Web Application Development

Software Process Models

Indian Institute of Information Technology Sri City, India

April 9, 2021

The Process Model

- The general process models are as follows
- The waterfall model
- Incremental Process Models
- Evolutionary Process Models
- Concurrent Models

The waterfall model

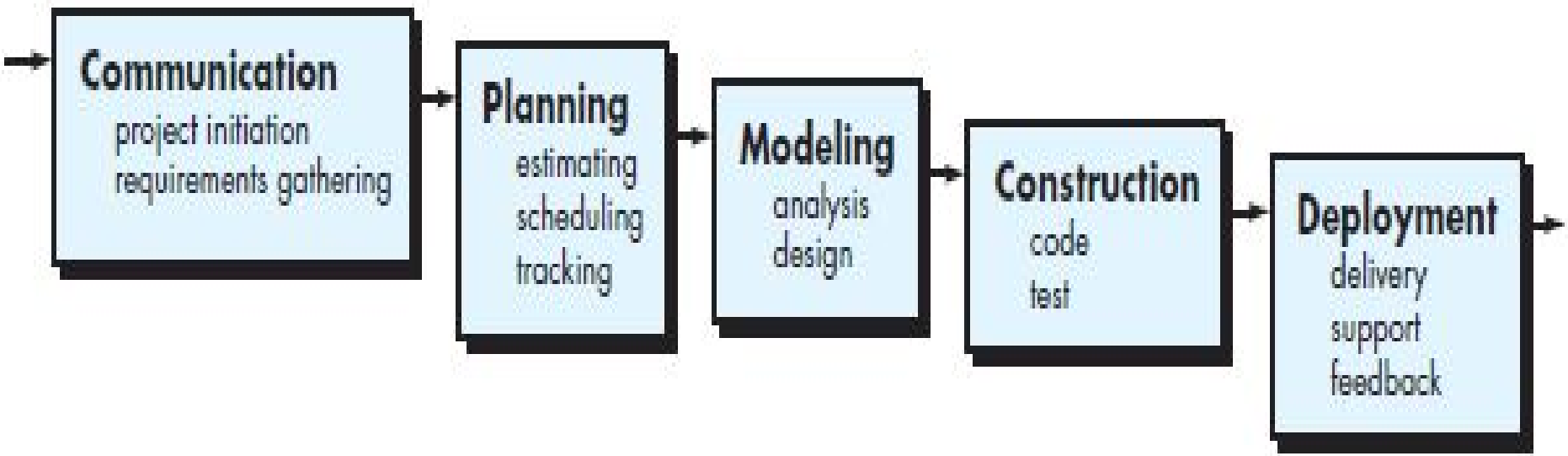

Figure 1. The waterfall model

The waterfall model

- The waterfall model, sometimes called the classic life cycle , suggests a systematic, sequential approach to software development that begins with customer specification of requirements and progresses through planning, modeling, construction, and deployment, culminating in on-going support of the completed software

- The waterfall model is the oldest paradigm for software engineering

The waterfall model

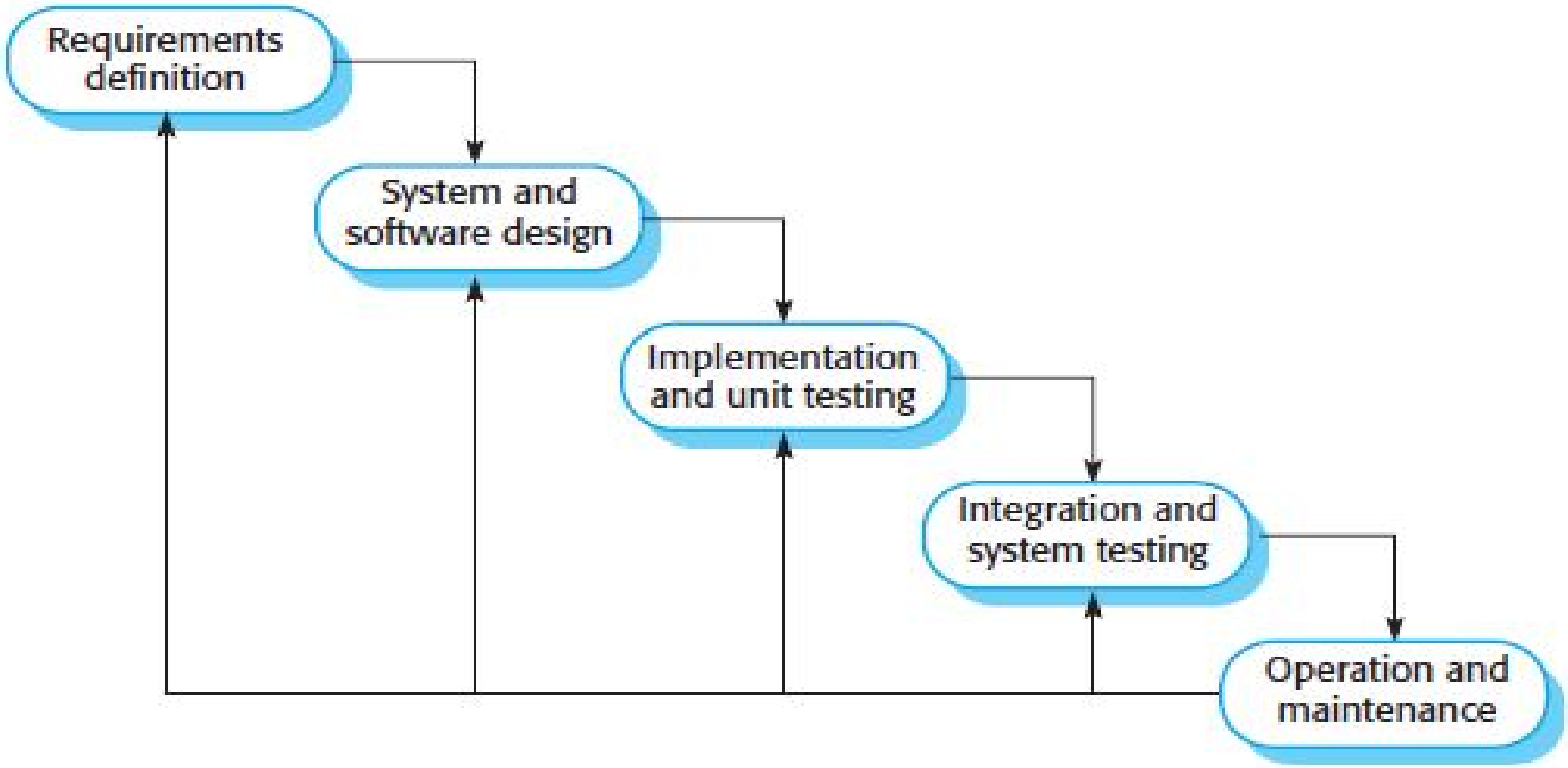

Figure 2. The waterfall model

The waterfall model

- The stages of the waterfall model directly reflect the fundamental software development activities:
- Requirements analysis and definition
- System and software design
- Implementation and unit testing
- Integration and system testing
- Operation and maintenance

The waterfall model

- **Requirements analysis and definition** The system's services, constraints, and goals are established by consultation with system users. They are then defined in detail and serve as a system specification.

- **System and software design** The systems design process allocates the requirements to either hardware or software systems. It establishes an overall system architecture. Software design involves identifying and describing the fundamental software system abstractions and their relationships.

- **Implementation and unit testing** During this stage, the software design is realized as a set of programs or program units. Unit testing involves verifying that each unit meets its specification.

The waterfall model

- **Integration and system testing** The individual program units or programs are integrated and tested as a complete system to ensure that the software requirements have been met. After testing, the software system is delivered to the customer.

- **Operation and maintenance** Normally, this is the longest life-cycle phase. The system is installed and put into practical use. Maintenance involves correcting errors that were not discovered in earlier stages of the life cycle, improving the implementation of system units, and enhancing the system's services as new requirements are discovered.

The waterfall model

- The waterfall model is appropriate for some types of system
- Embedded systems where the software has to interface with hardware systems. Because of the inflexibility of hardware, it is not usually possible to delay decisions on the software's functionality until it is being implemented.
- Critical systems where there is a need for extensive safety and security analysis of the software specification and design. In these systems, the specification and design documents must be complete so that this analysis is possible.
- Large software systems that are part of broader engineering systems developed by several partner companies. The hardware in the systems may be developed using a similar model, and companies find it easier to use a common model for hardware and software

The waterfall model

- The problems that are sometimes encountered when the waterfall model is applied are:
- Real projects rarely follow the sequential flow that the model proposes. Although the linear model can accommodate iteration, it does so indirectly. As a result, changes can cause confusion as the project team proceeds.
- It is often difficult for the customer to state all requirements explicitly. The waterfall model requires this and has difficulty accommodating the natural uncertainty that exists at the beginning of many projects.

The waterfall model

- The customer must have patience. A working version of the program(s) will not be available until late in the project time span. A major blunder, if undetected until the working program is reviewed, can be disastrous.

- The waterfall model is not the right process model in situations where informal team communication is possible and software requirements change quickly. Iterative development and agile methods are better for these systems.

Incremental Process Model

- The incremental model combines the elements' linear and parallel process flows
- The incremental model applies linear sequences in a staggered fashion as calendar time progresses.
- Each linear sequence produces deliverable "increments" of the software.

Incremental Process Model

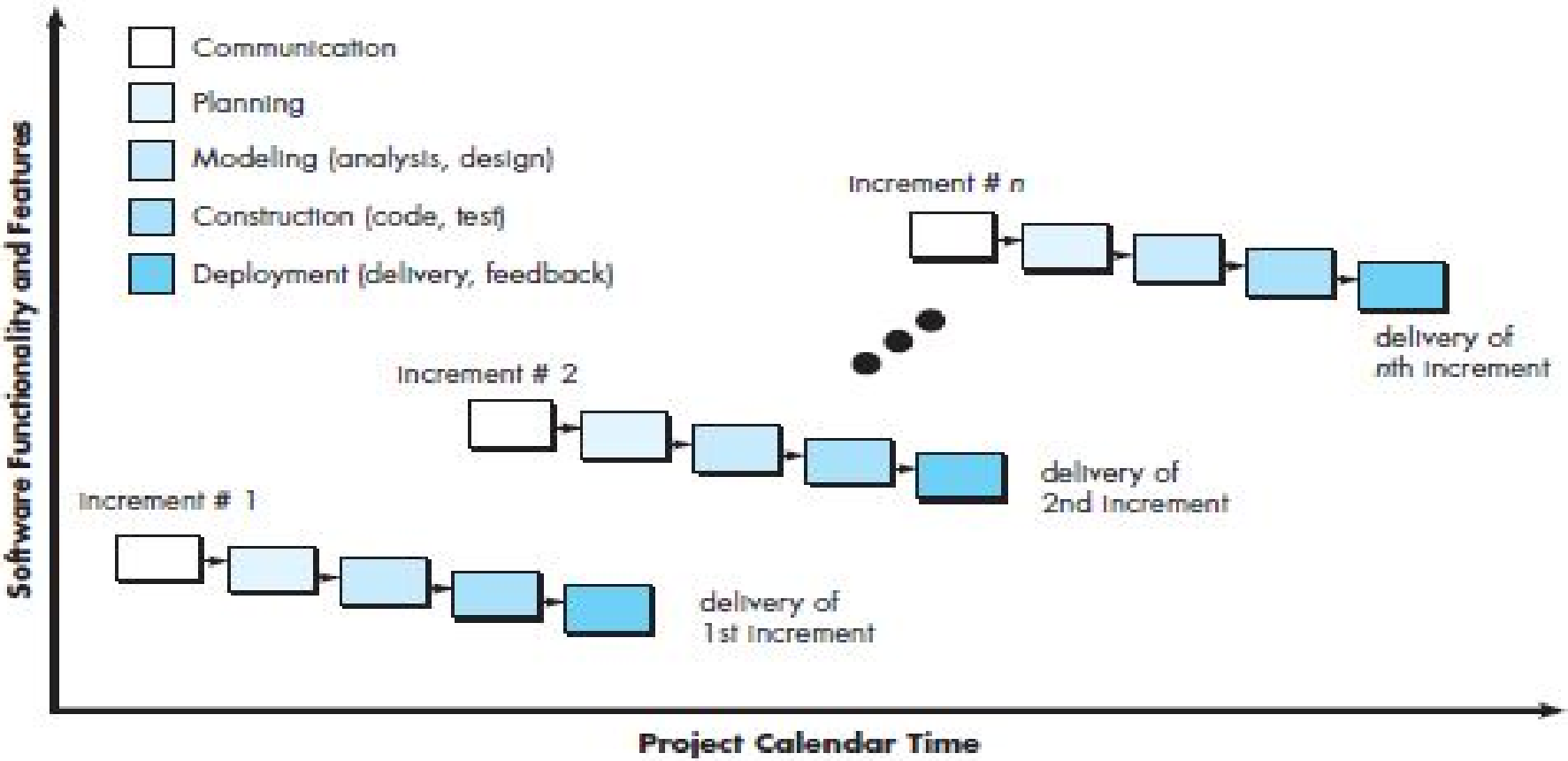

Figure 3. The incremental model

Incremental Process Models

- For example, word-processing software developed using the incremental paradigm might deliver basic fi le management, editing, and document production functions in the first increment
- More sophisticated editing and document production capabilities in the second increment
- Spelling and grammar checking in the third increment
- Advanced page layout capability in the fourth increment.

Incremental Process Models

- When an incremental model is used, the first increment is often a core product. That is, basic requirements are addressed but many supplementary features (some known, others unknown) remain undelivered
- The core product is used by the customer (or undergoes detailed evaluation).
- As a result of use and/ or evaluation, a plan is developed for the next increment. The plan addresses the modification of the core product to better meet the needs of the customer and the delivery of additional features and functionality.
- This process is repeated following the delivery of each increment, until the complete product is produced.

Incremental Process Models

- Incremental development is based on the idea of developing an initial implementation, getting feedback from users and others
- Evolving the software through several versions until the required system has been developed
- Specification, development, and validation activities are interleaved rather than separate, with rapid feedback across activities
- Incremental development in some form is now the most common approach for the development of application systems and software products.

The Incremental Model

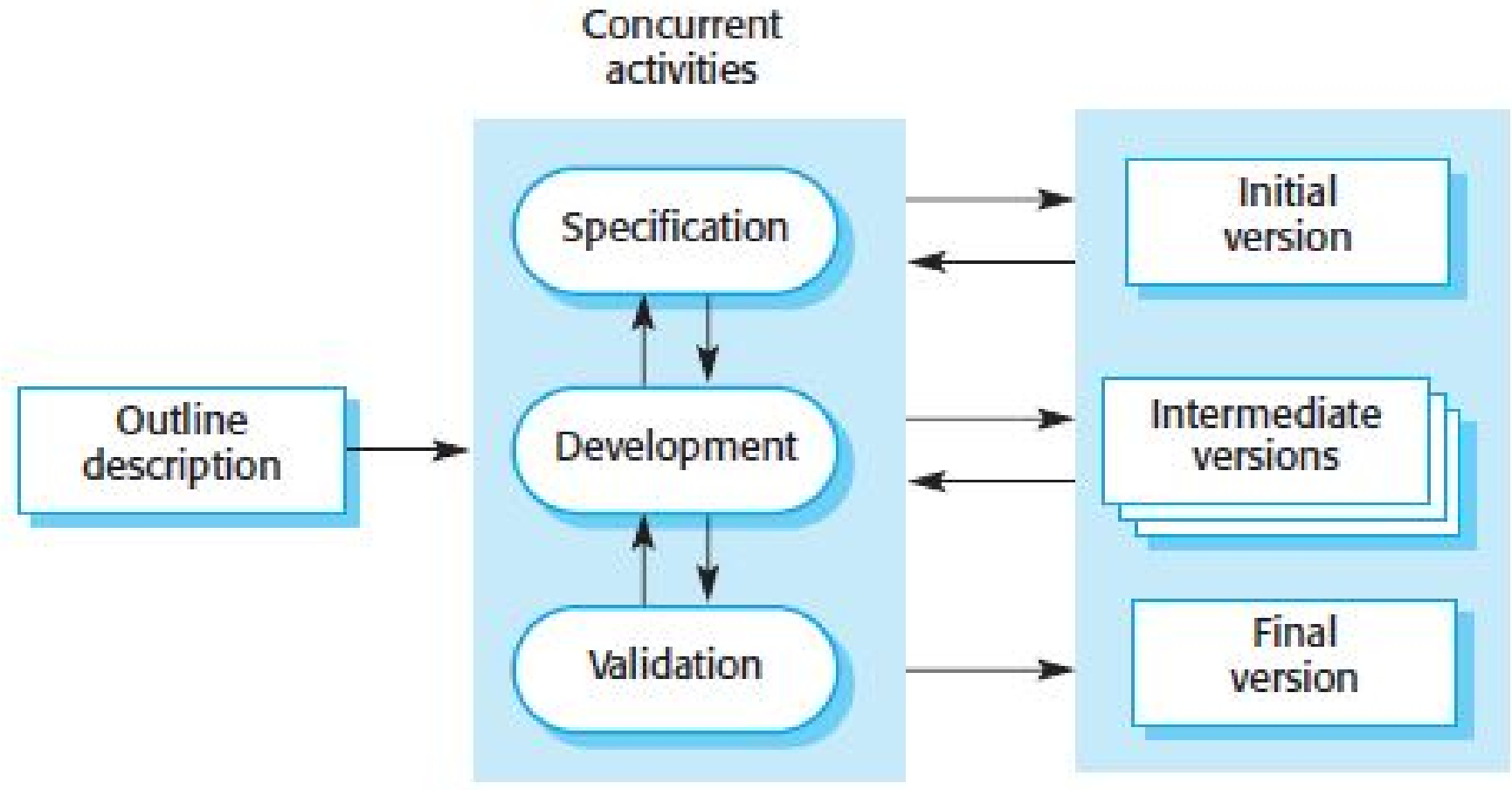

Figure 4. The incremental model

Incremental Process Models

- Incremental development has three major advantages over the waterfall model:
- The cost of implementing requirements changes is reduced. The amount of analysis and documentation that has to be redone is significantly less than is required with the waterfall model.
- It is easier to get customer feedback on the development work that has been done. Customers can comment on demonstrations of the software and see how much has been implemented. Customers find it difficult to judge progress from software design documents.
- Early delivery and deployment of useful software to the customer is possible, even if all of the functionality has not been included. Customers are able to use and gain value from the software earlier than is possible with a waterfall process.

Evolutionary Process Models

- Software, like all complex systems, evolves over a period of time. Business and product requirements often change as development proceeds, making a straight line path to an end product unrealistic
- Tight market deadlines make completion of a comprehensive software product impossible, but a limited version must be introduced to meet competitive or business pressure
- A set of core product or system requirements is well understood, but the details of product or system extensions have yet to be defined.

Evolutionary Process Models

- In these and similar situations, you need a process model that has been explicitly designed to accommodate a product that grows and changes.

- Evolutionary models are iterative. They are characterized in a manner that enables you to develop increasingly more complete versions of the software

Evolutionary Process Models

- Two common evolutionary process models.
- Prototyping Model
- The Spiral Model

AGILE Development

- What is Agile Development?
- Agile software engineering combines a philosophy and a set of development guidelines.
- The philosophy encourages customer satisfaction and early incremental delivery of software; small, highly motivated project teams; informal methods; minimal software engineering work products; and overall development simplicity.
- The development guidelines stress delivery over analysis and design (although these activities are not discouraged), and active and continuous communication between developers and customers.

AGILE Development

- Who does it
- Software engineers and other project stakeholders (managers, customers, end users) work together on an agile team—a team that is self-organizing and in control of its own destiny.
- An agile team fosters communication and collaboration among all who serve on it.

AGILE Development

- Why is it important
- The modern business environment that spawns computer-based systems and software products is fast-paced and ever-changing.
- Agile software engineering represents a reasonable alternative to conventional software engineering for certain classes of software and certain types of software projects.
- It has been demonstrated to deliver successful systems quickly.

AGILE Development

- What are the steps?
- Agile development might best be termed "software engineering lite."
- The basic framework activities— communication, planning, modeling, construction, and deployment.
- But they morph into a minimal task set that pushes the project team toward construction and delivery (some would argue that this is done at the expense of problem analysis and solution design).

Other Agile Process Models

- The other agile process models are as follows
- Scrum
- Dynamic Systems Development Method (DSSM)
- Agile Modeling (AM)
- Agile Unified Process (AUP).

Scrum

- Scrum (the name is derived from an activity that occurs during a rugby match) is an agile software development method that was conceived by Jeff Sutherland and his development team
- Scrum principles are consistent with the agile manifesto and are used to guide development activities within a process that incorporates the following framework activities:
- Requirements
- Analysis
- Evolution, and
- Delivery.

Scrum process flow

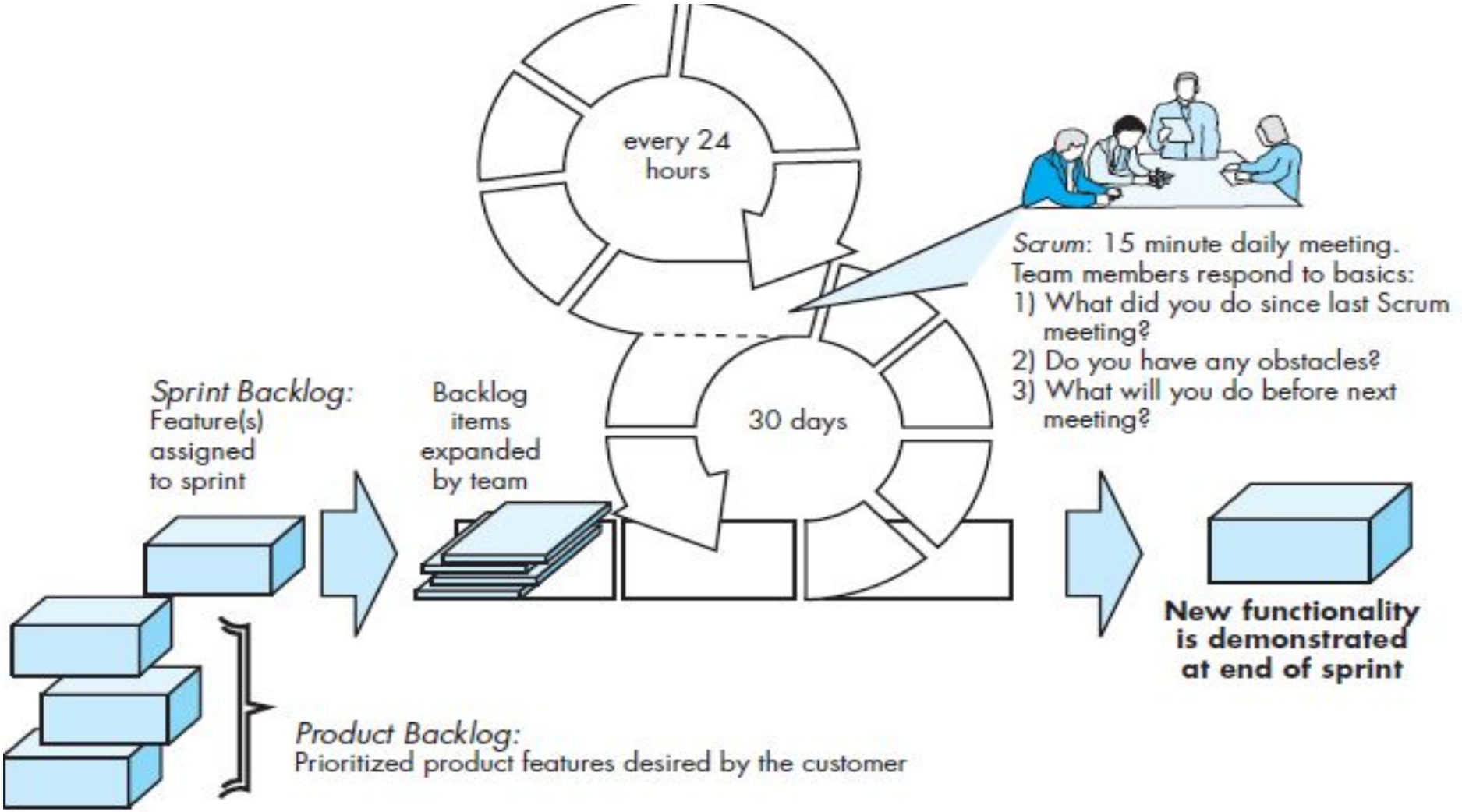

Figure 5. Scrum process flow

Scrum

- Within each framework activity, work tasks occur within a process pattern called a sprint.
- The work conducted within a sprint (the number of sprints required for each framework activity will vary depending on product complexity and size) is adapted to the problem at hand and is defined and often modified in real time by the Scrum team.
- Scrum emphasizes the use of a set of software process patterns that have proven effective for projects with tight timelines, changing requirements, and business criticality.

Scrum

- Each of these process patterns defines a set of development activities:
- **Backlog** —a prioritized list of project requirements or features that provide business value for the customer.
- Items can be added to the backlog at any time (this is how changes are introduced).
- The product manager assesses the backlog and updates priorities as required.

Scrum

- **Sprints** —consist of work units that are required to achieve a requirement defined in the backlog that must be fit into a predefined time-box (typically 30 days).
- Changes (e.g., backlog work items) are not introduced during the sprint.
- Hence, the sprint allows team members to work in a short-term, but stable environment.

Scrum

- **Scrum meetings** —are short (typically 15-minute) meetings held daily by the Scrum team.
- Three key questions are asked and answered by all team members
- What did you do since the last team meeting?
- What obstacles are you encountering?
- What do you plan to accomplish by the next team meeting?

Scrum

- A team leader, called a Scrum master, leads the meeting and assesses the responses from each person.
- The Scrum meeting helps the team to uncover potential problems as early as possible.
- Also, these daily meetings lead to “knowledge socialization” and thereby promote a self-organizing team structure.

Scrum

- **Demos** —deliver the software increment to the customer so that functionality that has been implemented can be demonstrated and evaluated by the customer.
- It is important to note that the demo may not contain all planned functionality, but rather those functions that can be delivered within the time-box that was established.

Book

- "Software Engineering" by Ian Sommerville