GLS UNIVERSITY
FACULTY OF COMPUTER APPLICATIONS & INFORMATION TECHNOLOGY

222304501 SOFTWARE ENGINEERING

UNIT -1 SOFTWARE ENGINEERING – AN INTRODUCTION

Introduction to Software Engineering

Software Process

Specification

Design and Implementation

Validation

Evolution

Process ISO Standards

Software Development Life Cycle Models

Waterfall Model

Prototyping Model

Software Development Challenge

Introduction to CI/ CD (Continuous Integration/ Continuous Delivery Deployment)

Introduction to Jenkins CI

INTRODUCTION TO SOFTWARE ENGINEERING

- **Software** is a program or set of programs containing instructions that provide desired functionality.
- Engineering is the process of designing and building something that serves a particular purpose and finds a cost-effective solution to problems.

What is Software Engineering?

- **Software Engineering** is the process of designing, developing, testing, and maintaining software.
- It is a systematic and disciplined approach to software development that aims to create high-quality, reliable, and maintainable software.

SOFTWARE PROCESS

- A software process is the set of activities and associated outcome that produce a software product. Software engineers mostly carry out these activities. These are four key process activities, which are common to all software processes. These activities are:

1. **Software specifications:** The functionality of the software and constraints on its operation must be defined.
2. **Software Design & Implementation:** The software to meet the requirement must be produced.
3. **Software validation:** The software must be validated to ensure that it does what the customer wants.
4. **Software evolution:** The software must evolve to meet changing client needs.

PROCESS ISO STANDARDS

- ISO (International Organization for Standardization) is a worldwide federation of national standards bodies.
- ISO is a nongovernmental organization that comprises standards bodies from more than 160 countries, with one standards body representing each member country.
- For example, the American National Standards Institute represents the United States.
- ISO members are national standards organizations that collaborate in the development and promotion of international standards for technology, scientific testing processes, working conditions, societal issues and more.

POPULAR ISO STANDARDS

- **ISO 27000** : These security standards define a process for developing and implementing information security policies and processes.
- **ISO 17799** : This security management standard specifies more than 100 best practices for business continuity, access control, asset management and more.
- **ISO 20000** : This ISO standard creates a technical specification and codifies best practices for IT service management.
- **ISO 12207 :** This ISO standard creates a consistent lifecycle management process for all software.
- **ISO 9000** : This family of standards defines how organizations can establish and maintain effective quality assurance systems for manufacturing and service industries.

WHY SHOULD BUSINESS BE ISO CERTIFIED?

- **Regulatory Requirements :** To meet the common standards, it is essential that the products of the business are certified.
- **Commercial standards :** Products and services that are certified to meet minimum standards are required by some industries when certification is not a regulatory requirement.
- **Customer Requirements :** Some customers like government agencies may prefer or require certification, even where there is an industry-standard or regulatory requirement for certification.
- **Improved Consistency :** Consistent quality assurance can be delivered across business units as well as across international borders with the help of the certification.
- **Customer Satisfaction :**A consistent performance by the product or the service is appreciated by the customers of the enterprise. The certified organization can also resolve customer issues by their compliance with standards.

SOFTWARE DEVELOPMENT LIFE CYCLE MODELS

- A software development is a structure imposed on the development of a software product.
- There are several models that describes the approaches or the way different tasks are done during the software development process.
- These models are called “**Software Development Life Cycle**” models.
- In the physical sense, software does not wear out.
- It has no replacement (spare) parts.
- However, still the software has a lifespan.

SOFTWARE DEVELOPMENT LIFE CYCLE MODELS

- The system for which the software is made and the system’s environment is never static.
- These keep changing due to development in technology and market condition.
- When the system is changed significantly, it may not be possible to update or maintain the software any longer.
- Hence, the software may be considered to have a lived its lifespan.
- This kicks of new software and this cycle continues.
- Hence, software development process models are also called as “**Software Development Life cycle** “ models.

WATERFALL MODEL

- The waterfall model proposed in 1970 is a traditional model for software development.
- It is sequential process model.
- According to this model the software development project is executed in some steps or phases.
- Waterfall model is also referred to as a linear-sequential life cycle model.
- In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

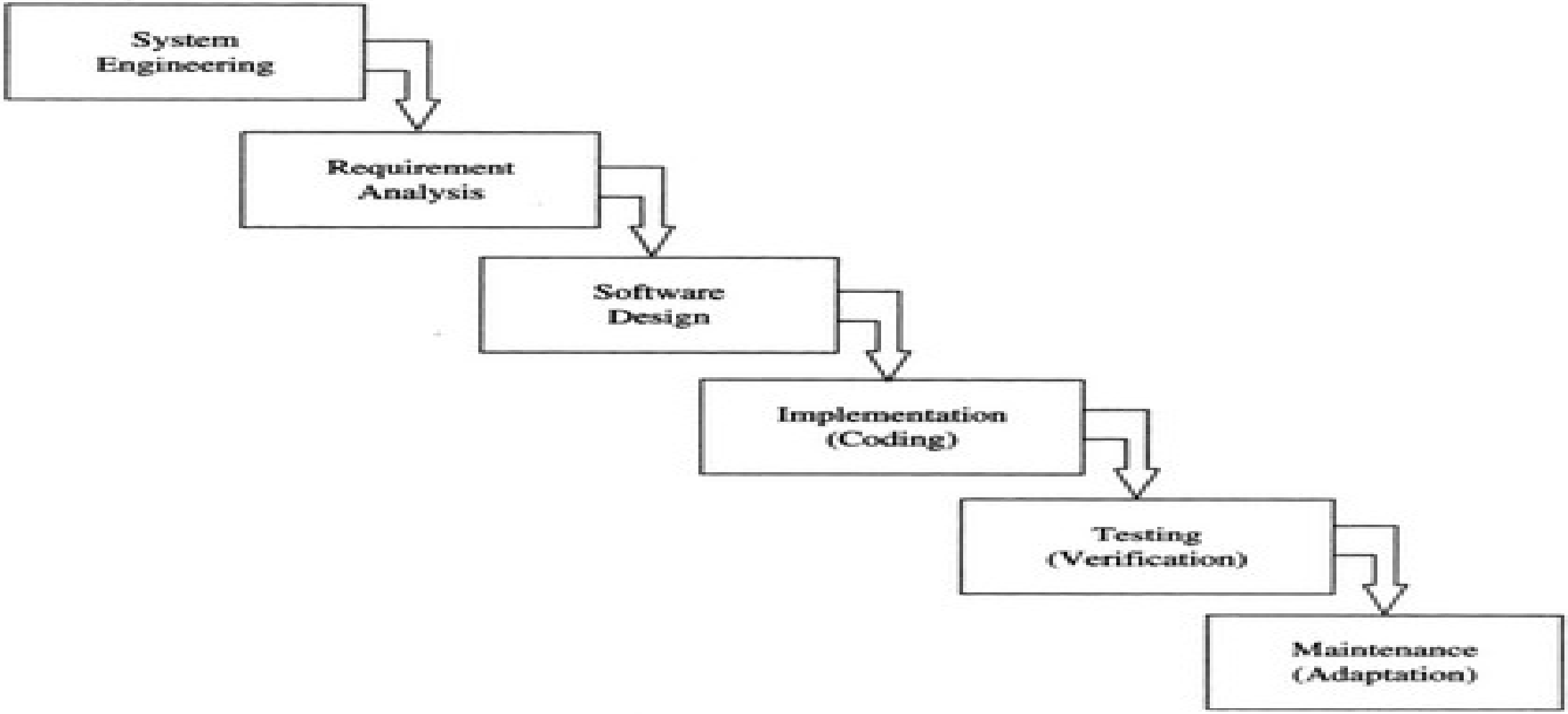
WATERFALL MODEL

- The waterfall Model illustrates the software development process in a linear sequential flow.
- This means that any phase in the development process begins only if the previous phase is complete.
- In this waterfall model, the phases do not overlap.

WATERFALL MODEL - DESIGN

- Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project.
- In "The Waterfall" approach, the whole process of software development is divided into separate phases.
- In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.
- The following illustration is a representation of the different phases of the Waterfall Model.

WATERFALL MODEL - DESIGN



WATERFALL MODEL -DESIGN

- The sequential phases in waterfall model are –
  - **System Engineering –** The software to be developed is always a part of a larger system. Hence, system engineering is the first stage in which requirements for the larger system are established.
  - **Requirement gathering and analysis** – requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
  - **System 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 -** 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.

WATERFALL MODEL - DESIGN

- **Integration and 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. Maintenance is done to deliver these changes in the customer environment.

WATERFALL MODEL APPLICATIONS

- Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors.
- Some situations where the use of Waterfall model is most appropriate are −
  - Requirements are very well documented, clear and fixed.
  - Product definition is stable.
  - Technology is understood and is not dynamic.
  - There are no ambiguous requirements.
  - Ample resources with required expertise are available to support the product.
  - The project is short.

WATERFALL MODEL - ADVANTAGES

- Some of the major advantages of waterfall are as follows –
  - Simple and easy to understand and use
  - Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
  - Phases are processed and completed one at a time.
  - Works well for smaller projects where requirements are very well understood.
  - Clearly defined stages.
  - Well understood milestones.
  - Easy to arrange tasks.
  - Process and results are well documented.

WATERFALL MODEL - DISADVANTAGES

- Some of the major disadvantages of waterfall model are as follows –
  - No working software is produced until late during the life cycle.
  - High amounts of risk and uncertainty.
  - Not a good model for complex and object-oriented projects.
  - Poor model for long and ongoing projects.
  - Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty is high with this process model.
  - It is difficult to measure progress within stages.
  - Cannot accommodate changing requirements.

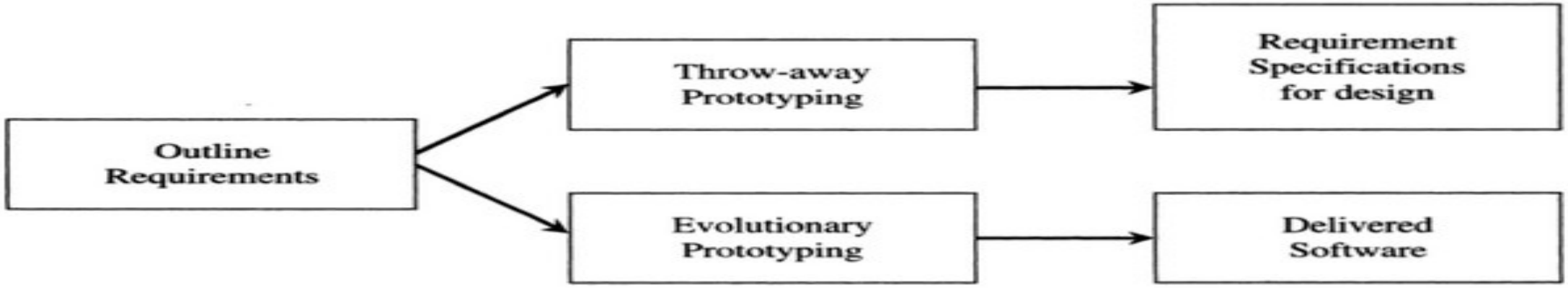
PROTOTYPING MODEL

- The users have a general view of what is expected from the software product.
- They are able to tell the general objectives of the project but not the detailed requirements such as various inputs, processing needs and output.
- Sometimes there is a communication gap between the users and the software developers because of the defferences in their professional background
- This problem is solved by the prototype model.

PROTOTYPING MODEL

- There are basically two main approaches to prototyping models.

1) Throw-away prototyping
2) Evolutionary prototyping



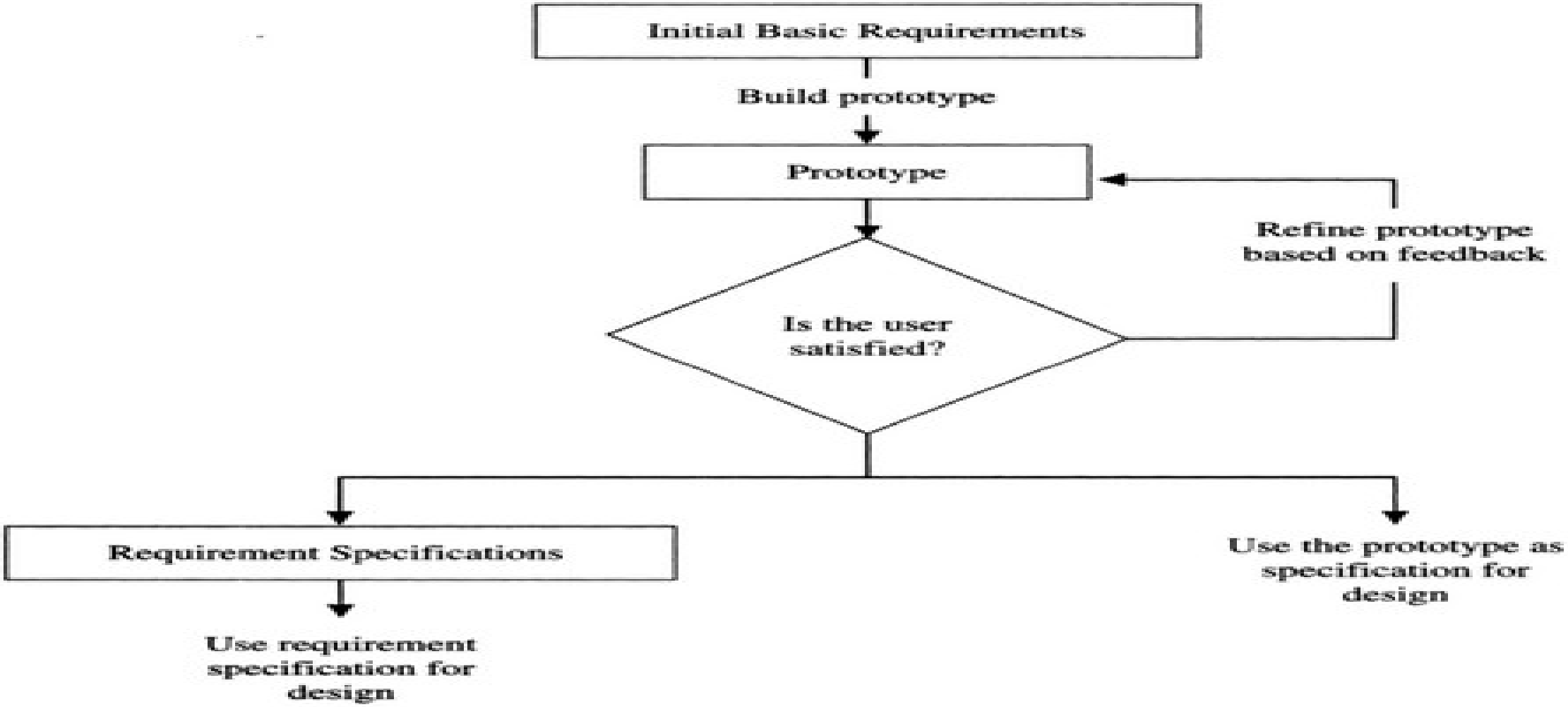
PROTOTYPING MODEL – THROW-AWAY PROTOTYPING

- In throw-away prototyping, a prototype is produced to help in determining the software requirement specification and then discarded.
- The software is then developed based on software requirement specification that was determined with the help of the prototype.

PROTOTYPING MODEL – EVOLUTIONARY PROTOTYPING

- Evolutionary prototype is used for the systems where the specification cannot be developed in advance.
- Verification is not possible in such cases for want of documented specifications.
- Hence, it may be better to develop software using an evolutionary approach rather than first build a throw-away prototype and build software the second time.

PROTOTYPING MODEL



PROTOTYPING MODEL – ADVANTAGES

- Instead of concetrating on documentation, more effort is placed in creating the actual software.
- The software is created by using lots of feedbacks from users. Every prototype is created to get the views and opinions about the software from users.
- Since the users are involved in software development, there is a greater chance of the resulting software being more acceptable to them..
- If something is unfavorable, it can be changed. The software is created with the customer in mind.

PROTOTYPING MODEL – DISADVANTAGES

- Often users feel that a few minor changes to the prototype will suffice for their needs. They do not realise that the prototype is meant only for the demonstration purposes and it is not the real software. A lot of other work such as algorithm design, coding, debugging and quality-related activities are done for developing the real software.
- The developers may lose focus on the real purpose of the prototype and compromise on the equality of the product. For example, they may retain some of the inefficient algorithms or inappropriate programming codes of prototype in the real software.
- A prototype has no legal standing in the event of any dispute. Hence, the protoype as the software specification is used only as a tool for a software development and not as a part of the contract.

INTRODUCTION TO CI/CD

- CI/ CD stands for Continuous Integration/ Continuous Delivery/ Deployment).
- CI And CD is the practice of automating the integration of code changes from multiple developers into a single codebase.
- It is a software development practice where the developers commit their work frequently to the central code repository (Github or Stash).

INTRODUCTION TO CI/CD

- Then there are automated tools that build the newly committed code and do a code review, etc as required upon integration.
- The key goals of Continuous Integration are to find and address bugs quicker, make the process of integrating code across a team of developers easier, improve software quality, and reduce the time it takes to release new feature updates.

INTRODUCTION TO CI/CD

- Some popular CI tools are Jenkins, TeamCity, and Bamboo.
- With Continuous Integration, developers frequently commit to a shared common repository using a version control system such as Git.
- We can configure the CI pipeline to be triggered every time there is a commit/merge in the codebase.

INTRODUCTION TO JENKINS CI

- Some popular CI tools are Jenkins, TeamCity, and Bamboo.
- With Continuous Integration, developers frequently commit to a shared common repository using a version control system such as Git.
- We can configure the CI pipeline to be triggered every time there is a commit/merge in the codebase.