SOFTWARE ENGINEERING

Anchor & co: Sowmya BP, Archana A & Anusha GB

Department of Computer Applications

SOFTWARE ENGINEERING

Introduction to Software Engineering

Anchor & co: Sowmya BP, Archana A & Anusha GB
Department of Computer Applications

PES UNIVERSITY

Why Software Engineering

- Software engineering exists to ensure that software systems are reliable, scalable, maintainable, and aligned with business and user needs.
- As software underpins critical functions across industries, informal or ad-hoc development approaches are insufficient.
- Software engineering provides the discipline required to manage complexity and risk.

PES UNIVERSITY

What is Software Engineering

Software engineering is the systematic, disciplined, and measurable application of engineering principles to the development, operation, maintenance, and evolution of software systems.

Software Engineering Example

Software engineering can be explained using a student management system. First, requirements are gathered, such as storing student details, recording marks, and generating reports. Next, the system is designed by deciding how data will be stored and how users will interact with the software. Then, developers write the code to implement these features. The software is tested to ensure it works correctly and produces accurate results. Finally, the system is deployed and maintained to fix errors and add new features. This shows how software engineering follows a clear, step-by-step process to build reliable software.

PES UNIVERSITY

Purpose of studying Software Engineering

1. To understand systematic methods for designing, developing, and maintaining software.
2. To manage complexity in large and real-world software systems effectively.
3. To improve software quality through proper testing, documentation, and standards.
4. To develop teamwork, communication, and project management skills.
5. To build reliable, scalable, and maintainable software that meets user and business needs.

PES UNIVERSITY

Course Objectives:

The objective of the course is to

- Understand the concepts of software processes and process models along with agile software development method.
- Introduce software requirements and the processes involved in discovering and documenting these requirements.
- Introduce some types of system model that may be developed as a part of requirements engineering and system design.
- Understand the stages of testing, during development to acceptance by customers.

PES UNIVERSITY

Course Outcomes:

At the end of the course, the student should be able to

- Apply these process model to plan software projects effectively and efficiently develop a quality software.
- Gather and document software requirements.
- Design software using appropriate notations and techniques.
- Generate test cases to test software and ensure its quality.

PES UNIVERSITY

Unit-I Software Processes and Agile software development

Introduction to software engineering, Software process models, Process activities, coping with change, the rational unified process. Agile software development – Plan driven and agile development, Extreme programming.

Experiential Learning: Case study on software processes.

14+7 Sessions

PES UNIVERSITY

Unit-II Requirements engineering

Functional and non functional requirements, the software requirement document, Requirements engineering processes, requirements elicitation and analysis, requirements validation. Requirements analysis through flow-oriented approach -Data flow diagram.

Experiential Learning: Build SRS document. Practice exercises on data flow diagrams.

14+7 Sessions

PES UNIVERSITY

Unit-III System Modeling and Design

System modelling –Interaction model- use case diagram, sequence diagram, Structural models – class diagrams, Behavioural models – state diagrams.

Architectural design – design decisions, architectural views, architecture patterns

Experiential Learning: Hands-on exercise on use case diagrams, sequence diagrams, class diagrams and state diagrams

14+7 Sessions

PES UNIVERSITY

Unit-IV Software Testing

Introduction, Development testing – unit testing, choosing unit test cases, component testing, system testing, test-driven development (TDD), Release testing – Requirements-based testing, Scenario testing, Performance testing, User testing – Acceptance testing.

Experiential Learning: Test case generation based on scenarios.

14+7 Sessions

PES UNIVERSITY

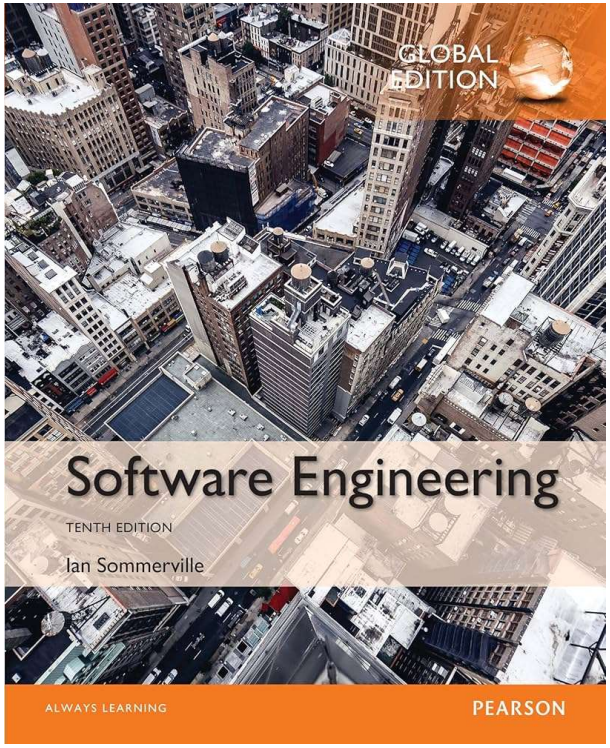
TextBook

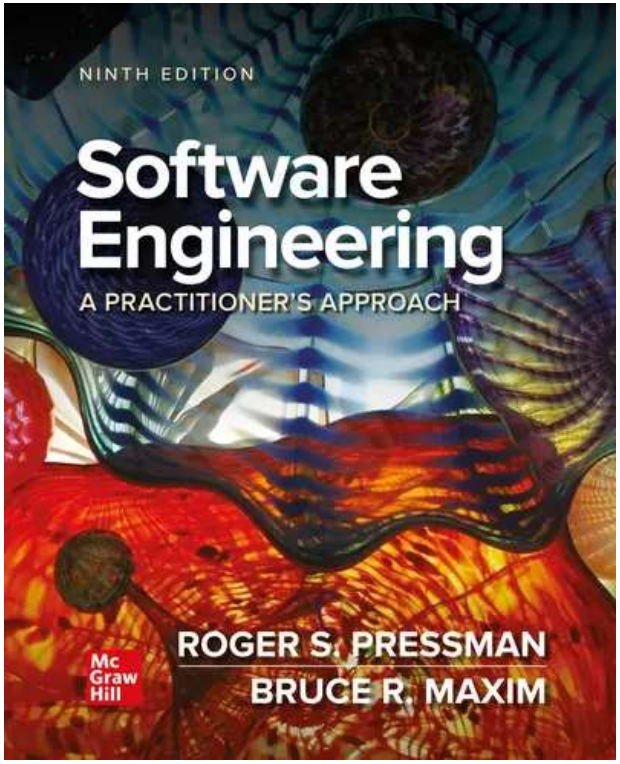
1. Software Engineering, Ian Sommerville, 10th Ed., Pearson Education Ltd., Reprint 2019 - No latest edition/reprint available.

Reference Book:

1. “Software Engineering – A Practitioner’s Approach”, Roger S Pressman, Tata McGraw Hill, 9th Edition, 2020

PES UNIVERSITY





ISA Pattern

Total Units: 4

Total ISA = 2, ISA-1=Unit1&2. ISA-2=Unit3&4

Test mode= Hybrid

Type of Questions	No of questions	Marks	Expected time to answer the question	Total Marks
MCQ / True or False	16 (including theory / assignment/case study/ experiential learning)	1 mark each	1 min each	16
Short Answer	4	2 marks each	3 mins each	8
Long Answer	4	4 marks each	6 mins each	16
Time for reviewing the answer			8 mins	
Total			1 Hour	40 Marks

SOFTWARE ENGINEERING

Anchor & co: Sowmya BP, Archana A & Anusha GB

Department of Computer Applications

SOFTWARE ENGINEERING

Introduction to Software Engineering

Anchor & co: Sowmya BP, Archana A & Anusha GB
Computer Applications

PES UNIVERSITY

Definition of Software and Software Engineering

We can **define software** as

- Software is a **product,** that software professionals build and then support over the long term.

- Software engineering encompasses
  - a **process**,
  - a collection of **methods (practice)** and
  - an array of **tools** that allow professionals to build high-quality computer software.

Types of Software Products

Software engineers are concerned with developing software products, that is, software that can be sold to a customer.

There are **two kinds** of software product:

1. **Generic products** These are stand-alone systems that are produced by a development organization and sold on the open market to any customer who is able to buy them.
- Examples of this type of product include **apps for mobile devices**, **software for PCs such as databases, word processors, drawing packages**, etc.

2. Customized (or bespoke) software –

These are systems that are commissioned by and developed for a particular customer.

- A software contractor designs and implements the software especially for that customer.

- **Examples** of this type of software include **control systems for electronic devices**, **systems written to support a particular business process**, and **air traffic control systems.**

Essential **attributes/characteristics** of good software

1. **Acceptability** - Software must be acceptable to the **type of users** for which it is designed. This means that it must be **understandable, usable**, and **compatible** with other systems that they use.

2. Dependability and security –

- Software dependability includes a range of characteristics including **reliability, security, and safety**.
- Software has to be secure so that **malicious users cannot access or damage** the system.

3. Efficiency

- Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, resource utilization, etc.

4. Maintainability

- Software should be written in such a way that it can **evolve to meet the changing needs of customers**.
- This is a critical attribute because software change is an inevitable requirement of a changing business environment.

PES UNIVERSITY

Software Engineering

Software engineering is an **engineering discipline** that is concerned with **all aspects of software production** **from the early stages of system specification** through to **maintaining the system** after it has gone into use.

In this definition, there are two key phrases:

1. **Engineering discipline** - Engineers make things work. They **apply** **theories, methods, and tools** where these are appropriate.
2. **All aspects of software production** - Software engineering is not just concerned with the technical processes of software development. It also **includes activities** such as **software project management** and **the development of tools**, **methods**, and **theories** to support software development.

software engineering **methods and techniques** are used to develop different type of application. They are :

1. **Stand-alone applications** – These are application systems that run on a personal computer or apps that run on a mobile device

Eg: **MS office applications on a PC**, **CAD programs**, **photo manipulation software**, **travel apps**, productivity apps, and so on

2. **Interactive transaction-based applications** – These are applications that execute on a remote computer and that are accessed by users from their own computers,phones, or tablets. Eg: e-commerce applications

3. Embedded control systems – These are software control systems that control and manage hardware devices. software in a microwave oven to control the cooking process

4. **Batch processing systems** – These are business systems that are designed to process data in large batches. Eg: periodic billing systems, such as phone billing systems, and salary payment systems

5. Entertainment systems – These are systems for personal use that are intended to entertain the user. Eg: handle **audio and video, video gaming, home theatre system** etc.

6. Systems for modeling and simulation –

These are systems that are developed by scientists and engineers to model physical processes or situations, which include many separate, interacting objects.

Eg: **weather forecasting, flight simulators used for training pilots** etc.

7. Data collection and analysis systems – Data collection systems are systems that collect data from their environment and send that data to other systems for processing.

Eg: **Big data analysis** may involve cloud-based systems carrying out **statistical analysis** and looking for relationships in the collected data.

8. Systems of systems – These are systems, used in enterprises and other large organizations, that are composed of a number of other software systems.

Eg: **ERP** system (Enterprise resource planning) – It is a software system that includes all the tools and processes required to run a successful company.

1. Confidentiality You should normally respect the confidentiality of your employers or clients regardless of whether or not a formal confidentiality agreement has been signed.

2. Competence You should not misrepresent your level of competence. You should not knowingly accept work that is outside your competence.

3. Intellectual property rights You should be aware of local laws governing the use of intellectual property such as patents and copyright. You should be careful to ensure that the intellectual property of employers and clients is protected.

4. Computer misuse You should not use your technical skills to misuse other people's computers. Computer misuse ranges from relatively trivial (game playing on an employer's machine) to extremely serious (dissemination of viruses or other malware).

SOFTWARE ENGINEERING

Anchor & co: Sowmya BP, Archana A & Anusha GB

Department of Computer Applications

SOFTWARE ENGINEERING

Introduction to Software Engineering

Anchor & co: Sowmya BP, Archana A & Anusha GB
Computer Applications

SOFTWARE ENGINEERING

What is SDLC ?

Anchor & co: Sowmya BP, Archana A & Anusha GB
Department of Computer Applications

What is SDLC ?

- SDLC is the abbreviation used for **Software Development Life Cycle**. It is also Known as Software Development Process.

- SDLC is a description of phases in the life cycle of a software application.

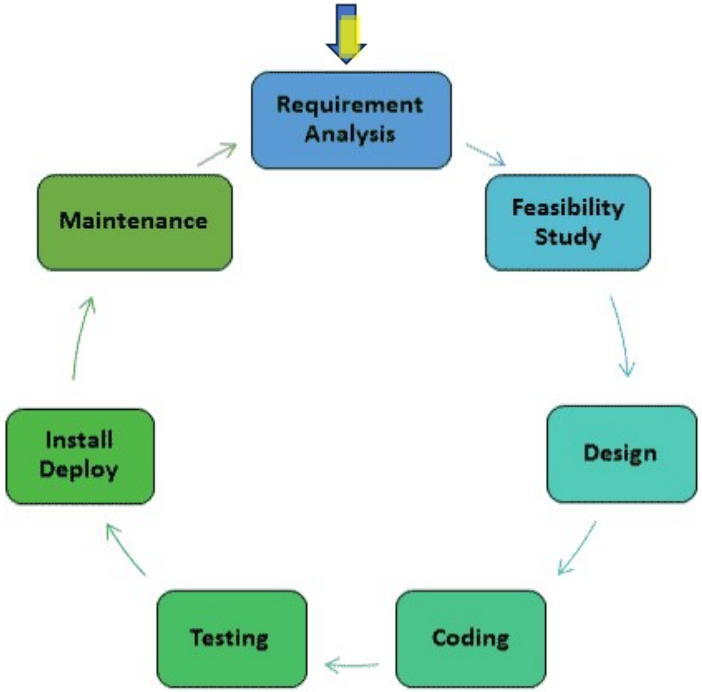
- It consists of a detailed plan as how to **develop**, **build** and **enhance** a specific software. Each phase of the SDLC lifecycle has its own process and deliverables that feed into the next phase.

.

SDLC Phases

Various phases of Software Development Life Cycle (SDLC) are:

- **Requirement Analysis**
- **Feasibility Study**
- **Design**
- **Coding**
- **Testing**
- **Install/Deploy**
- **Maintenance**



What is SDLC ?

- **Requirement Analysis**

  Stakeholders' needs are gathered, analyzed, and documented. This phase defines what the system must do and results in formal requirement specifications (e.g., SRS).

- **Feasibility Study**

  Evaluates the practicality of the project from technical, economic, operational, legal, and schedule perspectives. The outcome determines whether the project should proceed.

What is SDLC ?

- **Design**

  Translates requirements into a detailed system architecture and design. This includes high-level design (system architecture) and low-level design (modules, data structures, interfaces).

- **Coding (Implementation)**

  Developers write source code based on the design specifications, following coding standards and best practices.

- **Testing**

The software is verified and validated to ensure it meets requirements and is defect-free. This includes unit testing, integration testing, system testing, and acceptance testing.

- **Installation / Deployment**

The tested software is released into the production environment. This may involve configuration, data migration, user training, and rollout planning.

- **Maintenance**

Post-deployment activities to fix defects, improve performance, adapt to environment changes, and add enhancements. This phase typically consumes the longest portion of the software's lifecycle.

Example: Online Food Delivery Application

- **Requirement Analysis**

  Users should be able to browse restaurants, place orders, make online payments, and track delivery.

- **Feasibility Study**

  Analyze development cost, server infrastructure, payment gateway integration, and expected user demand.

- **Design**

  Design mobile app screens, database for users, orders, and restaurants, and the order-processing workflow.

- **Coding (Implementation)**

  Implement user login, restaurant listing, order placement, payment processing, and delivery tracking features.

- **Testing**

  Test that payments are processed securely and orders are not placed without successful payment.

- **Installation / Deployment**

  Deploy the app on cloud servers and publish it on app stores.

- **Maintenance**

  Fix bugs, improve performance during peak hours, and add features like discounts or loyalty programs.

SOFTWARE ENGINEERING

Anchor & co: Sowmya BP, Archana A & Anusha GB

Department of Computer Applications

SOFTWARE ENGINEERING

Introduction to Software Engineering

Anchor & co: Sowmya BP, Archana A & Anusha GB
Computer Applications

SOFTWARE ENGINEERING

Software Process

Anchor & co: Sowmya BP, Archana A & Anusha GB
Department of Computer Applications

- A process is a collection of **activities, actions and tasks** that are performed when some work product is to be created.
- **An activity** try towards achieving the broad objective regardless of specific domain.
- **Ex: communication with stakeholders – gathering requirements from different categories of users** (need not be all technically knowledgeable)

An *action* *encompasses* a **set of tasks** that produce a major work product.

- **Ex—Building an architectural design model – which includes various set of tasks in it.**

A *task* *focuses on a small, but well-defined objective* that produces a tangible outcome

- **Ex--*conducting a unit* test – which includes micro level testing**

A software process is a **set of related activities** that leads to the production of a software system.

The **four fundamental software engineering activities** are:

1. **Software specification:** The **functionality of the software** and **constraints on its operation** must be defined.
2. **Software development:** The software to meet the specification 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 customer needs**.

SOFTWARE ENGINEERING

Software Process Model

Anchor & co: Sowmya BP, Archana A & Anusha GB
Department of Computer Applications

Software Process Model

Software process model is a simplified representation of a software process. Each process model represents a **process from a particular perspective.**

The Generic process models are:

1. **Waterfall model,**
2. **Incremental model,**
3. **Prototyping,**
4. **Spiral model,**
5. **Rational Unified Process model.**

These are also known as **Traditional Process Models**

SOFTWARE ENGINEERING

1. Waterfall Model

Anchor & co: Sowmya BP, Archana A & Anusha GB
Department of Computer Applications

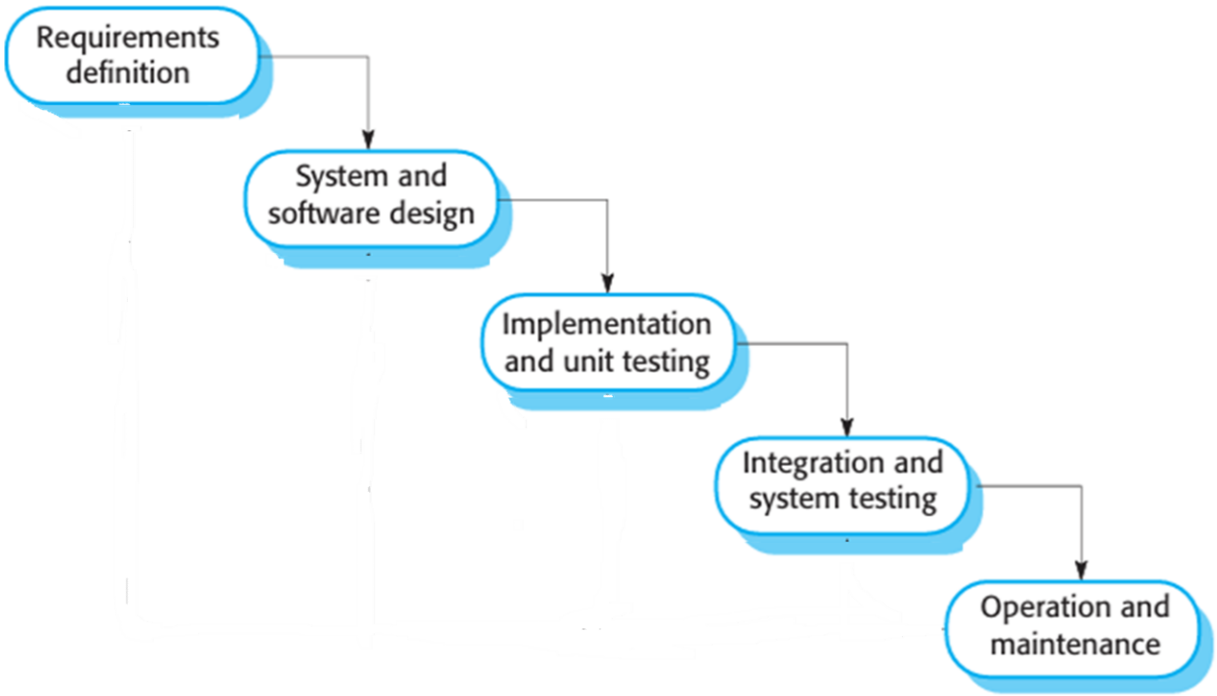
PES UNIVERSITY

The waterfall model

- This takes the fundamental process activities of **specification, development, validation, and evolution** and represents them **as separate process phases** such as :
- **Requirements specification,**
- **Software design,**
- **Implementation,**
- **Testing and**
- **Maintenance**.

Waterfall Model

Waterfall model is the first published model of the software development process. It presents the software development process as a number of stages, as shown in Figure



Waterfall Model

The stages of the waterfall model directly reflect the fundamental software development activities:

1. Requirements analysis and definition: The system's services, constraints, and goals are established by **consultation with system users**. They are then defined in detail and serve as a system specification.

2. System and software design: The systems design process allocates the requirements to either hardware or software systems.

- It establishes an **overall system architecture**.
- Software design involves identifying and describing the fundamental software system abstractions and their relationships.

3. Implementation and unit testing: During this stage, 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**.

4. Integration and system testing: The individual program units or programs are **integrated** and tested as a complete system to ensure that the software requirements have been met. **After testing, the software system is delivered to the customer.**

5. Operation and maintenance: Normally, this is the longest life-cycle phase. The system is installed and put into practical use.

Maintenance involves **correcting errors** that **were not discovered in earlier stages** of the life cycle, **improving the implementation** of system units, and **enhancing the system's services** as **new requirements** are discovered.

Waterfall Model

- All these phases are performed sequential to each other in which progress is seen as **flowing steadily downwards** (like a waterfall) through these phases.

- The next phase is started **only after** the defined set of goals are achieved for previous phase and it is signed off, so the name "**Waterfall Model**".

- In this model, **phases do not overlap** and **no backtracking** possible.

Advantages of Waterfall Model

The major **advantages** of the Waterfall Model are as follows −

1. Simple and easy to understand and also to use.
2. Easy to manage, Well understood milestones and Clearly defined stages.
3. **The Phases** are processed and completed **one at a time**.
4. Works well for **smaller projects** where requirements are **very well understood**.
5. Process and results are well documented.

Limitations of Waterfall Model

The major **disadvantages** of the Waterfall Model are as follows −

1. **No working software** is produced until late during the life cycle.
2. Not a good model for **complex and object-oriented projects**.
3. **Poor model for long and ongoing projects**.
4. Not suitable for the projects where requirements are at a moderate to high risk of changing.
5. Cannot measure the project progress during the development process.

SOFTWARE ENGINEERING

Anchor & co: Sowmya BP, Archana A & Anusha GB

Department of Computer Applications

PES UNIVERSITY

SOFTWARE ENGINEERING

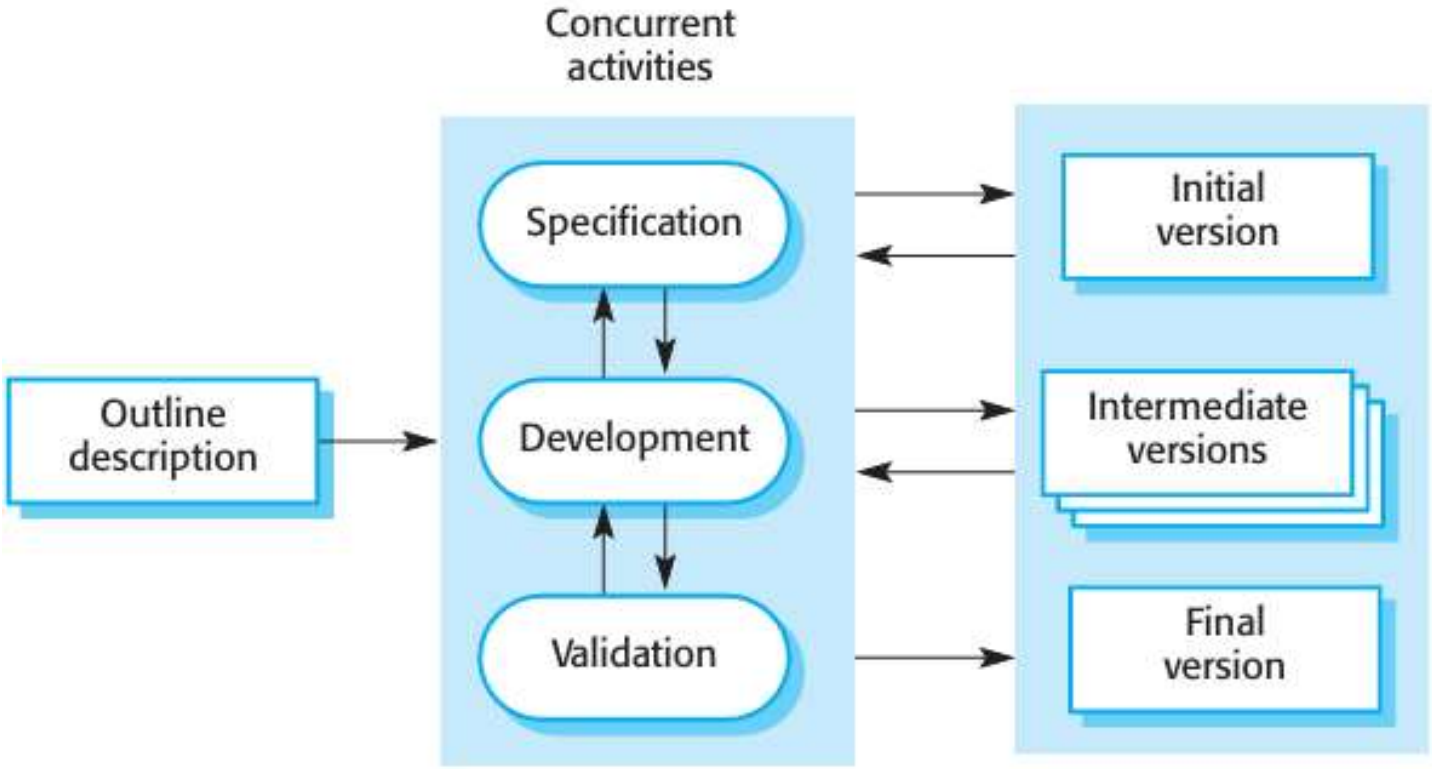
2. Incremental Development Model

Anchor & co: Sowmya BP, Archana A & Anusha GB
Department of Computer Applications

2. Incremental Model

- The incremental model delivers a **series of releases**, called **increments.**
- 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
- Incremental development is based on the idea of **developing an initial implementation**, **getting feedback from users** and others, and **evolving the software through several versions** until the required system has been developed.
- **Specification, development**, and **validation** activities are **interleaved** rather than separate, with rapid feedback across activities.

Incremental Model



Incremental Model

Incremental development in some form is now the most common approach for the development of application systems and software products.

This approach can be either plan-driven, agile or, more usually, a mixture of these approaches.

- In a plan-driven approach, the system increments are identified in advance;
- if an agile approach is adopted, the early increments are identified, but the development of later increments depends on progress and customer priorities.

Incremental software development, is **better than a waterfall** approach for systems whose **requirements** are **likely to change during the development process**.

Incremental Model

By developing the software **incrementally,** it is **cheaper and easier** to **make changes** in the software.

Each increment or version of the system incorporates some of the functionality that is needed by the customer.

- **First Increment** is often **core product:**
  - Includes **basic requirement,**
  - **Many supplementary features delivered in later increments**

- A plan of **next increment** is prepared
  Modifications of the first increment,
  Additional features of the first increment

Incremental Delivery

- Incremental delivery is an approach to software development where some of the **developed increments are delivered to the customer** and **deployed for use in their working environment.**

- In an incremental delivery process, **customers define which of the services are most important and which are least important** to them.

- Once the system increments have been identified, the requirements for the services to be delivered in the first increment are defined in detail and that increment is developed

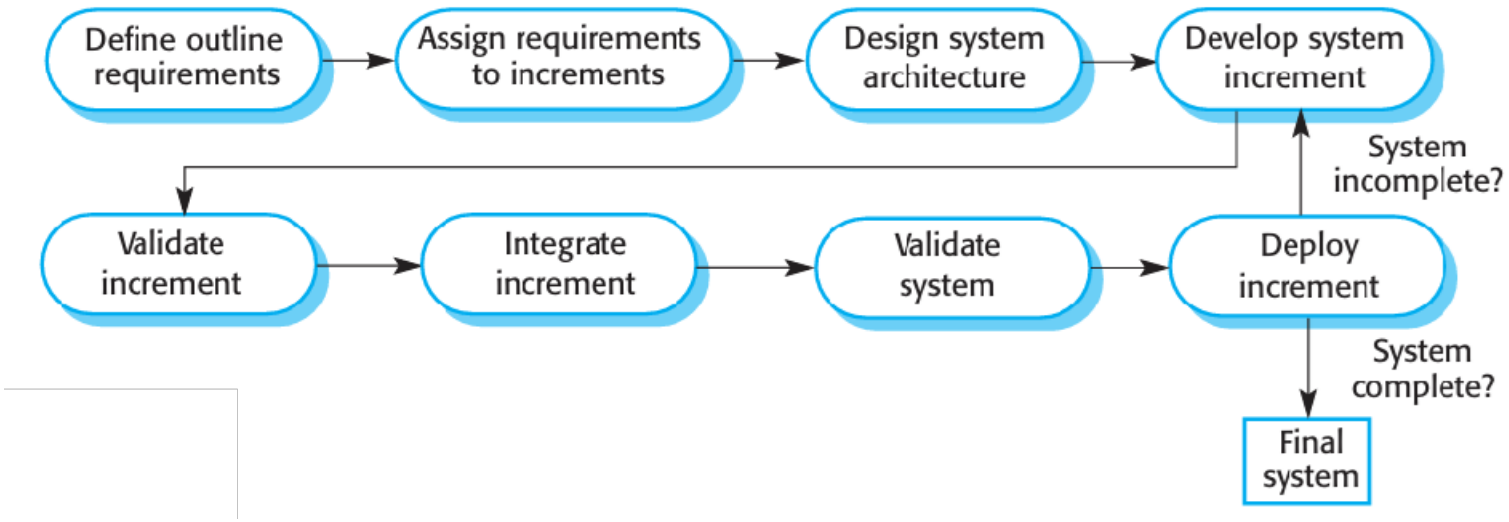
Incremental Model

During development, further requirements analysis for later increments can take place, but **requirements changes for the current increment are not accepted.**

Once an **increment is completed and delivered**, it is **installed in the customer's normal working environment**. As new increments are completed, they are **integrated with existing increments** so that system functionality improves with each delivered increment.

Incremental Model

Working of Incremental delivery is as shown below:



Incremental development has **three major advantages** over the waterfall model:

1. Implementation cost is **reduced.** The amount of analysis and documentation that has to be redone **is significantly less** than is required with the waterfall model.

2. 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.

3. **Early delivery** and **deployment of useful software** to the customer is possible, even if all of the functionality has not been included. Customers are able to use and gain value from the software earlier than is possible with a waterfall process

Incremental Model - disadvantages

From management perspective there are **two problems** with the model:

1. 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.

2. **System structure tends to degrade as new increments are added.**
   - Regular change leads to messy code as new functionality is added in whatever way is possible.
   - It becomes increasingly difficult and costly to add new features to a system.

SOFTWARE ENGINEERING

Anchor & co: Sowmya BP, Archana A & Anusha GB

Department of Computer Applications

SOFTWARE ENGINEERING

3. Prototyping Model

Archana A
Department of Computer Applications

3. Prototyping Model

A prototype is an **early version of a software system** that is used to **demonstrate concepts**, **try out design options**, and find out more about the **problem and its possible solutions.**

- System prototypes allow potential users to see **how well the system supports their work**.
  - They may get new ideas for requirements and find areas of strength and weakness in the software.
  - They may then propose new system requirements.

- Furthermore, as the prototype is developed, it may **reveal errors and omissions** in the system requirements.

PES UNIVERSITY

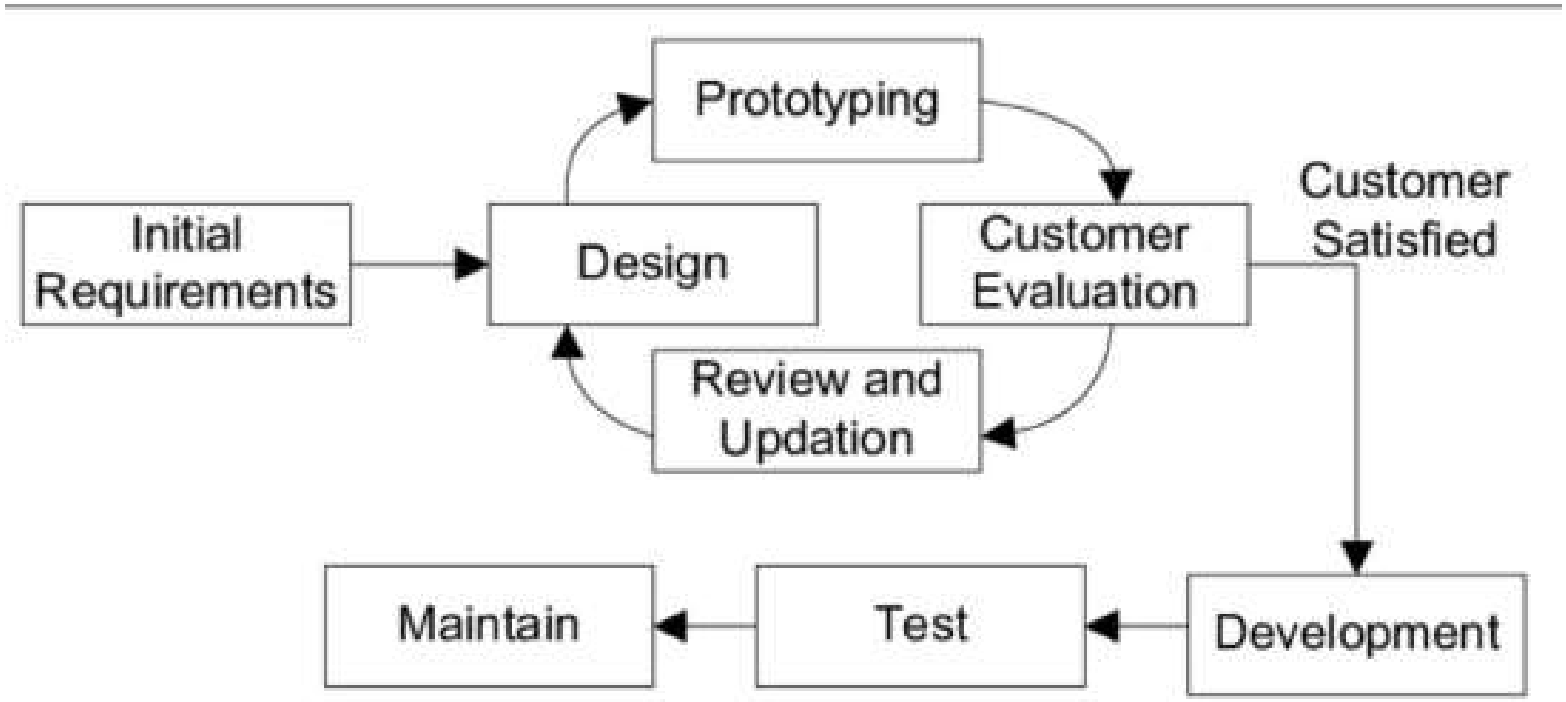
- **What is Prototyping?**

- **Prototyping is the process of quickly putting together a working model (a prototype) in order to test various aspects of a design, illustrate ideas or features and gather early user feedback**

- Prototyping is often treated as an integral part of system design process, where it is believed to reduce project risk and cost.

- **Prototyping model suggests that before carrying out development of actual software, a working prototype of the system is built.**

Prototyping Model

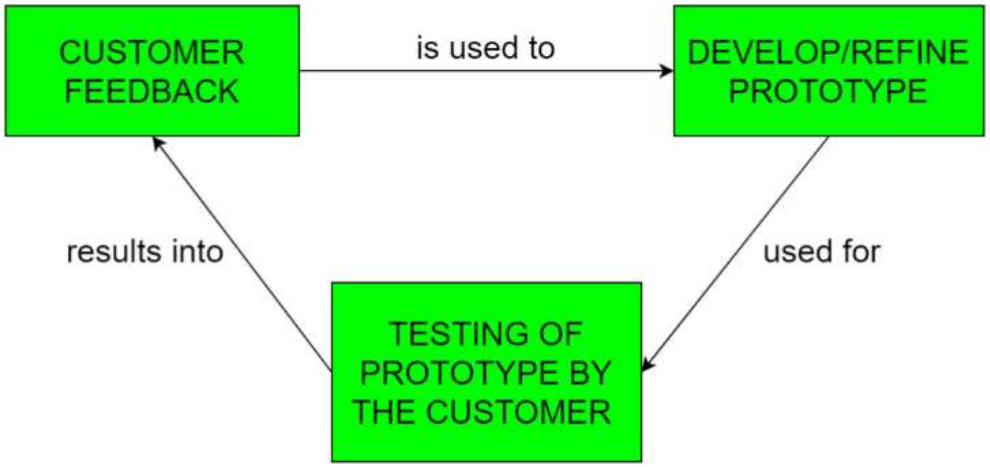


Prototype Model Phases:

The main focus of this model is to provide a system with overall functionality. The prototype model in software engineering has **six main phases**.

They are:

1. **Requirements and gathering**
2. **Quick design**
3. **Build a Prototype**
4. **Customer Evaluation**
5. **Refining Prototype**
6. **Engineering Product**



Phase 1: Requirements and gathering

In this phase, the requirements of the project are gathered from the customers. During this process**, customers have explained their expectations of the product**.

Phase 2: Quick Design

In this phase, **preliminary designs** of the system are created. It **explains the brief idea of the project to the customer**. It is created the simple design of the project.

Phase 3: Building Prototype

In this phase, the actual work of the design is started based on the **quick design** phase. This is required for a **small working model.**

Phase 4: Customer evaluation

In this phase, the enhanced system is served to the customers for an **initial evaluation**.

- It is required because it helps to find out the **pros and cons** of the working model. Also **collected the comments from the customers** and given them to the development team.

Phase 5: Refining prototype

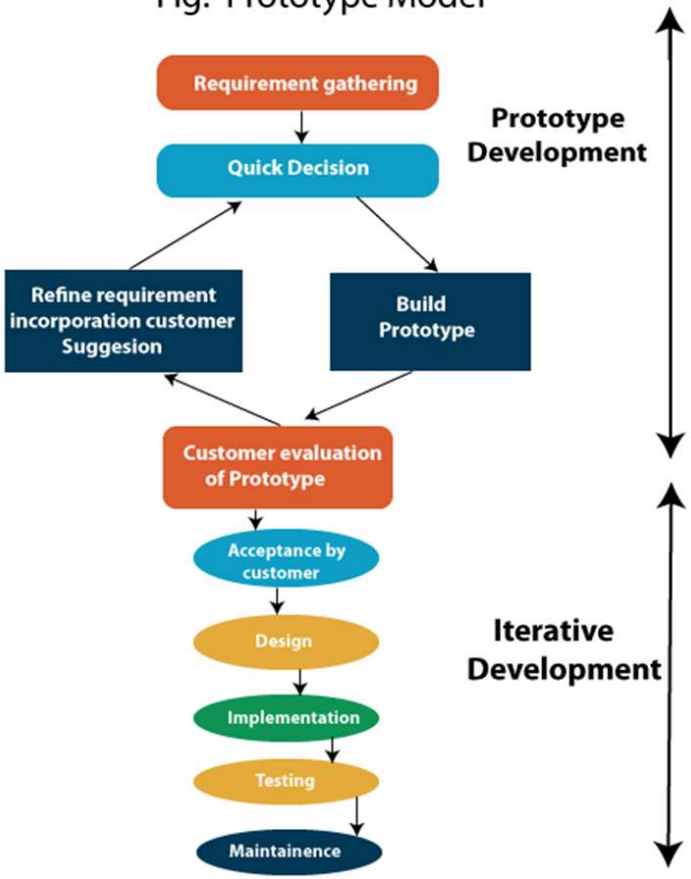
In this phase, If the customer is **not happy** with the prototype, it is **necessary to refine the prototype** according to the **customer's comments and feedback**. So when the customer will satisfy with the prototype, the system will be developed based on the approved prototype.

Phase 6: Engineering Product

In this last phase, the final product is developed based on the final prototype. It is **released on production after thoroughly testing the product**.

Prototyping Model - Summary

Fig: Prototype Model



PES UNIVERSITY

Prototype model advantages :

1. **Customers** are **actively involved** in the **development phase**.
2. Defects can be **detected at early**
3. **Confusion or difficulty in the requirements** can be identified
4. In this model, **missing features can be identified easily**.

Disadvantages are:

1. In this model, It is a **time taking process**.
2. It is **difficult to manage** due to the **client's on-demand changes**.
3. Documentations are **not prepared** well due to changes in the requirements.
4. Sometimes customers are not involved in the iteration cycles.

SOFTWARE ENGINEERING

Anchor & co: Sowmya BP, Archana A & Anusha GB

Department of Computer Applications

PES UNIVERSITY

SOFTWARE ENGINEERING

4. Spiral Model

Archana A
Department of Computer Applications

Spiral Model

A **risk-driven** **software process framework** (the spiral model) was proposed by **Boehm** (1988).

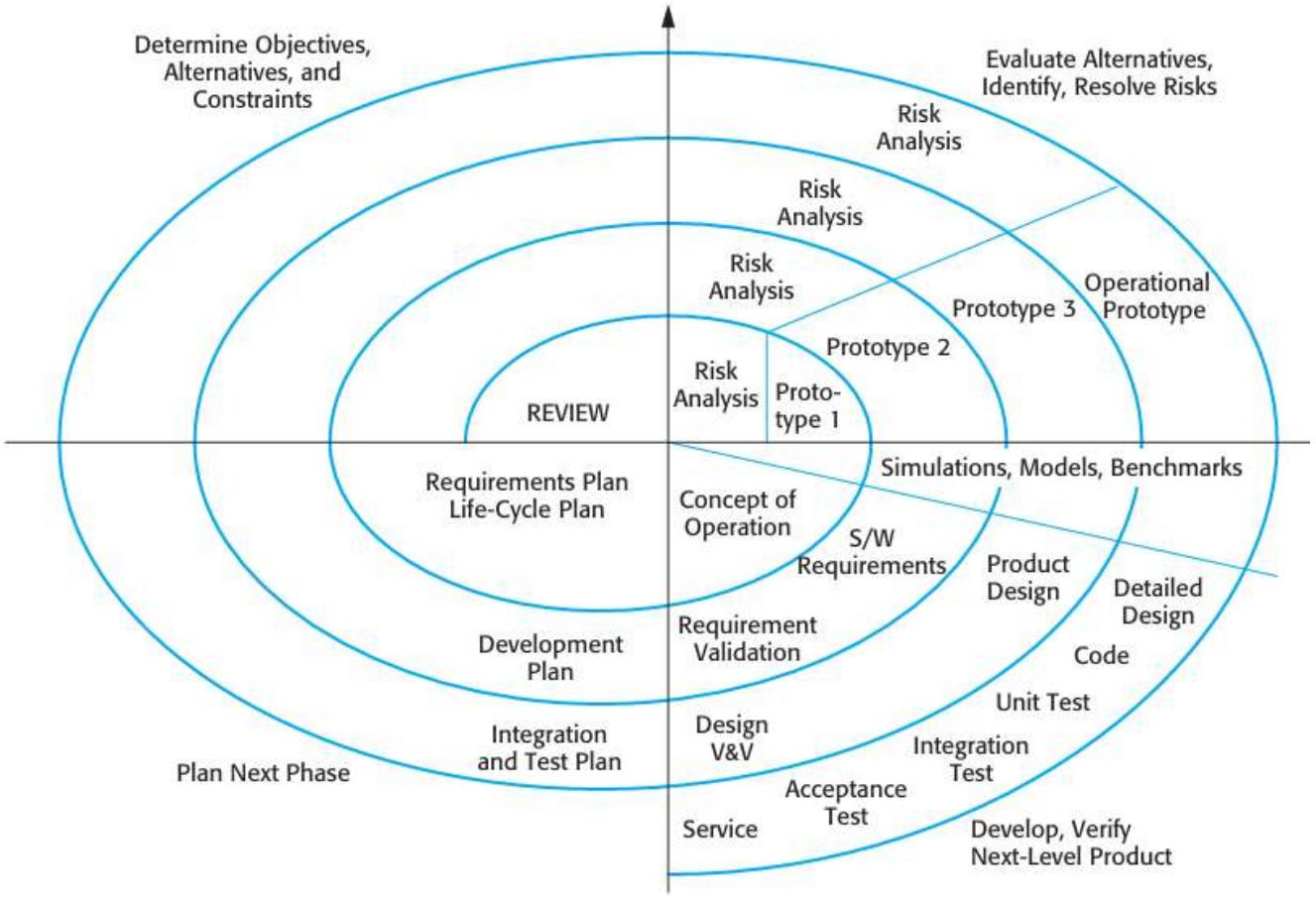
- Here, the **software process** is represented as a **spiral**, rather than a **sequence of activities** with some **backtracking** from one activity to another.

- Each loop in the spiral represents **a phase of the software process.** Thus, the **innermost** loop might be concerned with **system feasibility**, the **next loop with requirements definition**, the next loop **with system design**, and so on.

- The spiral model combines **change avoidance** with **change tolerance**. It assumes that changes are a **result of project risks** and includes explicit risk management activities to reduce these risks.

4. Spiral Model

:



Each loop in the spiral is split into **four sectors**:

1. Objective setting Specific objectives for that phase of the project are defined. Constraints on the process and the product are identified and a detailed management plan is drawn up. **Project risks are identified**. Alternative strategies, depending on these risks, may be planned.

2. Risk assessment and reduction For each of the identified project risks, a detailed analysis is carried out. Steps are taken to **reduce the risk**.

- For example, if there is a risk that the requirements are inappropriate, **a prototype** system may be developed

3. Development and validation After risk evaluation, a development model for the system is chosen.

- For example, **prototyping** may be the best development approach if **user interface risks** are dominant. If **safety risks** are the main consideration, development based on **formal transformations** may be the most appropriate process, and so on. If the main identified risk is sub-system integration, the waterfall model may be the best development model to use.

4. Planning The project is **reviewed and a decision made** whether to continue with a further loop of the spiral. If it is decided to continue, plans are drawn up for the next phase of the project.

4. Spiral Model

- The spiral model is **similar to the incremental model**, with more **emphasis** placed on **risk analysis.**
- The spiral model can be **adapted to apply throughout the entire life cycle** of an application, from concept development to maintenance.
- Spiral **models uses prototyping** as a **risk reduction mechanism** but, more important, enables the developer to apply the prototyping approach at each stage in the evolution of the product.

4. Spiral Model

- It maintains the **systematic stepwise approach** suggested by the classic life cycle (i.e, waterfall) but **also incorporates** it into an **iterative framework activity (**prototype approach**).**

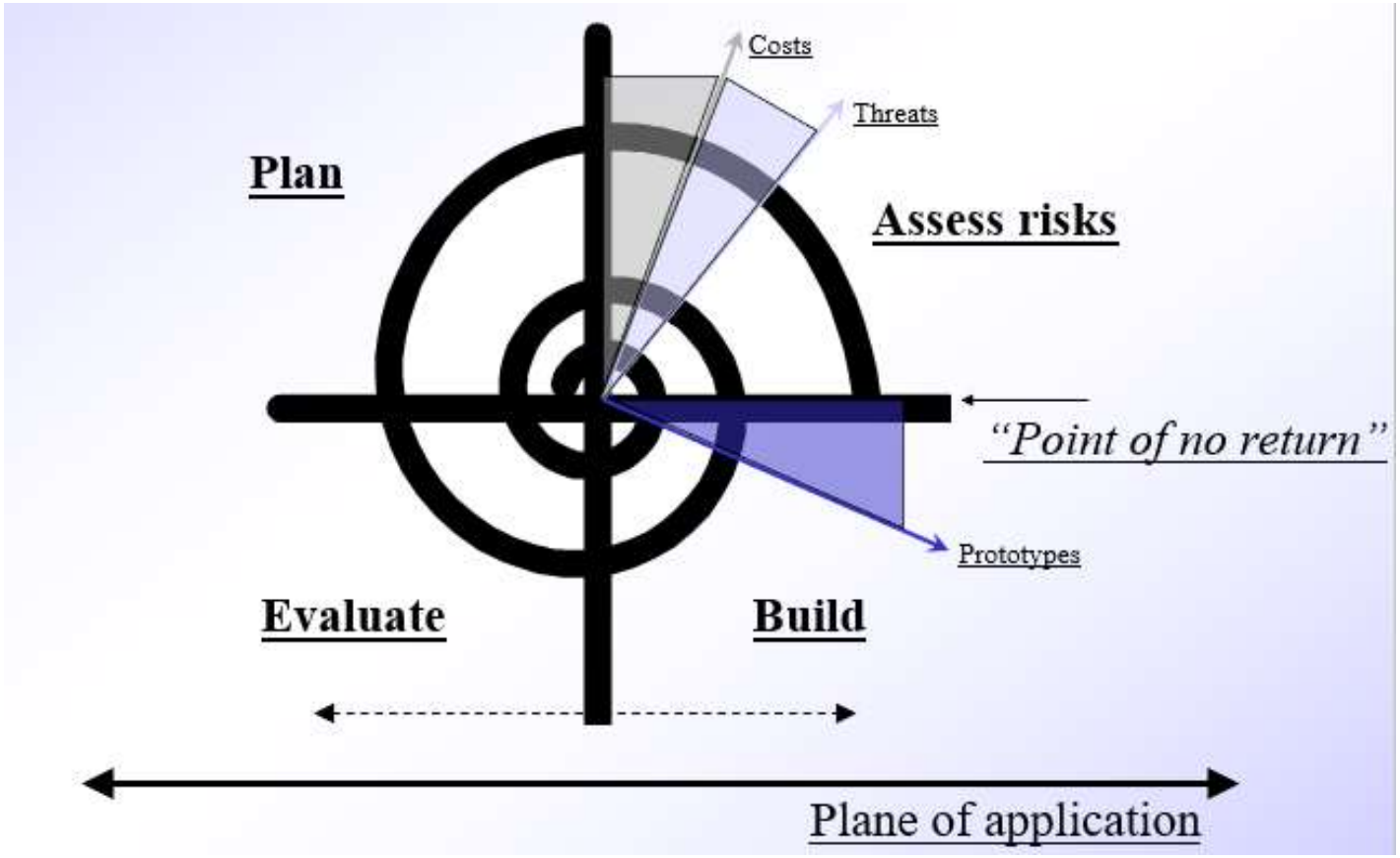
4. Spiral Model in detail

Spiral areas are:

- **Planning (Specification)**
  - Getting requirements
  - Project planning (based on initial requirements.)
  - Project planning (based on customer evaluation.)
- **Risk analysis**
  - Cost/Benefit and threats/opportunities analysis
  - Based on initial requirements and later on customer feedback
- **Engineering (implementation)**
  - Preview it
  - Do it
- **Customer evaluation (Validation)**

SOFTWARE ENGINEERING

4. Evolutionary Process Model - Spiral Model…



- Each phase of Spiral Model is divided into **four quadrants** as (*shown in the previous slide*). The functions of these four quadrants are:

1. Planning:- ***Objectives determination and identify alternative solutions*****.** Requirements are gathered from the customers and the objectives are **identified, elaborated** and **analyzed** at the start of every phase. Then alternative solutions possible for the phase are **proposed** in this quadrant.

2. Risk Analysis:- ***Identify and resolve Risks*****.** During the second quadrant all the possible solutions are evaluated to select the best possible solution.

- Then the **risks** **associated** with that **solution is identified** and the risks are **resolved** using the best possible strategy.

- At the end of this quadrant, Prototype is built for the best possible solution.

3. Engineering :- ***Develop next version of the Product.*** During the third quadrant, the **identified features are developed** and **verified through testing**. At the end of the third quadrant, the next version of the software is available.

4. Customer Evaluation:- ***Review and plan for the next Phase.*** In the fourth quadrant, the **Customers evaluate** the so far developed version of the software. In the end, planning for the next phase is started.

Advantages of Spiral Model…

1. **Risk Handling:** High amount of **risk analysis** hence, avoidance of Risk is enhanced.
2. **Good for large** and mission-critical (complex) projects.
3. Strong approval and documentation control.
4. **Flexibility in Requirements:** Additional **Functionality can be added** at a later date.
5. **Customer satisfaction: Software is produced early** in the software life cycle.

Disadvantages of Spiral Model…

1. **Expensive:** Can be a **costly model** to use.
2. To perform Risk analysis, it requires **highly specific expertise**.
3. **Doesn't work** well for **smaller** projects.
4. **Too much dependable on Risk Analysis:** Project's **success** is highly **dependent on the risk analysis phase.**
5. **Difficulty in time management:** As the number of phases is unknown at the start of the project, so time estimation is very difficult

SOFTWARE ENGINEERING

Anchor & co: Sowmya BP, Archana A & Anusha GB

Department of Computer Applications

SOFTWARE ENGINEERING

5. Rational Unified Process Model

Archana A
Department of Computer Applications

Rational Unified Process Model

The Rational Unified Process (RUP) brings together elements of all of the general process models discussed earlier and **supports prototyping** and **incremental delivery** of software.

The Rational Unified Process (RUP) is an example of a modern process model that has been **derived from work on the UML** and the associated Unified Software Development Process

The RUP is normally described from **three perspectives**:

- a **dynamic perspective** that shows the **phases of the model in time**,
- a **static perspective** that shows **process activities**, and
- a **practice perspective** that suggests **good practices to be used in the process**

Rational Unified Process Model

The Rational Unified Process (RUP), which was developed by **Rational, a U.S. software engineering company.**

- The RUP is a flexible model. Its goal is to ensure the **production of high-quality software** that meets the needs of its end-users, within a predictable **schedule and budget.**

- The RUP has been adopted by some large software companies (**notably IBM**), but it has not gained widespread acceptance

The phases of Unified Process are:

1. The ***Inception*** *phase*
2. The ***Elaboration*** *phase*
3. The ***Construction*** *phase*
4. The ***Transition*** *phase*

Rational Unified Process Model

A generic software process model that presents software development as a four phase iterative activity,

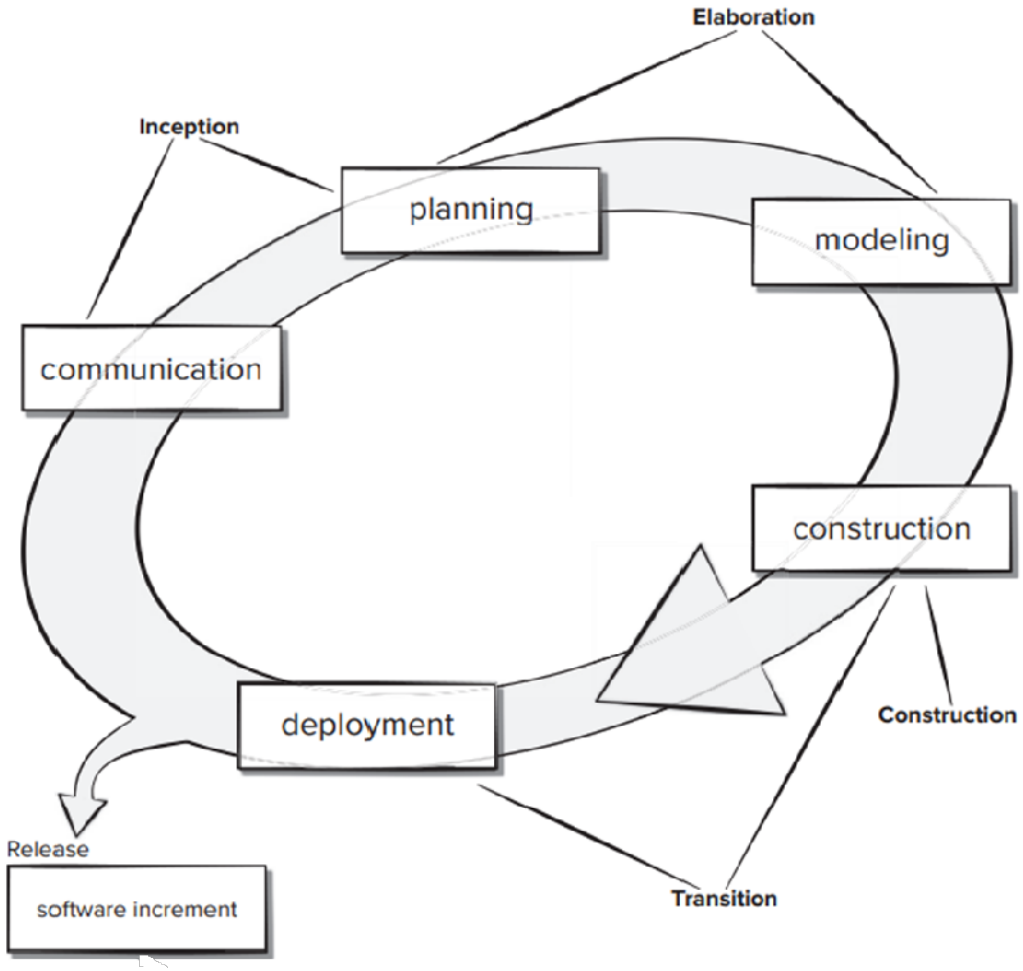
- The phases are :

1. The ***Inception*** *phase*
2. The ***Elaboration*** *phase*
3. The ***Construction*** *phase*
4. The ***Transition*** *phase*

1. Inception establishes a business case for the system,
2. elaboration defines the architecture,
3. construction implements the system, and
4. transition deploys the system in the customer's environment

SOFTWARE ENGINEERING

The Phases of Unified Process Model…



Phases of RUP Model

1. **Inception** The goal of the inception phase is to **establish a business case** for the system.

- First **identify all external entities** (people and systems) that will interact with the system and **define these interactions**.

- Then use this information to **assess the contribution that the system makes to the business**. If this contribution is minor, then the project may be cancelled after this phase.

Phases of RUP Model

2. Elaboration The goals of the elaboration phase are :

- **to develop an understanding of the problem domain,**
- **establish an architectural framework for the system,**
- **develop the project plan, and**
- **identify key project risks**.

- On completion of this phase we should **have a requirements model** for the system, which may be a **set of UML use-cases**, an **architectural description**, and a **development plan** for the software

PES UNIVERSITY

3. Construction

- The construction phase involves **system design, programming, and testing**.
- Parts of the system are developed in parallel and integrated during this phase.
- On completion of this phase, we should have a **working software system** and **associated documentation** that is ready for delivery to users.

4. Transition

- The final phase of the RUP is concerned with **moving the system from the development community to the user community** and **making it work in a real environment**.

- This is something that is ignored in most software process models but is, in fact, an **expensive and sometimes problematic activity**.

- On completion of this phase, we should have a **documented software system** that is working correctly in its operational environment.

The Unified Process Model…

Advantages:

- Well-documented and complete methodology.
- Open and Public.
- Training readily available.
- Changing Requirements.
- Reduced Integration Time and Effort.
- Higher Level of Re-use

Disadvantages:

- Process is too complex.
- Does not capture the sociological aspects of software development.

PES UNIVERSITY

Prepare a comparative study on the generic process models (Waterfall, Incremental, Prototyping, Spiral and RUP) in a tabular form.

SOFTWARE ENGINEERING

Ms. Sowmya BP, Archana A & Akshatha PR
Department of Computer Applications

SOFTWARE ENGINEERING

Agile Software Development

Ms. Sowmya BP, Archana A & Akshatha PR
Department of Computer Applications

What is Agile ?

Agile **combines** a **philosophy** and a set of **development guidelines**.

The **philosophy** encourages

- **customer satisfaction and early incremental delivery of software,**
- **Small but highly motivated project teams,**
- **informal methods,**
- **minimal software engineering work products,**
- **and overall development simplicity.**

- The **development guidelines** more focus on
  - **Delivery over analysis and design ,**
  - **Active and continuous communication between developers and customers.**

Agile Values

- The philosophy behind agile methods is reflected in the agile manifesto that was agreed on by many of the leading developers of these methods.
- This manifesto states: We are uncovering better ways of developing software by doing it and helping others do it.

- Through this work we have come to value:
- **Individuals and interactions over processes and tools**
- **Working software over comprehensive documentation**
- **Customer collaboration over contract negotiation**
- **Responding to change over following a plan**

That is, while there is value in the items on the right, we value the items on the left more.

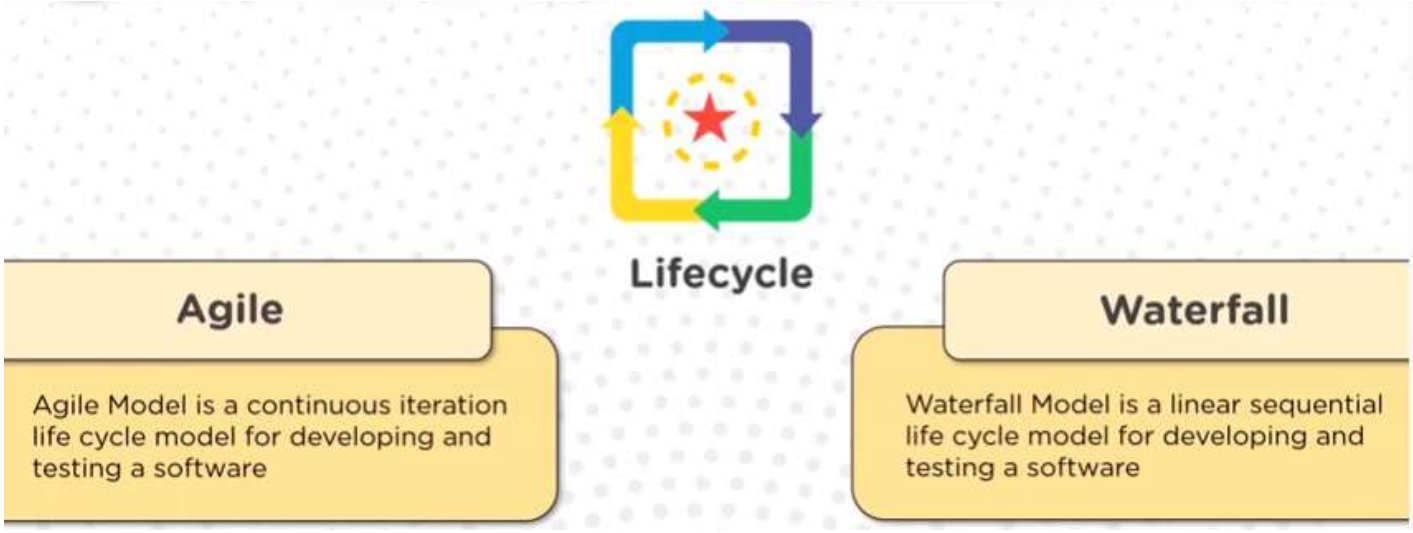
Why we move towards Agile?

- **Disadvantages of Waterfall model**

Why Agile?

Difference between Traditional waterfall model to Agile -

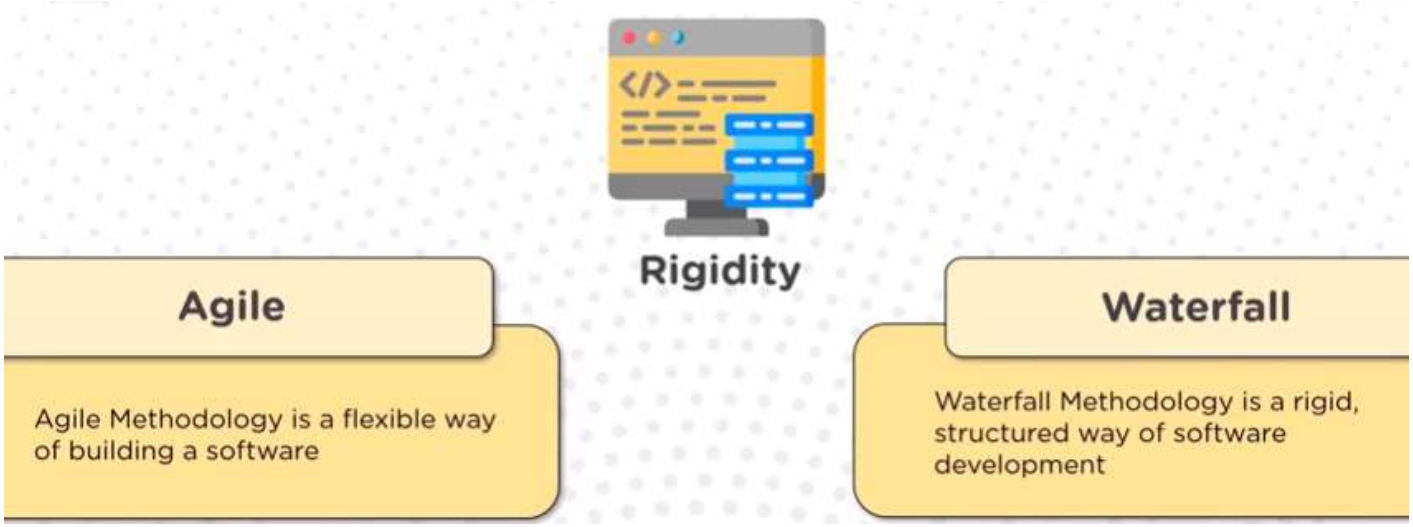
1. Lifecycle



Why Agile?

Difference between Traditional waterfall model to Agile -

2. Rigidity



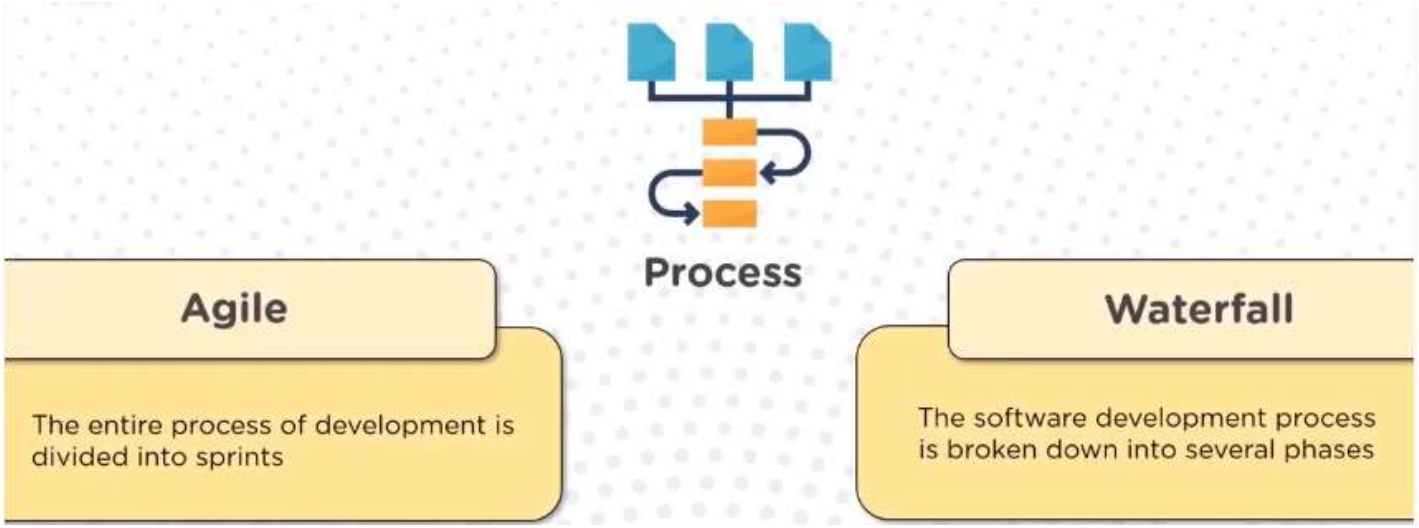
PES UNIVERSITY

Difference between Traditional waterfall model to Agile-

3. Collaboration

Difference between Traditional waterfall model to Agile-

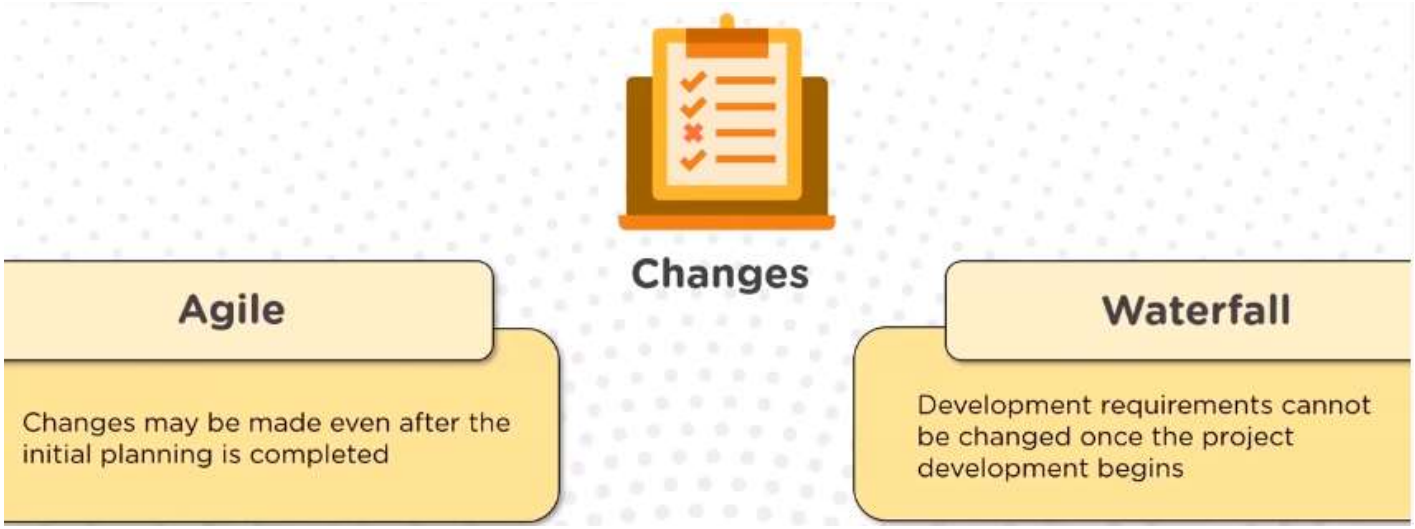
4.Process



Why Agile?

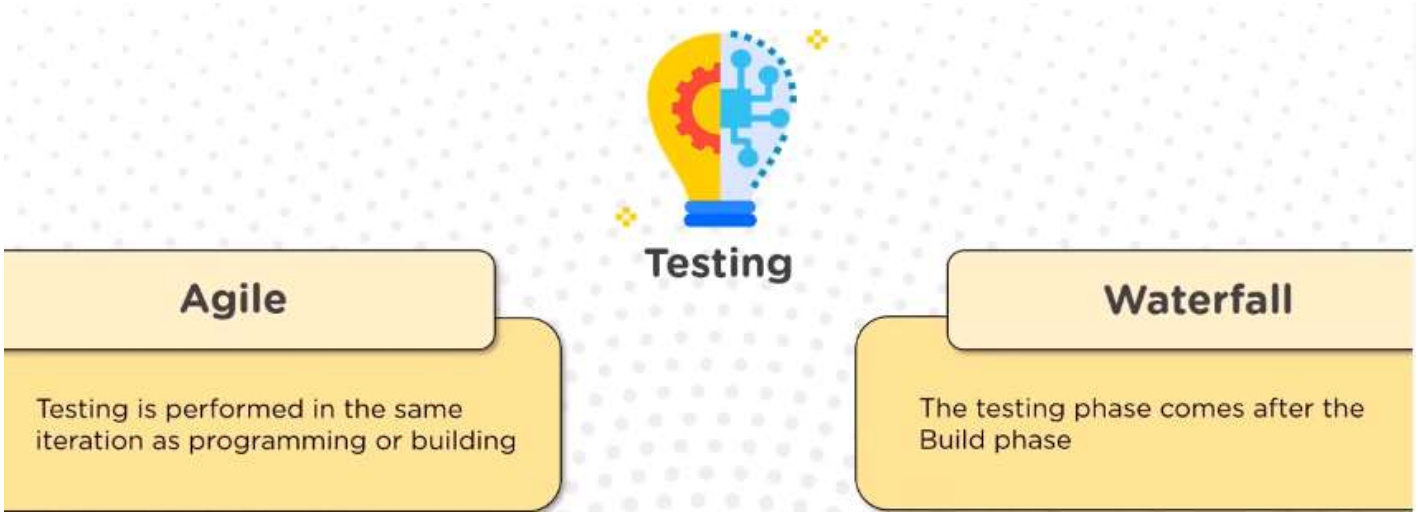
Difference between Traditional waterfall model to Agile –

5. Changes



Difference between Traditional waterfall model to Agile –

6. Testing



SOFTWARE ENGINEERING

Principles of Agile

Ms. Sowmya BP, Archana A & Akshatha PR
Department of Computer Applications

1. Customer Satisfaction – Highest priority is given to satisfy the requirements of customers through **early and continuous delivery of valuable software.**

2. Welcome Change – Changes are inevitable during software development. Ever-changing requirements should be welcome, even late in the development phase. Agile processes should work to increase customers' competitive advantage.

3. Deliver a Working Software – Deliver a working software frequently, ranging from a few weeks to a few months, considering shorter time-scale.

4. Collaboration – Business people and developers must work together during the entire life of a project.

5. Motivation – Projects should be built around motivated individuals. Provide an environment to support individual team members and trust them so as to make them feel responsible to get the job done.

6. Face-to-face Conversation – Face-to-face conversation is the most efficient and effective method of conveying information to and within a development team.

7. Measure the Progress as per the Working Software – Working software is the key and it should be the primary measure of progress

8. Maintain Constant Pace – Agile processes aim towards sustainable development. The business, the developers, and the users should be able to maintain a constant pace with the project.

9. Monitoring – Pay regular attention to technical excellence and good design to enhance agility.

10. Simplicity – Keep things simple and use simple terms to measure the work that is not completed.

11. Self-organized Teams – An agile team should be self-organized and should not depend heavily on other teams because the best architectures, requirements, and designs emerge from self-organized teams.

12. Review the Work Regularly – Review the work done at regular intervals so that the team can reflect on how to become more effective and adjust its behavior accordingly.

SOFTWARE ENGINEERING

Ms. Sowmya BP, Archana A & Akshatha PR
Department of Computer Applications

Agile Methods

- All agile methods suggest that software should be developed and **delivered incrementally**.
- These methods are based on different agile processes but they share a set of principles, based on the agile manifesto, and so they have much in common.

- Agile methods have been particularly successful for **two kinds of system development**.

1. **Product development** where a software company is developing a small or medium-sized product for sale. Virtually all software products and apps are now developed using an agile approach.

2. Custom system development within an organization, where there is a clear commitment from the **customer to become involved in the development process** and where there are few external stakeholders and regulations that affect the software

Agile methods work well in these situations because it is possible to have **continuous communications between the product manager or system customer** and **the development team.**

SOFTWARE ENGINEERING

Agile Software Development

Ms. Sowmya BP, Archana A & Akshatha PR
Department of Computer Applications

Plan-driven Development

Plan-driven development is a software engineering approach characterized by a strong emphasis on **detailed upfront planning and a structured**, **sequential execution of development activities.** It **contrasts with agile methodologies**, which prioritize **flexibility and iterative** development.

Key Characteristics:

1. Detailed Planning:

- A comprehensive **project plan is created** at the outset, outlining the entire development process, including **timelines, milestones, resource allocation, and risk assessment.**
- This plan serves as a roadmap for the entire project and is rigorously adhered to.

Plan driven Development - Characteristics

2. Sequential Phases: Development typically follows a linear or waterfall-like progression through distinct phases:

- **Requirements Gathering:** Detailed elicitation, analysis, and documentation of user needs and system requirements.
- **Design:** Creation of architectural and detailed designs, specifying system components, interfaces, and algorithms.
- **Implementation:** Coding and development of the software based on the approved design.
- **Testing:** Rigorous testing of the software at various levels (unit, integration, system) to ensure quality and conformance to requirements.
- **Deployment:** Release of the software to the end-users.
- **Maintenance:** Ongoing support, bug fixes, and enhancements after deployment.

Plan-driven Development - Characteristics

3. Emphasis on Documentation: Extensive documentation is produced throughout the development process, including **requirements specifications**, **design documents**, **test plans, and user manuals**.

- This documentation serves as a crucial artifact **for communication, coordination, and future reference.**

4. Change Control: Changes to the plan or requirements are carefully managed and controlled to minimize disruptions and maintain project stability.

- A formal change request process is often implemented to evaluate and approve any proposed modifications

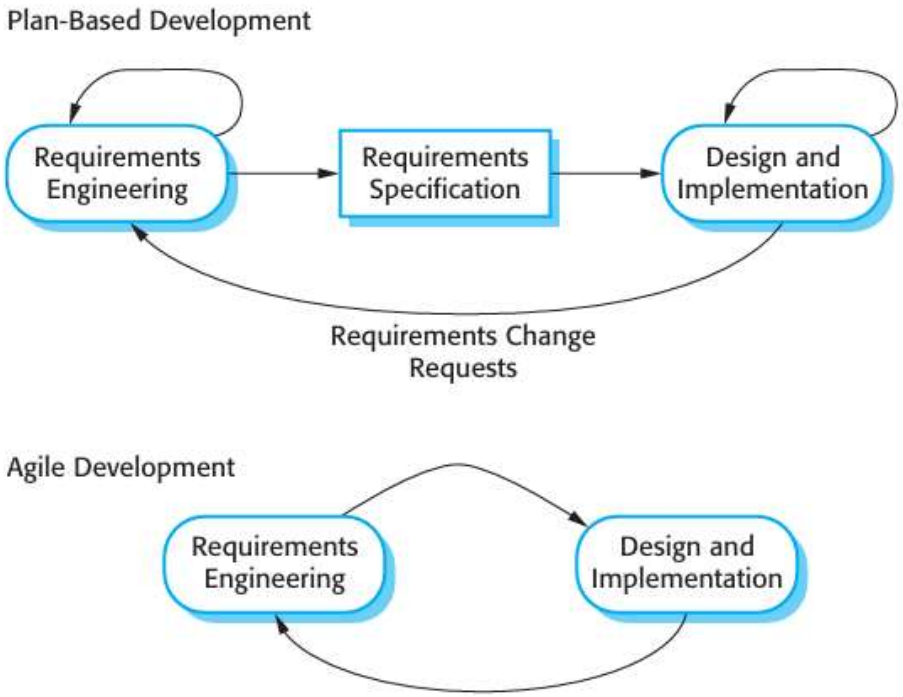
5. Risk Management: Potential risks are identified and mitigated proactively through risk assessment and mitigation plans.

Plan-driven Development - Characteristics

In a plan-driven approach, **iteration occurs within activities** **with formal documents** used to communicate between stages of the process.

- For example, the requirements will evolve and, ultimately, a requirements specification will be produced.
- This is then an input to the design and implementation process.

In an agile approach, **iteration occurs across activities**. Therefore, the requirements and the design are developed together, rather than separately.



The ideas underlying agile methods were developed around the same time by a number of different people in the 1990s.

The most significant approach to changing software development culture was the development of **Extreme Programming (XP).**

The name was coined by **Kent Beck** (Beck 1998) because the approach was developed by pushing recognized good practice, such as **iterative development**, to **"extreme" levels**.

- For example, in XP, several new versions of a system may be developed by different programmers, integrated, and tested in a day.

SOFTWARE ENGINEERING

Ms. Sowmya BP, Archana A & Akshatha PR
Department of Computer Applications

SOFTWARE ENGINEERING

Extreme Programming (XP)

Ms. Sowmya BP, Archana A & Akshatha PR
Department of Computer Applications

Extreme Programming (XP)

- **XP** is the **most prominent** Agile Software development method.
- **Prescribes** a set of **daily stakeholder practices**.
- "**Extreme**" **levels** of practicing leads to more **responsive software**.
- Changes are more **realistic**, **natural**, **inescapable**.

XP Values

- A set of **five values** that establish a foundation for all work performed as part of XP—
  1. **Communication,**
  2. **Simplicity,**
  3. **Feedback,**
  4. **Courage, and**
  5. **Respect**

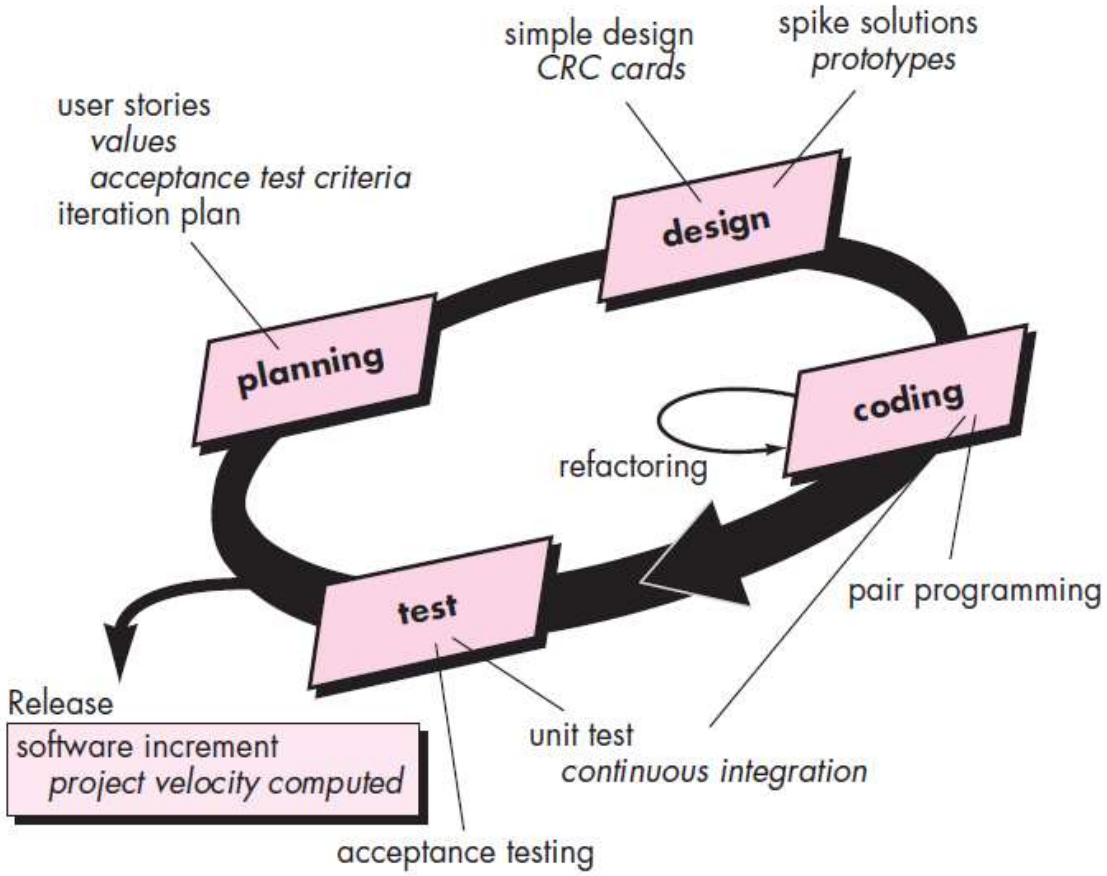
- Each of these values is used as **a driver** for specific XP activities, actions, and tasks.

PES UNIVERSITY

XP Process

- The most widely used agile process, originally proposed by **Kent Beck**.
- Extreme Programming uses an **object-oriented approach.**
- Encompasses **a set of rules and practices** that occur within the context of **four framework activities**: **planning, design, coding, and testing.**

Extreme Programming (XP) - XP Process



1. XP Planning:

- The planning activity *begins with* ***listening*** , Listening leads to the creation of a set of **"stories"** *that describe* ***required output,*** **features, and functionality** for software to be built.

  - Begins with the creation of "**user stories**".
  - Agile team assesses each story and assigns a **cost.**
  - Stories are grouped for a **deliverable increment.**
  - A **commitment** is made on delivery date.
  - After the first increment "**project velocity**" is used to help define subsequent delivery dates for other increments.

2. XP Design:

- Encourage the use of **CRC cards** as an **effective mechanism** for thinking about the software in an **object-oriented context**.

- CRC (class-responsibility-collaborator) cards **identify and organize the object-oriented classes** that are relevant to the current software increment.

2. XP Design…

Class-Responsibility Collaborator(CRC).

Class: *Librarian*	
Responsibilities	**Collaborators**
check in book	*Book*
check out book	*Book, Borrower*
search for book	*Book*
knows all books	
search for borrower	*Borrower*
knows all borrowers	

Class CardReader	
Responsibilities	Collaborators
Tell ATM when card is inserted	ATM
Read information from card	Card
Eject card	
Retain card	

2. XP Design…

- For difficult design problems, XP suggests the creation of "**spike solutions**"—a design prototype.
- A spike solution is a very simple program to explore potential solution.
- **Difficult design** should be modeled using **prototype.**

3. XP Coding:

- Recommends the **construction of a unit test** for a story ***before*** **coding commences**. This helps developer to better focus on what must be implemented to pass the test.

- Encourages "**pair programming**". This provides a mechanism for real time problem solving, real time quality assurance as it is created.

- As pair programmers complete their work, the code they develop is integrated with the work of other.

- All **unit tests** are executed daily

3. XP Coding…

- XP encourages "**refactoring**"—an iterative refinement of the internal program design.
    - Refactoring is a process of changing a software system in such a way that **it does not alter the external behavior** of the code, **yet improves the internal structure**.

4. XP Testing:

- All **unit tests are executed daily.**
- "**Acceptance tests**" are defined by the customer and executed to assess customer visible functionality.
    - XP *acceptance tests, also called* ***customer tests****, are specified by the customer and* focus on **overall system features and functionality** that are visible and reviewable by the customer.

THANK YOU

Ms. Sowmya BP, Archana A & Akshatha PR
Department of Computer Applications

archana@pes.edu

+91 80 6666 3333 Extn 392

Assignment Questions for Unit-I Unit Test

Note: Find the solutions for the following questions. 4 questions were asked in the unit test. Each question carries 5 marks. Totally 20 marks in 1 hour.

1) What is software? Explain different types of Software
2) List and explain characteristics of software
3) What is software Engineering? Explain different type of application used by software engineering
4) Discuss the phases of SDLC with a neat diagram
5) Define the following:
   i) Activities
   ii) Actions
   iii) Task
6) What are four fundamental software engineering activities
7) Explain Waterfall model with a neat diagram
8) Explain Incremental model with a neat diagram
9) Explain Prototyping with a neat diagram
10) Explain Spiral model with a neat diagram
11) Explain Rational Unified Process model with a neat diagram
12) List advantage and disadvantages of water fall model and incremental model
13) List advantage and disadvantages of Prototyping model and spiral model
14) List advantage and disadvantages of RUP model
15) What is agile methodology
16) Compare agile with waterfall model
17) Explain agile principles
18) Discuss agile two methods
19) What is Plan-driven Development and explain key characteristics
20) Compare plan-driven approach and agile approach with a neat diagram
21) What is Extreme Programming? Explain with XP values
22) Explain working of Extreme Programing with a neat diagram