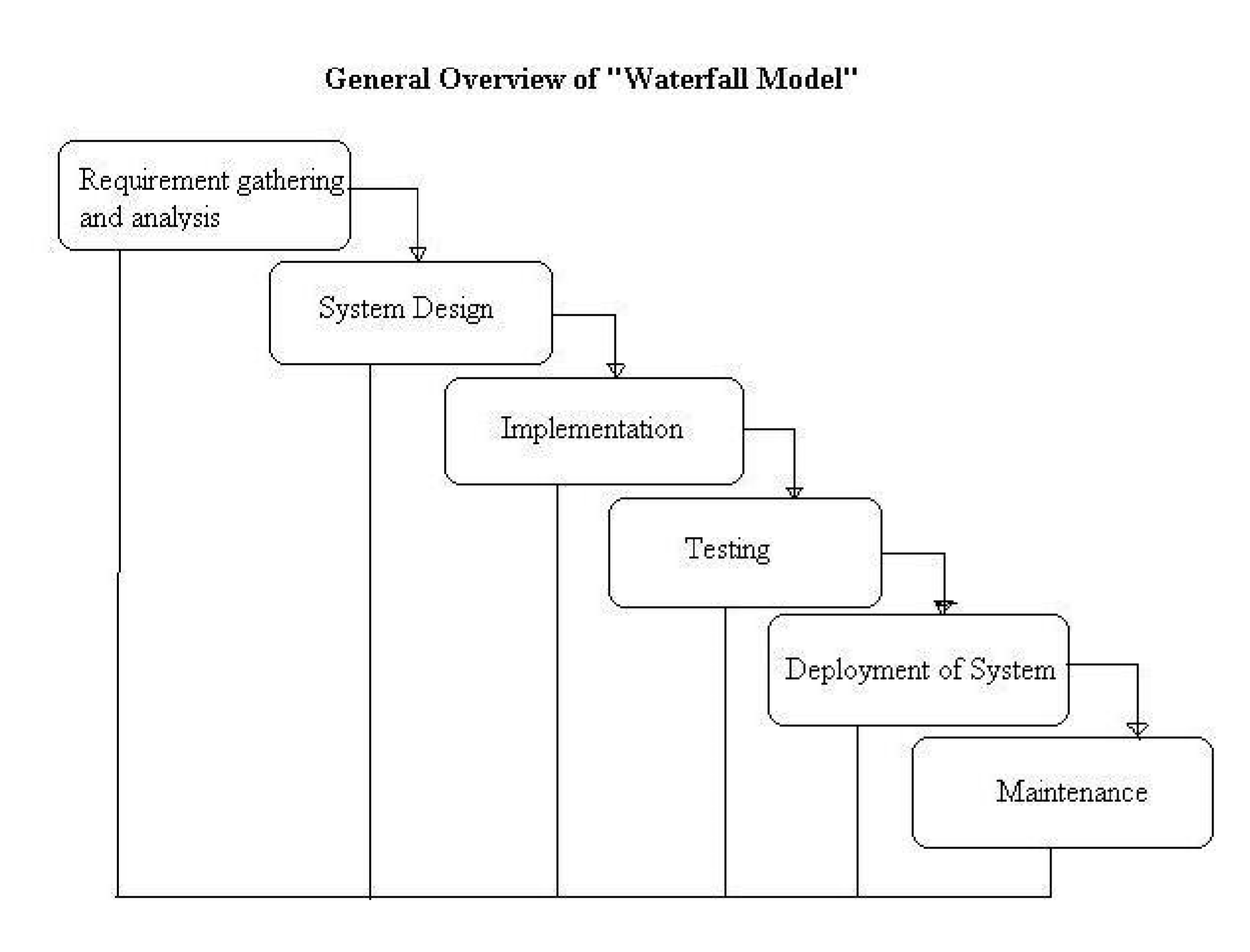
**Chapter4: Implementation**

# 4.1 Model

**Waterfall Model :**



**Figure4.1.1.Waterfall Model**

* Requirements– defines needed information, function, behavior, performance and interfaces.
* Design– The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
* Implementation and unit testing - With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
  + Integration and system Testing **-** All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
  + Deployment of system **-** Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
  + Maintenance **-** There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released.

Strengths of Waterfall Model:

* 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)
* Works well when quality is more important than cost or schedule

Deficiencies of Waterfall Model:

1.Idealised, does not match reality well.

* 1. Doesn’t reflect iterative nature of exploratory development.
  2. Unrealistic to expect accurate requirements so early in project
  3. Software is delivered late in project, delays discovery of serious errors.
  4. Difficult to integrate risk management
  5. Difficult and expensive to make changes to documents,”swimming upstream”.
  6. Significant administrative overhead, costly for small teams and projects.

When to use Waterfall Model:

* Requirements are very well known
* Product definition is stable
* Technology is understood
* New version of an existing product
* Porting an existing product to a new platform.

# 4.2 Data Flow diagram:

**4.2.1.Data Flow Diagram(Level 0):**

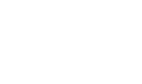
OWNER

USER

A



A



Request for

login

Request for

registration

Response

Response

**Figure 4.1.2 DataFlow**

Drycleaning

And

Laundry

System

A context diagram is a top level (also known as "Level 0") data flow diagram. It only contains one process node ("Process 0") that generalizes the function of the entire system in relationship to external entities. DFD Layers. Draw data flow diagrams can be made in several nested layers.

**2. Data Flow diagram(level 1):**

Wash

Details

Order info

Record

invoice

D1-Customer

customer

Recipt

Transition

Details

Wash

Clothes

Copy

invoice

Transition

Details

order

Details

D2-Transition

**Figure 4.1.3 DataFlow(level1)**

A data flow diagram is a graphical representation of the "flow" of data through an information system, modelling its process aspects. A data flow diagram is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated. DFDs can also be used for the visualization of data processing(structured design).

A DFD shows what kind of information will be input to and output from the system, how the data will advance through the system, and where the data will be stored. It does not show information about process timing or whether processes will operate in sequence or in parallel, unlike a traditional structured flowchart which focuses on control flow, or a UML activity workflow diagram, which presents both control and data flows as a unified model.

# 4.3 Usecase Diagram:

Login

Registration

Maintain clothes detail

Maintain user detail

View clothing detail

View transition details

Place Order

Delivery of order

Feedback

User

DEO

Administrator

**Figure4.1.4.Usecase Diagram**

use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use case[s](https://en.wikipedia.org/wiki/Use_case) in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well.

While a [use case](https://en.wikipedia.org/wiki/Use_Case) itself might drill into a lot of detail about every possibility, a use-case diagram can help provide a higher-level view of the system. It has been said before that "Use case diagrams are the blueprints for your system". They provide the simplified and graphical representation of what the system must act