SE - Notes


V2V EDTECH LLP
Online Coaching at an Affordable Price.
OUR SERVICES:
• Diploma in All Branches, All Subjects
• Degree in All Branches, All Subjects
• BSCIT / CS
• Professional Courses
+91 93260 50669
V2V EdTech LLP
v2vedtech.com
v2vedtech

Unit - I Basics of Software Engineering

📌 Software

Definition: Software is a collection of programs, data, and documentation that perform specific tasks on a computer system. It enables users to interact with hardware and perform desired functions.

Computer software is the product that software professionals build and then support over the long term.

It also includes a set of documents, such as the software manual, meant for users to understand the software system. Today's software comprises the source code, Executable, Design Documents, Operations and System Manuals and installation and Implementation Manuals.

Software is described by its capabilities. The capabilities relate to the functions it executes, the features it provides and the facilities it offers.

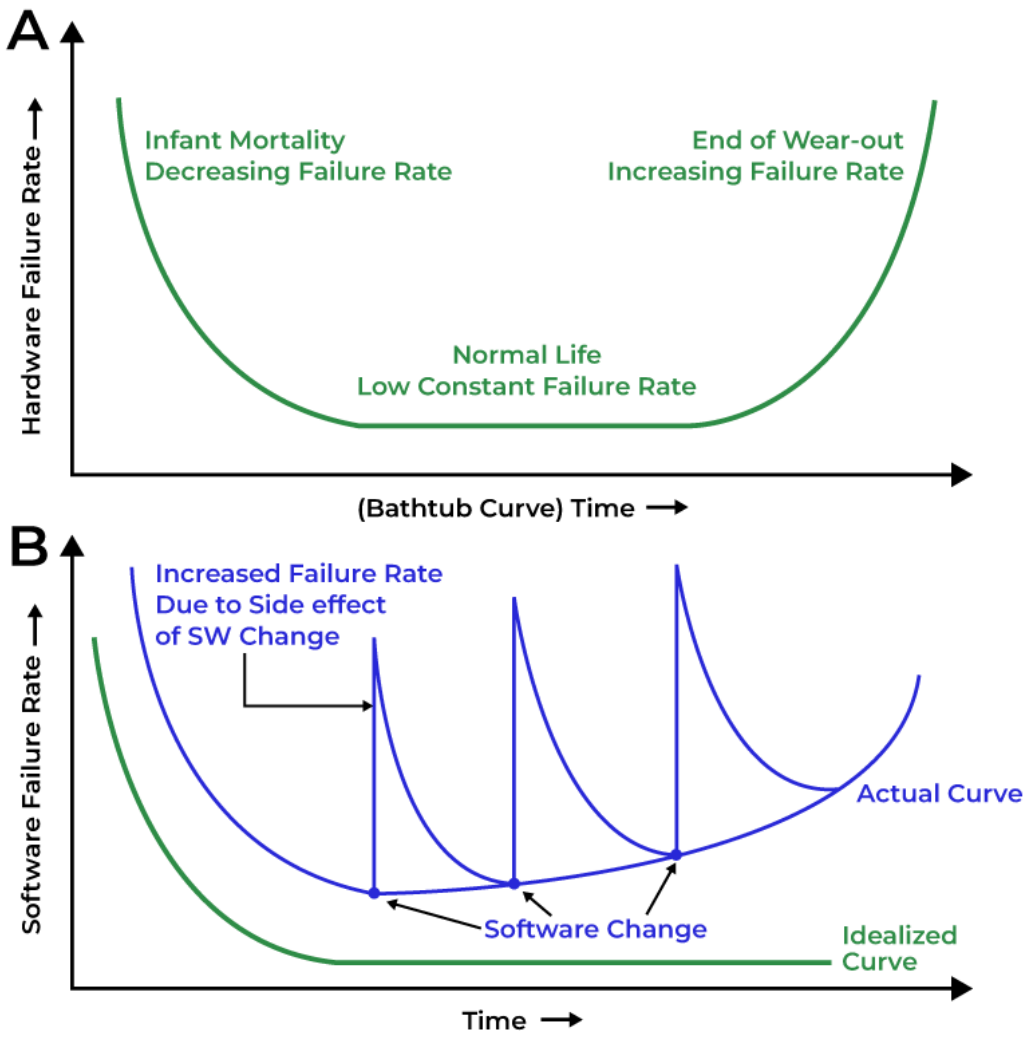
Characteristics of Software:

1. **Software is Developed or Engineered, Software is not manufactured:**
   It is developed through design and coding.The two activities (software development and hardware manufacturing) are fundamentally different. In both activities, high quality is achieved through good design, but the manufacturing phase for hardware can introduce quality problems.

2. **Software doesn't "wear out" like hardware and it is not degradable over a period.**

   The following figure depicts failure rate as a function of time for hardware. The relationship often called the "bathtub curve", indicates that hardware exhibits relatively high failure rates early in its life.The failure rate curve for software should take the form of an "idealized curve" as shown in the above figure. Undiscovered defects will cause high failure rates early in the life of a program. However, these are corrected and the curve flattens as shown. Hence the software doesn't wear out, but it does deteriorate.

3. **Most of the Software are Custom-built or generic rather than assembled from existing components:** Software can be tailored or



commercially available.Although the industry is moving toward component – based construction, most software continues to be custom built. A software component should be designed and implemented so that it can be reused in many different programs.

Other Characteristics are

4. **Complex:** Often contains millions of lines of code.
5. **Easy to modify:** Can be updated with new features or bug fixes.
6. **Intangible**

Advantages of Software :

- Automates manual processes
- Increases accuracy and efficiency
- Enhances decision-making
- Reduces operational costs

Disadvantages of Software:

- May contain bugs or vulnerabilities
- Needs regular updates and maintenance
- Can be expensive to develop
- Complex systems may be difficult to manage

Applications:

- Education (e-learning platforms)
- Business (ERP, accounting software)
- Healthcare (EMR systems)
- Entertainment (games, streaming)
- Communication (chat apps, emails)

📌 Software Engineering as a Layered Approach

1 Quality Focus (Base Layer)

This is the **foundation** of all software engineering activities.

- It ensures the **final product meets customer requirements** and performs reliably.
- Quality is maintained through activities like **quality assurance (QA), reviews, and testing**.
- Without this layer, all other efforts may lead to unreliable or unsatisfactory software.
- Total quality management, six sigma and similar philosophies foster a continuous process improvement culture.
- The bedrock that supports software engineering is a quality focus.

---

2 Process Layer

This layer defines the **framework** for managing and controlling the software development process.

- It provides a **structured approach** using Software Development Life Cycle (SDLC) models like **Waterfall, Agile, Spiral, etc.**
- Ensures **planning, monitoring, and evaluation** of all phases in software development.
- The software process forms the basis for management of software projects.

---

3 Methods Layer

This layer includes the **technical methods** used to build software.

- Covers **requirement analysis, system design, coding, testing, and maintenance**.
- Methods guide **how** software is developed technically to ensure correctness and efficiency.

---

4 Tools Layer

This is the **topmost layer**, consisting of **automated tools** that support the methods and processes.

- Tools improve **productivity, accuracy, and efficiency**.
- Examples include **IDEs (like VS Code), testing tools (like Selenium), version control systems (like Git)**.

📌 3. Attribute of Good Software are

1. **Functionality**: The ability to perform its intended task
2. **Reliability:** The probability of failure-free software operation
3. **Usability:** Ease with which users can operate the software
4. **Efficiency:** Optimal use of system resources5.
5. **Maintainability:** Ease of modification and updates
6. **Portability:** Ability to work across different environments
7. **Scalability:** Can handle growth in users or data

📌 4. Types of Software

1. System Software:

It is a collection of programs written to service other programs. Some system software (e.g., compilers, editors, and file management utilities) processes complex, but determinate, information structures. Other systems applications (e.g., operating system components, drivers, networking software, telecommunications processors) process largely indeterminate data.

2. Application Software:

Stand-alone programs that solve a specific business need. Applications in this area process

business or technical data in a way that facilitates business operations or management/technical decision making

3. Programming Software:

Used by developers to write code.
Examples: Compilers, editors, debuggers.

4. Embedded Software:

Embedded software can perform limited functions (e.g., key pad control for a microwave oven) or provide significant function and control capability (e.g., digital functions in an automobile such as fuel control, dashboard displays, and braking systems).
Used in embedded systems (non-PC devices).
Examples: Software in washing machines, traffic lights.

5. Web-based Software:

In their simplest form, WebApps can be little more than a set of linked hypertext files that present information using text and limited graphics.
Runs on web browsers and accessed via the internet.
Examples: Gmail, Google Docs.

6. AI Software:

Designed to simulate intelligence.
Examples: Chatbots, recommendation systems.

7. Engineering/scientific software:

Applications range from astronomy to volcanology, from automotive stress analysis to space shuttle orbital dynamics, and from molecular biology to automated manufacturing

Example: CAD software

7. Product-line software:

Designed to provide a specific capability for use by many different customers.

Product-line software can focus on a limited marketplace (e.g., inventory control products) or address mass consumer markets (e.g., word processing, spreadsheets,database management).

📘 Software Development Framework

A **software development framework** defines a **standard structure and set of activities** to guide the software development life cycle (SDLC). It ensures that software is built in an **organized, systematic, and efficient** manner.

✅ 1. Generic Process Framework Activities

These are the **core activities** that are part of almost every software development process, regardless of the development model (Waterfall, Agile, Spiral, etc.).

🔹 Communication

- Involves interacting with stakeholders to understand the **requirements**.
- Includes activities like **requirement gathering, meetings, and interviews**.

🔹 Planning

- Defines a **roadmap** for the project.
- Includes **estimating time, cost, resources**, and setting milestones.

🔹 Modeling

- Focuses on **designing the architecture and system structure**.
- Includes **data modeling, process modeling, interface design**, etc.

🔹 Construction

- Involves **actual coding and testing** of the software.
- Focuses on **writing source code, unit testing, integration testing**, etc.

🔹 Deployment

- Involves **delivering the final product** to the end-user.
- May include **installation, user training, and feedback collection**.
- Updates and patches are handled during **post-deployment maintenance**.

☂️ Umbrella Activities in Software Engineering

Umbrella activities are **supportive tasks** that occur **throughout the software development life cycle (SDLC)**. They ensure that the core development activities are well-managed and high in quality.

✅ 1. Software Project Tracking and Control

- Monitors project progress to ensure it's on time and within budget.
- Helps take corrective actions if deviations occur.

✅ 2. Risk Management

- Identifies, analyzes, and prepares for **potential risks** that could impact the project.
- Plans strategies to **reduce or avoid risks**.

✅ 3. Software Quality Assurance (SQA)

- Ensures the software meets the **defined quality standards**.
- Includes activities like audits, reviews, and testing.

✅ 4. Formal Technical Reviews (FTRs)

- Structured reviews of software artifacts (design, code, documents).
- Helps in early **error detection and correction**.

✅ 5. Measurement and Metrics

- Collects **quantitative data** to assess process and product quality.
- Used for **improvement, estimation, and evaluation**.

✅ 6. Software Configuration Management (SCM)

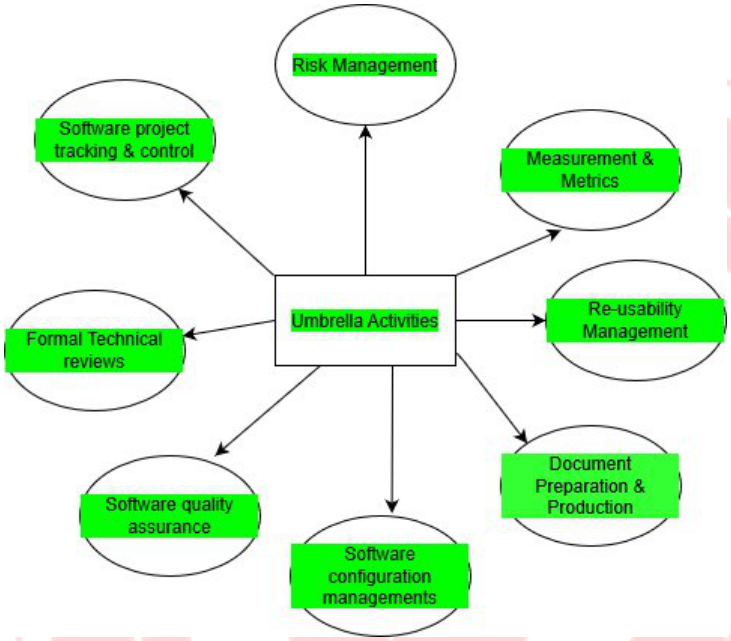
- Manages changes in software versions and components.
- Maintains integrity and traceability of project artifacts.

✅ 7. Reusability Management

- Identifies and promotes **reuse of existing software components**.
- Saves time and improves reliability.

✅ 8. Documentation

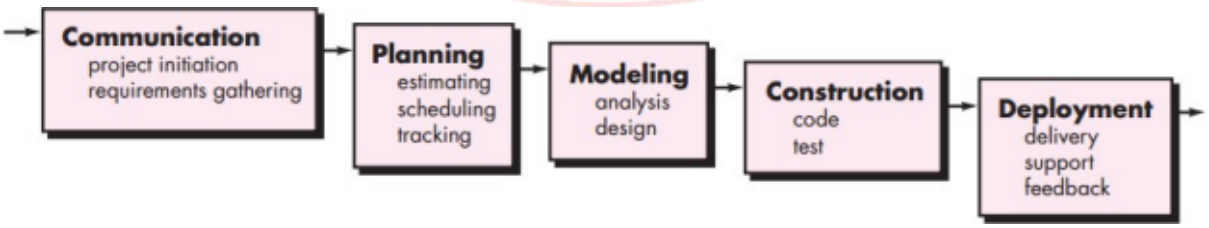
- Ensures proper recording of all development activities, plans, and decisions.
- Helps in **maintenance, training, and future development**.



Prescriptive process models:

Prescriptive models are **well-defined, structured software development models** that guide the step-by-step process of software development. They are useful when project requirements are clear and stability is required.

✅ 1. Waterfall Model



Definition: The Waterfall Model is a **linear and sequential approach** where each phase (requirements, design, implementation, testing, deployment) is completed before the next begins.Easy to manage due to its rigid structure and documentation.Works well for projects with fixed and well-understood requirements.Testing starts late, only after the build phase.

Advantages:

- Simple and easy to use
- Well-documented and structured
- Ideal for small or well-defined projects
- No feedback until final stage
- Difficult to handle changes once a phase is complete

Disadvantages:

- No flexibility once the project is in the testing phase
- No overlapping of stages; each must be completed before the next begins.
- Poor model for long or complex projects
- Difficult to handle changes in requirements

Applications:

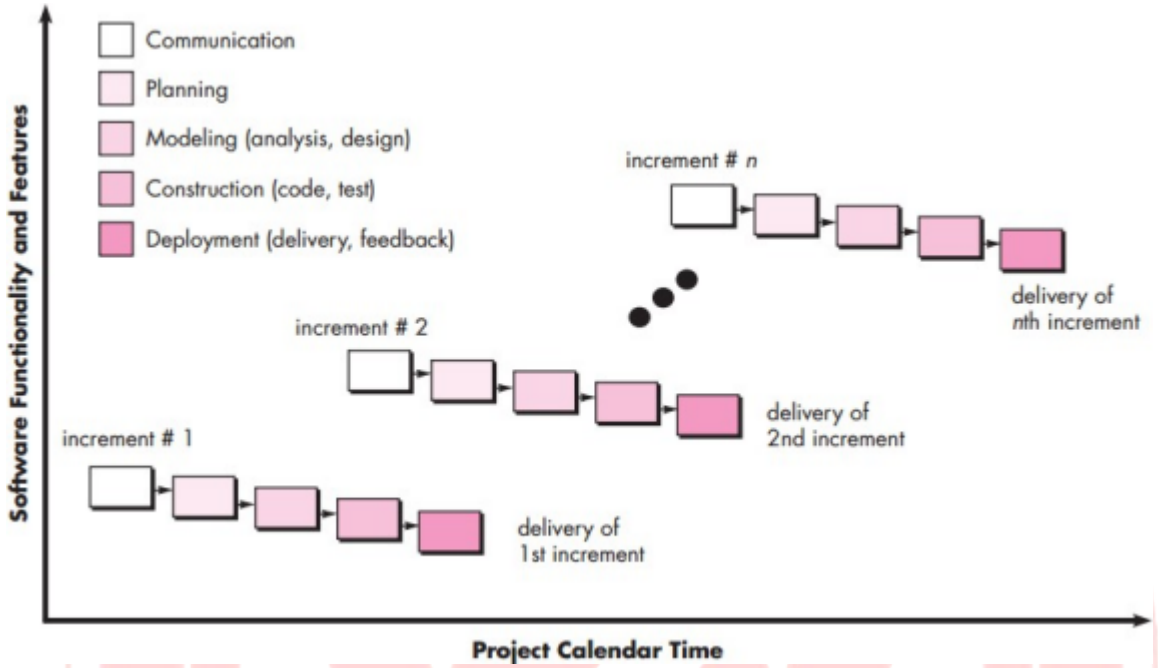
- Government and defense projects
- Projects with clear, fixed requirements
- Simple applications like calculators, tools

✅ 2. Incremental Model

Definition: The Incremental Model develops software in **small, manageable parts (increments)**. Each increment adds functionality until the final product is complete.Easier to

test and debug.Early delivery of partial functionality.Each increment includes design, coding, and testing.Useful when requirements are well understood, but may evolve.

Helps reduce risk by identifying issues early.



Advantages:

- Early partial product delivery — some features are usable early in the process.
- Easier testing and debugging — smaller pieces of code are tested incrementally.
- Flexible to changes — easier to adapt based on feedback from earlier increments.
- Lower initial delivery cost — functionality is delivered step-by-step.

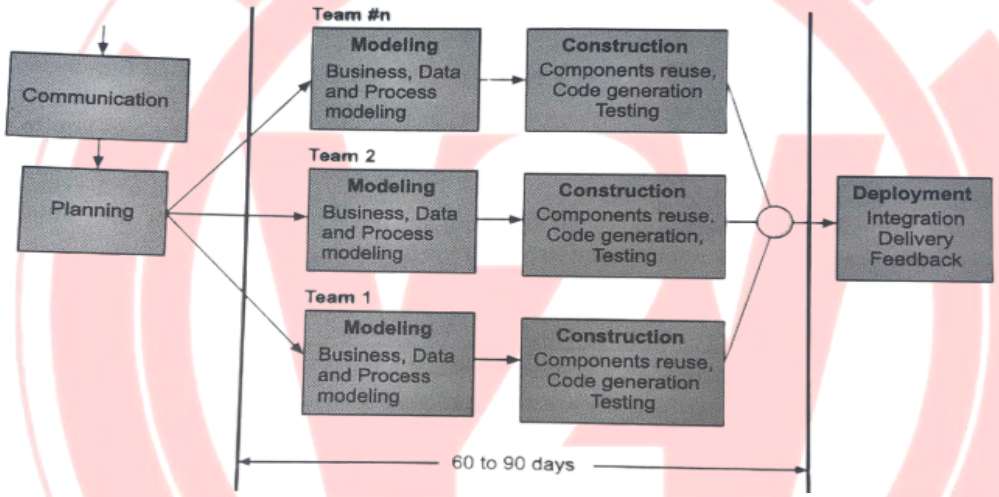
Disadvantages:

- Requires proper planning and design
- Integration can become complex
- May not be suitable for large systems with tight deadlines

Applications:

- Web applications
- E-learning platforms
- Projects with evolving requirements

✅ 3. RAD Model (Rapid Application Development)



Definition: RAD focuses on **quick development and delivery** by using reusable components, prototyping, and minimal planning. It emphasizes **rapid iterations with user feedback**.Very fast development using reusable components.Requires active user involvement.Best for small to medium-sized projects

Advantages:

- Faster development due to component reuse and parallel tasks.
- Customer involvement is high with constant feedback and review.
- Prototyping helps refine requirements early in the project.
- Highly flexible to changing requirements.

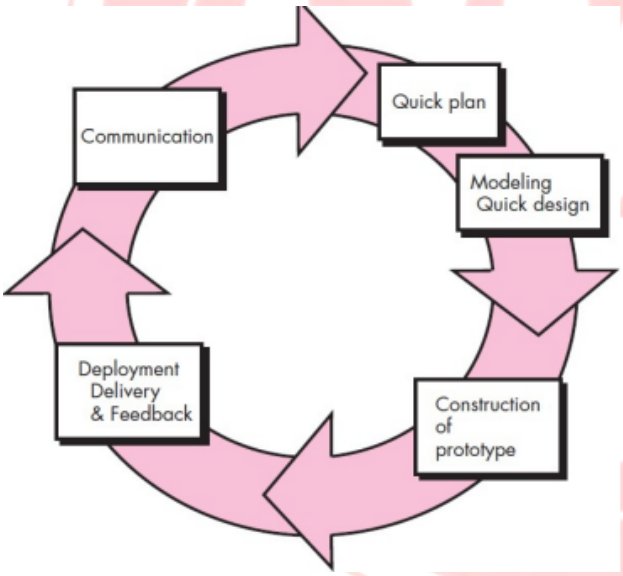
Disadvantages:

- Not suitable for large-scale or complex systems
- Requires skilled developers and designers
- Poor documentation due to rapid pace

Applications:

- Short-term projects
- Mobile apps, internal tools
- UI/UX-heavy applications needing quick changes

✅ 4. Prototyping Model



Definition: The Prototyping Model builds a **working prototype** early in the process to understand user requirements. Based on user feedback, the prototype is refined until the final product is developed.

Encourages user involvement.Reduces risk of misunderstanding requirements.Useful when requirements are unclear.Prototype may be discarded or evolved into final system

Advantages:

- Improves requirement clarity by visualizing the system early.
- Reduces misunderstandings between developers and users.
- User involvement is high, leading to a better-aligned final product.
- Faster identification of missing or unclear features.

Disadvantages:

- Prototype may be mistaken for final product
- Frequent changes increase cost
- Not suitable for large-scale or structured projects

Applications:

- Complex systems with unclear requirements
- Software involving new technology
- User-interface intensive applications

✅ 5. Spiral Model

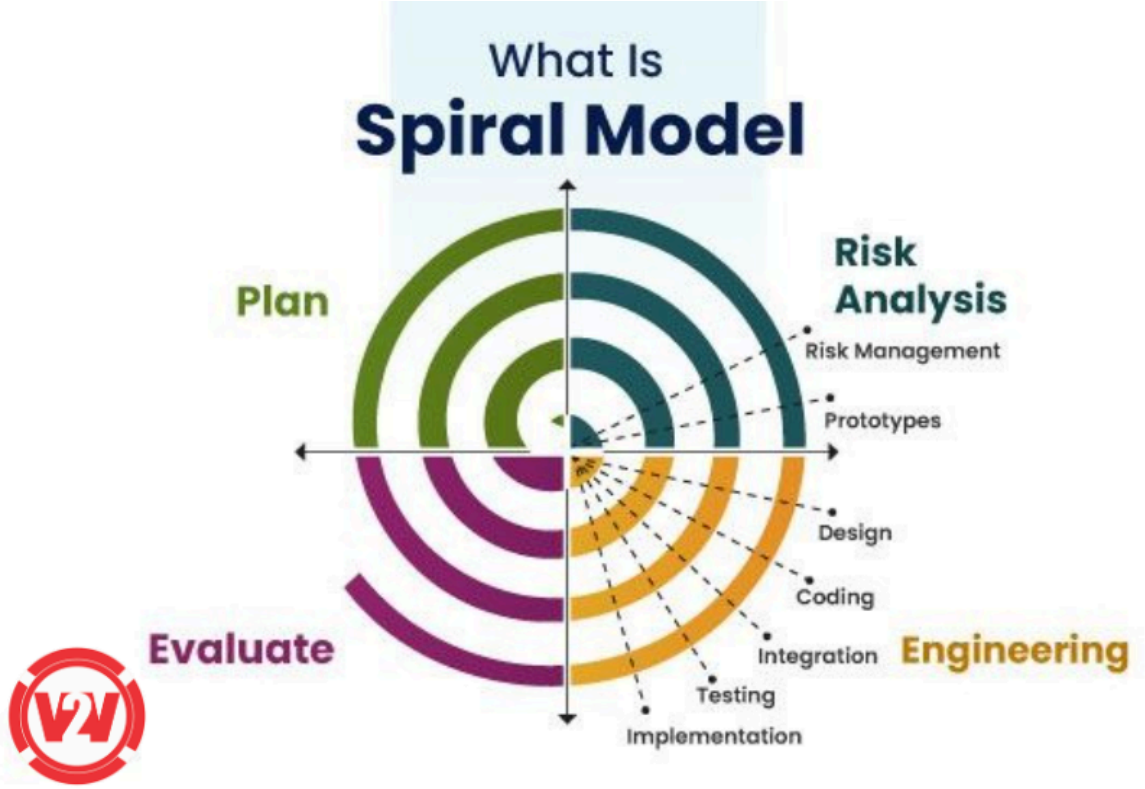
- **Definition:** The Spiral Model combines features of **Waterfall and Prototyping**, focusing on **risk analysis**. It involves repeated cycles (spirals), each with planning, risk analysis, development, and evaluation.
  Excellent for large, complex, high-risk projects.Allows changes at any stage.More costly and complex to manage.

Advantages:

- Focuses on risk analysis and mitigation at every phase.
- Supports flexible and iterative development.
- Well-suited for large, high-risk, and complex projects.
- Allows progressive refinement of requirements.

Disadvantages:

- Complex to manage and implement
- Expensive for small projects
- Requires expertise in risk assessment



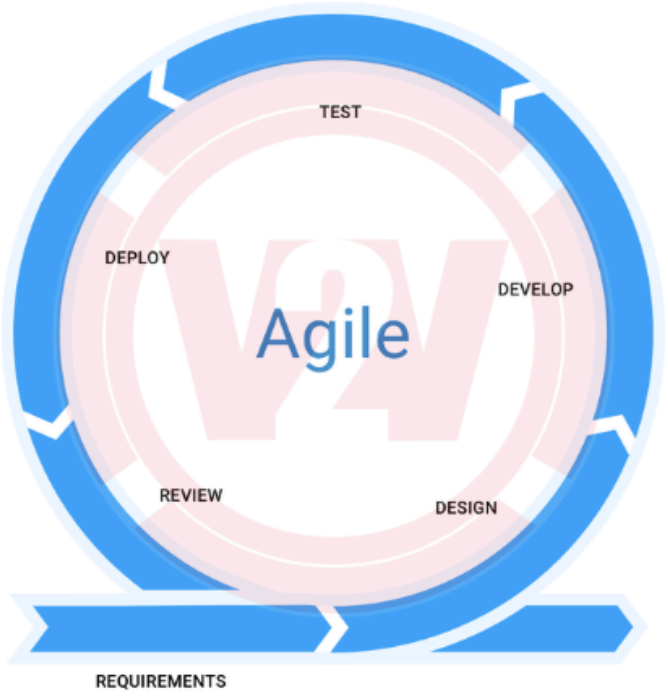
Applications:

- Aerospace, military, and safety-critical software
- Long-term or high-budget systems
- R&D projects

🌀 Agile Software Development

Agile is a flexible, iterative, and collaborative software development methodology.
Agile is a **lightweight and iterative software development methodology** focused on **flexibility**, **customer collaboration**, **frequent delivery**, and **responding to change**. Instead of a rigid plan, Agile encourages **continuous improvement**, **short development cycles (sprints)**, and **early delivery**.Divides work into small units called iterations or sprints (typically 1–4 weeks).Continuous planning, testing, and integration.

Agile model



✅ 2. Core Principles of Agile (Based on Agile Manifesto)

1. **Individuals and interactions** over processes and tools
2. **Working software** over comprehensive documentation
3. **Customer collaboration** over contract negotiation
4. **Responding to change** over following a plan

✅ 3. Importance of Agile

- **Fast Delivery:** Delivers usable software early and frequently.
- **Customer Satisfaction:** Involves customer feedback at every stage.
- **Flexibility:** Easily adapts to changing requirements.
- **Improved Quality:** Regular testing during each iteration.
- **Better Team Collaboration:** Promotes face-to-face communication and teamwork.

- **Risk Reduction:** Problems are identified early, reducing the risk of failure.

✅ 4. Agile Process – Key Activities

1. **Requirement gathering** in the form of **user stories**
2. **Planning** for short time-boxed iterations (called sprints)
3. **Design and Development** in cycles (each sprint produces working software)
4. **Testing and Review** after each sprint
5. **Retrospective** to reflect and improve in the next iteration

---

🔧 Extreme Programming (XP)

Definition: XP is an Agile method focused on improving software quality and responsiveness to customer requirements using frequent releases and technical practices.Prioritizes code simplicity, communication, and continuous improvement.

Practices include Test-Driven Development (TDD), Pair Programming, Continuous Integration, and Refactoring.

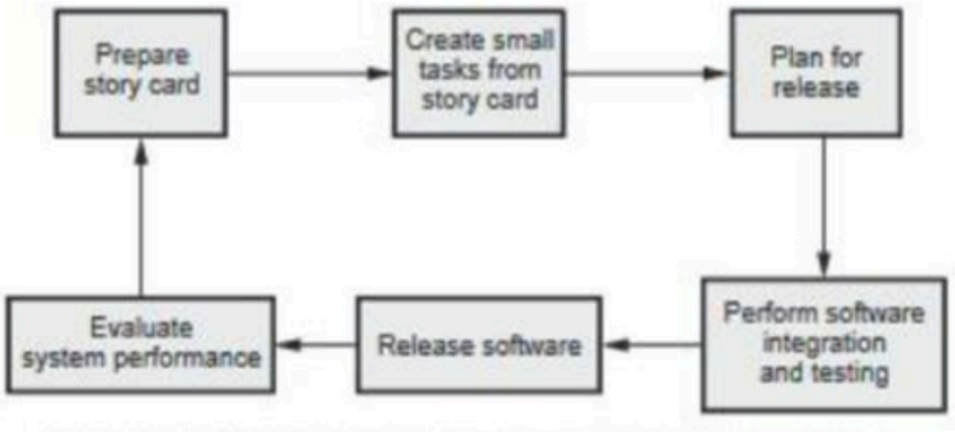


Fig. 1.8.1 Extreme programming release cycle

✅ Core Practices of XP:

- **Pair Programming** – Two developers code together on the same computer
- **Test-Driven Development (TDD)** – Write tests before coding
- **Continuous Integration** – Code is integrated and tested frequently
- **Refactoring** – Regular code improvement
- **Small Releases** – Deliver functional software often
- **Simple Design** – Avoid unnecessary complexity

✅ Advantages of XP:

- High-quality code due to frequent testing and continuous integration.
- Enables fast and flexible responses to changing requirements.
- Promotes strong team communication through pair programming and stand-ups.
- Frequent releases keep customers engaged and informed.

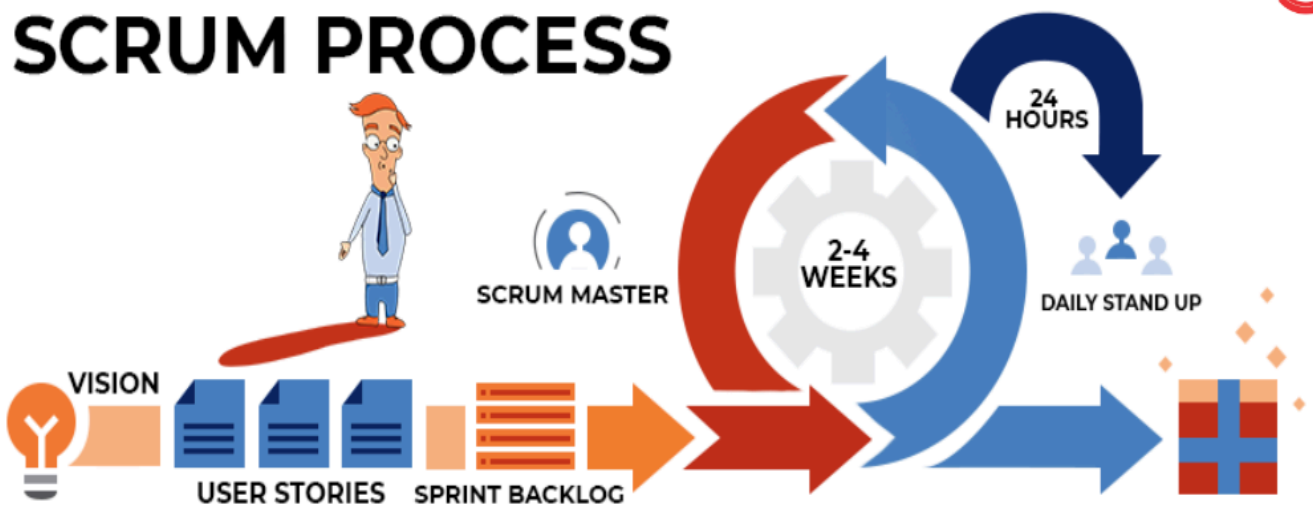
❌ Disadvantages of XP:

- Requires high discipline and skilled developers
- Continuous feedback may be overwhelming
- Not ideal for very large teams or long-term planning

🎯 Applications:

- Small to medium-sized software projects
- Real-time and evolving applications
- Startups needing quick MVP (Minimum Viable Product)

🧩 Scrum



Definition: Scrum is an Agile framework that **organizes development in short iterations** (called sprints), typically lasting 1–4 weeks. The team works collaboratively to complete selected features during each sprint.

✅ Key Roles in Scrum:

- **Product Owner:** Represents the customer, defines product backlog
- **Scrum Master:** Facilitates the process, removes blockers
- **Development Team:** Self-organizing team that delivers the product

✅ Key Elements

- **Product Backlog:** List of features or requirements
- **Sprint Backlog:** Tasks to be completed in a sprint

- **Sprint Planning:** Decide what to deliver in the sprint
- **Daily Stand-up (Daily Scrum) :** 15-minute daily progress meeting
- **Sprint Review & Retrospective:** Evaluate work and process after each sprint

✅ Advantages of Scrum:

- Faster and incremental delivery of features.
- Encourages continuous feedback and improvement.
- Quick identification and resolution of problems through daily stand-ups.
- Promotes transparency and accountability within the team.

❌ Disadvantages of Scrum:

- Not suitable for uncooperative teams
- Scope creep can occur if backlog is not managed well
- Requires experienced team for best results

🎯 Applications:

- Web and mobile app development
- Product-based companies with regular releases
- Projects requiring frequent updates or customer involvement

Adaptive Software Development (ASD)

- Adaptive Software Development (ASD) is a technique for building complex software and systems. ASD is a design principle for the creation of software systems. The principle focuses on the rapid creation and evolution of software systems.
- The goal of the ASD model is to consider midstream changes as the natural flow of projects and to incorporate them into the development model.

  Adaptive software development is made of three steps, each revolving around the coding of a program.

- The first step in ASD is speculation. During this phase, coders attempt to understand the exact nature of the software and the requirements of the users. This phase relies on bug and user reports to guide the project.
    -
- The second step is collaboration. Collaboration. in ASD is a balance between managing the doing and creating and maintaining the collaborative environment needed for output.
    - First, he or she has to decide which parts of the project are predictable.
    - For parts that are unpredictable, a manager has to establish an environment in which a team basically opens, collaborates..

Learning is the last step in adaptive software development. Learning activities expose those products to a variety of stakeholders to ascertain value.

Advantages of ASD:

1. The ASD model works very well for smaller teams (four to eight people) when the requirements technologies are uncertain or domain knowledge is missing.

2. The ASD allows the team to speculate about the final product without providing a detailed plan up front. In this way, the team can continuously adjust the final product over the course of the project.

3. The ASD includes processes for tracking progress and identifying problem areas.

Disadvantages of ASD:

1. In areas of predictable requirements and technologies, the traditional methods will probably work more efficiently than the Adaptive model.

2.The project is very different from managing projects developed with the traditional models because the Adaptive approach is result oriented rather than flow oriented

Dynamic System Development Method (DSDM)

The DSDM is an agile software development approach that provides a framework for building and maintaining systems, which meet tight time constraints through the use of incremental prototyping in a controlled project environment.

DSDM is based on nine key principles that basically revolve around business needs and values, active user involvement, empowered teams, frequent delivery, integrated testing, and stakeholder collaboration.

The main **features of the DSDM** method are as follows:

1. User involvement.
2. Iterative and incremental development.
3. Increased delivery frequency.
4. Integrated tests at each phase.
5. The acceptance of delivered products depends directly on fulfilling requirements.

DSDM specifically calls out "fitness for business purpose" as the primary criteria for delivery and acceptance of a system, focusing on the useful 80% of the system that can be deployed in 20% of the time.

DSDM projects are prioritized using MOSCOW Rules which is stated as follows:

M-Must have requirements.

S-Should have if at all possible.

C-Could have but not critical.

W-Would not have this time, but potentially later.

All critical work must be completed in a DSDM project.

Within each time-box, less critical items are included so that if necessary, they can be removed to keep from impacting higher priority requirements on the schedule.

The DSDM project framework is independent of, and can be implemented in conjunction with, other iterative methodologies such as Programming (XP) and the Rational Unified Process (RUP).
The phases of DSDM in Fig. 1.22 are explained below

1. Feasibility Study:

- It is about whether the proposed method can be applied or not and thorough research is carried out to find out the existing problems.

2. Business Study:

- It establishes the functional and information requirements that will allow the application to provide business value.
- It is about acquiring a clear understanding of the business flow and how the processes are related to each other. It involves identifying the stakeholders and those who are involved in the project.

3. Functional Model Iteration:

- Functional model iteration produces a set of incremental prototypes that demonstrate functionality for the customer.
- In this phase risk has to be identified and a plan on how to deal with risk for future developments.

4. System Design and Build Iteration:

- This phase revisits prototypes built during functional model iteration to ensure that each has been developed.
- In this phase the actual system is built based on the non-functional requirements carried out in the previous phase and the built-in system is implemented in the next phase once the testing is done.

5.Implementation:

This is the final phase in the methodology where the built-in system is moved into the production environment from the developed environment.

Advantages of the DSDM:

1. DSDM provides a technique-independent process and is flexible in terms of requirement evolution.

2. DSDM is designed from the ground up by business people, so business value is identified and expected to be the highest priority deliverable.

3. DSDM incorporates stakeholders into the development process.

4. An emphasis as DSDM on testing is so strong that at least one tester is expected to be on each project team. Which improves software quality and performance.

Disadvantages of the DSDM:

1. DSDM focuses on RAD, which can lead to decrease in code robustness.

2. It requires significant user involvement.

3. DSDM requires a skilled development team in both the business and technical areas.

Crystal:

Crystal focuses on the first Agile principle, individuals and interactions over processes and tools.

The Crystal methodology is an agile software development approach that focuses on people and their interactions over processes and tools.

Characteristics of Crystal include, fast and continuous delivery, less administration and higher involvement of customers in the process, leading to higher customer satisfaction.

The Crystal agile methodology is an agile framework which puts focus on the interaction among individuals/peoples (teams) rather than on the processes and tools used.

Crystal methodology is considered to be among the lightest weighed and versatile approaches in the software development process.

To make Crystal methodology successful, the essential components are teamwork, proper communication, and simplicity.

Adjust and improving technique also plays a vital role in the project's success. Crystal also encourages frequent, incremental deliverables of software and discourages distractions and hurdles.

Agile Crystal methodology focuses on the interaction between people rather than the processes involved or tools used.

Features of Crystal:

1. People-Centric: Emphasizes team collaboration, direct communication and minimal bureaucracy.

2. Lightweight and Flexible: Unlike Scrum Crystal doesn't have a fixed set of rules. It adapts based on project size, priorities, and team dynamics.

3. Frequent Delivery: Encourages regular releases of working software to get early feedback.

4. Reflective Improvement: Teams continuously review and improve their processes .

Crystal is a family of methodologies, with different variations (e.g., Crystal Clear, Crystal Yellow, Crystal Orange, Crystal Red) tailored to different team sizes and project complexities.

Crystal Clear: Constitutes of less than 8 team members

Crystal Yellow: Constitutes of 10 to 20 team members

Crystal Orange: Constitutes of 20 to 50 team members

Crystal Red: Constitutes of 50 to 100 team members

Advantages Crystal Agile Framework:

1. Flexibility: It is highly adaptable, enabling the team to adjust to changing requirements.

2. Faster Delivery

3. Adaptability: It lets the team respond well to the demanding requirements.

4. Higher Quality: It emphasizes on quality, enabling the team to detect and fix defects early in the development process, resulting in a higher quality product.

5. Improved Customer Satisfaction: The framework promotes customer involvement, enabling the team to deliver products that meet customer needs, resulting in higher customer satisfaction.

6. Reduced Risk: The framework promotes risk management, enabling the team to identify and mitigate potential risks early in the development process, reducing the likelihood of project failure.

7. Increased Productivity: The framework enables the team to focus on delivering the highest value features, which can increase productivity and reduce waste.

Disadvantages of Crystal Agile Framework:

1. Limited Scalability
2. Ambiguity
3. Inability to handle Regulatory Requirements
4. Lack of Predictability
5. Lack of Documentation
6. Dependence on Team Expertise

Agile Unified Process (AUP)

Unified Process (UP) is a software development process framework that provides guidelines to design and build high-quality softwares.

Agile Unified Process (AUP) is a simplified version of the Rational Unified Process (RUP). It describes a simple, easy to understand approach to developing business application software using agile techniques.

The AUP is an iterative, and adaptable software development methodology that combines the best practices of agile development with the disciplined approach of the Unified Process (UP). AUP is designed to deliver high-quality software that meets the changing needs of the stakeholders in an efficient and effective manner.
RUP is an iterative and incremental approach to software development

The unified process is a "use-case driven, architecture-centric, iterative and incremental" software process designed as a framework for UML methods and tools.

Phases of AUP

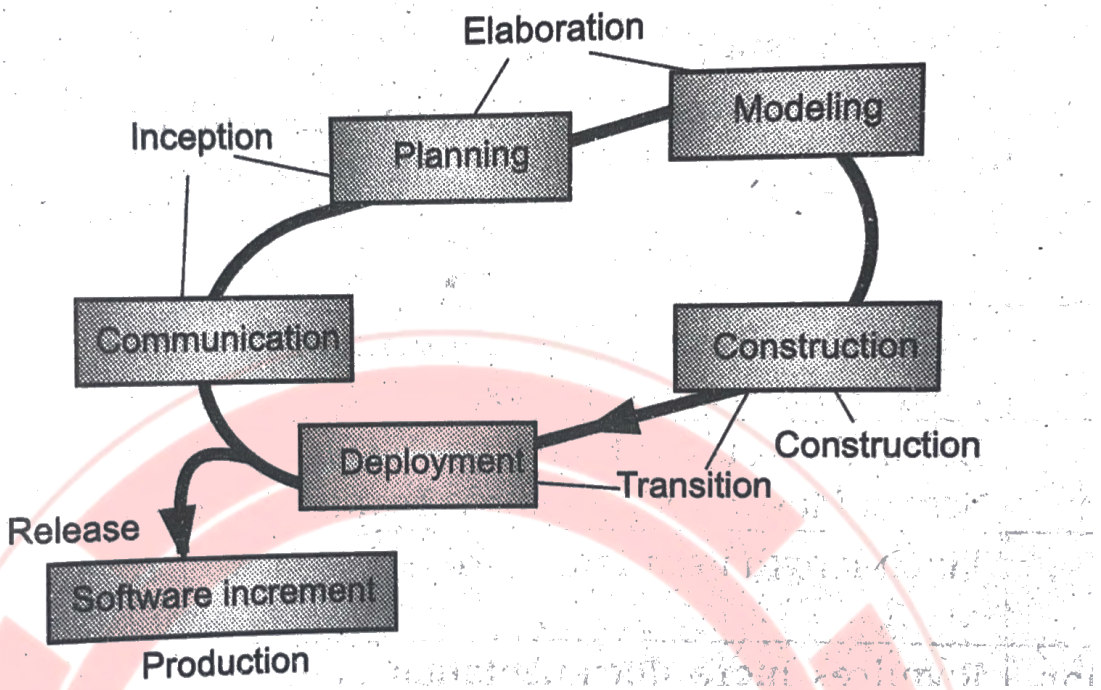
.**1.** An inception phase that encompasses both customer communication and planning activities and emphasizes the development and refinement of use-cases as a primary model. 2.

2. An elaboration phase that encompasses the customer communication and modeling activities focusing on the creation of analysis and design models with an emphasis on class definitions and architectural representations.

3. A construction phase that refines and then translates the design model into implemented software components.

4. A transition phase that transfers the software from the developer to the end-user for beta testing and acceptance.

5. A production phase in which on-going monitoring and support are conducted



Advantages of AUP:

1. Adaptability
2. Flexibility
3. High-Quality Software
4. Faster Time-to-Market
5. Risk Management

Disadvantages of AUP:

1. Lack of Formal Process: AUP may not provide the level of structure and formal process that some organizations require.

2. Lack of Documentation: AUP is a lightweight methodology that emphasizes working software over comprehensive documentation.
3. Limited Scalability: AUP may not be suitable for large, complex projects that require a high degree of coordination and management.

4. Dependency on Team Experience: AUP relies on the experience and expertise of the development team to deliver high-quality software.