CH.2

All software processes 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.

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 processes are processes where all of the process activities are planned in advance and progress is measured against this plan. **(Before)**

Agile processes, planning is incremental and it is easier to change the process to reflect changing customer requirements. **(with in)**

×There are no right or wrong software processes.

The 3 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.
- ✧ **Reuse-oriented software engineering**
  - The system is assembled from existing components. May be plan-driven or agile.

Waterfall model phases :

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

- **The main problem(draw back) of the waterfall model** is the difficuilty to change after the process is underway

Waterfall problems :

✧ Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements.

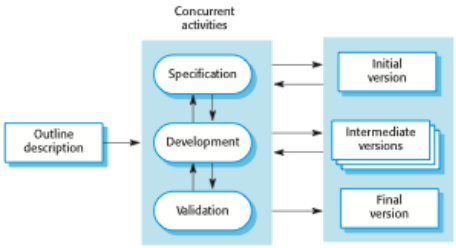
- Therefore, this model is only appropriate when the requirements are well-understood and changes will be fairly limited during the design process.
- Few business systems have stable requirements.

✧ 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.

Incremental development benefits :

- ✧ **The cost of accommodating changing customer requirements is reduced.**
  - The amount of analysis and documentation that has to be redone is much 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.
- ✧ **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.



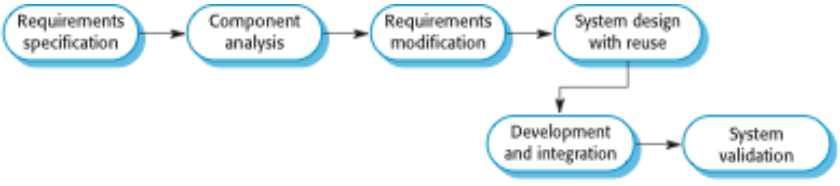
Incremental development problems :

- ✧ **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.

Reuse-oriented software engineering : Based on systematic reuse where systems are integrated from existing components or COTS (Commercial-off-the-shelf) systems.

Reuse-oriented software engineering process stages (phases) :

- Component analysis;
- Requirements modification;
- System design with reuse;
- Development and integration.



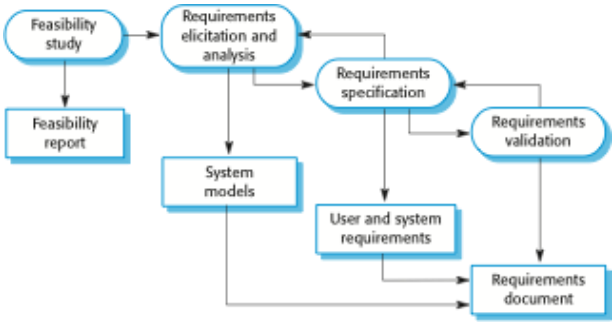
Types of software component :

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

The four basic process activities 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 **inter-leaved.**

Software specification : is The process of establishing what services are required and the constraints(القيود) on the system's operation and development.

- ✧ **Requirements engineering process**
  - **Feasibility study (دراسة جدوى)**
    - Is it technically and financially feasible to build the system? (هل مقدور عليها)
  - **Requirements elicitation and analysis (استحضار و تحليل المتطلبات)**
    - What do the system stakeholders require or expect from the system? (وش يتوقعون المستثمرين)
  - **Requirements specification**
    - Defining the requirements in detail
  - **Requirements validation**
    - Checking the validity of the requirements



Design activities :

1) Architectural design : is where you identify the overall structure of the system, the principal components (sometimes called **sub-systems or modules**), their relationships and how they are distributed.

2) Interface design *:* is where you define the interfaces between system components.

3) Component design : is where you take each system component and design how it will operate.

4) Database design : is where you design the system data structures and how these are to be represented in a database.

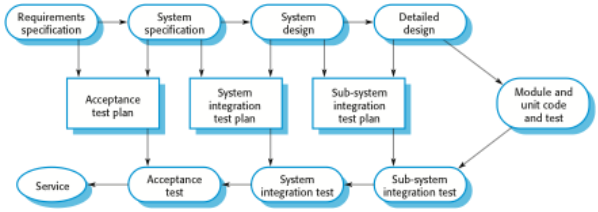
Software verification and validation :

- ✧ 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**.
- ✧ 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.

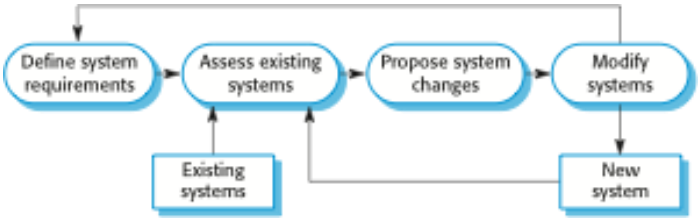
Testing stages :

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

Testing phases in a plan-driven software process :



System evolution :



- **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.

Coping with change :

- Business changes lead to new and changed system requirements
- New technologies open up new possibilities for improving implementations
- Changing platforms require application changes

- **Change leads to rework so the costs of change include both rework as well as the costs of implementing new functionality**

How to Reduce the costs of rework :

- **Change avoidance**, where the software process includes activities that can anticipate possible changes before significant rework is required. (**Prototype**)
- **Change tolerance**, where the process is designed so that changes can be accommodated at relatively low cost. (**Incremental development**).

Software prototyping :

- ✧ A prototype is an initial version of a system used to demonstrate concepts and try out design options.
- ✧ A prototype can be used in:
    - The requirements engineering process to help with requirements elicitation and validation;
    - In design processes to explore options and develop a UI design;
    - In the testing process to run back-to-back tests.

Benefits of prototyping :

1. Improved system usability.
2. A closer match to users' real needs.
3. Improved design quality.
4. Improved maintainability.
5. Reduced development effort.

Prototype development :

- Prototype should focus on areas of the product that are not well-understood;
- Error checking and recovery may not be included in the prototype;
- Focus on functional rather than non-functional requirements such as reliability and security

Throw-away prototypes :

- ✧ Prototypes should be discarded after development as they are not a good basis for a production system:
    - It may be impossible to tune the system to meet non-functional requirements;
    - Prototypes are normally undocumented;
    - The prototype structure is usually degraded through rapid change;
    - The prototype probably will not meet normal organizational quality standards.

✧ **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 and delivery:

✧ **Incremental development**

- Develop the system in increments and evaluate each increment before proceeding to the development of the next increment;
- Normal approach used in agile methods;
- Evaluation done by user/customer proxy.

✧ **Incremental delivery**

- Deploy an increment for use by end-users;
- More realistic evaluation about practical use of software;
- Difficult to implement for replacement systems as increments have less functionality than the system being replaced.

Incremental delivery 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.

Incremental delivery problems :

✧ Most systems require a set of basic facilities that are used by different parts of the system.

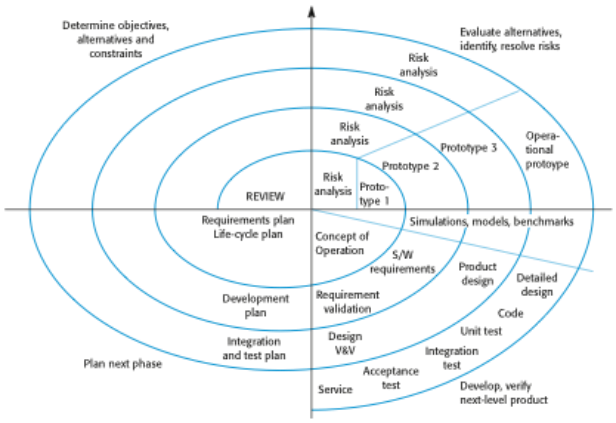
- As requirements are not defined in detail until an increment is to be implemented, it can be hard to identify common facilities that are needed by all increments.

✧ The essence of iterative processes is that the specification is developed in conjunction with the software.

- However, this conflicts with the procurement model of many organizations, where the complete system specification is part of the system development contract.

Boehm's spiral 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.



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 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.

The Rational Unified Process is a modern generic process model that is organized into phases **(inception, elaboration, construction and transition)** but **separates activities** (**requirements, analysis and design, etc**.) from these phases.

The Rational Unified Process :

- A modern generic process derived from the work on the UML and associated process.
- Normally described from 3 perspectives
    - A dynamic perspective that shows phases over time;
    - A static perspective that shows process activities;
    - A practive perspective that suggests good practice

RUP phases :

- Inception
    - Establish the business case for the system.
- Elaboration
    - Develop an understanding of the problem domain and the system architecture.
- Construction
    - System design, programming and testing.
- Transition
    - Deploy the system in its operating environment.

RUP iterations :

- **In-phase iteration**
    - Each phase is iterative with results developed incrementally.
- **Cross-phase iteration**
    - As shown by the loop in the RUP model, the whole set of phases may be enacted incrementally.

Static workflows in the Rational Unified Process (سير العمل) (Activities):

1. **Business modelling** is The business processes are modelled using business use cases
2. **Requirements** are Actors who interact with the system are identified and use cases are developed to model the system requirements.
3. **Analysis and design** is a design model is created and documented using architectural models, component models, object models and sequence models.
4. **Implementation** The components in the system are implemented and structured into implementation sub-systems. Automatic code generation from design models helps accelerate this process.
5. **Testing** is an iterative process that is carried out in conjunction with implementation. System testing follows the completion of the implementation.
6. **Deployment** A product release is created, distributed to users and installed in their workplace
7. **Configuration and change management** This supporting workflow managed changes to the system
8. **Project management** This supporting workflow manages the system development
9. **Environment** This workflow is concerned with making appropriate software tools available to the software development team

RUP good practice:

- ✧ Develop software iteratively
    - Plan increments based on customer priorities and deliver highest priority increments first.
- ✧ Manage requirements
    - Explicitly document customer requirements and keep track of changes to these requirements.
- ✧ Use component-based architectures
    - Organize the system architecture as a set of reusable components.
- ✧ Visually model software
    - Use graphical UML models to present static and dynamic views of the software.
- ✧ Verify software quality
    - Ensure that the software meet's organizational quality standards.
- ✧ Control changes to software
    - Manage software changes using a change management system and configuration management tools.