UNIT-1

Introduction to Software Engineering

- Software crisis started in the mid of the late 1960s and the early 1970s.
- The importance of software, software industry, and software users has evolved rapidly.
- The fields of computing have become complex and diverse in the modern information society.
- The main focus of practitioners from the computing outset was to provide solutions to the complexity barriers of producing software, setting up the software industry, and escalating the number of software users.
- The dependency of business organizations on software and technology has increased.
- Small-scale and large-scale business organizations have automated their business processes for increased ease and effectiveness.
- The dynamic nature of changing software technology forces the adoption of software construction and maintenance processes according to the suitability of the application.
- Software companies are moving toward component-based development, where components are assembled rather than developed from the scratch
- The mobile nature of software allows changes in requirements as and when required but changing requirements during development needs a systematic process to incorporate the changes into software work products.
- Along with development and maintenance, software project management also plays an important role in the project success.
- Apart from the process of development, maintenance, management, and planning, some software engineering approaches aim to improve the process itself.
- The ultimate goal of software practitioners is to produce faster, better, and cost-effective products

Software

- Software is a collection of computer programs that when executed together with data provide desired outcomes.
- A computer program is a set of instructions written in a programming language.
- Programs run within specified constraints and environment, with defined rules of execution.
- Each standard program has a certain procedure for its execution and operation.
- Data play an important role in program execution for providing useful information in some form.
- The documentation is important in understanding the software code, design, constraints, customer needs, and specification for further maintenance.

- *Software is a collection of computer programs, together with data, procedure, rules, and associated documentation, which operate in a specified environment with certain constraints to provide the desired outcomes (IEEE).*
- Software concentrates more on quality issues, such as interoperability, portability, usability, reliability, and robustness; and it is sometimes referred to as *industrial quality software*

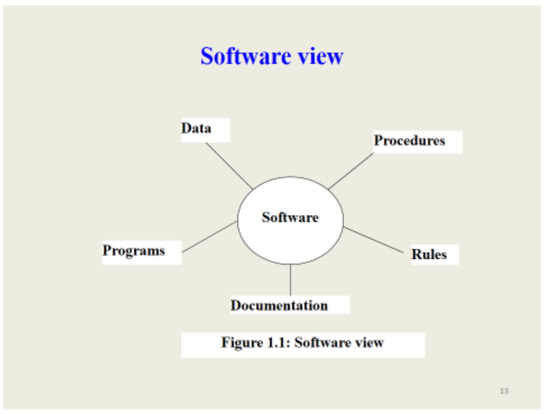


Figure 1.1: Software view

Software characteristics

- Software has logical properties rather than physical.
- Software is mobile to change.
- Software is produced in an engineering manner rather than in classical sense.
- Software becomes obsolete but does not wear out or die.
- Software has a certain operating environment, end user, and customer.
- Software development is a labor-intensive task

Software Classifications:

- Generic software products are developed for general purpose, regardless of the type of business.
- Customized software products are developed to satisfy the need of a particular customer in an organization.
- System software is the computer software that is designed to operate the computer hardware and manage the functioning of the application software running on it.
- e.g., device drivers, boot program, operating systems, servers, utilities, and so on.
- Application software is designed to accomplish certain specific needs of the end user.
- e.g., video editing software, word processing software, database software, and simulation software are some examples of application software etc.
- Embedded software is a type of software that is built into hardware systems. Embedded software is used to control, monitor, or assist the operation of equipment, machinery, or plant.

- Many of the advanced functions that are common in modern devices are used in daily life, such as in washing machines, cars, mobiles, etc.
- There are certain characteristics of embedded systems, such as naive , timeliness, concurrency, liveness, reactivity, and heterogeneity.
- Controllers, real time operating systems, communication protocols are some examples of embedded software.
- *Product-line software* is a set of software intensive systems that share a common, managed set of features to satisfy the specific needs of a particular market segment or mission.
- Product line software improves time to market, productivity, quality, and other business drivers. At the same time, it reduces product cost.
- It can also enable rapid market entry and flexible response, and provide a capability for mass customization.
- Some common applications are multimedia, database software, word processing software, etc. Reuse-based sSoftware eEngineering bBusiness (RSEB) promotes product-line software.
-

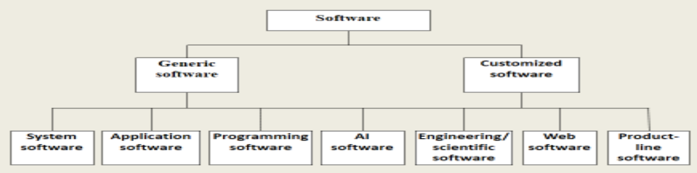


Figure 1.2: Software classification

Engineering Discipline:

- Engineering is a disciplined approach with some organized steps in a managed way to construction, operation, and maintenance of software.
- Engineering of a product goes through a series of stages, i.e., planning, analysis and specification, design, construction, testing, documentation, and deployment.
- The disciplined approach may lead to better results.
- The general stages for engineering the software include feasibility study and preliminary investigation, requirement analysis and specification, design, coding, testing, deployment, operation, and maintenance.

Software Crisis:

- Software crisis, the symptoms of the problem of engineering the software, began to enforce the practitioners to look into more disciplined software engineering approaches for software development.
- The software industry has progressed from the desktop PC to network-based computing to service-oriented computing nowadays.

- The development of programs and software has become complex with increasing requirements of users, technological advancements, and computer awareness among people.
- *Software crisis symptoms*
    - complexity,
    - hardware versus software cost,
    - Lateness and costliness,
    - poor quality,
    - unmanageable nature,
    - immaturity,
    - lack of planning and management practices,
    - Change, maintenance and migration,
    - etc.

What is Software Engineering?

- *The solution to these software crises is to introduce systematic software engineering practices for systematic software development, maintenance, operation, retirement, planning, and management of software.*
- The systematic means the methodological and pragmatic way of development, operation and maintenance of software.
- Systematic development of software helps to understand problems and satisfy the client needs.
- Development means the construction of software through a series of activities, i.e., analysis, design, coding, testing, and deployment.
- Maintenance is required due to the existence of errors and faults, modification of existing features, addition of new features, and technological advancements.
- Operational software must be correct, efficient, understandable, and usable for work at the client site.
- IEEE defines
    - *The systematic approach to the development, operation, maintenance, and retirement of software.*

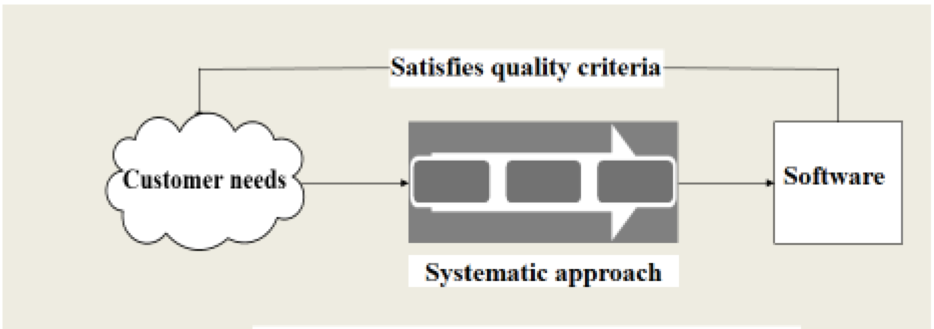


Figure 1.3: Software engineering view

Evolution of Software Engineering Methodologies:

- A *software engineering methodology* is a set of procedures followed from the beginning to the completion of the development process.
- Software engineering methodologies have evolved with increasing complexities in programming and advancements in programming technologies.
- The most popular software methodologies are:
  - *Exploratory methodology*
  - *Structure-oriented methodology*
  - *Data-structure-oriented methodology*
  - *Object-oriented methodology*
  - *Component-based development methodology*

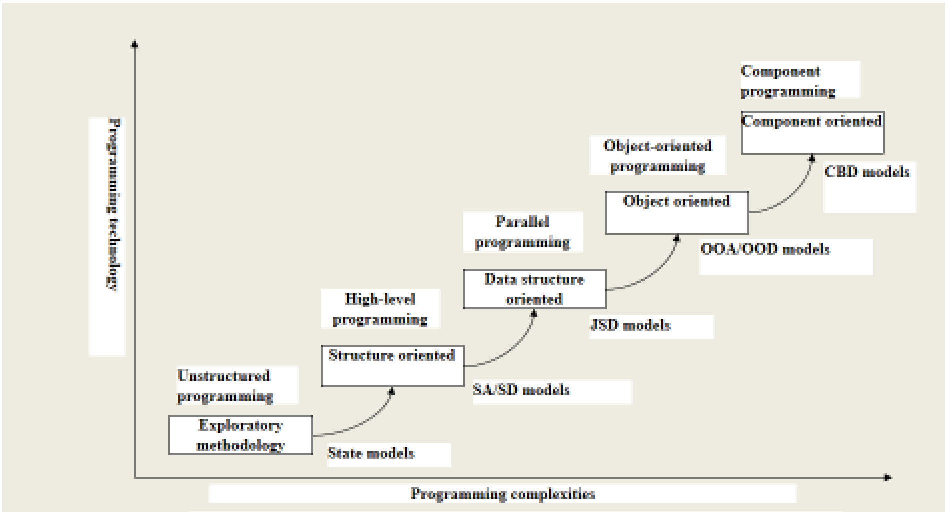


Figure 1.4: Evolution of software engineering methodologies

Exploratory Methodology

- Exploratory style of software engineering is a methodology that applies to the development of programs whose requirements are initially unclear.

- It involves experimentation and exploring the programs through step-by-step programming.
- Exploratory style is sufficient to develop software to test research hypothesis but it is unable to meet reliability, extensibility, and maintainability goals.
- In exploratory style, errors are detected only during the final product testing.
- Maintenance is very difficult and costly because of the lack of documentation and multiplicity of changes in the initial proposal.
- Exploratory style uses unstructured programming or design heuristics for program writing, where the focus is given on global data items.

Structure-Oriented Methodology

- Structured methodology focuses on procedural approach, which concentrates on developing functions or procedures,
- It has three basic elements, *Sequence, Selection, Iteration.*
- Structure-oriented methodology uses a variety of notations, such as Data Flow Diagrams (DFD), data dictionary, Control Flow Graphs (CFG), Entity Relationship (ER) diagrams, etc., to design the solution of the software.
- Structure-oriented methodology is suitable for all types of projects. Procedural programming languages (for example, C, COBOL, BASIC, FORTRAN, etc.) are actually very powerful and easy to understand.
- Structure-oriented approach is preferred in scripts and embedded systems with small memory requirements and high speed.

Data-Structure-Oriented Methodology

- Data-structure-oriented methodology concentrates more on designing data structures rather than on procedures and control.
- Jackson Structured Design (JSD) methodology developed by Michael Jackson in 1970 is a famous data-structure-oriented methodology that expresses how functionality fits in with the real world.
- JSD is a useful methodology for concurrent software, real time software, microcode, and for programming parallel computers because JSD emphasizes actions more and attributes less.
- Though JSD is good for shaping real world scenario, it is complex and difficult to understand.

Object-Oriented Methodology

- Object-oriented methodology emphasizes the use of data rather than functions.
- Data and procedures are built around these objects. The real world entities are treated as *objects.* The objects having characteristics in common are grouped into *classes*.
- An object involves p*roperties and methods*
- Object-oriented methodology has three important concepts: *modularity, abstraction,* and *encapsulation*.

- Object-Oriented Analysis (OOA) and Object-Oriented Design (OOD) techniques are used in object-oriented methodology.
- OOA is used to understand the requirements by identifying objects and classes, their relationships to other classes, their attributes, and the inheritance relationships among them.
- OOD creates *object models* and maps the real world situation into the software structure.
- Object-oriented methodology is the latest and the most widely-used method for the development of applications in a variety of domains.

Component-Based Development Methodology:

- Component-Based Development (CBD) is a significant methodology for communication among different stakeholders, and for large-scale reuse.
- CBD is a system analysis and design methodology that has evolved from the object-oriented methodology.
- It is largely based on its focus on reuse.
- Its proponents promise faster time to market, cost reduction, better quality, flexibility, and scalability. It places large, independently-packaged, reusable components at the core of software development.
- **Software Engineering Challenges**
- Problem Understanding
- Quality and Productivity
- Cycle Time and Cost
- Reliability
- Change and Maintenance
- Usability and Reusability
- Repeatability and Process Maturity
- Estimation and Planning

Software Engineering Principles

- Focus on customers' problems, needs, priorities, and expectations
- Choose appropriate process model
- Decomposition and modularity
- Abstraction
- Encapsulation
- Incremental development

- Understandability
- Consistency and completeness
- Generality
- Perform verification and validation to maintain quality
- Follow-up the scope statement, deadlines, and early product delivery
- Design for change
- Follow disciplined and mature process
- Take responsibility and commit
- Better planning and management rather than technology

Conclusion:

- The adoption of systematic software engineering approach has been a great effort in the software industry to produce quality software.
- Software crisis in today’s scenario has certain symptoms, such as complexity, hardware versus software cost, lateness, costliness, poor quality, unmanageable nature, immaturity, and lack of planning and management practices.
- IEEE defines software engineering as a systematic approach to development, operation, maintenance, and retirement of software.
- The main goal of software engineering is to understand customer needs and develop software with improved quality, on time and within budget.
- There are certain other important challenges, such as understanding the customer requirements, frequently changing technology, changing customer requirements, increasing market of reuse business, platform independency, and so on.

Software Processes

Software Process

- A software process is a coherent set of activities, procedures, policies, organizational structures, constraints, technologies, and artifacts that are needed to develop software products.
- The main goal of a software process is to produce quality software with defined constraints.
- A *software process* is a set of ordered activities carried out to produce a software product.
- An *activity* is a specified task performed to achieve the process objectives.
- Each activity of software process involves tools and technologies, procedures, and artifacts.
- A software process is a complex entity in which each activity is executed with the supporting tools and techniques.
- A *software project* is an entity, with defined start and end, in which a software process is being used.
- A successful project is the one that conforms with the project constraints (cost, schedule, and quality criteria).
- A *product* is the outcome of a software project produced through processes.
- A project can have more than one product called *work products*. A work product is the intermediate outcome of processes.

- Software process, project, and products are interrelated to each other for the development of software.

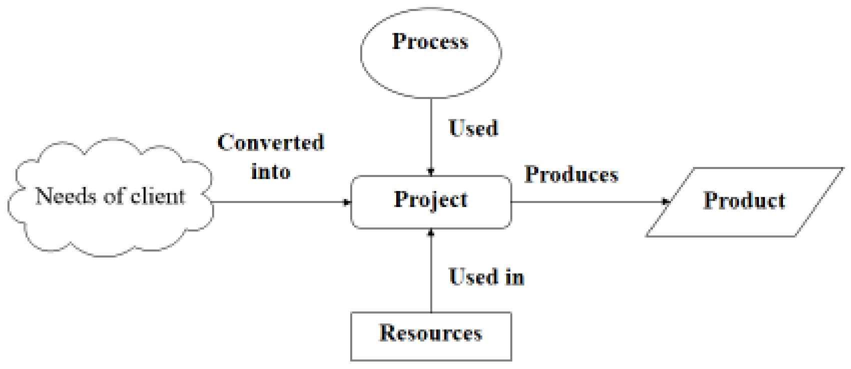


Figure 2.1: Process, project and product

Software Process Model:

- A *software process model* is a generic representation of a software process instantiated for each specific project.
- A process model is a set of activities that have to be accomplished to achieve the process objectives.
- A process model can be made practical by executing the concept, technologies, implementation environment, process constraints, and so on.
- These models may be related to development, management, improvement, and maintenance.
- Process models specify the activities, work products, relationships, milestones, etc.

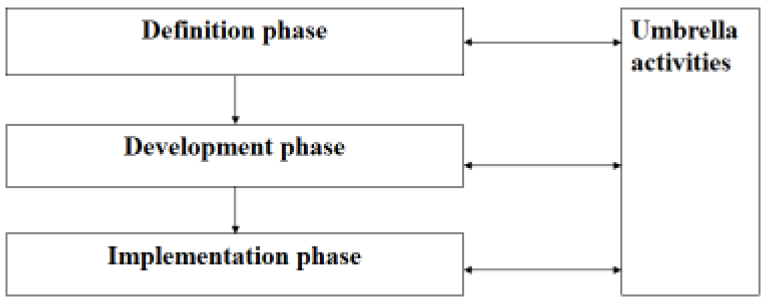


Figure 2.2: Generic representation of process model

Generic representation of process model

- **Definition phase** concentrates on understanding the problem and planning for the process model.
  - The activities may include problem formulation, problem analysis, system engineering, and project planning for the process.
- **Development phase** focuses on determining the solution of the problem with the help of the umbrella activities.
  - The main activities of this phase are designing the architecture and algorithms of the system, writing codes, and testing the software.
- **Implementation phase:** Deployment, change management, defect removal, and maintenance activities are performed in this phase.
- The *umbrella activities* are responsible for ensuring the proper execution of definition, development, and implementation phases.
  - The umbrella activities are project management, quality assurance, configuration management, risk management, work products preparation and deployment, and process improvement.

Elements of Software Process

- **Artifacts** are tangible work products produced during the development of software.
- **Activity** specifies the tasks to be carried out implicitly or explicitly.
- **Constraint** refers to the criteria or condition that must be met or possessed by a software product
- **People** are persons or stakeholders who are directly or indirectly involved in the process.

Tools and Technology provide technical support to the methods or techniques to be used for performing the activities.

Characteristics of Software Process

- Understandability
- Effectiveness
- Predictability
- Maintainability
- Reliability
- Changeability
- Improvement
- Monitoring and Tracking
- Rapidity
- Repeatability
- *quality,*
- *adoptability,*
- *acceptability,*
- *visibility,*
- *supportability,*

Process Classification

- Software processes may be classified as
    - *product development process,*
    - *project management process,*
    - *change management process,*
    - *process improvements,*
    - *Quality management process.*

<u>Product Development Process</u>

- Product development processes focus mainly on producing software products.
- These processes involve various techniques, tools, and technologies for developing software.
- Such processes include various activities like conceptualization, designing, coding, testing, and implementation of a new or existing system.

- There are certain work products of these activities, such as software requirements specifications (SRS), design models, source codes, test reports, and documentation.
- The most widely used software development process models are the waterfall model, prototyping model, spiral model, agile model, RUP, and so on models.

Project Management Process

- Project management processes concentrate on planning and managing projects in order to achieve the project objectives.
- The goal of these processes is to carry out the development activities within time, budget, and resources.
- There are various project management processes, which are scope, budget, schedule, quality, information, team, risk, and contracts.
- Initiating, planning, coordinating, controlling, executing, and terminating are the main activities of a general project management process.

Process Improvement Process

- These processes are involved in improving the process itself.
- The ultimate goal of improvement in a process is to enable the organization to produce more quality products.
- Process improvement is an incremental improvement of process, which is used for software development.
- There exist various process improvement process models, such as CMMI, QIP, continuous quality improvement (QI), total quality management (TQM), Six Sigma, and so on.
- All these process models provide improvement guidelines and standards for improving software processes.

Configuration Management Process

- Changes may occur in projects, processes, and products as these entities are evolutionary in nature.
- Changes may arise due to either change in the customer requirements or discrepancies in the work products or procedures from developer's side.
- There is a need for a lot of efforts and systematic procedures for performing these changes.
- Configuration management includes various activities for performing changes, such as identification of configuration items, devising mechanisms for performing changes, controlling changes, and tracking the status of those changes.

Quality Management Process

- A quality management process provides metrics, feedback, and guidelines for the assurance of product quality.

- Quality system ensures that a proper business system is working with the help of various monitors and controls.
- Software quality organization gives information and expertise to development and management process for quality production.
- The main activities of software quality groups are verification and validation, acceptance testing, measurement and metrics, process consulting, and so on.
- ISO 9000 is a framework that provides certain guidelines for the quality system.

Phased Development Life Cycle

- Product development process is carried out as a series of certain activities for software production. Each activity in the process is also referred to as a *phase*.
- General activities include feasibility study, analysis, design, coding, testing, implementation, and maintenance.
- Collectively, these activities are called the *software development life cycle (SDLC)* or simply *software life cycle* and each of these activities are called *life cycle phase*.
- The SDLC provides a framework that encompasses the activities performed to develop and maintain software.
- Some of these models are waterfall, prototyping, spiral, incremental, agile process, RUP process model, and so on.

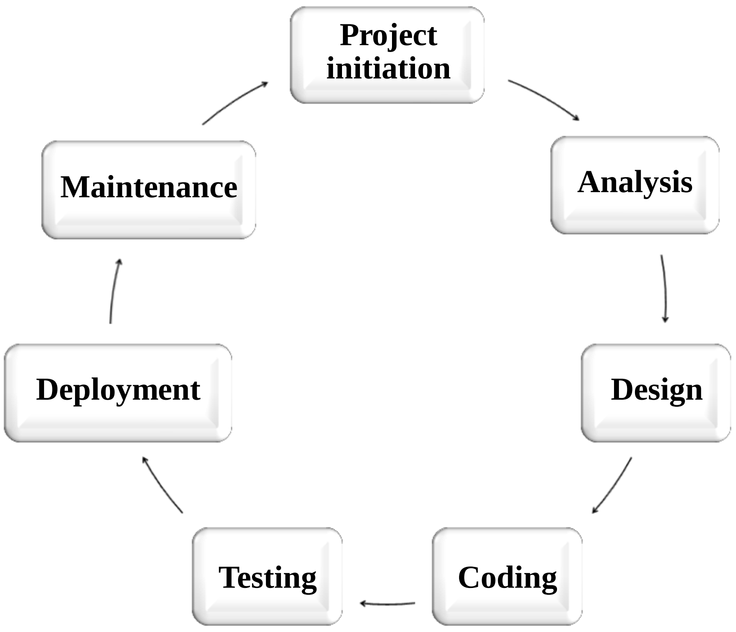


Figure 2.3: Software Development Life Cycle (SDLC) activities

- Project Initiation
  - The aim of project initiation is to study the existing system; determine the feasibility of a new system; and define the scope, key elements, and a plan for the successful completion of the project.
  - Project initiation involves preliminary investigation, feasibility study, and a project plan.
  - *Preliminary investigation (PI)* is the initial step that gives a clear picture of what actually the physical system is.
  - PI goes through problem identification, background of the physical system, and the system proposal for a candidate system.
- Project Initiation
  - The purpose of the *feasibility study* is to determine whether the implementation of the proposed system will support the mission and objectives of the organization.
  - Feasibility study ensures that the candidate system is able to satisfy the user needs; promotes operational, effective use of resources; and is cost effective.
  - There are various types of feasibility study performed, such as technical, economical, operational, and so on
  - Feasibility report is prepared and submitted to the top-level management. A positive report leads to project initiation.
  - A detailed project plan is actually prepared after knowing the requirements
- Requirements Analysis
  - Requirement analysis is the process of collecting factual data, understanding the processes involved, defining the problem, and providing a document for further software development.
  - Requirement analysis is a systematic approach to elicit, organize, and document the requirements of a system.
  - The requirement analysis phase consists of three main activities: *requirements elicitation, requirements specification*, and *requirements verification and validation*.
- Software Design
  - Software design focuses on the solution domain of the project on the basis of the requirement document prepared during the analysis phase.
  - The goal of the design phase is to transform the collected requirements into a structure that is suitable for implementation in programming languages.
  - The design phase has two aspects: *physical design* and *logical design*.
  - Physical design concentrates on identifying the different modules or components in a system that interact with each other to create the architecture of the system.

  - In logical design, the internal logic of a module or component is described in pseudo code or in an algorithmic manner.

- Coding
  - The coding phase is concerned with the development of the source code that will implement the design.
  - This code is written in a formal language called a programming language, such as assembly language, C++, Java, etc.
  - Good coding efforts can reduce testing and maintenance tasks.
  - Programs must be modular so that they can help in rapid development, maintenance, and enhancements of the system.
  - The programs written during the coding phase must be easy to read and understand.
- Testing
  - Before the deployment of the software, testing is performed to remove the defects in the developed system
  - Testing covers various errors at the requirements, design, and coding phases.
  - Testing is performed at different levels: unit testing, integration testing, system testing, and acceptance testing.
  - Various special tests are also performed to check the functionality of the system, such as recovery testing, performance testing, load testing, security testing, and so on.
  - Testing is an important technique of software quality assurance.
- Deployment
  - The purpose of software deployment is to make the software available for operational use.
  - It includes various activities to make a system available for assembly and to transfer it to the customer site.
  - Required resources are procured to operate at the customer site and important information is collected for the deployment process.
  - During deployment, all the programs files are loaded onto user's computer. After installation of all the modules of the system, training of the user starts.
  - Documentation is prepared in the form of a user manual or system operation process which ensures the continuity of the system.
- Maintenance
  - The maintenance phase comes after the software product is released and put into operation through the deployment process.

- Software maintenance is performed to adapt to changes in a new environment, correct bugs, and enhance the performance by adding new features.
- The maintenance activities can be classified as adaptive (changes in the software environment), perfective (new user requirements), corrective (fixing errors), preventive (prevent problems in the future).
- The software will age in the near future and enter the retirement stage.
- In extreme cases, the software will be reengineered onto a different platform.

Software Development Process Models:

- Classical waterfall model
- Iterative waterfall model
- Prototyping model
- Incremental model
- Spiral model
- Agile process model
- RUP process model

Classical Waterfall Model

- The waterfall model is a classical development process model proposed by R. W. Royce in 1970.
- In this model, software development proceeds through an orderly sequence of transitions from one phase to the next in order (like a waterfall).
- It is the simplest and the most widely used model in development.
- This model produces standard outputs at the end of every phase, which is called *work products*.
- This model was enhanced with a feedback process, which is referred to as an *iterative model*.

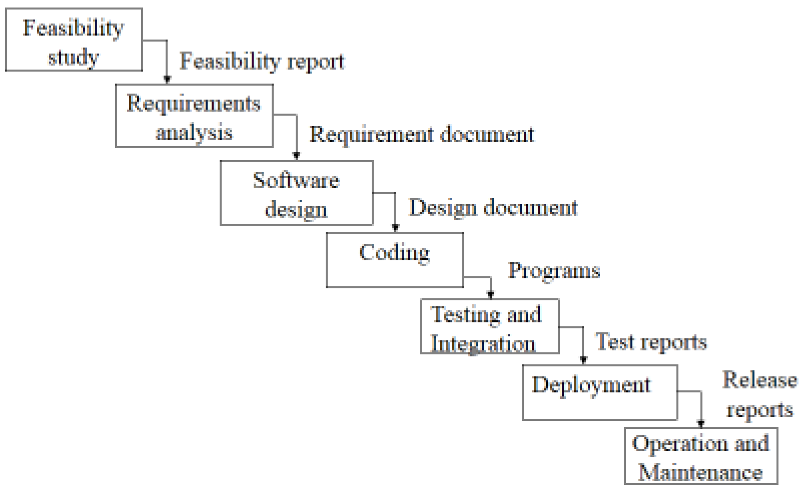


Figure 2.4: Classical waterfall model

- Advantages
  - The main advantage of the waterfall model is that it is easy to understand and implement.
  - Due to the straightforward organization of phases, it is fit for other engineering process models, such as civil, mechanical, etc.
  - It is a document-driven process that can help new people to transfer knowledge.
  - Milestones and deliverables at each stage can be used to monitor the progress of the project.
  - This model works well on large and mature products. It is not well suited for small teams and projects.
  - Where the requirements are well understood and the developers are confident, the waterfall model works well.
- Disadvantages
  - The model assumes that the requirements will not change during the project. Sometimes, it is unrealistic to expect accurate requirements early in a project.
  - It is very difficult to estimate the time and cost in the waterfall model.
  - There may be difference of opinions among the specialists because there are specialized teams for each individual phase.
  - The people mentally ready to work in a phase will have to wait until its previous phase is completed.
  - Due to so much emphasis on documentation, sometimes people may become irritated.
  - Fixing the software and hardware technology early may create problems for larger projects.
  - There is no inherent risk management policy in this model.

- This model works well on large and mature products.
- It is not well suited for small teams and projects.

Iterative Waterfall Model

- The iterative waterfall model is an extended waterfall model with backtracking at each phase to its preceding phases.
- The life cycle phases are organized similar to those in the classical waterfall model. The only difference is backtracking of phases on detection of errors at any stage.
- The errors can come at any stage of the development life cycle.
- Once a defect is detected there is a need to go back to the phase where it was introduced.
- On defect detection at the source, some of the work performed during that phase and all the subsequent phases may be revised.
- On error detection at any phase, it may be required that the preceding and succeeding phases be changed.
- Detecting and removing defects earlier in the process of software development is called *phase containment of errors*.
- Removing defects in early phases of development reduces testing and maintenance efforts.
- The iterative waterfall model is the most widely used model and it is simple to apply it in projects.
- Still it is a document-driven model.

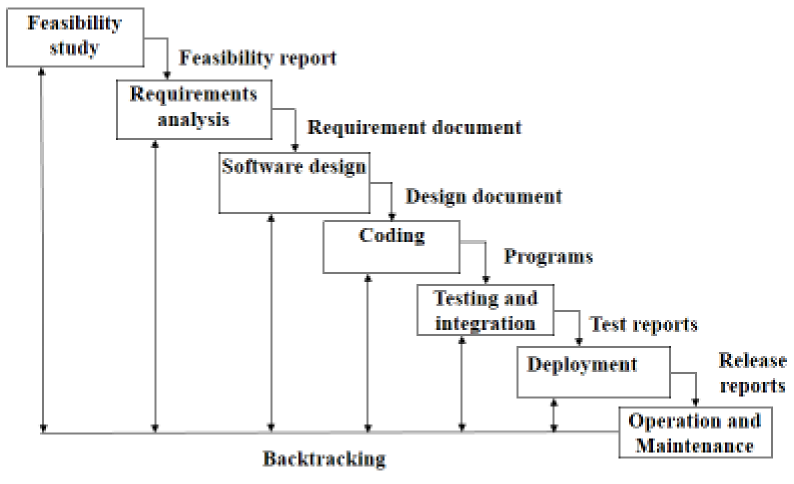


Figure 2.5: Iterative waterfall model

- It is very difficult to manage changes between the phases.
- There is a possibility of blocking states, which can slow down the productivity and efficiency of the process.
- Risks are not addressed in this model.

- This model is most suitable for simple projects where the work products are well defined and their functioning is understood.

Prototyping Model

- Prototyping is an alternative in which partial working software (i.e. a prototype) is initially developed instead of developing the final product.
- IEEE defines prototyping as “a type of development in which emphasis is placed on developing prototypes early in the development process to permit early feedback and analysis in support of the development process.”
- Prototype development is a toy implementation, which provides a chance to the customer to give feedback for final product development.
- A prototype provides limited functionalities, low reliability, and insufficient performance as compared to the actual software.
- A prototype helps customer to understand the requirements that can further reduce the possibility of requirement changes.
- The prototype model is well suited for projects where requirements are difficult to understand and the customer is not confident in illustrating and clarifying the requirements.
- It fits best where the customer risks are related to the changing requirements (software and hardware requirements) of the projects.
- But this model requires exclusive involvement of the customer, which is not always possible. Sometimes bad design decisions during prototype development may propagate to the real product

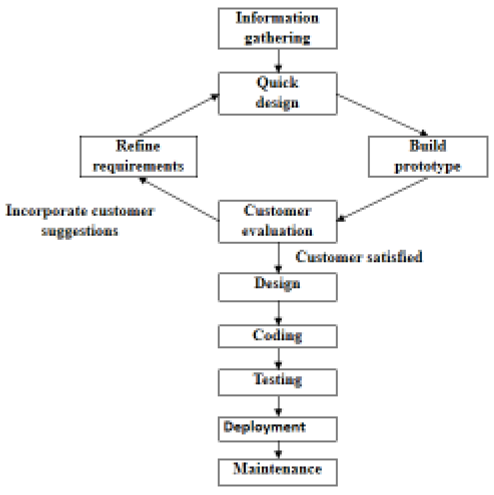


Figure 2.6: Prototyping model

Incremental Model

- The incremental model is an intuitive approach to the waterfall model with fewer restrictions.
- The activities are performed in the same order as in the waterfall model, but they are conducted in several iterations.

- Each iteration releases a fully functional work product by providing additional functionalities in successive releases.
- The final iteration releases the complete product

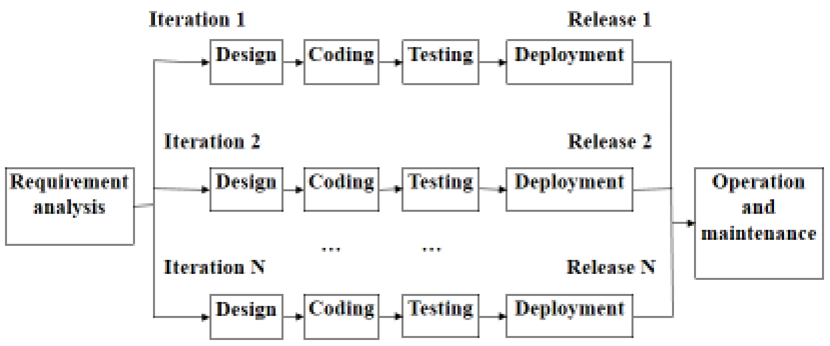


Figure 2.7: Incremental Model

- The requirements are functionally divided and prioritized according to the needs of the customers.
- A blueprint of the product based on the prioritized requirements is also designed, which describes dependency of tasks on each other.
- A *project control list* is prepared, describing the order of the tasks to be performed and outcomes of the task to be released.
- Each task in the project control list is treated as a mini project. Each item in the list is a module, which is to be developed in increments.
- Each task is removed from the project control list and developed using the waterfall model in a sequential manner.
- Each new release is integrated with the existing increments to provide enhanced functionality with each delivered increment.
- The length of iterations is generally kept very short and fine. For example, an iteration can be a duration of 6 weeks, 10 weeks, etc.
- The number of iterations depends upon the nature of the project and the features it supports.
- The main advantage of the incremental model is the early production of working software during the software life cycle.
- Because each module is tested thoroughly, there is little possibility to change scope and requirements in the final software.
- Due to incremental development, testing and debugging of each module become easier.

- This model is also helpful in handling risks (technical, requirements, usability, etc.) because risky modules are identified and handled in a separate iteration.
- This model is suitable for larger projects where requirements are somewhat clear and which need phase-wise implementation.
- Mostly this model is used in web applications, object-oriented development projects, and product-based companies.
- Also, it is widely used by many commercial software companies and system vendors.
- Each phase of an iteration is rigid and does not overlap another. Therefore, coordination between iterations is required for better quality products.
- Sometimes, an initial upfront cost is required for parallel development of various modules.

Spiral Model

- The spiral model is an iterative software development approach, which was proposed by Boehm in 1988.
- In this model, activities are organized as a spiral with many loops.
- Each loop in the spiral represents a phase of software development.
- The exact number of loops in the spiral is not fixed.
- The main focus of this model is identification and resolution of potential risks (product risks, project risks, and process risks).

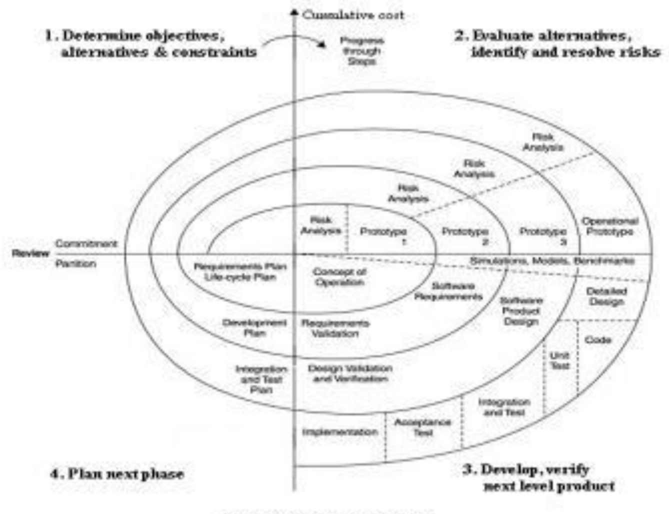


Figure 2.8: Spiral Model

- Each loop in the spiral is split into four quadrants. Each of these four quadrants is used for the development of each phase.
  - *Determine objectives, alternatives, and constraints.*
  - *Evaluate alternatives; identify and resolve risks.*
  - *Develop, verify the next level product.*

  - *Plan for the next phase.*

- The redial dimension represents the cumulative cost incurred so far for the development of phases in a project.
- The angular dimension indicates the progress made so far in completing each cycle.
- It is considered a *Meta model* because it incorporates the features of all other models, which are the waterfall, prototyping, incremental, simulation, and performance models.
- The performance of prototype development is evaluated using benchmarking tools, simulation models, and customer feedback.

Agile Process Model

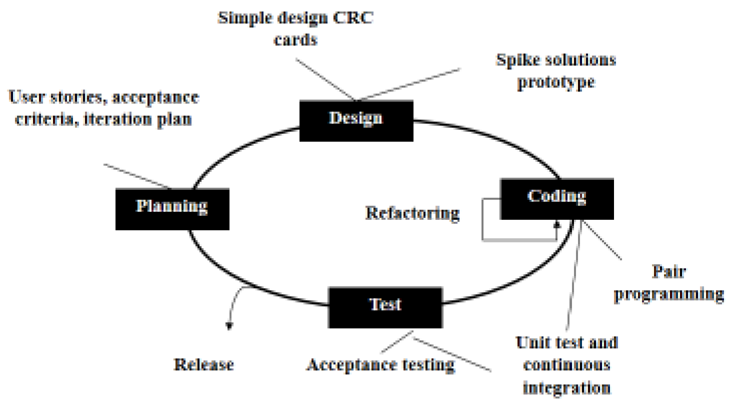
- The agile process model is a group of software development methodologies based on iterative and incremental development.
- In February 2001, 17 software developers published the manifesto of agile software development to define the approach.
- Some of the manifesto's authors formed the agile alliance. The manifesto of agile software development is as follows:
- *"We are uncovering better ways of developing software by doing it and helping others do it."*
  This work focuses to value the following:
- *Individuals and interactions* over processes and tools
- *Working software* over comprehensive documentation
- *Customer collaboration* over contract negotiation
- *Responding to change* over following a plan
- Agile software development methods
  - extreme programming (XP),
  - Scrum,
  - dynamic systems development method (DSDM),
  - adaptive software development (ASD),
  - Crystal, feature-driven development (FDD),
  - test-driven development (TDD),
  - pair programming,
  - refactoring,
  - agile modeling,
  - Internet speed development, and so on
- Extreme Programming (XP) (initiated by Kent Beck)

- XP is a system of practices that is being evolved by a community of software developers to address the problems of quickly delivering quality software.
- Extreme programming process is an iterative development process which consists of planning, design, coding, and test phases.
- The XP process is the most suitable practice for dynamically changing requirements, projects having risks, small developer groups, and non-fixed scope or price contract.
- However , XP is difficult to get representative of customers, who can sit with the team and work with them daily.
- Also, there is a problem of architectural design because the incremental style of development means that inappropriate architectural decisions are made at an early stage of the process.

- The XP process is the most suitable practice for dynamically changing requirements, projects having risks, small developer groups, and non-fixed scope or price contract.

- This practice produces good quality products for the regular involvement of customers.

- However , XP is difficult to get representative of customers, who can sit with the team and work with them daily.
- Also, there is a problem of architectural design because the incremental style of development means that inappropriate architectural decisions are made at an early stage of the process.



Simple design CRC cards
Spike solutions prototype
User stories, acceptance criteria, iteration plan
Design
Planning
Refactoring
Coding
Pair programming
Test
Release
Acceptance testing
Unit test and continuous integration

Figure 2.9: Extreme programming process

Scrum:

- Benefits and drawbacks to Scrum process
  - It is a completely developed and tested feature in short iterations.
  - It is a simple process with clearly defined rules. It increases productivity and the self-organizing team member carries a lot of responsibility.
  - It improves communication and combination with extreme programming.
  - It has no written documentation and sometimes violation of responsibilities.

Scrum

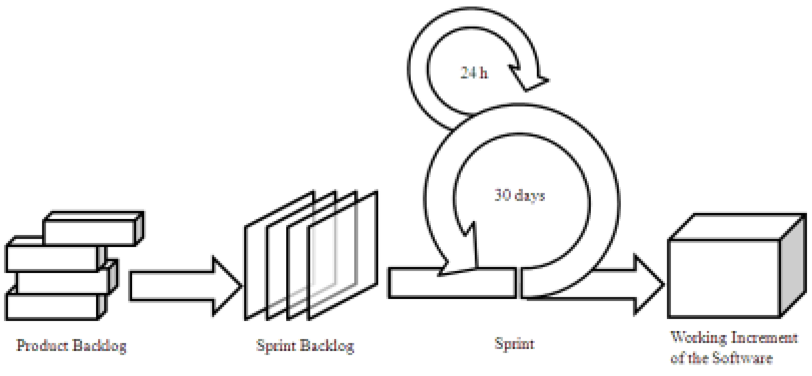


Figure 2.10: The Scrum Process

RUP Process Model

- The Rational Unified Process (RUP) is a use-case driven, architecture-centric, iterative, and incremental process model.
- The RUP focuses on creating and maintaining *models* rather than documentation.
- It is derived from Unified Modeling Language (UML), which is an industry-standard language that helps to clearly communicate requirements, architectures, and designs.
- The RUP is supported by tools which automate most of the activities of the process.
- The RUP divides the development cycle into four consecutive phases; namely, *inception, elaboration, construction,* and *transition.*

- Inception phase:
  - It establish the business case for the system and delimit the project scope.
  - The business case includes success criteria, risk assessment, and estimate of the resources needed; and a phase plan showing dates of major milestones.
  - This phase produces vision document of the project, initial use-case model, initial risk assessment, business model, and a project plan showing the phases and iterations.

  - At this stage, customers will be clear with their requirement and life cycle objectives milestone will be produced.

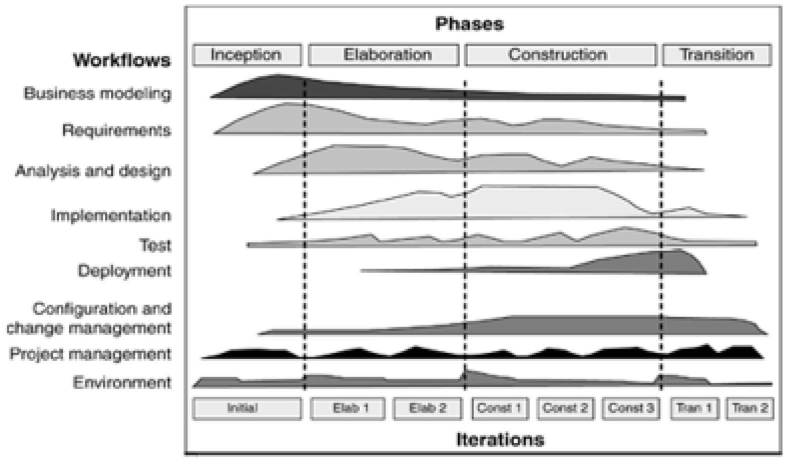


Figure 2.11: The RUP Process Model

- Elaboration phase*:*
  - It analyzes the problem domain, establish an architectural framework, develop the project plan, and eliminate the highest risk elements of the project.
  - The architectural decisions have to be made with an understanding of the whole system: its scope, major functional and nonfunctional requirements.
  - In the elaboration phase, an executable architecture prototype is built in one or more iterations, depending on the scope, size, risk, and novelty of the project.
  - At the end of the elaboration phase, the second important project milestone will be the life cycle architecture milestone.
- Construction phase:
  - During the construction phase, all application features are developed, integrated, and thoroughly tested.
  - This phase also focuses on the user manuals and the current release details.
  - At the end of this phase, a beta release becomes operational for the customers.
- Transition phase:
  - The purpose of the transition phase is to move the software product to the user community for working.
  - This phase includes several iterations, including beta releases, general availability releases, as well as bug-fix and enhancement releases.
  - Effort is made in developing user-oriented documentation, training users, supporting users in their initial product use, and reacting to user feedback.
- Each phase in the RUP can be further broken down into *iterations*.

- Each iteration in the RUP mitigates risks, manage changes, provide reuse, and produces better quality products as compared to the traditional waterfall process.
- The RUP is suitable for small development teams as well as large development organizations.
- It can be found in a simple and clear process architecture that provides commonality across a family of processes.
- *Workflow* represents the sequence of activities that produces a result of the observable value.
- Workflows are divided into six core workflows (business modeling workflow, requirements workflow, analysis and design workflow, implementation workflow, test workflow, deployment workflow) and three supporting workflows (project management workflow, configuration and change management workflow, and environment workflow).
- The RUP is a complete methodology in itself that emphasizes documentation.
- It helps to proactively resolve project risks related with changing requirements.
- The RUP process is openly published, distributed, and supported for operation.
- It requires less time for integration of reusable components as the process of integration goes on throughout the software development life cycle.
- However, some expertise is required to develop software using this methodology.
- The development process is very complex and not well organized.

Conclusion:

- A software process is a coherent set of activities, procedures, policies, organizational structures, constraints, technologies, and artifacts that are needed to develop software products.
- A systematic software process is the prerequisite to the development of high quality software.
- A software process model is an abstract representation of a software process instantiated for each specific project.
- A generic process model involves the definition phase, development phase, and implementation phase that are coordinated and supported by umbrella activities.
- Software processes may be classified as product development process, project management process, change management process, process improvements, and quality management process.