Department Of Computer Science

Lecture 02

Process Models

Objectives

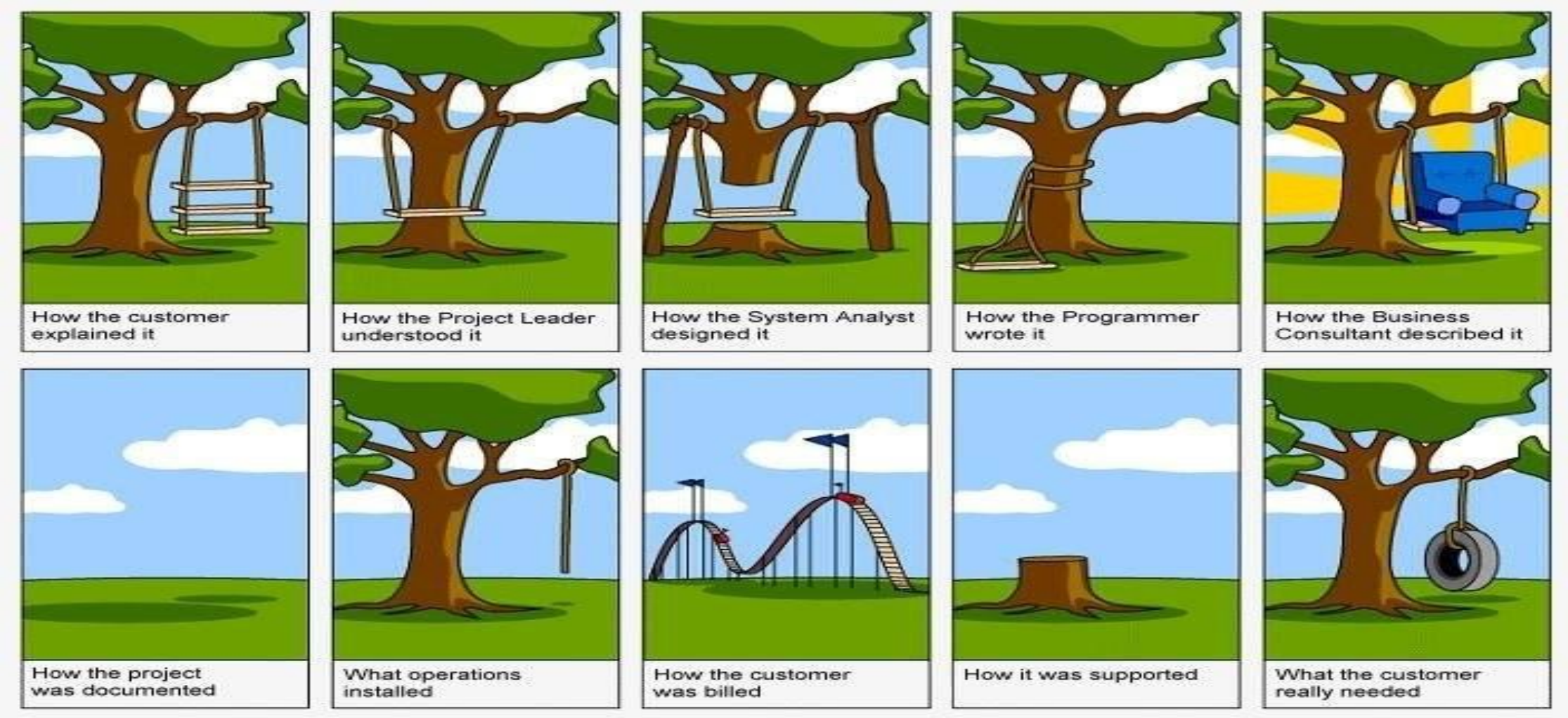
- To introduce software process models
- To describe three generic process models and when they may be used
- To describe outline process models for requirements engineering, software development, testing and evolution

Topics Covered

- Steps in Software Development Process
- Software process Models
- **Sequential Process Model:**
  - Waterfall Model
  - Incremental Model
  - V-Model
- **Evolutionary Development:**
  - Prototyping
  - The Spiral Model
- **Agile Process Model:**
  - RUP
  - RAD
- Process iteration
- Agile SDLC Models

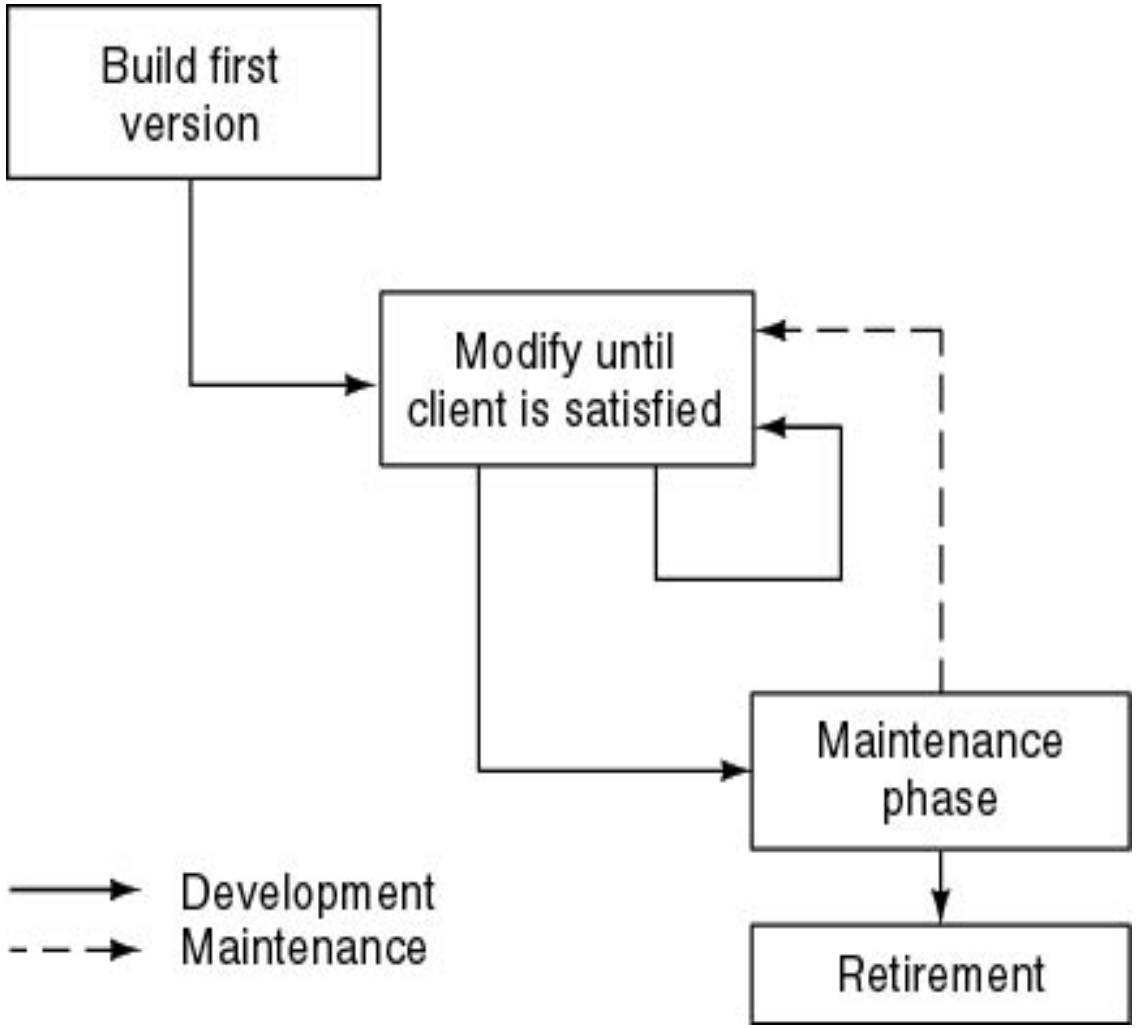
- **Build and Fix Model**

Importance of Software Engineering



Build and Fix Model

- Problems
  - No specifications
  - No design
- Totally unsatisfactory
- Need a life-cycle model
  - Phases
  - Milestones



What is a Software Process?

- Software Process: "The systematic approach that is used in software engineering is sometimes called a software process."

- "A set of activities whose goal is the development or evolution of software".

- Generic activities in all software processes are:
  - **Software specification**
  - **Software development:**
  - **Software validation:**
  - **Software evolution:**

Steps in Software Development Process

What is a Software Process?

- "**A set of activities whose goal is the development or evolution of software**".

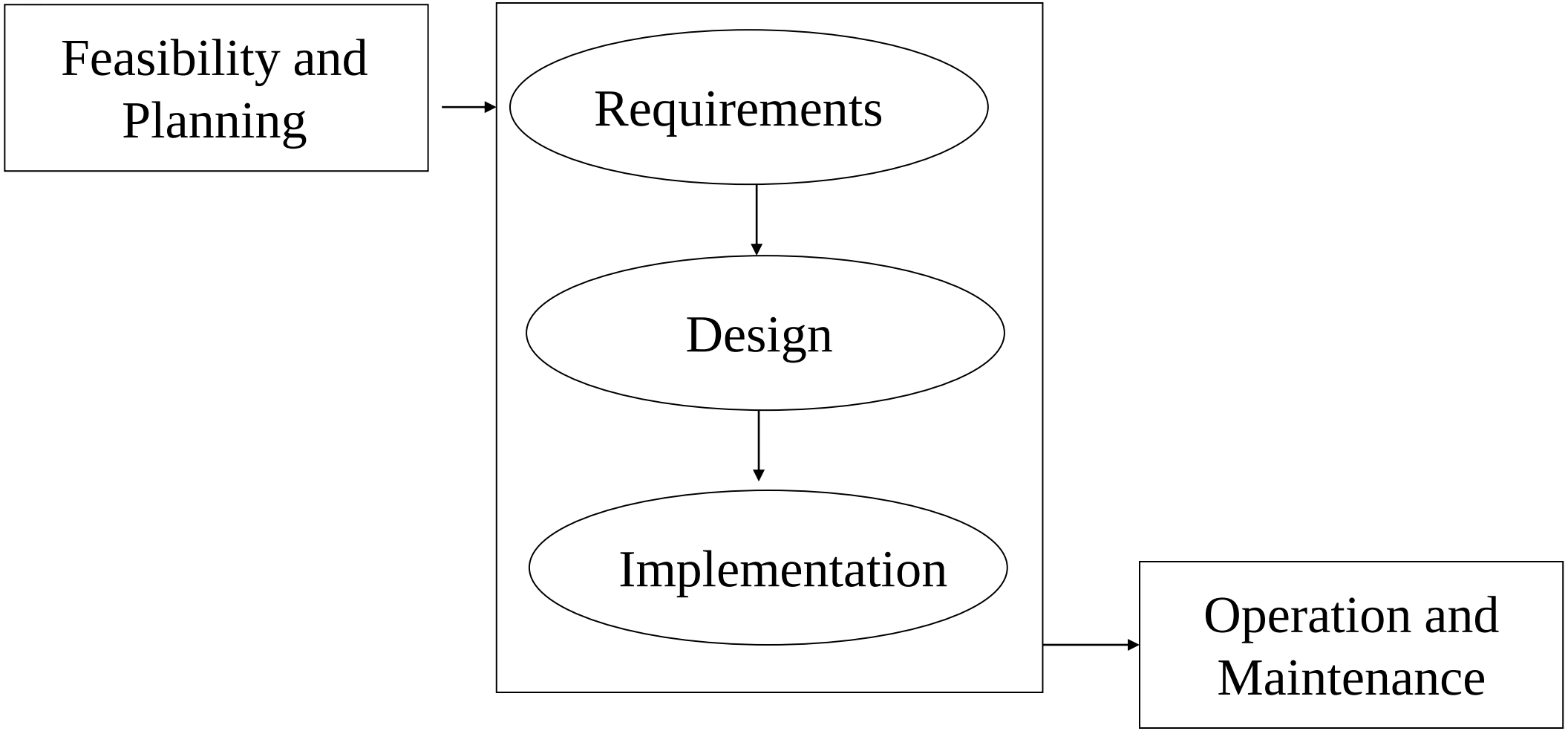
- Generic activities in all software processes are:

  - **Specification**
  - **Development**
  - **Validation**
  - **Evolution**

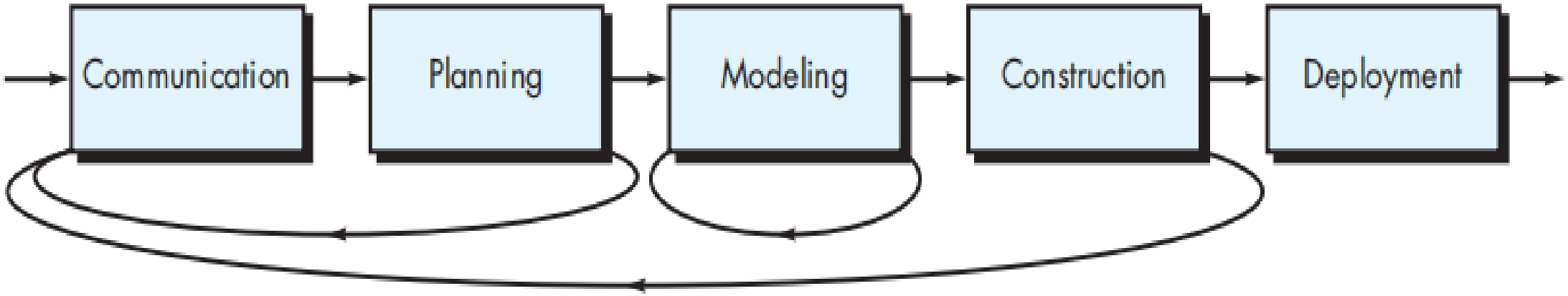
Activities in Software Process?

- Generic activities in all software processes are:
  - **Software Specification** - what the system should do and its development constraints
    - where customers and engineers define the software that is to be produced and the constraints on its operation
  - **Software Development** - production of the software system
    - where the software is designed and programmed
  - **Software Validation** - checking that the software is what the customer wants
    - where the software is checked to ensure that it is what the customer requires.
  - **Software Evolution** - changing the software in response to changing demands
    - where the software is modified to reflect changing customer and market requirements.

The Software Process (Simplified)



Iterative Process Flow



(b) Iterative process flow

Incremental Development

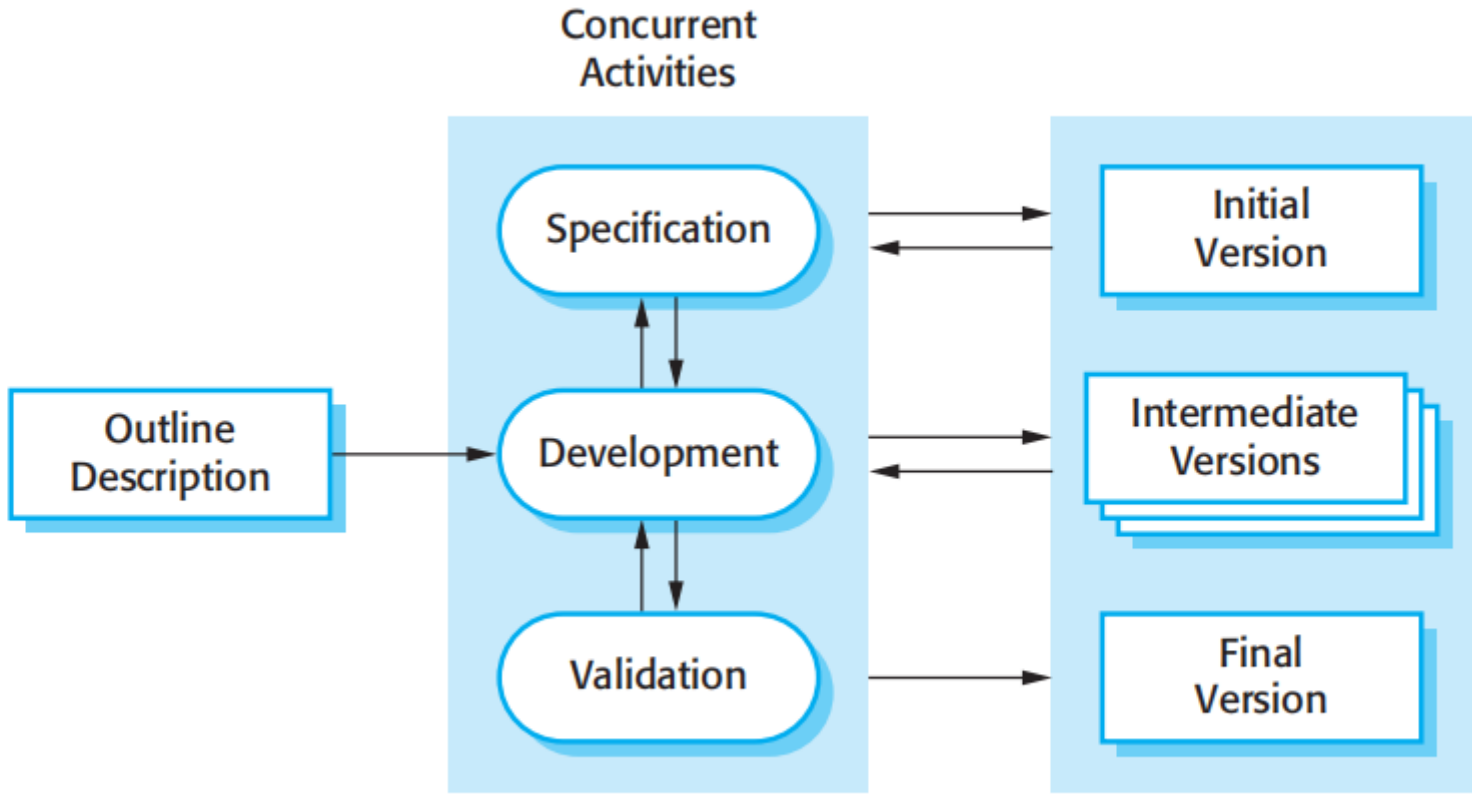
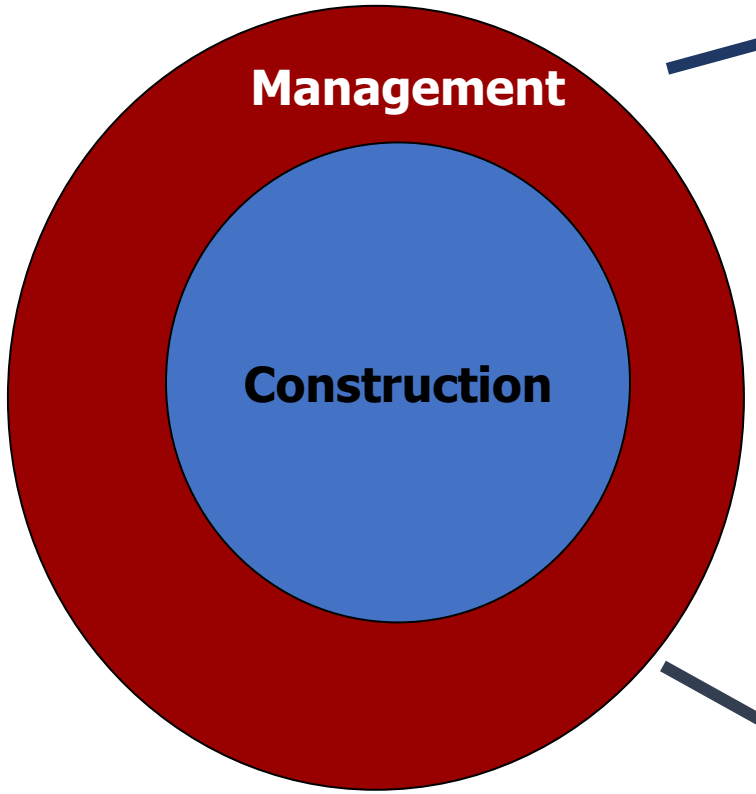


Figure 2.2 Incremental development

Software Development Activities

- Project Management
- Software Quality Assurance
- Software Configuration Management
- Software Integration
- Requirement Engineering
- Design
- Coding
- Testing
- Rest of the activities

Software Development Activities



- Project planning and Management
- Configuration Management
- Quality Assurance
- Installation and Training etc.

- Requirements
- Design
- Coding
- Testing
- Maintenance etc.

Software Process

- **Fundamental Assumption:**
  - Good processes lead to good software
  - Good processes reduce risk
  - Good processes enhance visibility
  - Good processes enable teamwork

Variety of Software Processes

- Software products are very varied. Therefore, there is no standard process for all software engineering projects
- BUT successful software development projects always need to address similar issues. This creates a number of process steps that should be part of all software projects.

Basic Process Steps in all Software Development

- **1-** Feasibility and Planning

- **2-** Requirements
- **3-** System and Program design
- **4-** Implementation
- **5-** Acceptance and Release

These steps may be repeated many times during the development cycle.

- **6-** Operation and Maintenance

Process Step-1: Feasibility

- A feasibility study precedes the decision to begin a project.
- What is the scope of the proposed project?
- Is the project technically feasible?
- What are the projected benefits?
- What are the costs, Numerable?
- Are the resources available?
- What are the risks and how can they be managed?
- A feasibility study leads to a decision: go or no--go.

Process Step-2: Requirements

- Requirements define the function of the system from the client's viewpoint.
- The requirements establish the system's functionality, constraints, and goals by consultation with the client, customers, and users.
- They must be developed in a manner that is understandable by both the client and the development staff.

Process Step-2: Requirements

- **This step is occasionally divided into:**
  - **Requirements Gathering**
  - **Requirements Analysis**
  - **Requirements Definition**
  - **Requirements Specification**
- The requirements may be developed in a limited manner , or may emerge incrementally.
- Failure to agree on the requirements and define them adequately is one of the biggest cause of software projects failing.

Process Step-3: System Design

- Design describes the system from the software developers' viewpoint
- **System design:**
- Establish a system architecture, both hardware and software, that matches the requirements
- **Program design:**
- Represent the software functions in a form that can be transformed into one or more executable programs
- Preliminary user testing is often carried out as part of the design step.
- Models are used to represent the requirements, system architecture, and program design.

Process Step-4: Implementation

- **Implementation (coding)**
- The software design is realized as a set of programs or program units.
- These software components may be written by the development team, acquired from elsewhere, or modified from existing components.

- **Program testing**
- Program testing by the development staff is an integral part of implementation.
- Individual components are tested against the design.
- The components are integrated and tested against the design as a complete system.

Process Step-5: Acceptance and Release

- **Acceptance testing**
- The system is tested against the requirements by the client, often with selected customers and users.

- **Delivery and release**
- After successful acceptance testing, the system is delivered to the client and released into production or marketed to customers.

Process Step-6: Operation and Maintenance

- Operation:

  The system is put into practical use.

- Maintenance:

  Errors and problems are identified and fixed.

- Evolution:

  The system evolves over time as requirements change, to add new functions, or adapt to a changing technical environment.

- Phase out:

  The system is withdrawn from service.

- **This is called the Software Life Cycle**

Quality Control Steps in all Software Development

- Validating the requirements
- Validating the system and program design
- Usability testing
- Program testing
- Acceptance testing

- Bug fixing and maintenance

Some of these steps will be repeated many times during the life of the system

Sequence of Process Steps

- Every software project will include these basic processes, in some shape or form, but:
  - **The steps may be formal or informal**
  - **The steps may be carried out in various sequences**

Sequence of Process Steps

- **Major alternatives**
- We will look at three categories of software development processes:
- **Sequential:**
- As far as possible, complete each process step before beginning the next. Waterfall model.
- **Iterative:**
- Go quickly through all process steps to create a rough system, then repeat them to improve the system. Iterative refinement.
- **Incremental:**
- A variant of iterative refinement in which small increments of software are placed in production (sprints). Agile development.

Software Process Models

Heavyweight and Lightweight Software Development

Heavyweight and Lightweight Software Development

- In a heavyweight process, the development team works through the process steps slowly and systematically, with the aim of fully completing each process step and deliverables for that step that will need minimal changes and revision.
- **Example:** Modified Waterfall Model

- In a lightweight process, the development team releases working software in small increments, and develops the plans incrementally, based on experience. Each increment includes all the process steps. There is an expectation that changes will be made based on experience.
- **Example:** Agile Software Development

Deliverables

- **Deliverables**
- A deliverable is some work product that is delivered to the client.
- In a heavyweight process, each process step creates a deliverable, usually documentation,

  e.g., a requirements specification.
- In a lightweight process, the deliverables are incremental working code, with minimal supporting documentation.

- **Software Process Models**

Software Life-Cycle Models

- Life-cycle model (formerly, process model)

- SDLC Model: A framework that describes the activities performed at each stage of a software development project.

What is a Software Process Model?

- *"A simplified representation of a software process, presented from a specific perspective"*.

- **Generic process models:**
  - Waterfall
  - Evolutionary development
  - Component-based software engineering
  - Iterative development

Process Models

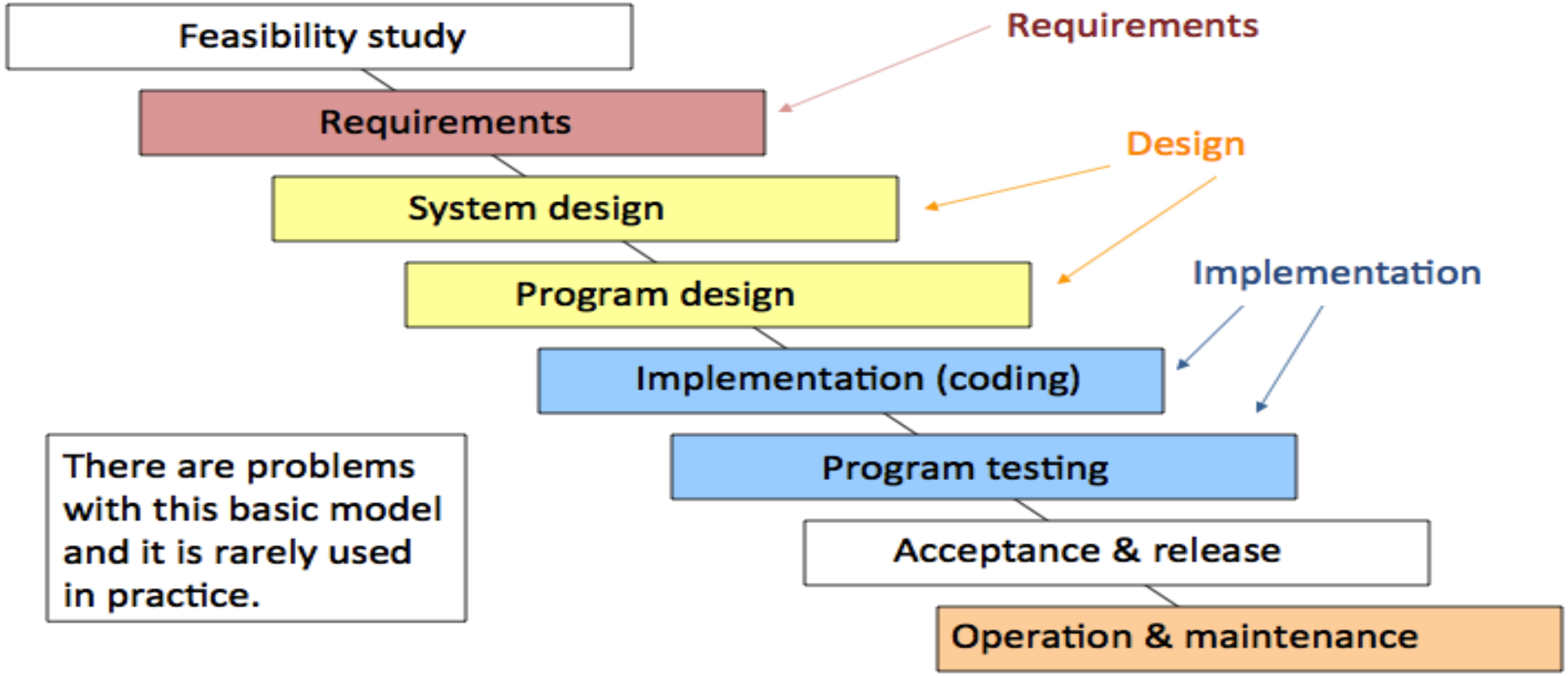
- **Perspective Process Models**
  - The Waterfall Model
  - Incremental Process Models
  - V-Shaped Model
- **Evolutionary Process Models**
    - Prototyping
    - The Spiral Model
  - Concurrent Models
- **Specialized Process Models**
  - Component-Based Development
  - The Formal Methods Model
  - Aspect-Oriented Software Development

Generic Software Process Models

- Sequential Model: The waterfall model
  - Separate and distinct phases of specification and development.
- Evolutionary development
  - Specification, development and validation are interleaved.
- Component-based software engineering
  - The system is assembled from existing components.

- **Sequential Models: Waterfall Model**

➢ Sequential Model: Waterfall Model

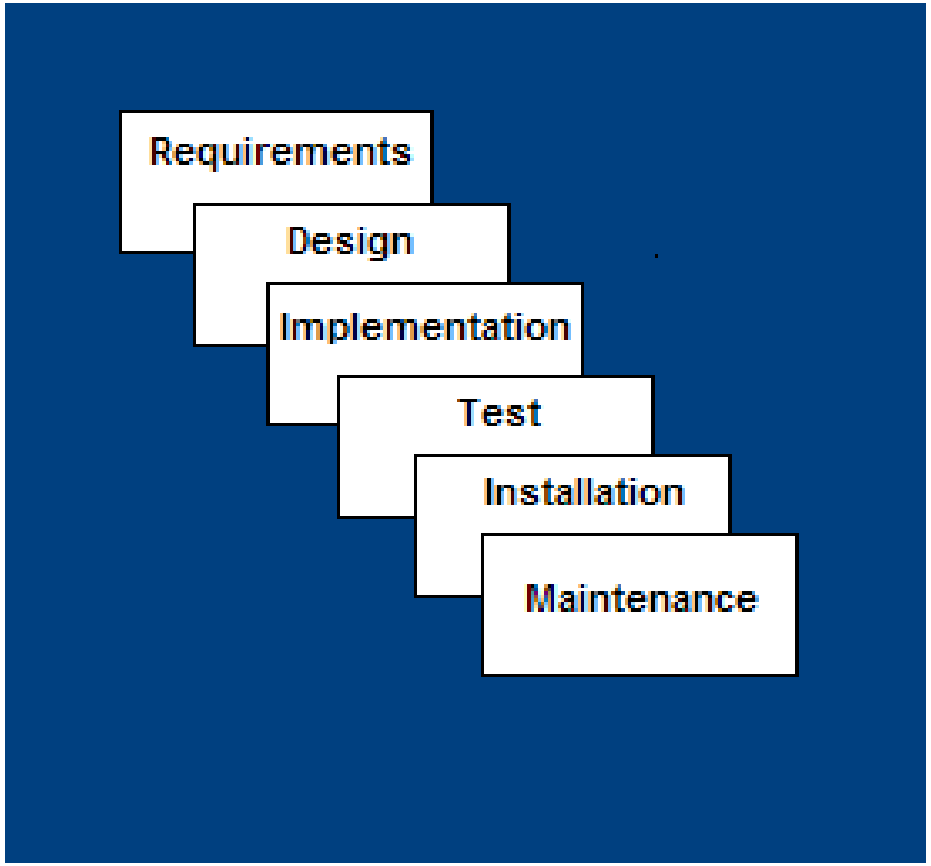


Waterfall Model Phases

- **Requirements analysis and definition**
- **System and software design**
- **Implementation and unit testing**
- **Integration and system testing**
- **Operation and maintenance**

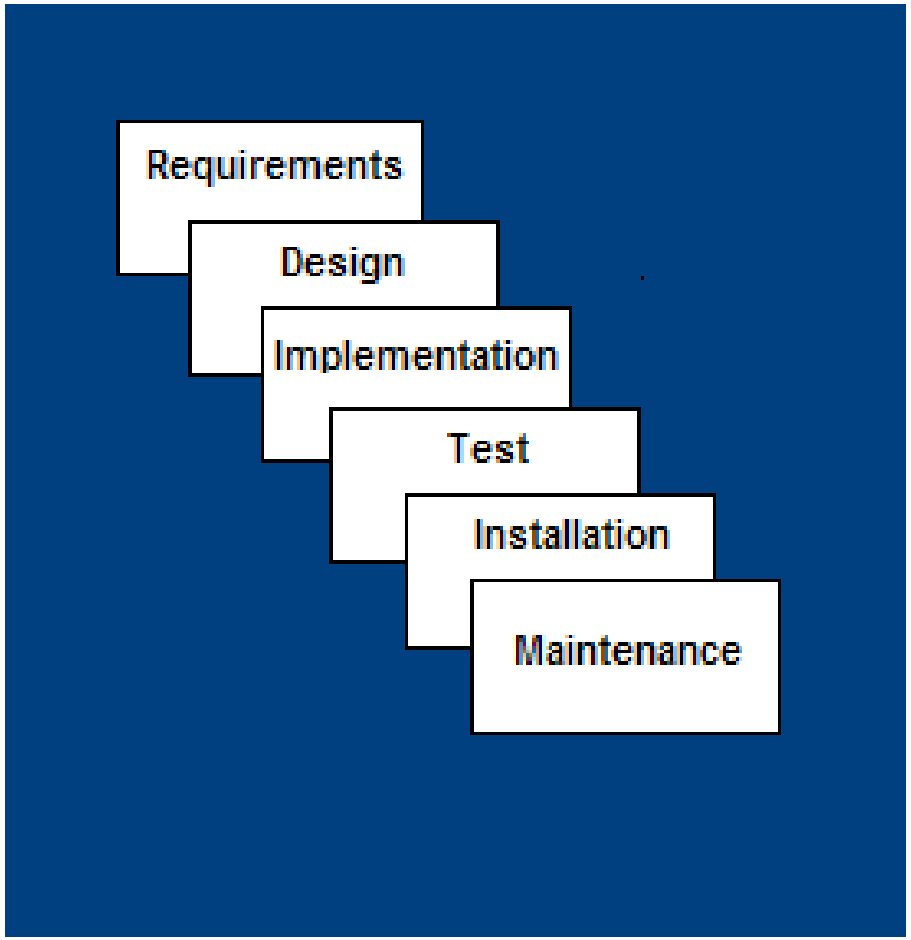
- **The main drawback** of the waterfall model is the difficulty of accommodating change after the process is underway. One phase has to be complete before moving onto the next phase.

Waterfall Model



- **Requirements** – defines needed information, function, behavior, performance and interfaces.
- **Design** – data structures, software architecture, interface representations, algorithmic details.
- **Implementation** – source code, database, user documentation, testing.

Waterfall Model



- **Test** – check if all code modules work together and if the system as a whole behaves as per the specifications.
- **Installation** – deployment of system, user-training.
- **Maintenance** – bug fixes, added functionality (an on-going process).

The Water Fall Model

Advantages

- Easy to understand, easy to use
- Provides structure to inexperienced staff
- Milestones are well understood
- Sets requirements stability
- Good for management control (plan, staff, track)

Disadvantages

- All requirements must be known upfront
- Difficult for the customer to state all requirements explicitly
- Integration is one big bang at the end
- Little opportunity for customer to preview the system (until it may be too late)

Waterfall Strengths

- Easy to understand, easy to use
- Provides structure to inexperienced staff
- Milestones are well understood
- Sets requirements stability
- Good for management control (plan, staff, track)

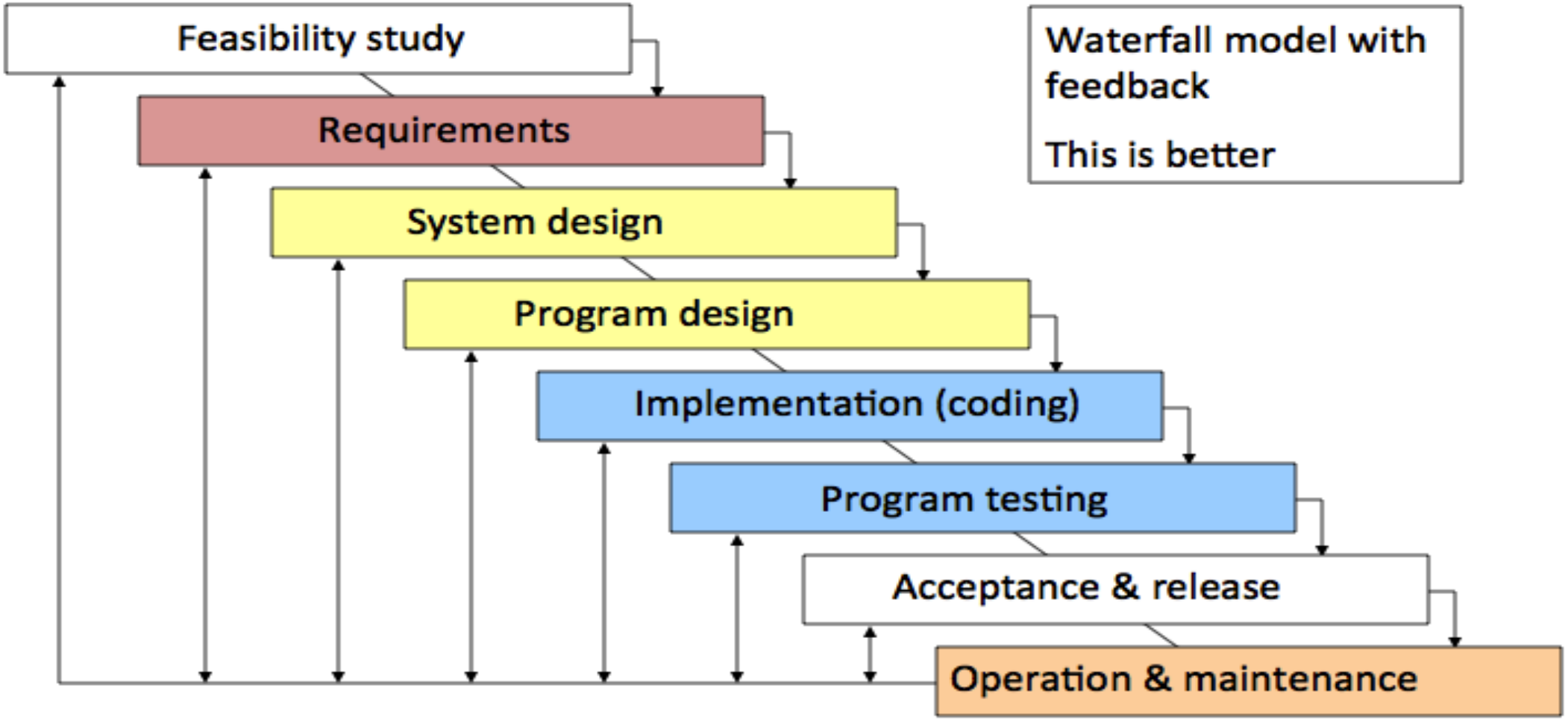
Waterfall Deficiencies

- All requirements must be known upfront
- Deliverables created for each phase are considered frozen – inhibits flexibility
- Does not reflect problem-solving nature of software development – iterations of phases
- Integration is one big bang at the end
- Little opportunity for customer to preview the system (until it may be too late)

When to use Water Fall Model

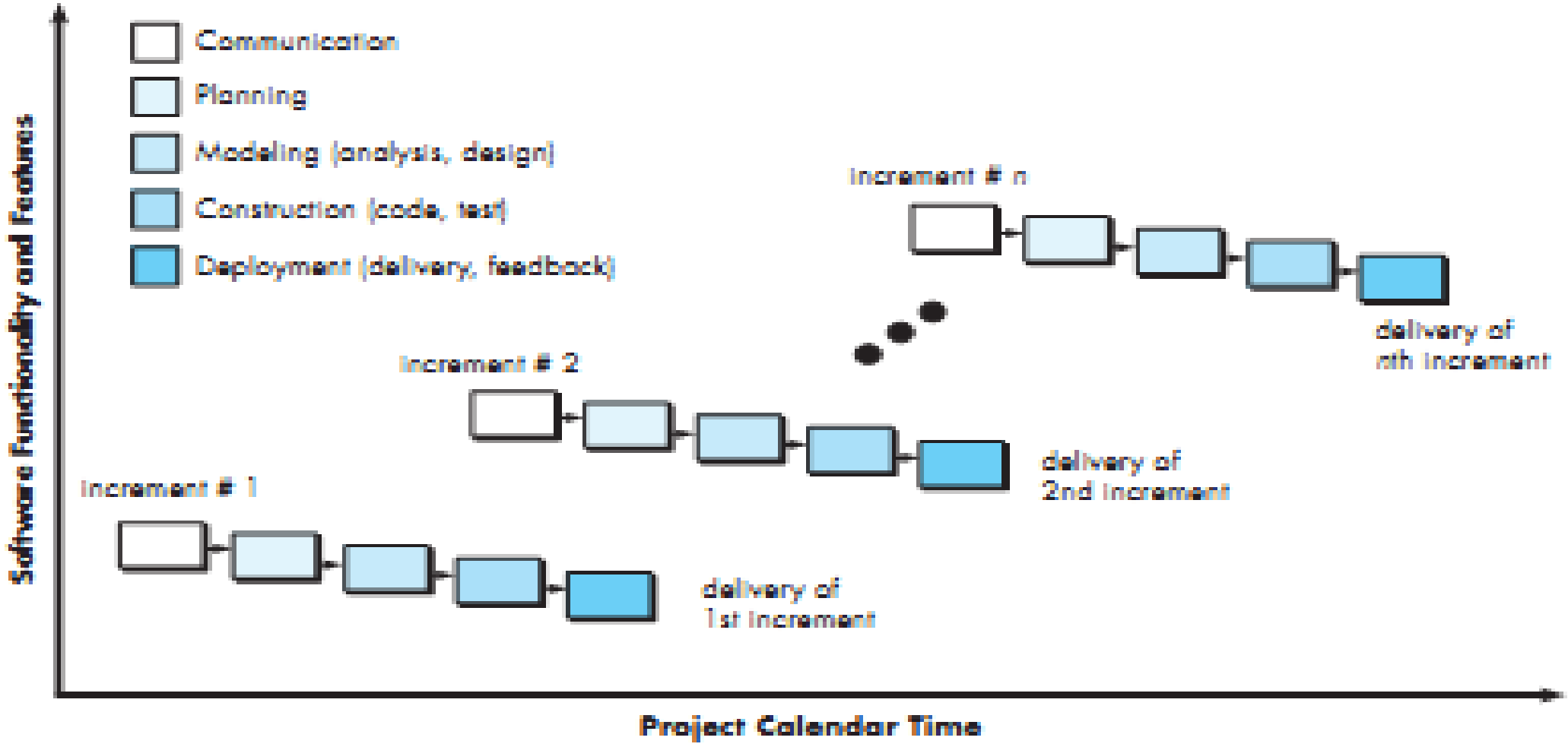
- **Requirements are** very well known
- **When it is possible to produce** a stable design
  - E.g. a new version of an existing product
  - E.g. porting an existing product to a new platform.

Modified Waterfall Model



Sequential Models: Incremental Process Model

Incremental Process Model



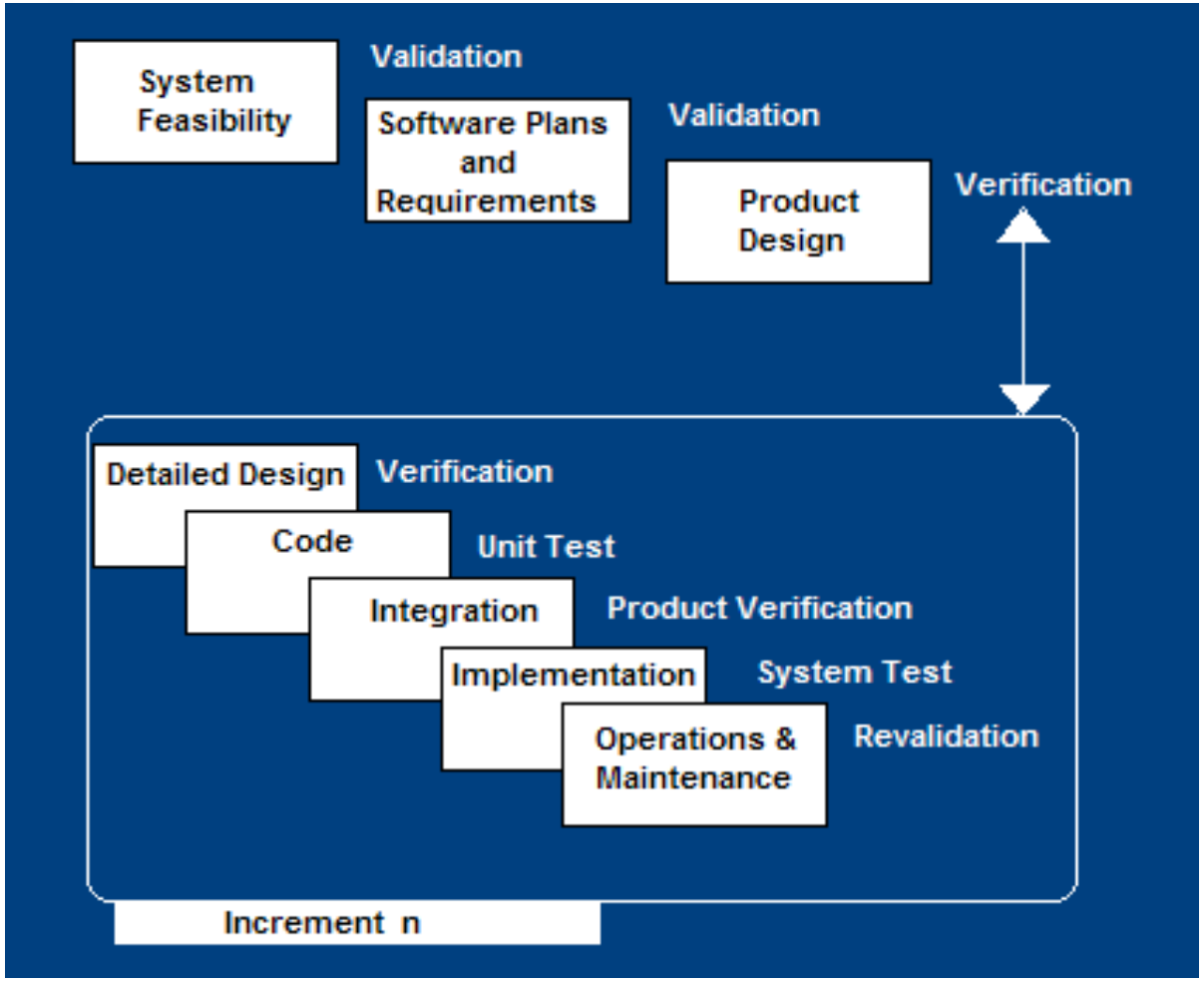
Incremental Process Model

- Combines elements of the **waterfall model applied in an iterative fashion.**
- **Incremental applies linear sequences** as the calendar time progress.
- **Limited software functionalities** and expands functionalities in later software release.

Incremental Model

- **Each** linear sequence produces a deliverable increment of the software
  - E.g. Word processing software
- **First increment** is often a core product
- **Basic requirements are** addressed, but many supplementary remains undelivered.
- **Customers evaluate** this core product, and the next increment is planned based on suggestions and next set of features
- **The plan addresses** the modification of core product and delivery of additional features and functionalities.
- **Process is repeated** until complete product is produced.

Incremental Model



- Construct a partial implementation of a total system
- Then slowly add increased functionality
- The incremental model prioritizes requirements of the system and then implements them in groups.
- Each subsequent release of the system adds function to the previous release, until all designed functionality has been implemented.

Incremental Model Strengths

- Develop high-risk or major functions first
- Each release delivers an operational product
- Customer can respond to each build
- Uses "divide and conquer" breakdown of tasks
- Lowers initial delivery cost
- Initial product delivery is faster
- Customers get important functionality early
- Risk of changing requirements is reduced

Incremental Model Weaknesses

- Requires good planning and design
- Requires early definition of a complete and fully functional system to allow for the definition of increments
- Well-defined module interfaces are required (some will be developed long before others)
- Total cost of the complete system is not lower

Incremental Model

Advantages

- **Generates working** software quickly and early during the software life cycle.
- **It is easier to test** and debug during a smaller iteration.
- **Customer can respond** to each built.
- **Lowers initial delivery cost**.
- **Easier to manage risk** because risky pieces are identified and handled during iteration.

Disadvantages

- **Needs good planning** and design.
- **Needs a clear** and complete definition of the whole system before it can be broken down and built incrementally.
- **Total cost** is higher than **waterfall**.

When to use the Incremental Model

- Risk, funding, schedule, program complexity, or need for early realization of benefits.
- Most of the requirements are known up-front but are expected to evolve over time
- A need to get basic functionality to the market early
- On projects which have lengthy development schedules
- On a project with new technology

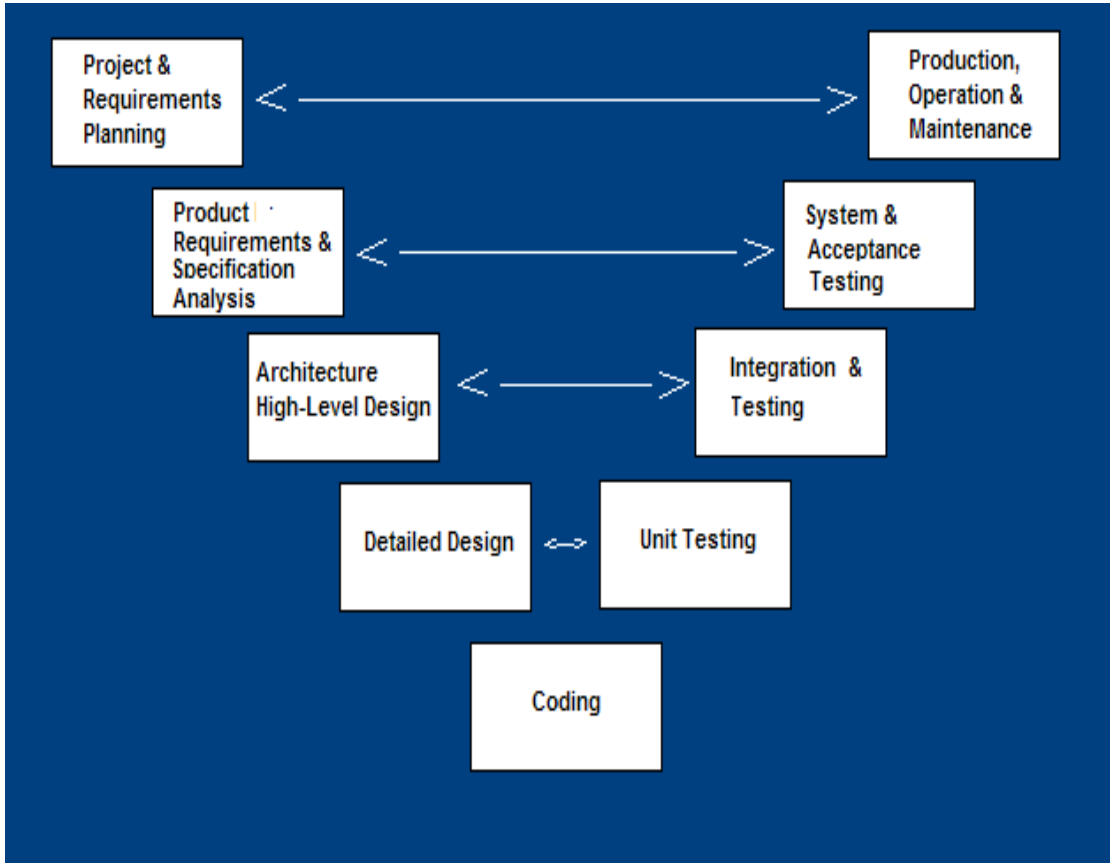
- **Useful when**
  - **Staff is unavailable** for complete implementation and deadline is tight
  - **If core product is well received**, additional staff can implement next increment
  - **Increment can be planned** to manage technical risks
  - **Partial functionalities** can be delivered to end – user without inordinate delay.

Incremental development advantages

- Customer value can be delivered with each increment so system functionality is available earlier.
- Early increments act as a prototype to help elicit requirements for later increments.
- Lower risk of overall project failure.
- The highest priority system services tend to receive the most testing.

- **V-Shaped Model**

V-Shaped Process Model



- A variant of the Waterfall that emphasizes the verification and validation of the product.
- Testing of the product is planned in parallel with a corresponding phase of development

V-Shaped Steps

- Project and Requirements Planning – allocate resources
- Product Requirements and Specification Analysis – complete specification of the software system
- Architecture or High-Level Design – defines how software functions fulfill the design
- Detailed Design – develop algorithms for each architectural component
- Coding – transform algorithms into software
- Unit testing – check that each module acts as expected
- Integration and Testing – check that modules interconnect correctly
- System and acceptance testing – check the entire software system in its environment
- Production, operation and maintenance – provide for enhancement and corrections

V-Shaped Strengths

- Emphasize planning for verification and validation of the product in early stages of product development
- Each deliverable must be testable
- Project management can track progress by milestones
- Easy to use

V-Shaped Weaknesses

- Does not easily handle concurrent events
- Does not handle iterations or phases
- Does not easily handle dynamic changes in requirements

When to use the V-Shaped Model

- Excellent choice for systems requiring high reliability – hospital patient control applications
- All requirements are known up-front
- When design is stable
- Solution and technology are known

- **Evolutionary Development**

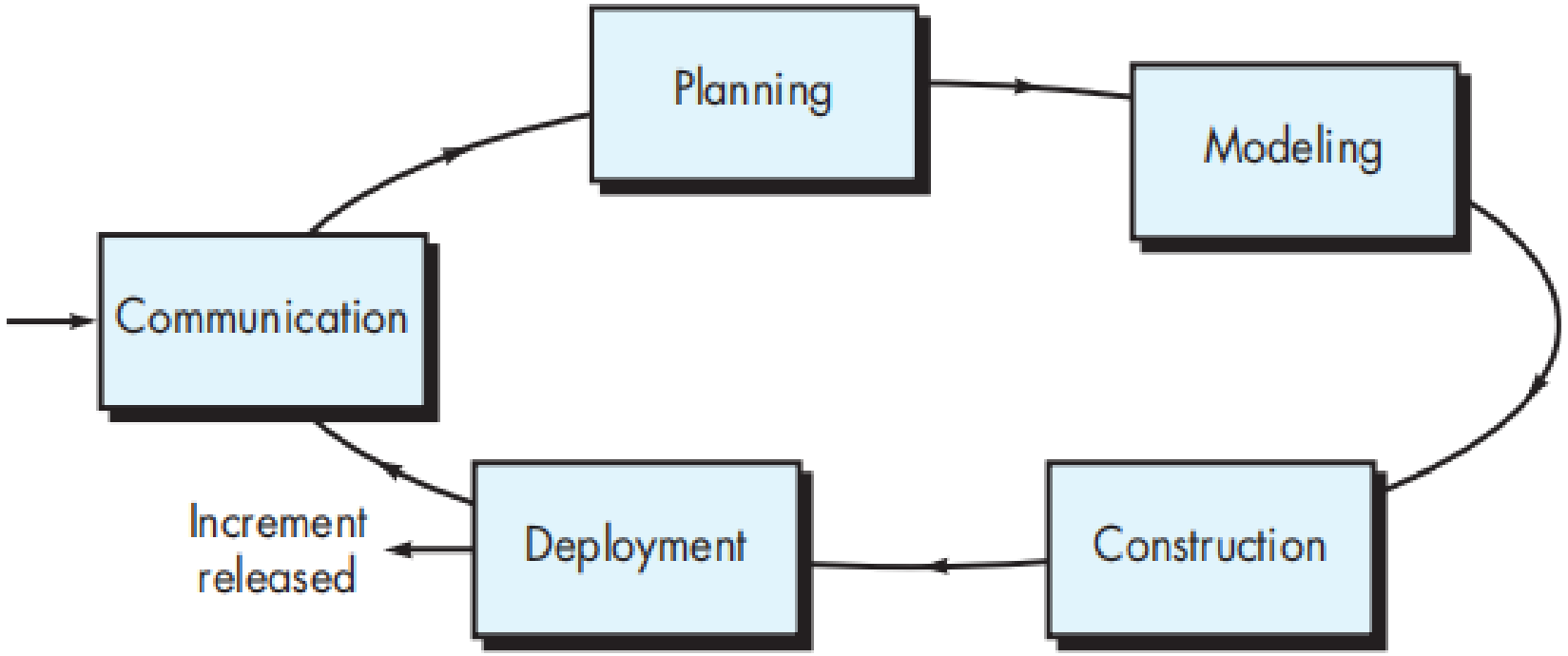
Evolutionary Process Model

- These models are **more suited to object oriented systems**.
- They are **iterative**.
- They **enable the software developer to** develop increasingly more complex versions of the software
- **Like all complex systems, software evolves over a period** of time and hence evolutionary models are more suited to software development.
- **Requirements change** while software gets developed.

Evolutionary development

- **Exploratory development**
  - Objective is to work with customers and to evolve a final system from an initial outline specification. Should start with well-understood requirements and add new features as proposed by the customer.
- **Throw-away prototyping**
  - Objective is to understand the system requirements. Should start with poorly understood requirements to clarify what is really needed.

Evolutionary development



(c) Evolutionary process flow

Evolutionary Development

- **Problems**
  - Lack of process visibility;
  - Systems are often poorly structured;
  - Special skills (e.g. in languages for rapid prototyping) may be required.
- **Applicability**
  - For small or medium-size interactive systems;
  - For parts of large systems (e.g. the user interface);
  - For short-lifetime systems.

Evolutionary Process Models

- **Evolutionary Process Models**
  - Prototyping
  - The Spiral Model

- **Prototyping Model**

Prototyping Model

- Developers build a prototype during the requirements phase
- Prototype is evaluated by end users
- Developer may be unsure about efficiency of algorithm, adaptability of OS, or human-machine interaction
- Users give corrective feedback
- Developers further refine the prototype
- When the user is satisfied, the prototype code is brought up to the standards needed for a final product.
  - **Used as standalone Process model**
  - **Used as a technique implemented in any process model**

Prototyping Steps

- A preliminary project plan is developed
- A partial high-level paper model is created
- The model is source for a partial requirements specification
- A prototype is built with basic and critical functions
- The designer builds
  - the database
  - user interface
  - algorithmic functions
- The designer demonstrates the prototype, the user evaluates for problems and suggests improvements.
- This loop continues until the user is satisfied

Process Of Prototype Development

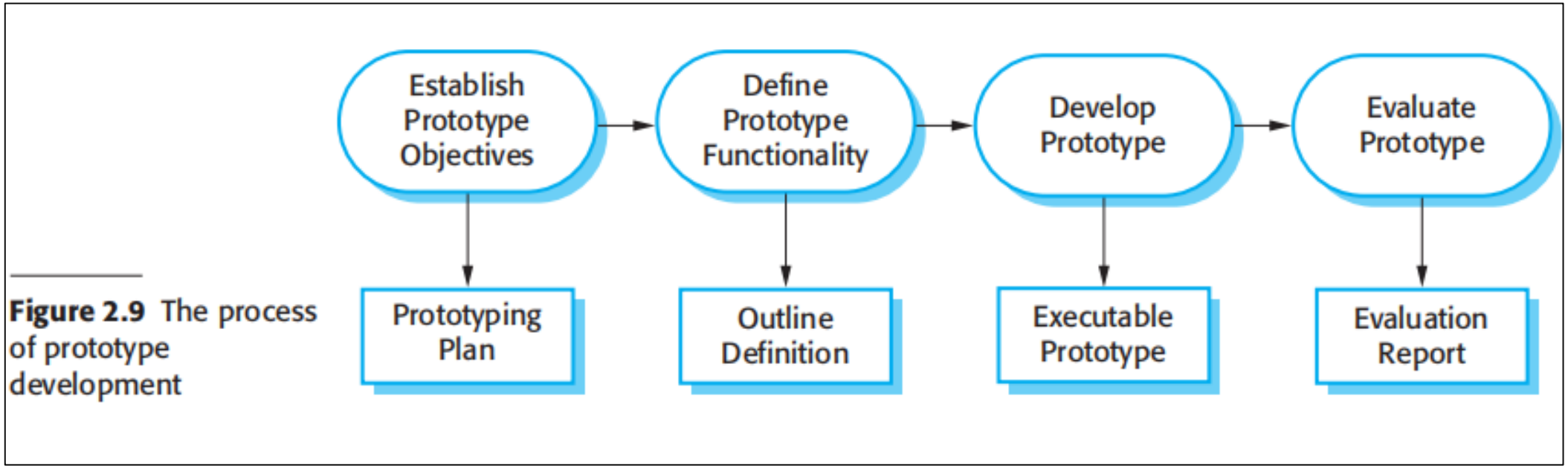
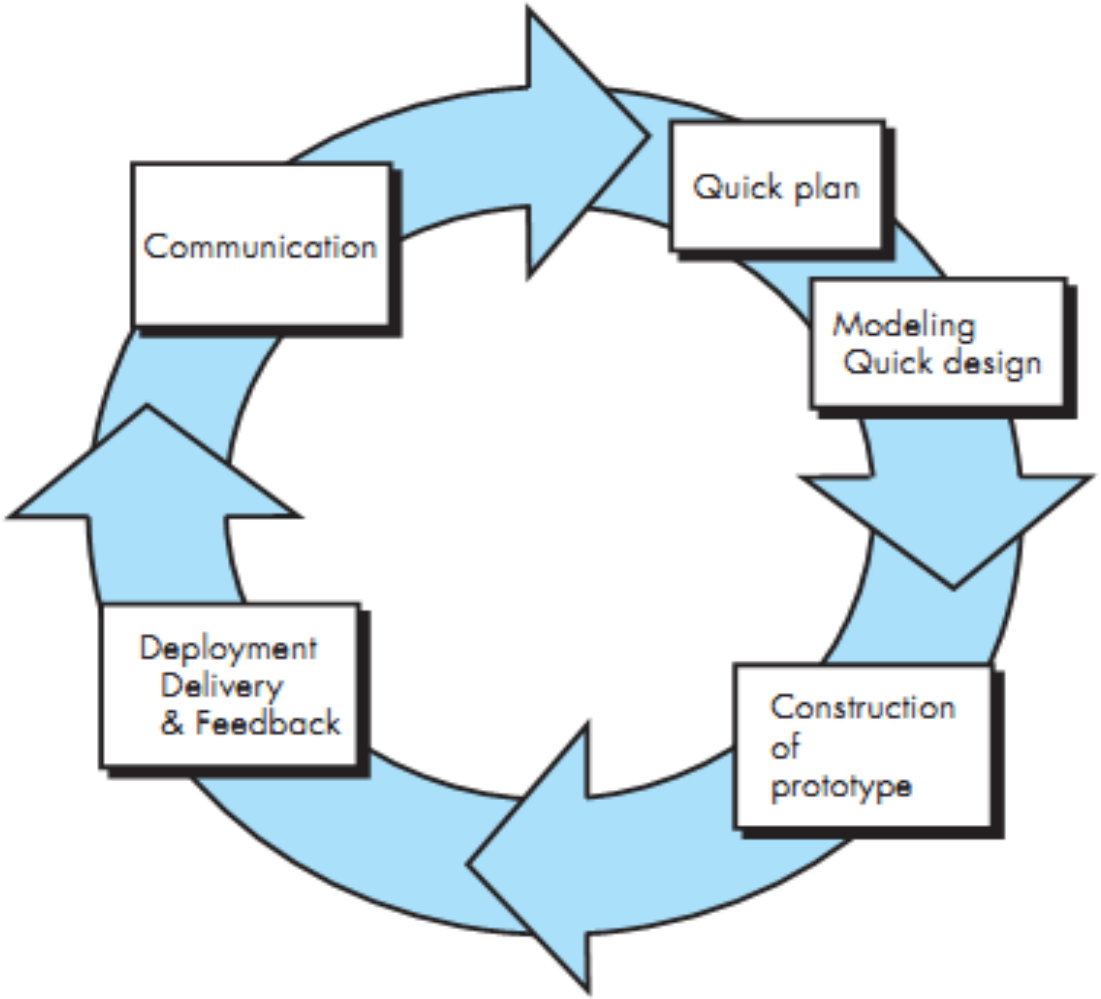


Figure 2.9 The process of prototype development

Prototype Model



Prototyping Strengths

- Customers can "see" the system requirements as they are being gathered
- Developers learn from customers
- A more accurate end product
- Unexpected requirements accommodated
- Allows for flexible design and development
- Steady, visible signs of progress produced
- Interaction with the prototype stimulates awareness of additional needed functionality

Prototyping Weaknesses

- Tendency to abandon structured program development for "code-and-fix" development
- Bad reputation for "quick-and-dirty" methods
- Overall maintainability may be overlooked
- Process may continue forever (scope creep)
- It is a thrown away prototype when the users are confused with it

When to use Prototyping

- Requirements are unstable or have to be clarified
- As the requirements clarification stage of a waterfall model
- Develop user interfaces
- New, original development

- **Spiral Model**

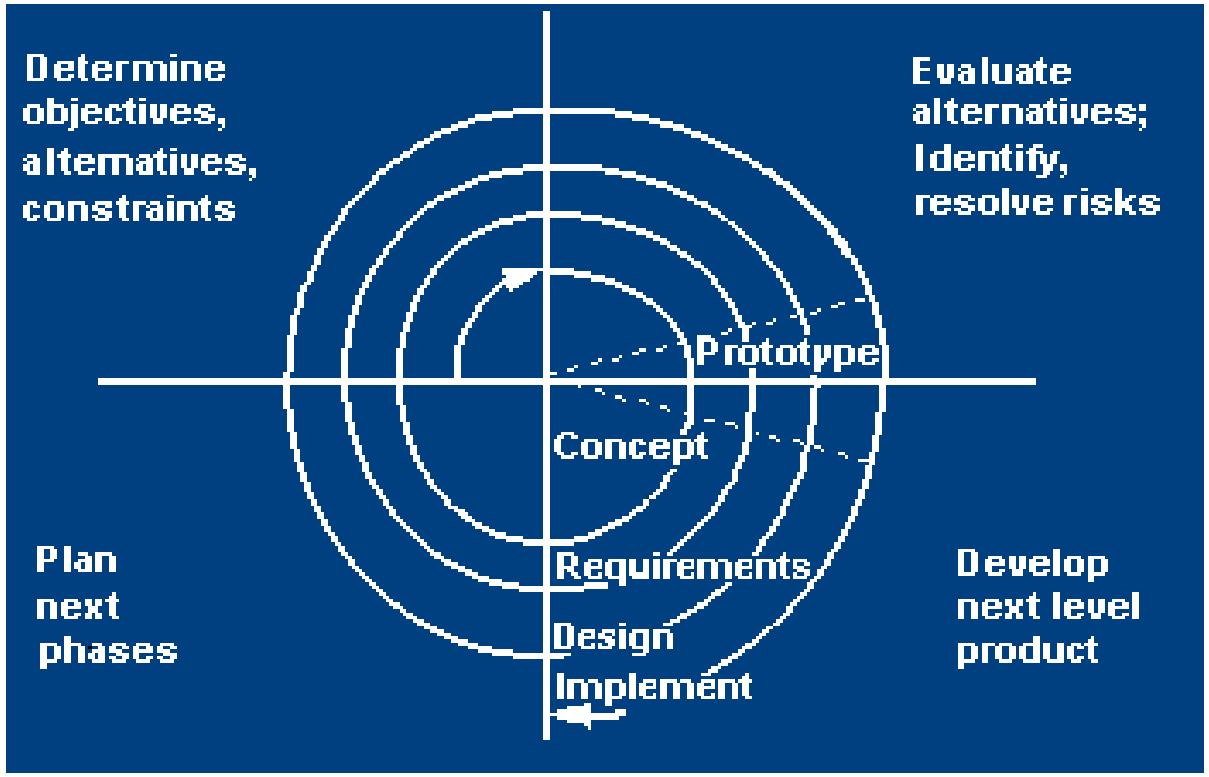
Spiral Model

- Proposed by Barry Boehm - Evolutionary software process model
- Couples Iterative nature of prototyping + waterfall model
- Potential for rapid development of increasingly more complete version of software
- Spiral Model - software is developed in a series of evolutionary releases.
- Early iteration - Paper model / prototype (trial product)
- Later increasingly more complete version of engineering software is produced

Spiral Development

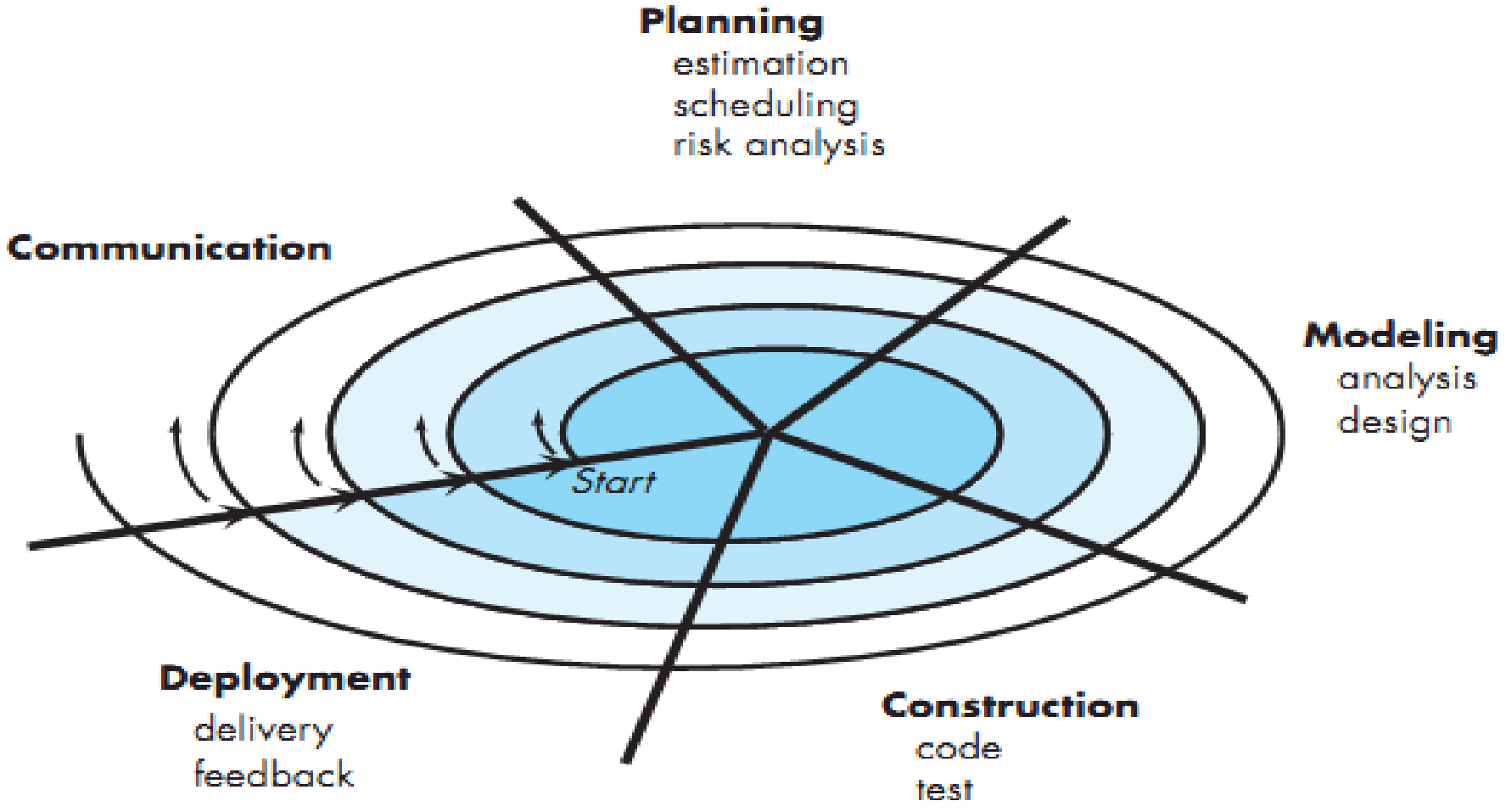
- Process is represented as a spiral rather than as a sequence of activities with backtracking.
- Each loop in the spiral represents a phase in the process.
- No fixed phases such as specification or design - loops in the spiral are chosen depending on what is required.
- Risks are explicitly assessed and resolved throughout the process.

Spiral Model



- Adds risk analysis, and 4gl RAD prototyping to the waterfall model
- **Each cycle involves the same sequence of steps as the waterfall process model**

Spiral Model



Spiral Model Sectors

- **Objective setting**
  - Specific objectives for the phase are identified.
- **Risk assessment and reduction**
  - Risks are assessed and activities put in place to reduce the key risks.
- **Development and validation**
  - A development model for the system is chosen which can be any of the generic models.
- **Planning**
  - The project is reviewed and the next phase of the spiral is planned.

Spiral Model

Advantage:

- Realistic approach of development in large scale
- Developer and customer satisfaction
- Reacts to risk at evolutionary level
- Uses prototype - Risk reduction mechanism
- Enables developer to apply prototype in any stage
- Direct consideration of technical risk at all step.

Disadvantage

- Difficult to convince customers
- Demands considerable risk assessment expertise
- Model has not been used as widely as the linear sequential or prototype paradigm
- It is not used for small projects

Spiral Model Strengths

- Provides early indication of insurmountable risks, without much cost
- Users see the system early because of rapid prototyping tools
- Critical high-risk functions are developed first
- The design does not have to be perfect
- Users can be closely tied to all lifecycle steps
- Early and frequent feedback from users
- Cumulative costs assessed frequently

Spiral Model Weaknesses

- Time spent for evaluating risks too large for small or low-risk projects
- Time spent planning, resetting objectives, doing risk analysis and prototyping may be excessive
- The model is complex
- Risk assessment expertise is required
- Spiral may continue indefinitely
- Developers must be reassigned during non-development phase activities
- May be hard to define objective, verifiable milestones that indicate readiness to proceed through the next iteration

When to use Spiral Model

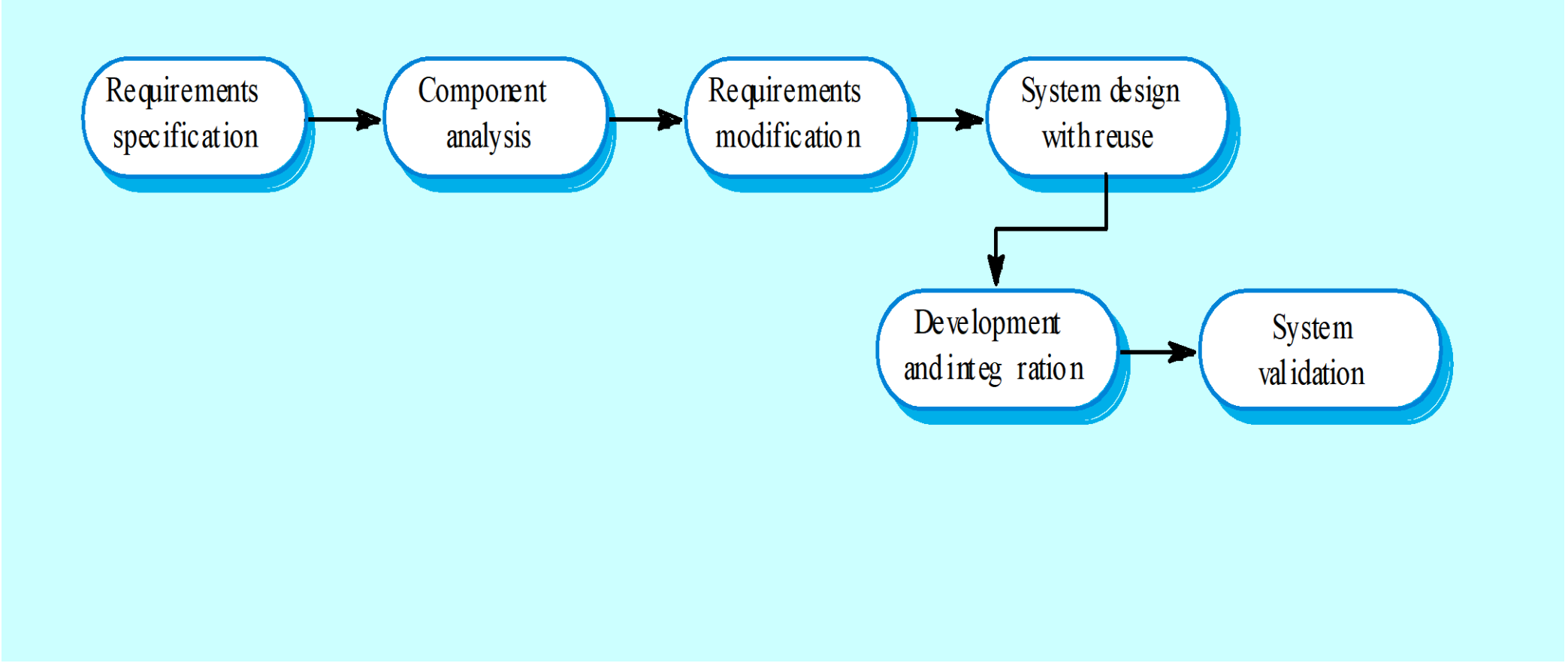
- When creation of a prototype is appropriate
- When costs and risk evaluation is important
- For medium to high-risk projects
- Long-term project commitment unwise because of potential changes to economic priorities
- Users are unsure of their needs
- Requirements are complex
- New product line
- Significant changes are expected (research and exploration)

- **Component-based Software Engineering**

Component-based software engineering

- Based on systematic reuse where systems are integrated from existing components or COTS (Commercial-off-the-shelf) systems.
- Process stages
  - **Component analysis;**
  - **Requirements modification;**
  - **System design with reuse;**
  - **Development and integration.**
- This approach is becoming increasingly used as component standards have emerged.

Reuse-oriented development



- **Choosing a Software Process Model**

Choosing a Software Process

Changes during the software development process are expensive.

- **If the requirements are poorly understood,** or expected to change, select a process that keeps flexibility. Iterative refinement, sprints, phased implementation.
- **If a big software systems has many inter-related components**, avoid major changes to the design of a system during development. Sequential process, such as the modified waterfall model.
- **If the market for the software is poorly understood**, use a process that gets operational software in front of customers as quickly as possible. Incremental, sprints.

Tailored SDLC Models

- No single model fits all projects
- If there is no suitable model for a particular project, pick a model that comes close and modify it for your needs.
  - If project should consider risk but complete spiral model is too much – start with spiral and simplify it
  - If project should be delivered in increments but there are serious reliability issues – combine incremental model with the V-shaped model
- Each team must pick or customize a SDLC model to fit its project

Corporate Processes

Large software development organizations have their own internal processes that are designed for their needs. For example:

- Amazon.com (Internet commerce) makes extensive use of sprints. Most software development is divided into increments of about four weeks elapsed time.
- SAP (business software) emphasizes the functionality that is seen by their business customers. Much of the development is suitable for a sequential process.
- Microsoft (PC software) places great emphasis on testing with a very wide variety of equipment and backward compatibility. Much of the development uses a spiral process.

- **Process iteration**

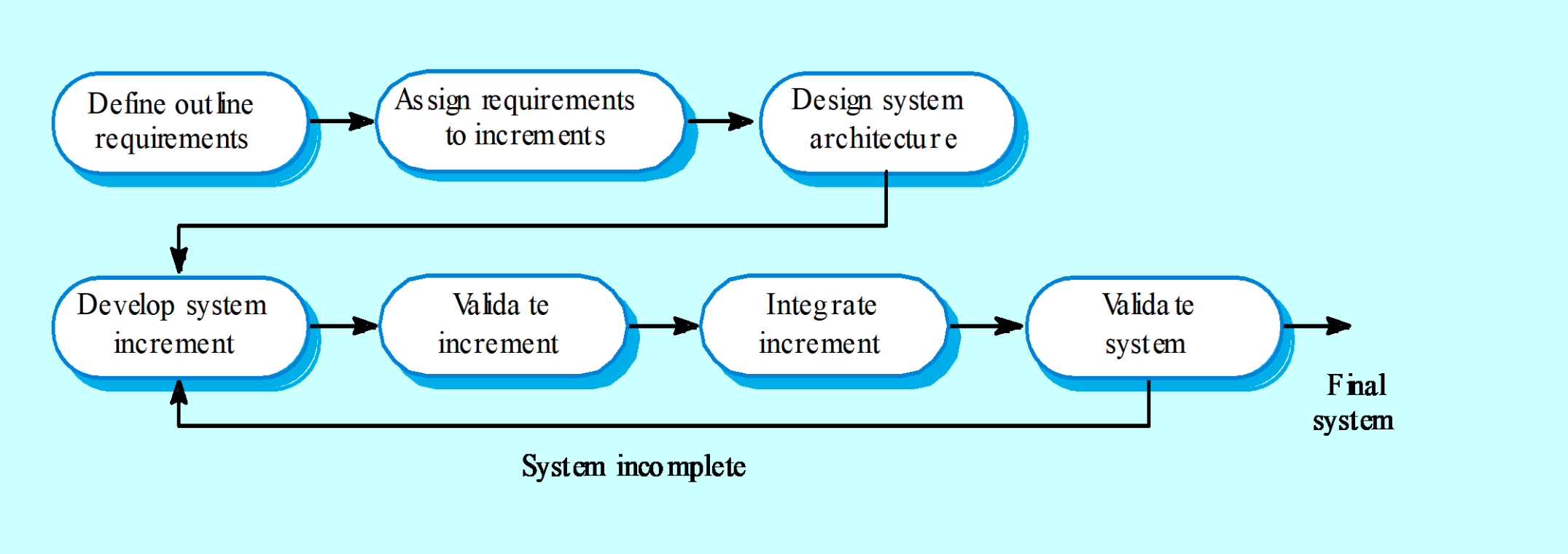
Process iteration

- System requirements ALWAYS evolve in the course of a project so process iteration where earlier stages are reworked is always part of the process for large systems.
- Iteration can be applied to any of the generic process models.
- Two (related) approaches
  - Incremental delivery
  - Spiral development.

Incremental delivery

- Rather than deliver the system as a single delivery, the development and delivery is broken down into increments with each increment delivering part of the required functionality.
- User requirements are prioritised and the highest priority requirements are included in early increments.
- Once the development of an increment is started, the requirements are frozen though requirements for later increments can continue to evolve.

Incremental development



Product and Process

- **If the process is weak, the end product will undoubtedly suffer**
- As creative software professional, you should also derive as much satisfaction from the process as the end product

Conclusions

- **Different life-cycle models**
  - Each with its own strengths
  - Each with its own weaknesses
- **Criteria for deciding on a model include:**
  - The organization
  - Its management
  - Skills of the employees
  - The nature of the product
- **Best suggestion**
  - “Mix-and-match” life-cycle model


Thank
you