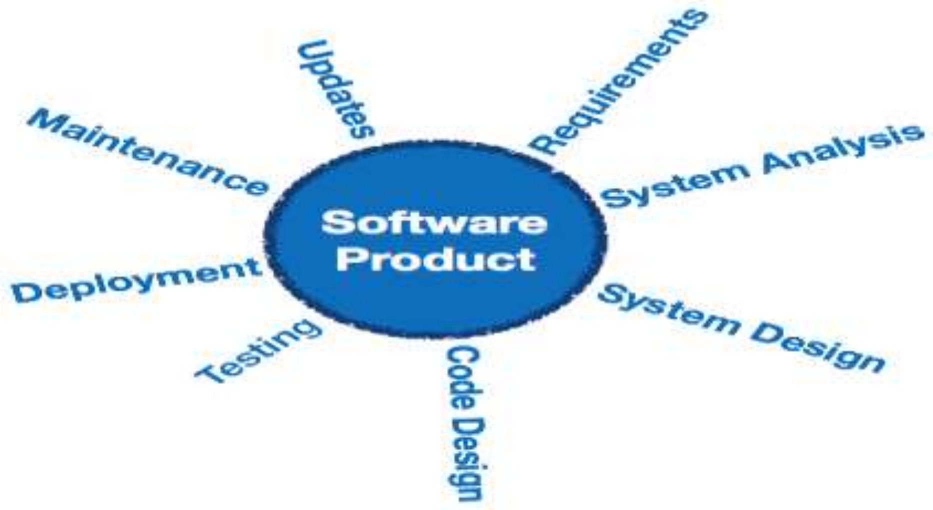
Software Engineering

INTRODUCTION

Introduction

- **Software** is more than just **a program code**. **A program is an executable code**, which serves some computational purpose. Software is considered to be **collection of executable programming code**, associated **libraries** and **documentations**. Software, when made for a specific requirement is called **software product.**
- **Engineering** on the other hand, is all about **developing products**, using **well-defined, scientific principles** and **methods.**



Introduction

- **Software engineering** is an engineering branch associated with **development of software product using well-defined scientific principles, methods** and procedures. The outcome of software engineering is an efficient and reliable software product.
- **Definitions:**
- The application of a **systematic, disciplined, quantifiable approach** to the development, operation and maintenance of software; that is, the application of engineering to software.
- Software engineering is the establishment and use of sound **engineering principles in order to obtain economically software** that is reliable and work efficiently on **real machines.**

Need of software Engineering

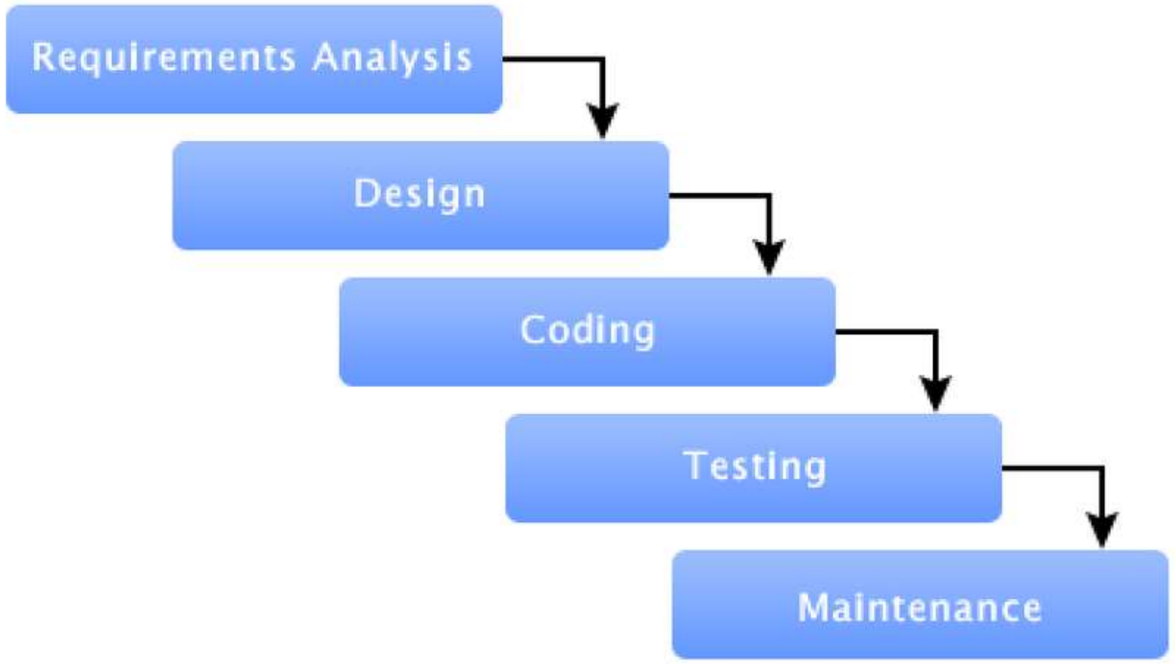
- **Large software:**
- It is **easier to build** a wall than to a **house or building,** likewise, as the **size of software become large engineering.**
- **Scalability:**
- easier to **re-create new software**
- **Cost:**
- **H**ardware, cost of software remains high
- **Dynamic nature :**
- **A**dapting nature of software
- **Quality Management :**
- software development provides **better and quality** software product.

software Engineering Process Paradigms

- **Process model (or)software engineering paradigm**
- Waterfall model
- Prototyping model
- Rapid application Development model

Waterfall model

- **Linear sequential model** or classic life cycle or **waterfall model.**
- Systematic ,sequential approach
- System levels :
- Requirements Analysis
- Design
- Coding
- Testing
- Maintenance



Waterfall model

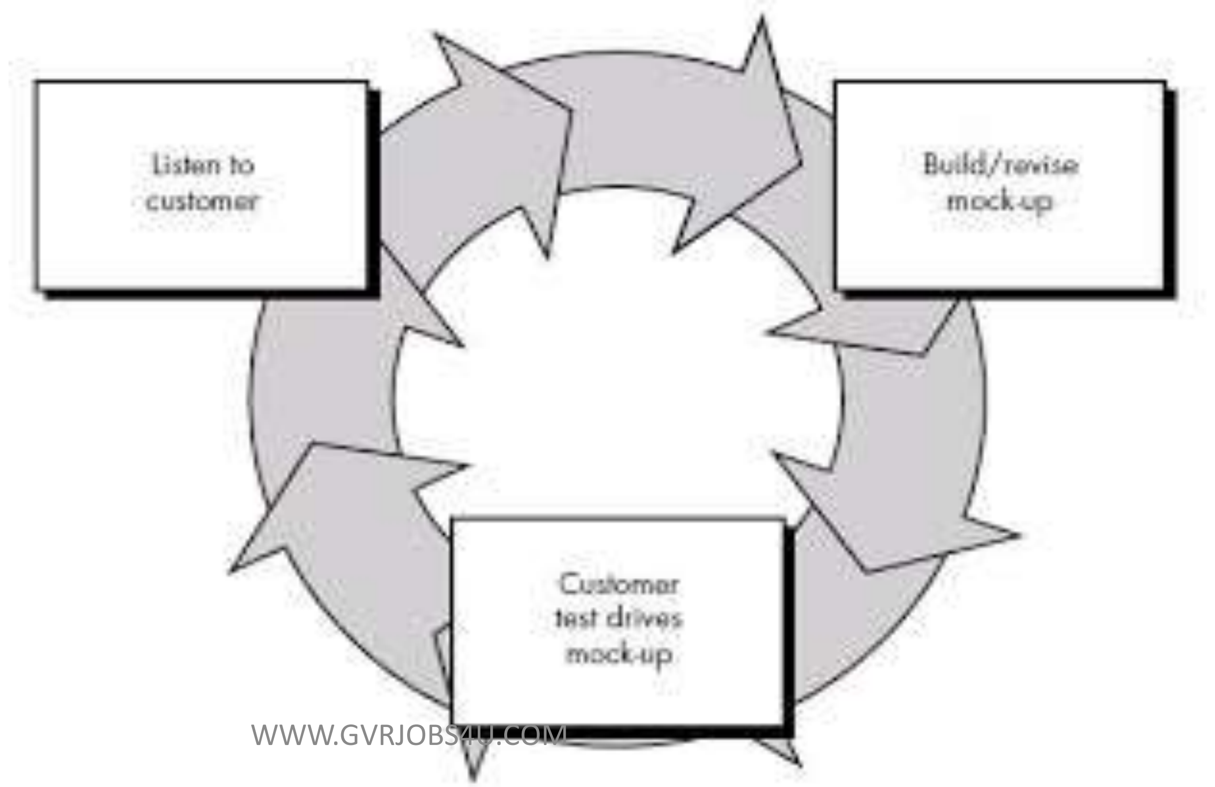
- **Requirements Analysis:**
- **->** information domain ,function ,behavioral.
- **Design** :
- ->Data structure ,software architecture ,interface representation ,algorithmic details.
- **Coding :**
- **->**program ,design is translated into machine readable form.
- **Testing:**
- **->** uncover errors ,fix the bugs ,execution of all paths.
- **Maintenance:**
- **->**longest life cycle phase.

Waterfall model

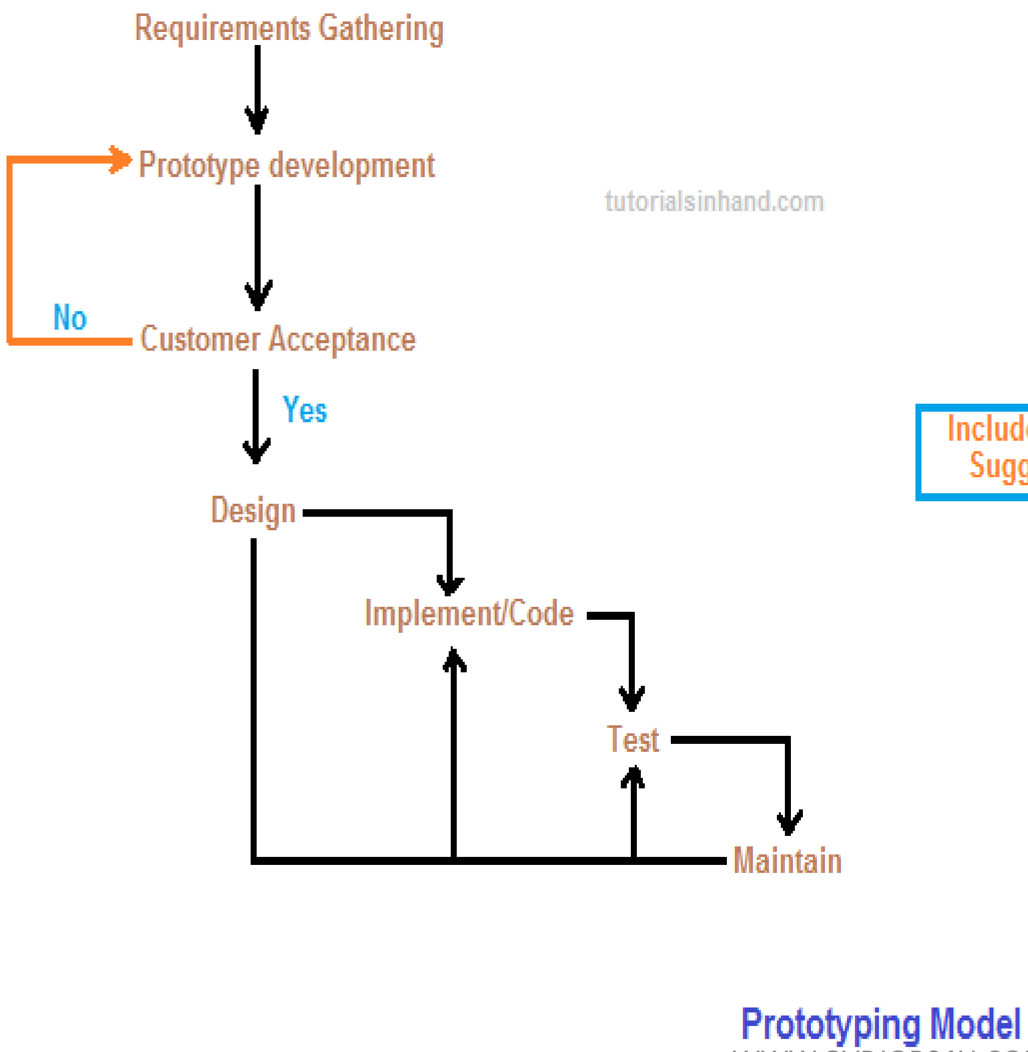
- **Advantages :**
- **Simple and easy to understand**
- Each phase has specific **deliverables and review process**
- Phases are processed and **complete one at a time .phase don’t overlap.**
- **Smaller projects.**
- **Disadvantages :**
- Once an application is in the testing stage ,it is **very difficult to go back and change.**
- Produced **until late during the life cycle.**
- **High amount** of risk and uncertainty
- Not good for **complex and object oriented projects**
- **Poor model** for long projects
- Not suitable for moderate to **high risk of changing.**

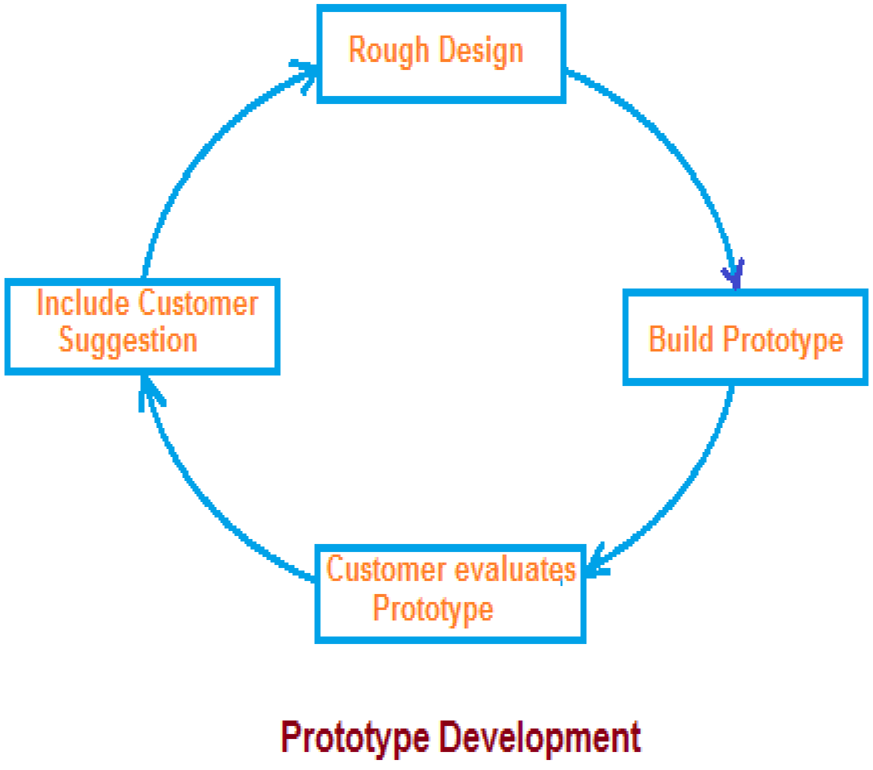
Prototyping Model

- Prototyping paradigm begins with **requirements gathering.**
- **Developer and customer meet** and define the overall objectives for the software , identity whatever requirement are known and **outline areas** .
- A **quick design** then occurs.



Prototyping Model





Prototyping Model

- **Advantages:**
- **User are actively involved** in the development
- Errors can be detected **much earlier**
- Quicker user **feedback is available** leading better solutions
- Missing functionality can be **easily identified**
- Users get a **better understanding**
- **Disadvantages:**
- **Repairing** way of building system
- Practically **may increase the complexity** of the system
- **Incomplete applications** may cause applications.