)
  - **Common focus.** (deliver working software increment)
  - **Collaboration.** (peers and stakeholders)
  - **Decision-making ability.** (freedom to control its own destiny)
  - **Fuzzy problem-solving ability.** (ambiguity and constant changes, today's problem may not be tomorrow's problem)
  - **Mutual trust and respect.**
  - **Self-organization.** (themselves for work done, process for its local environment, work schedule)