Software Engineering ?

Importance of Software Processes ?

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.
- A **software process model** is an **abstract representation** of a process. It presents a description of a process from some particular perspective.

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.

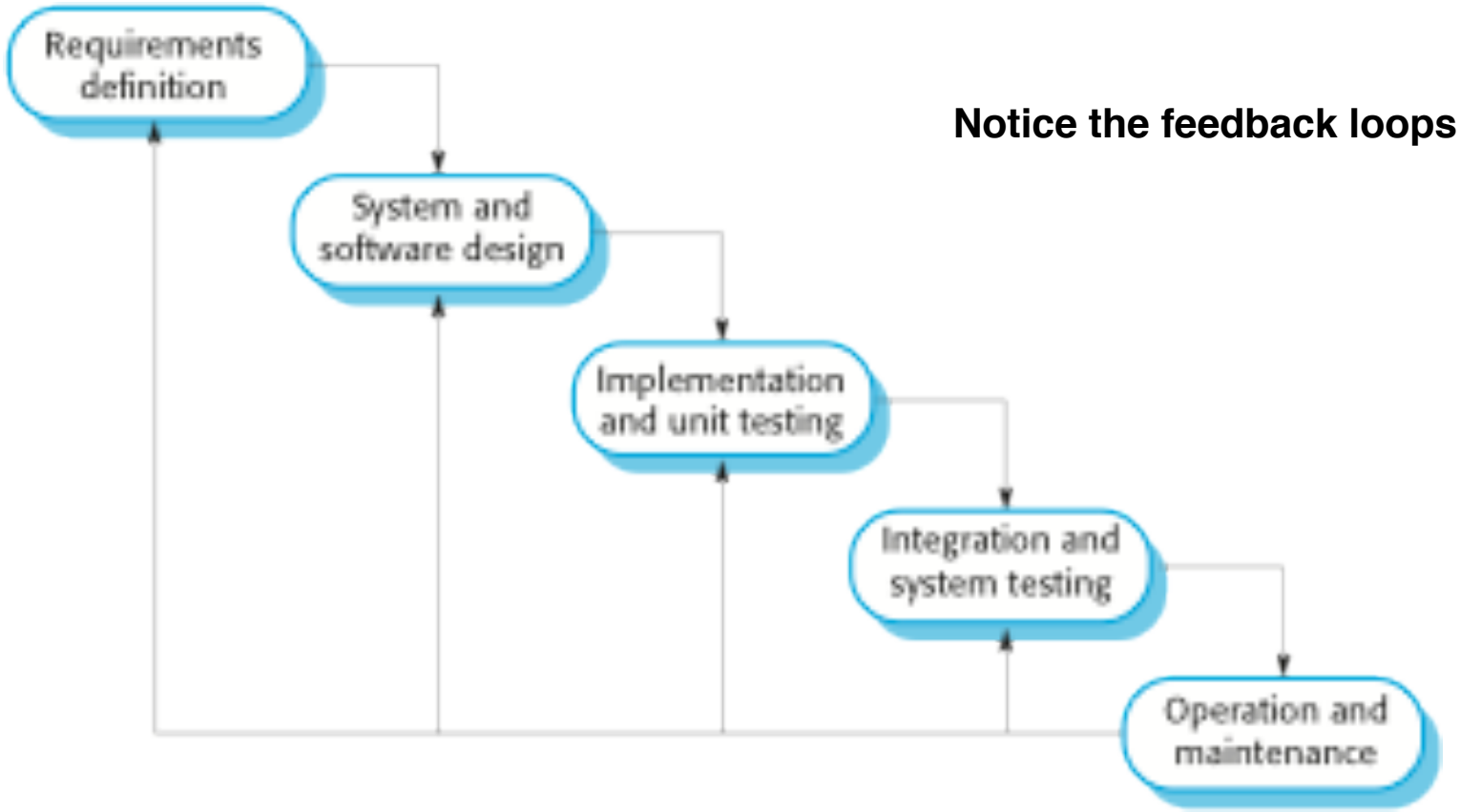
Plan-driven and Agile Processes

- **Plan-driven processes** are processes where all of the process activities are planned in advance and progress is measured against this plan.
  - Plan drives everything!
- In Agile Processes planning is incremental and it is **easier to change** the process to reflect changing customer requirements.

- **In practice, most practical processes include elements of both plan-driven and agile.**

- **There are no right or wrong software processes**.

The Waterfall Model



Notice the feedback loops

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
- The **main drawback** of the waterfall model is the difficulty of accommodating **change** after the process is underway. In principle, a phase has to be complete before moving onto the next phase.
- But many issues are true too, such as **risk addressing.**
- War Stories but true.

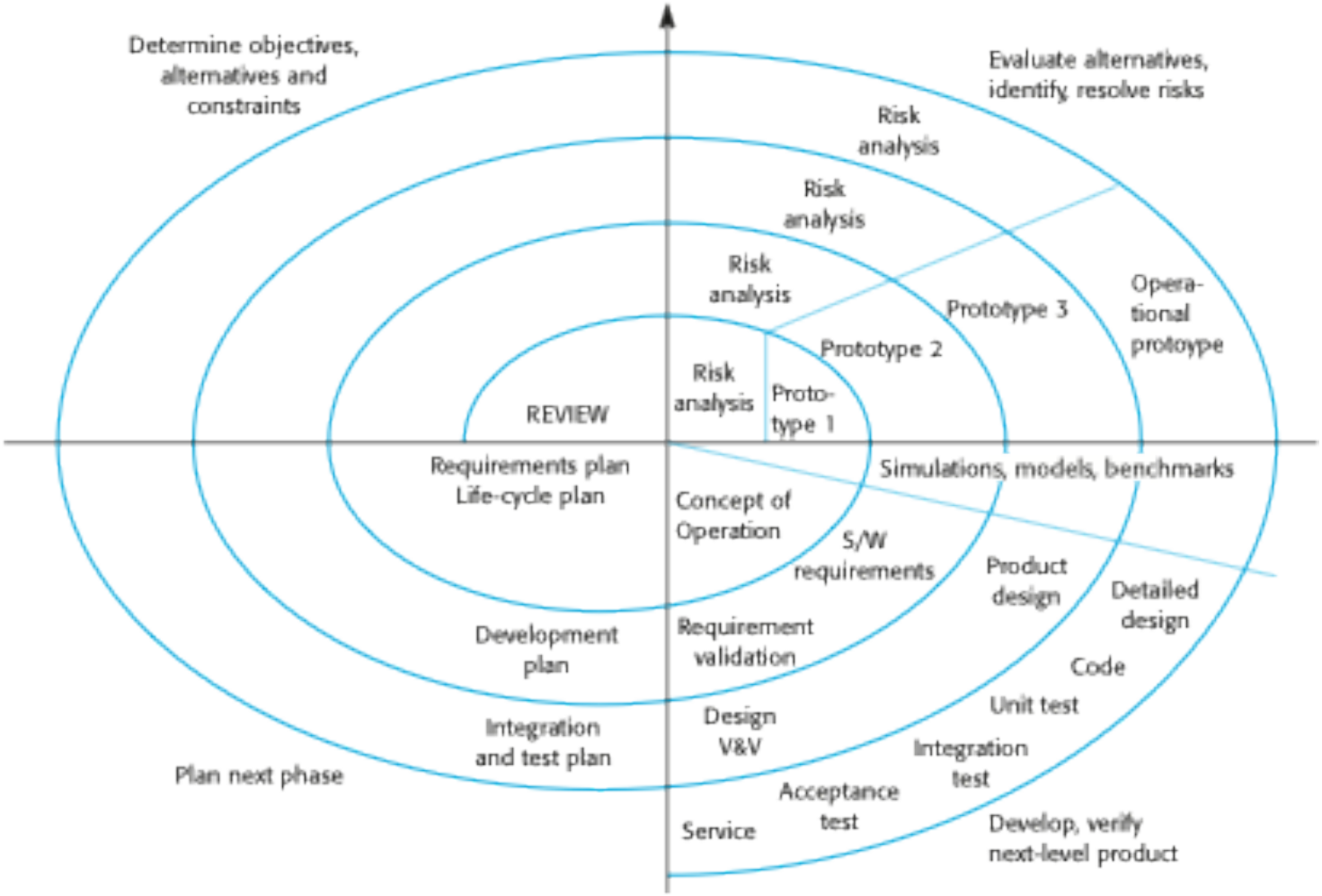
Waterfall Model Problems

- Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements.
  - Therefore, this model is only appropriate when the requirements are **well-understood** and changes will be **fairly limited** during the design process.
  - Few business systems have stable requirements.
- The waterfall model is mostly used for **large systems engineering** projects where a system is developed at several sites.
  - In those circumstances, the plan-driven nature of the waterfall model helps coordinate the work.

Boehm's Spiral Model - heavyweight model

- 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.
  - This was the motivation behind developing the Spiral Model - Risk

Boehm's Spiral Model of the Software Process



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. Development takes place.
- Planning
  - The project is reviewed and the next phase of the spiral is planned.

Spiral Model Usage

- Spiral model has been very influential in helping people think about **iteration** in software processes and introducing the **risk-driven approach** to development.

- In practice, however, the model is rarely used as published for **practical** software development.

Process Activities

- Real software processes are **inter-leaved sequences** of technical, collaborative and managerial activities with the overall goal of specifying, designing, implementing and testing a software system.

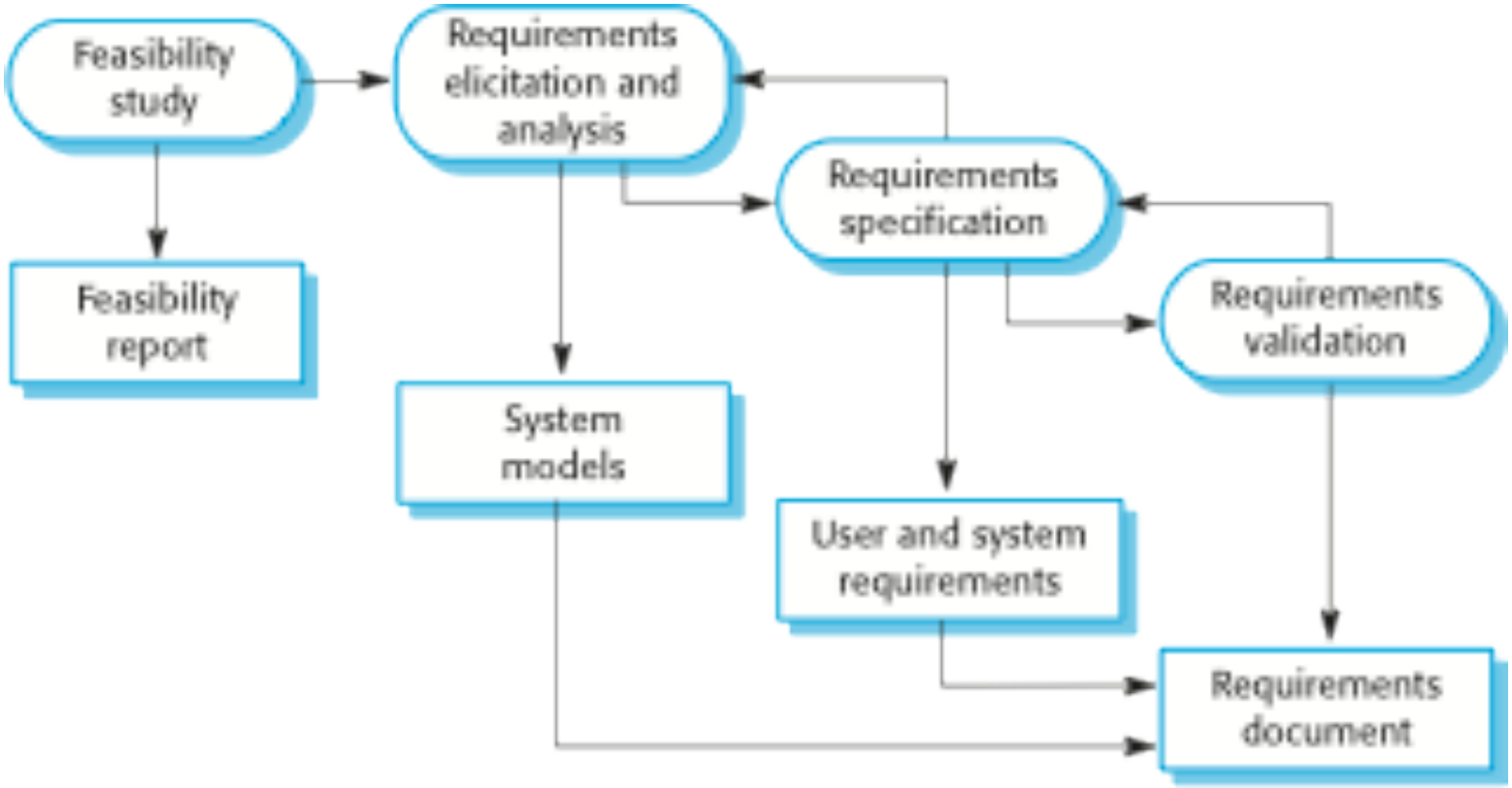
- The four basic process activities (specified in your book) of specification, development, validation and evolution are **organized differently in different development processes**.

- In the waterfall model, they are organized in **sequence**, whereas in incremental development they are i**nter-leaved**.

Software Specification

- **Is:** The process of establishing what services are **required** and the **constraints** on the system's operation and development.
- Requirements Engineering Process
  - **Feasibility study**
    - Is it technically and financially feasible to build the system?
    - ROI, product Vision, what market share are we after? Urgency of development?
  - **Requirements elicitation and analysis**
    - What do the system stakeholders **require or expect** from the system?
    - Recognition that the **client must be satisfied**. Have checkbook.
  - **Requirements Specification**
    - Defining the requirements in detail. **These are the 'whats' of a system!!!**
  - **Requirements Validation**
    - Checking the validity of the requirements
    - Are they feasible, testable, sufficient, necessary, ...

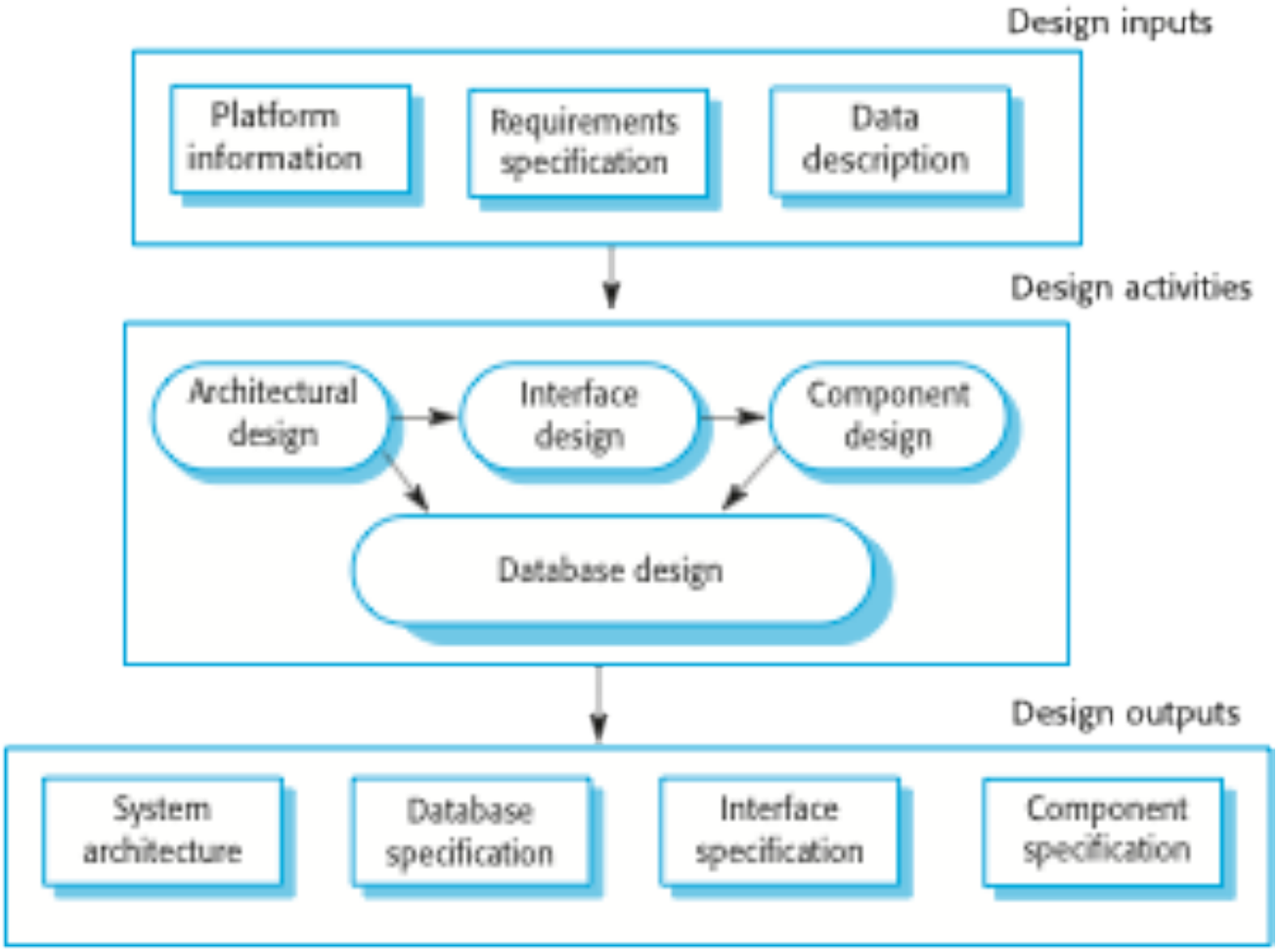
The Requirements Engineering Process



Software Design and Implementation

- The process of **converting** the system specification into an **executable system**.
- **Software Design** (inputs: specifications)
  - Design a software structure that **realizes** the specification;
- **Implementation**
  - Translate this model / structure into an executable app;
  - **Programming is the implementation of the design.**
- The activities of design and implementation are closely related and may be inter-leaved and definitely are using many modern development processes. (more to come)

A General Model of the Design Process



Design Activities

- ***Architectural design,*** where one identifies the **overall structure** of the system, the **principal components** (sometimes called sub-systems or modules or layers, etc…), their **relationships** and how they are distributed.

  Example: Such as Model View Controller (MVC)pattern; others.
- ***Interface design,*** where one defines the interfaces between system components. (interface to controller; controller to database; external device (sensor) to analyzer…)
- Recognize that to the end user, the UI **is** the application.
- ***Component design,*** where one takes each system component and design how it will operate.
- This is what we are often used to
- ***Database design,*** where one designs the system data structures and how these are to be represented in a database.

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.
  - Look up the difference between **verification** and **validation. Be able to explain clearly.**
- 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

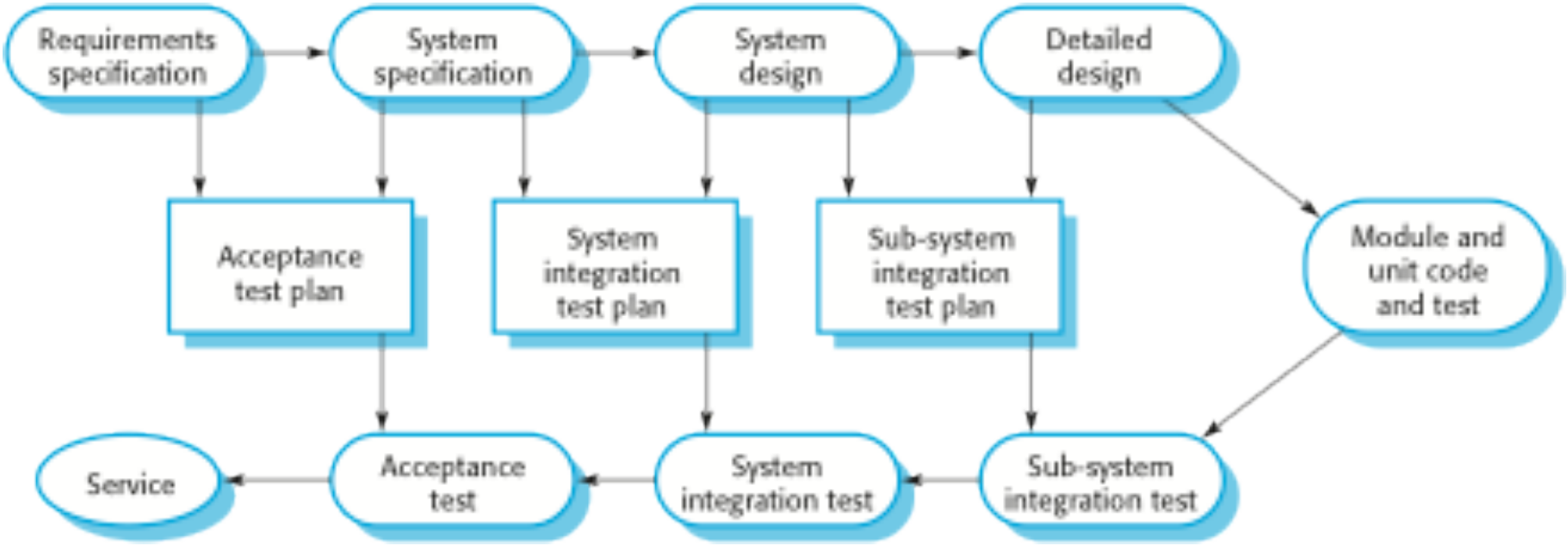
What does this mean??

Testing Stages

- **Development or component testing**. (**Unit 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.
- **Acceptance testing**
  - Testing with **customer** data to check that the system meets the customer's needs

- **Some models have various other kinds of testing; subsystem testing, integrated system testing, and much more.**

Testing Phases in a Plan-driven Software Process



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 (greenfield).

Key points

- Understand the principles of software engineering
- Understand what we mean by a software process
- Note the two major classifications of processes and note also that there are numerous hybrid classifications within these.
- Software processes are the activities involved in producing a software system.
- Software process models are abstract representations of these processes.

Key points

- Requirements engineering is the process of developing a software specification.
- Design and implementation processes are concerned with transforming a requirements specification into an executable software system.
- Software validation is the process of checking that the system conforms to its specification and that it meets the real needs of the users of the system.
- Software evolution takes place when you change existing software systems to meet new requirements. The software must evolve to remain useful.