Chapter 2 – Software Processes

Topics covered

- ✧ Software process models
- ✧ Process activities
- ✧ Coping with change
- ✧ Process improvement

فرایند

✧فرهنگ فارسی معین:

(فَ یَ) مجموعه عملیات و مراحل لازم برای رسیدن به یک هدف مشخص

مثال: فرایند تعویض لاستیک پنچر شده، فرایند پخت دلمه، فرایند ثبت‌نام دانشجو، فرایند تولید نرم‌افزار

The software process

- A structured set of activities required to develop a software system.

- Many different software processes but all involve:
  - Specification – defining what the system should do;
  - Design and implementation – defining the organization of the system and implementing the system;
  - Validation – checking that it does what the customer wants;
  - Evolution – changing the system in response to changing customer needs.

Software process descriptions

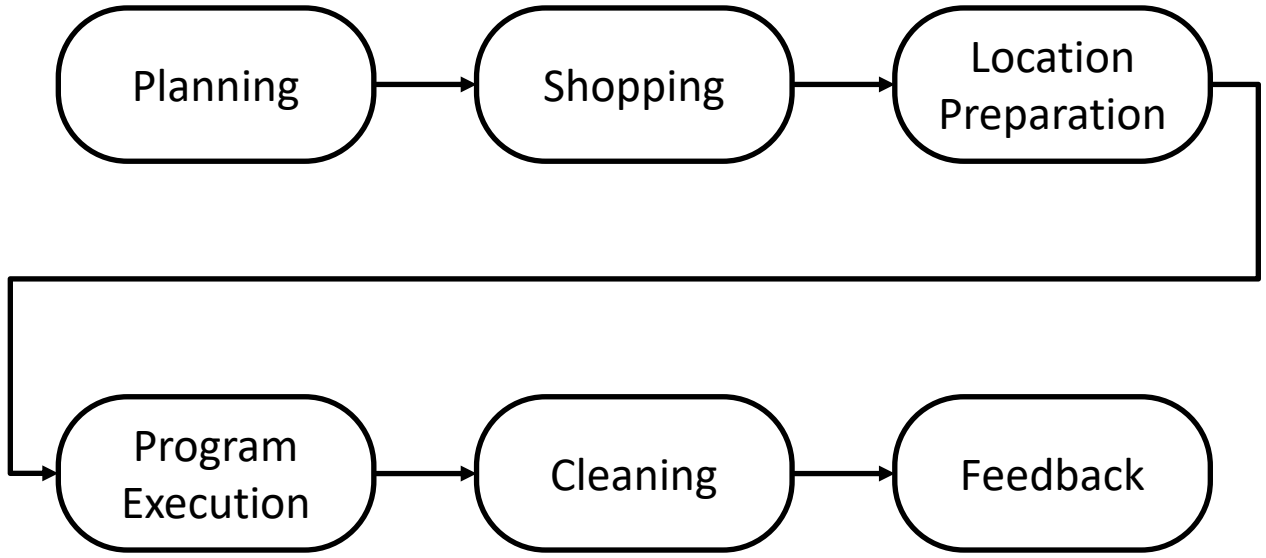
- When we describe and discuss processes, we usually talk about the activities in these processes such as specifying a data model, designing a user interface, etc. and the ordering of these activities.
- Process descriptions may also include:
  - Products, which are the outcomes of a process activity;
  - Roles, which reflect the responsibilities of the people involved in the process;
  - Pre- and post-conditions, which are statements that are true before and after a process activity has been enacted or a product produced.

Example of a process

✧ Let’s practice!

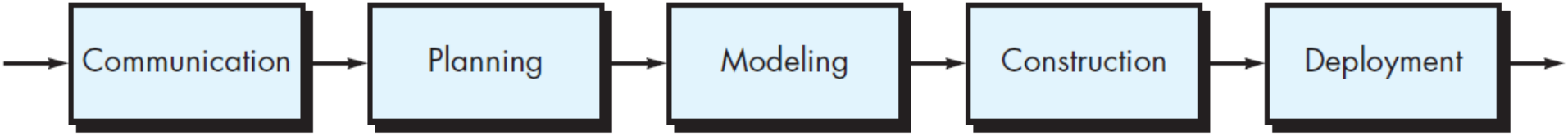
✧ Preparing for a birthday party

Birthday party



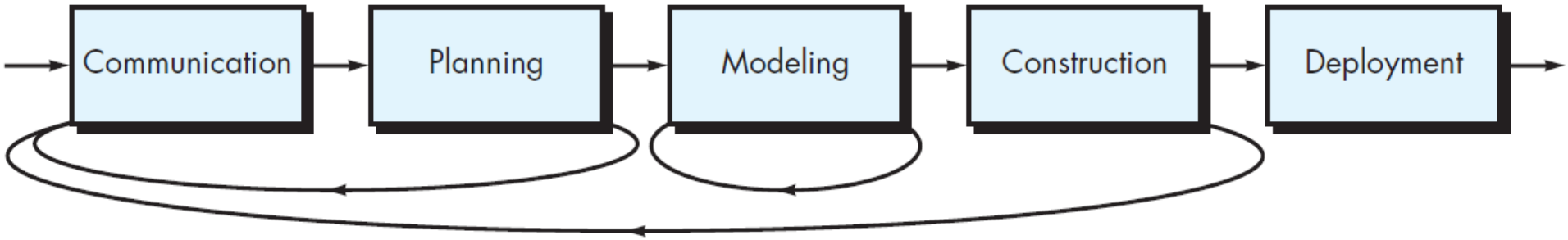
Process flow – linear process flow

- A *linear process flow* executes each of the activities in sequence, beginning with communication and culminating with deployment



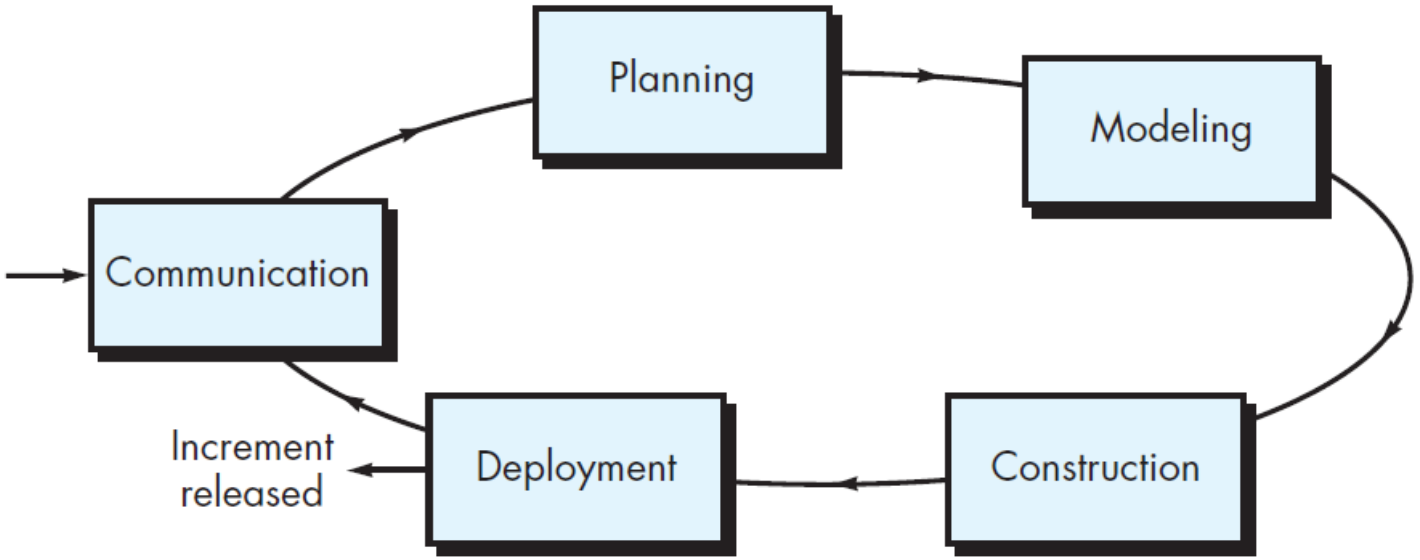
Process flow – iterative process flow

✧ An *iterative process flow* repeats one or more of the activities before proceeding to the next



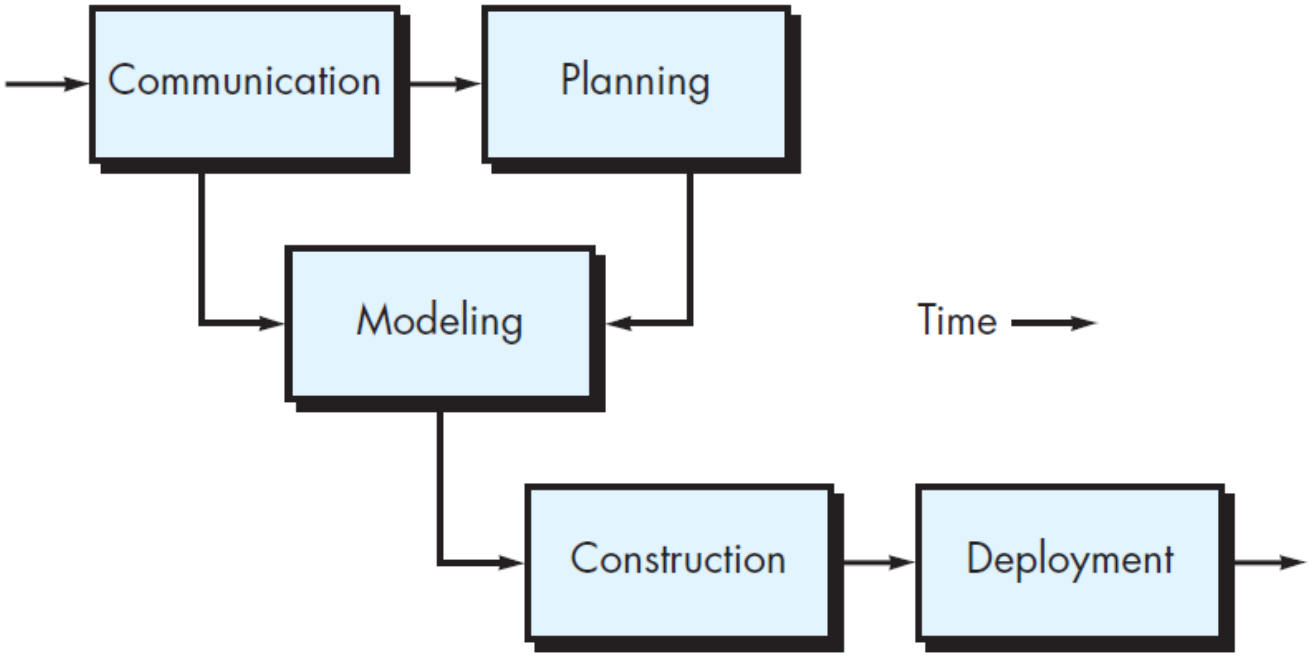
Process flow – evolutionary process flow

✧ An *evolutionary process flow* executes the activities in a "circular" manner. Each circuit through the five activities leads to a more complete version of the software.



Process flow – parallel process flow

- A *parallel process flow* executes one or more activities in parallel with other activities



Plan-driven and agile processes

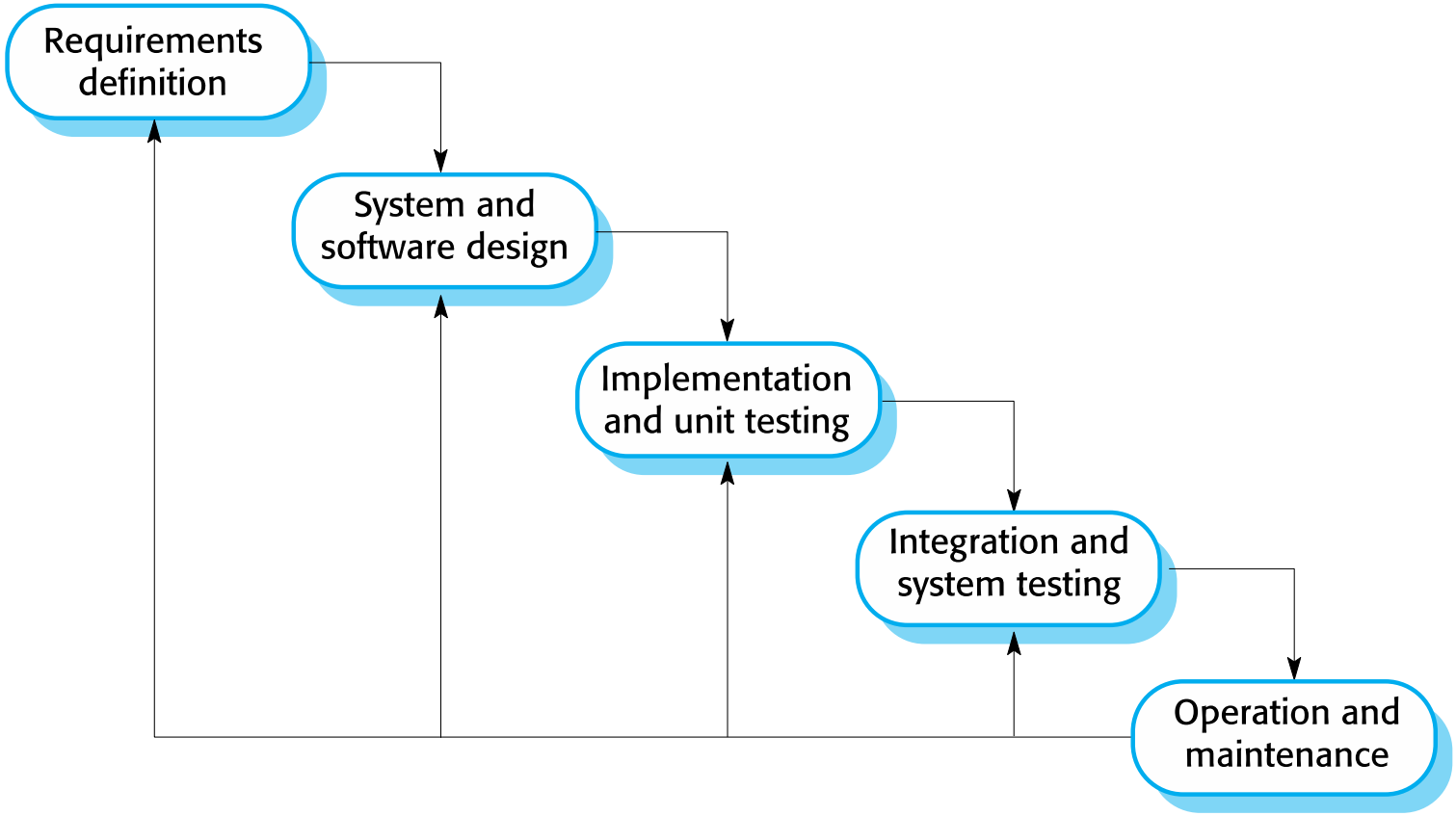
- Plan-driven processes
  - All activities are planned in advance and progress is measured against this plan.
- Agile processes
  - planning is incremental
    - Easier to change the process to reflect changing customer requirements.
- In practice, most practical processes include elements of both plan-driven and agile approaches.
- There are no right or wrong software processes.

Software process models

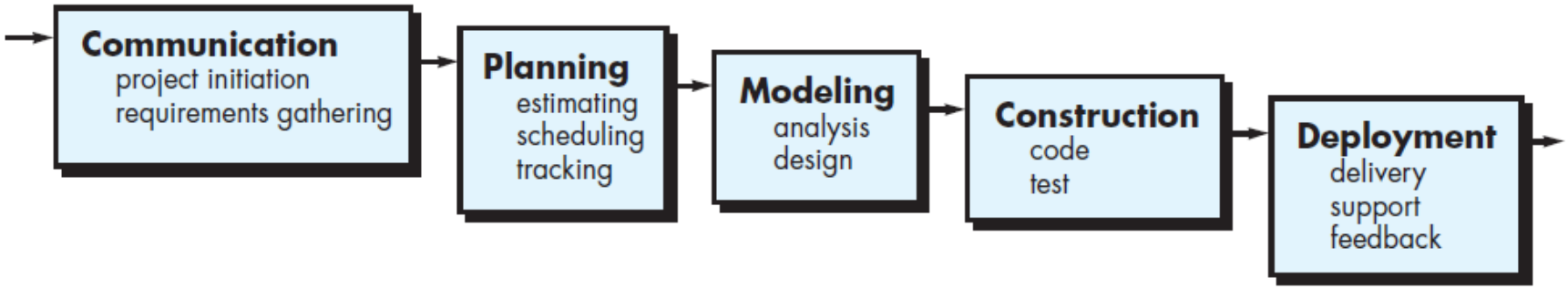
Software process models

- The waterfall model
  - Plan-driven model. Separate and distinct phases of specification and development.
- Incremental development
  - Specification, development and validation are interleaved. May be plan-driven or agile.
- Integration and configuration
  - The system is assembled from existing configurable components. May be plan-driven or agile.
- In practice, most large systems are developed using a process that incorporates elements from all of these models.

The waterfall model (classic life cycle)



The waterfall model (classic life cycle)



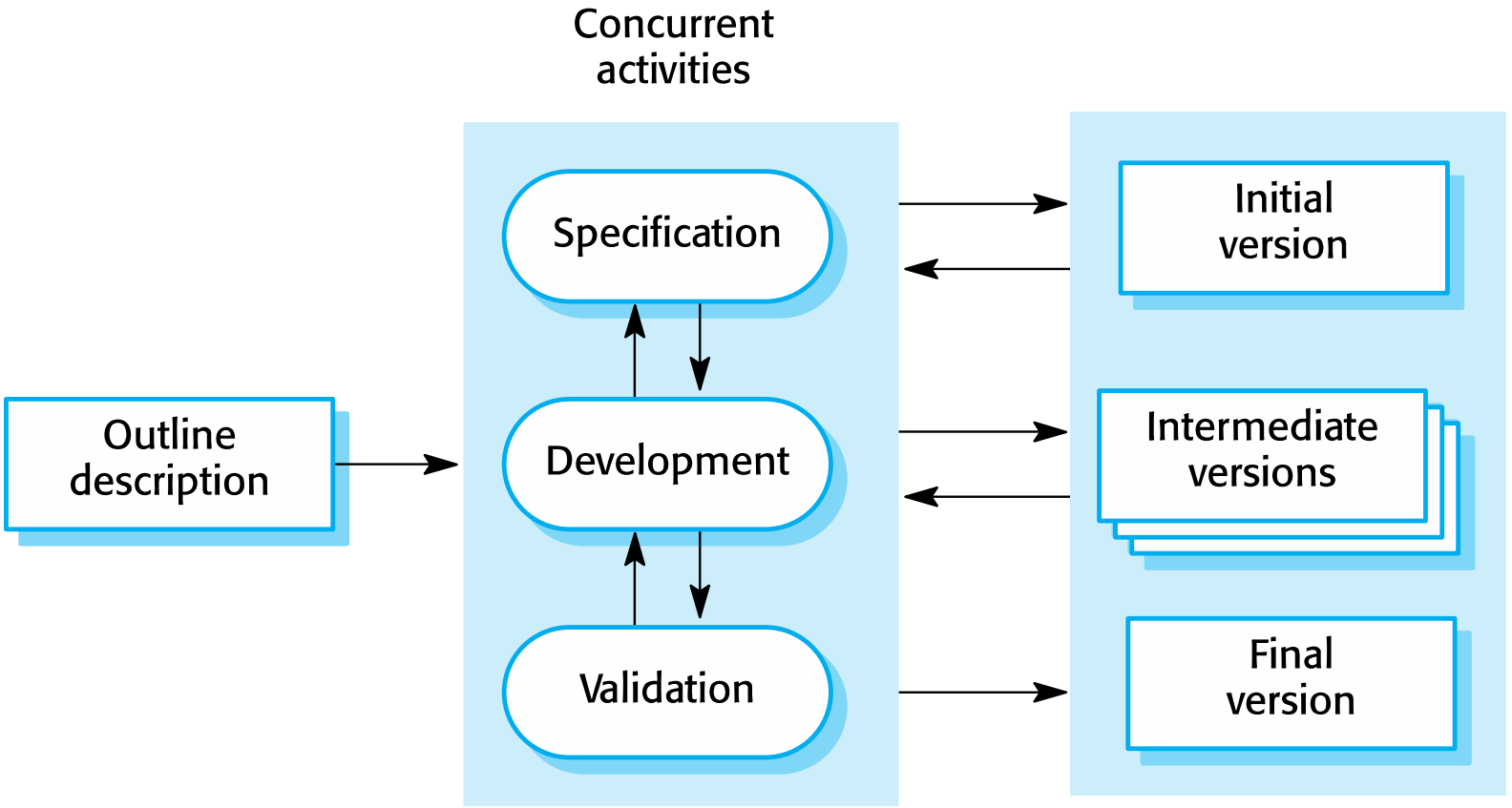
Waterfall model phases

- There are separate identified phases in the waterfall model:
  - Requirements analysis and definition
  - System and software design
  - Implementation and unit testing
  - Integration and system testing
  - Operation and maintenance
- Main drawback
  - Difficulty of accommodating change after the process is underway.
    - A phase has to be complete before moving onto the next phase.

Waterfall model problems

- Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements.
  - Appropriate when
    - Requirements are well-understood
    - Changes will be fairly limited during the design process
  - Few business systems have stable requirements.
- Mostly used for
  - Large systems engineering projects
    - System is developed at several sites
    - The plan-driven nature of the waterfall model helps coordinate the work

Incremental development



Incremental development benefits

- Lower cost to accommodate changes.
  - The amount of analysis and documentation that has to be redone is much less than is required with the waterfall model.
- 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.
- More rapid delivery and deployment of useful software to the customer is possible.
  - Customers are able to use and gain value from the software earlier than is possible with a waterfall process.

Incremental development problems

- The process is not visible.
  - Managers need regular deliverables to measure progress. If systems are developed quickly, it is not cost-effective to produce documents that reflect every version of the system.
- System structure tends to degrade as new increments are added.
  - Unless time and money is spent on refactoring to improve the software, regular change tends to corrupt its structure. Incorporating further software changes becomes increasingly difficult and costly.

Incremental development problems

- ✧ Less suitable for systems that are
  - Large
  - Complex
  - Long-lifetime
  - Different teams develop different parts
- ✧ Large complex systems need
  - Stable framework
  - Clear division of responsibilities

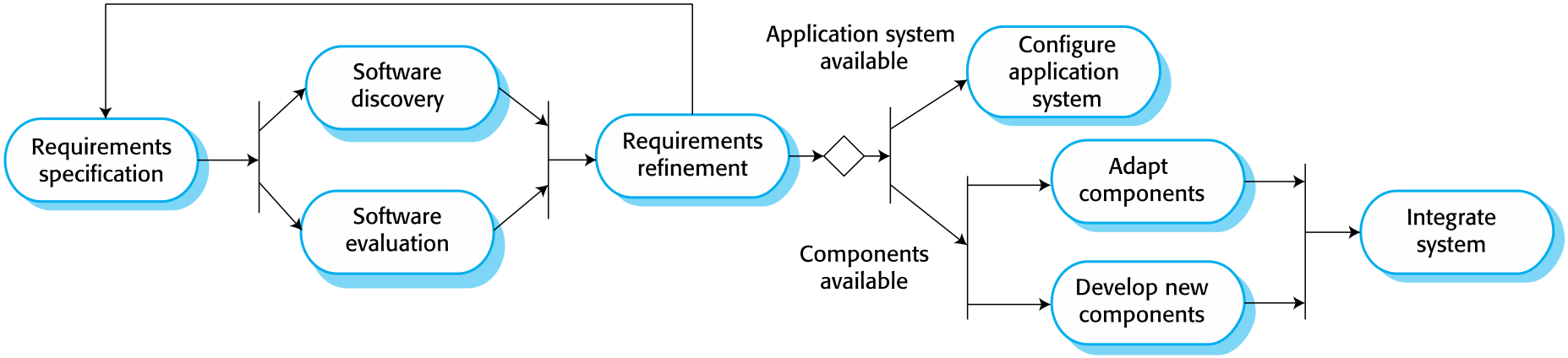
Integration and configuration

- Based on software reuse where systems are integrated from existing components or application systems (sometimes called COTS -Commercial-off-the-shelf) systems).
- Reused elements may be configured to adapt their behaviour and functionality to a user's requirements
- Reuse is now the standard approach for building many types of business system.

Types of reusable software

- Stand-alone application systems (COTS) that are configured for use in a particular environment.
- Collections of objects that are developed as a package to be integrated with a component framework such as .NET or J2EE.
- Web services that are developed according to service standards and which are available for remote invocation.

Reuse-oriented software engineering



Key process stages

- Requirements specification
- Software discovery and evaluation
- Requirements refinement
- Application system configuration
- Component adaptation and integration

Reuse-oriented software engineering

✧ Benefits:

- Reduces amount of software to be developed
  - Reduced cost
  - Reduced risk
  - Usually results in faster delivery of software

✧ Problems:

- Requirements compromises
  - Software may not meet user needs
  - No control over component evolution

Process activities

Process activities

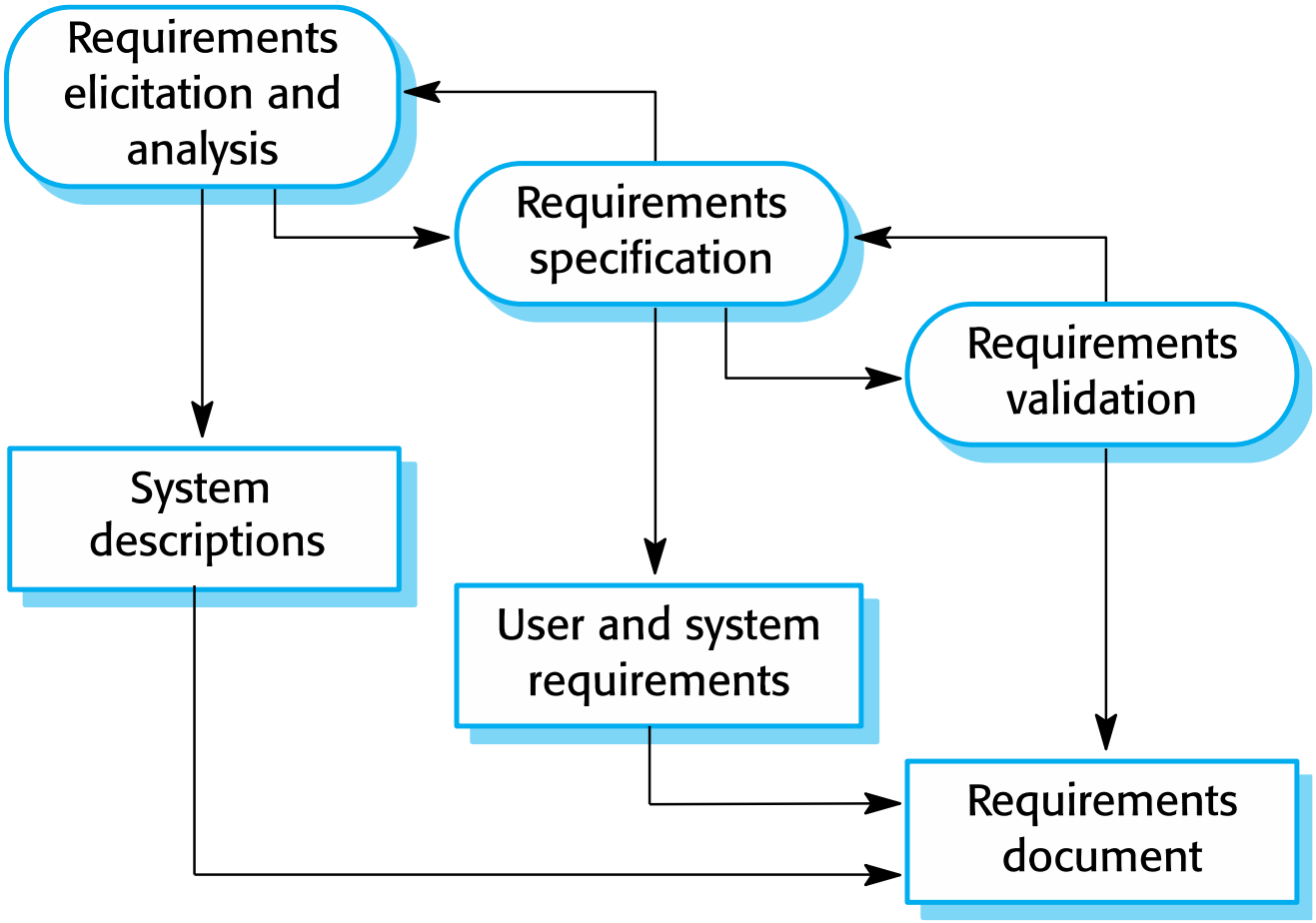
- Four basic process activities:
  - Specification, development, validation and evolution are organized differently in different development processes.
  - In the waterfall model, they are organized in sequence
  - In incremental development, they are inter-leaved

- Real software processes are inter-leaved sequences
  - Technical, collaborative, managerial activities
  - To develop a software

Software specification

- The process of establishing what services are required and the constraints on the system's operation and development.
- Requirements engineering process
  - Requirements elicitation and analysis
    - What do the system stakeholders require or expect from the system?
  - Requirements specification
    - Defining the requirements in detail
  - Requirements validation
    - Checking the validity of the requirements

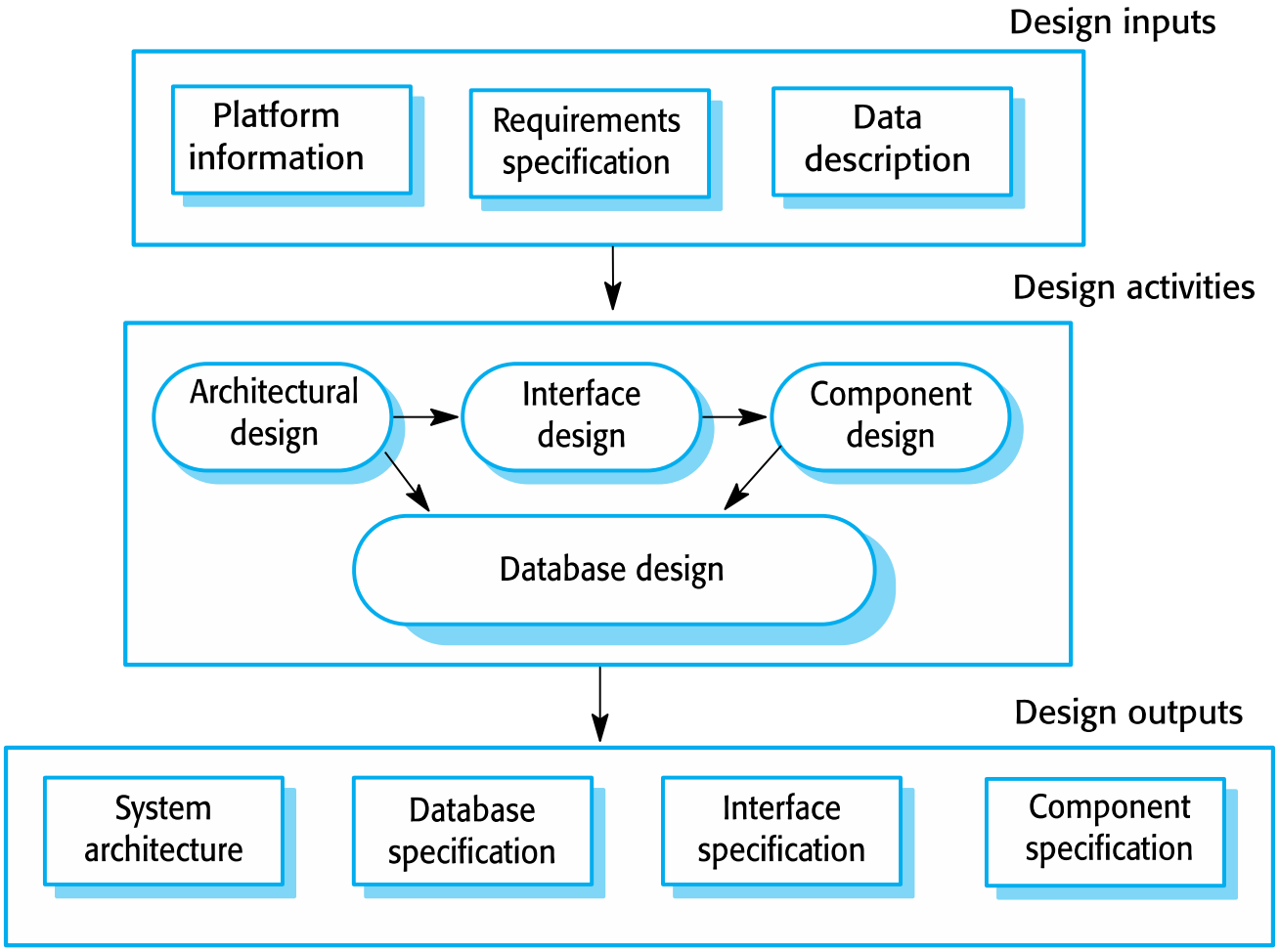
The requirements engineering process



Software design and implementation

- The process of converting the system specification into an executable system.
- Software design
  - Design a software structure that realises the specification;
- Implementation
  - Translate this structure into an executable program;
- The activities of design and implementation are closely related and may be inter-leaved.

A general model of the design process



Design activities

- *Architectural design,* where you identify the overall structure of the system, the principal components (or subsystems), their relationships and how they are distributed.
- *Interface design,* where you define the interfaces between system components.
- *Database design,* where you design the system data structures and how these are to be represented in a database.
- *Component selection and design,* where you search for reusable components. If unavailable, you design how it will operate.

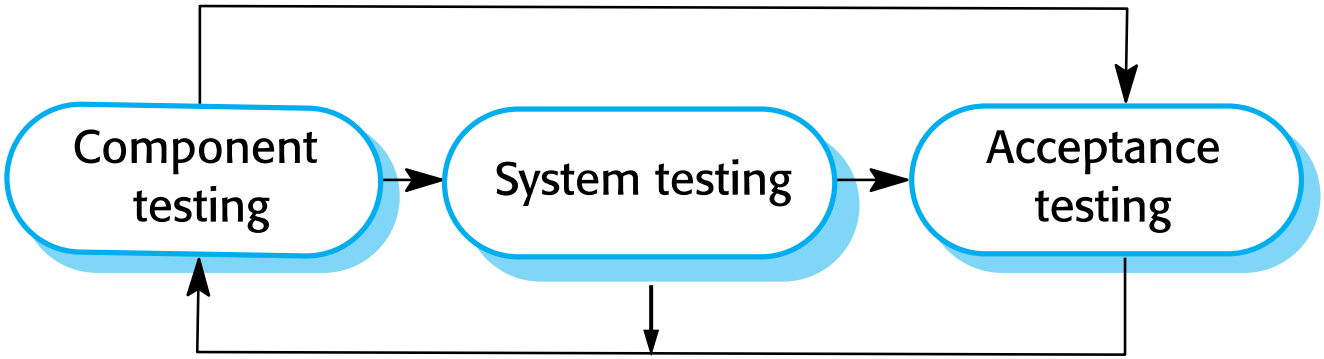
System implementation

- The software is implemented either by developing a program or programs or by configuring an application system.
- Design and implementation are interleaved activities for most types of software system.
- Programming is an activity with no standard process.
- Debugging is the activity of finding program faults and correcting these faults.

Software validation

- Verification and validation (V & V) is intended to show that a system conforms to its specification and meets the requirements of the system customer.
- Involves checking and review processes and system testing.
- System testing involves executing the system with test cases that are derived from the specification of the real data to be processed by the system.
- Testing is the most commonly used V & V activity.

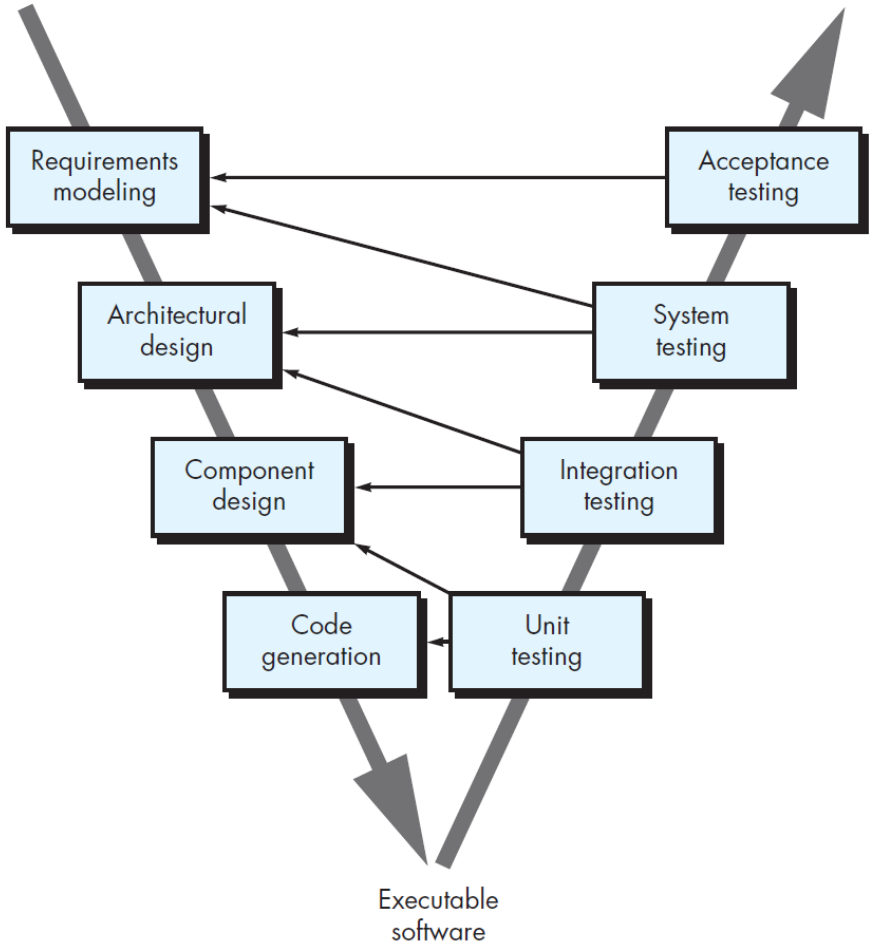
Stages of testing



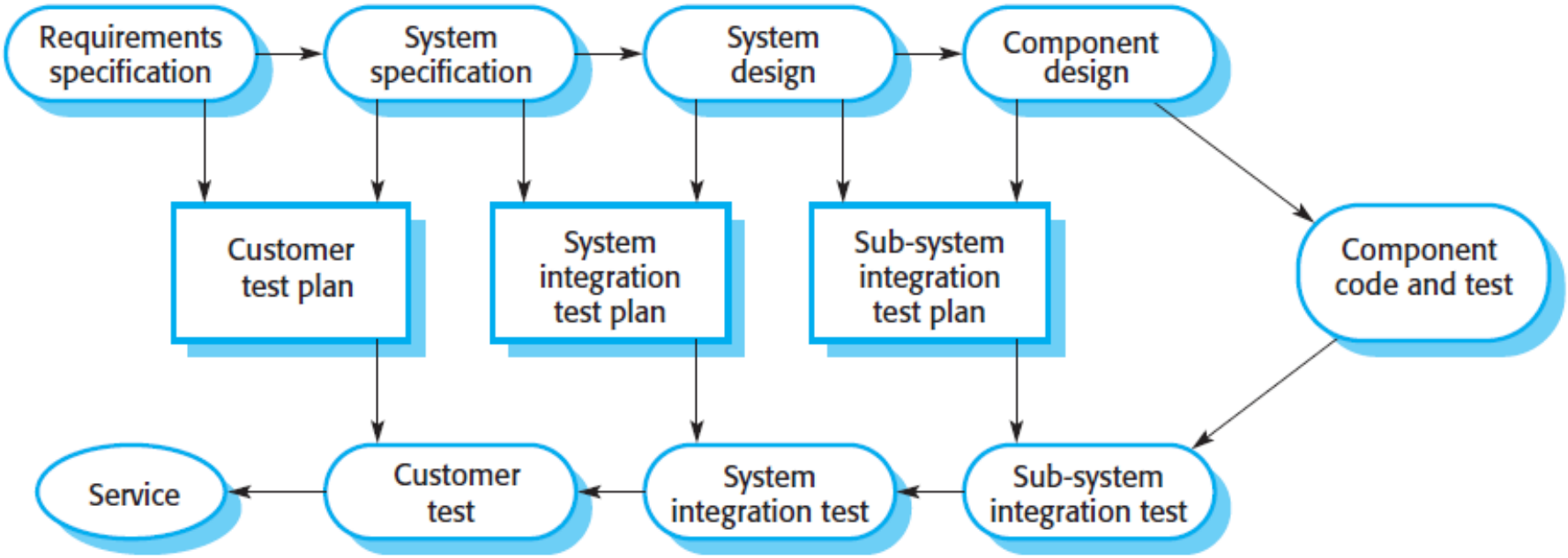
Testing stages

- ✧ Component testing
  - Individual components are tested independently;
  - Components may be functions or objects or coherent groupings of these entities.
- ✧ System testing
  - Testing of the system as a whole. Testing of emergent properties is particularly important.
- ✧ Customer testing
  - Testing with customer data to check that the system meets the customer's needs.

Testing phases in a plan-driven software process (V-model)



Testing phases in a plan-driven software process (V-model)



Software evolution

- Software is inherently flexible and can change.
- As requirements change through changing business circumstances, the software that supports the business must also evolve and change.
- Although there has been a demarcation between development and evolution (maintenance) this is increasingly irrelevant as fewer and fewer systems are completely new.

System evolution