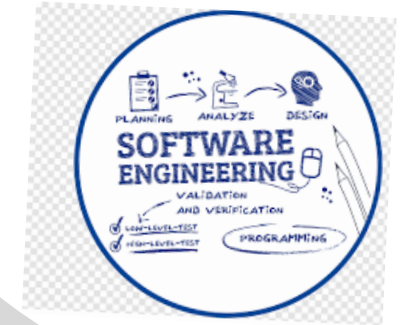




Dr. Vishwanath Karad
**MIT WORLD PEACE
UNIVERSITY** | PUNE
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

Unit-II



Introduction to Software Engineering, Software Process and Feasibility study



Prof. Shankar Mali

Computer Hardware



System Software

File Mngnt
Tools

Operating
System

Utilities

Assembler

Debugger

Compilers

Application Software

Word
Processing

Spreadsheets

Graphics

Games

Communications

Databases

Software Application Domains

- 1) **System software : operating systems, device drivers etc.**
- 2) **Application software : Google map, ERP**
- 3) **Engineering/Scientific software :MATLAB, AUTOCAD**
- 4) **Embedded software :keypad of microwave, washing machine**
- 5) **Web applications : e-commerce websites**
- 6) **Artificial intelligence software : Alexa**

Program

- A program is a set of instructions that help a computer or other machines to work/function automatically (completing a task).
- Programs are written in programming languages like C, C++, Java, etc. It is similar to a script that is compiled in a language that the computer can decode and perform.
- The program has to be developed and complied with every time a user wants to generate some output from it. Some programs are made up of many different files, and some are made up of one file.

What is software?

- Software is a collection of programs/data that are built to instruct a computer for its working.
- Software is the digital portion that runs on the hardware. It is said that software is directly related and responsible for managing a variety of independent hardware.
- The size of the software ranges from megabyte to gigabyte.
- The software takes more time to build, and its functionality depends on the operating system.



Engineering is the application of science and maths to solve problems. While scientists and inventors come up with innovations, it is engineers who apply these discoveries to the real world.

Legacy Software

- ✓ Some software's are old or even older. Ex. Software's delivered to government, Industry or individuals.
- ✓ These older programs are known as Legacy Software.
- ✓ Legacy software systems are old software systems ,developed decades ago and are continuously modified to meet changing computing platforms and business requirement
- ✓ Organizations can face problems like costly maintenance and risk to get evolved
- ✓ Most of the legacy software found supporting and useful for core business functions they are indispensable to the business

Reasons legacy software's should get evolved over the period of time

- ✓ **Need to be modified to cater the new requirements of user and technology or computing environments**
- ✓ **Software need to be enhanced to implement new or future business requirements**
- ✓ **To get join with the modern systems and database software should be extended**
- ✓ **The software should be re-architecture or re-build to make viable or workable within a network environment**

What was the Software Crisis?

- ✓ **It was in the late 1960s when many software projects failed.**
- ✓ **Many software became over budget. Output was an unreliable**
- ✓ **software which is expensive to maintain.**
- ✓ **Larger software was difficult and quite expensive to maintain.**
- ✓ **Lots of software not able to satisfy the growing requirements of the customer.**
- ✓ **Complexities of software projects increased whenever its hardware capability increased.**

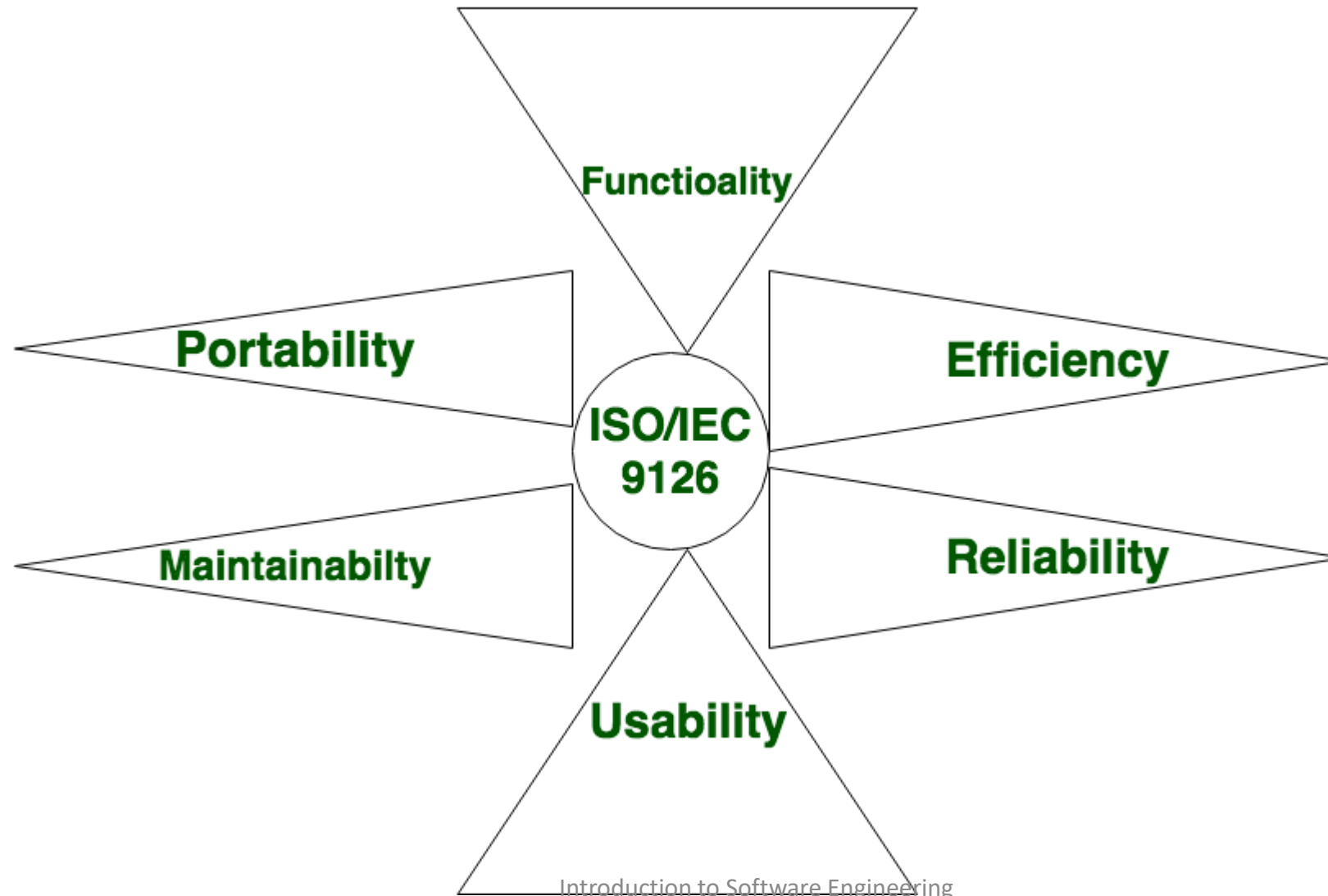
Why software Engineering

- ✓ **To manage Large software**
- ✓ **For more Scalability**
- ✓ **Cost Management**
- ✓ **To manage the dynamic nature of software**
- ✓ **For better quality Management**

Software Engineering

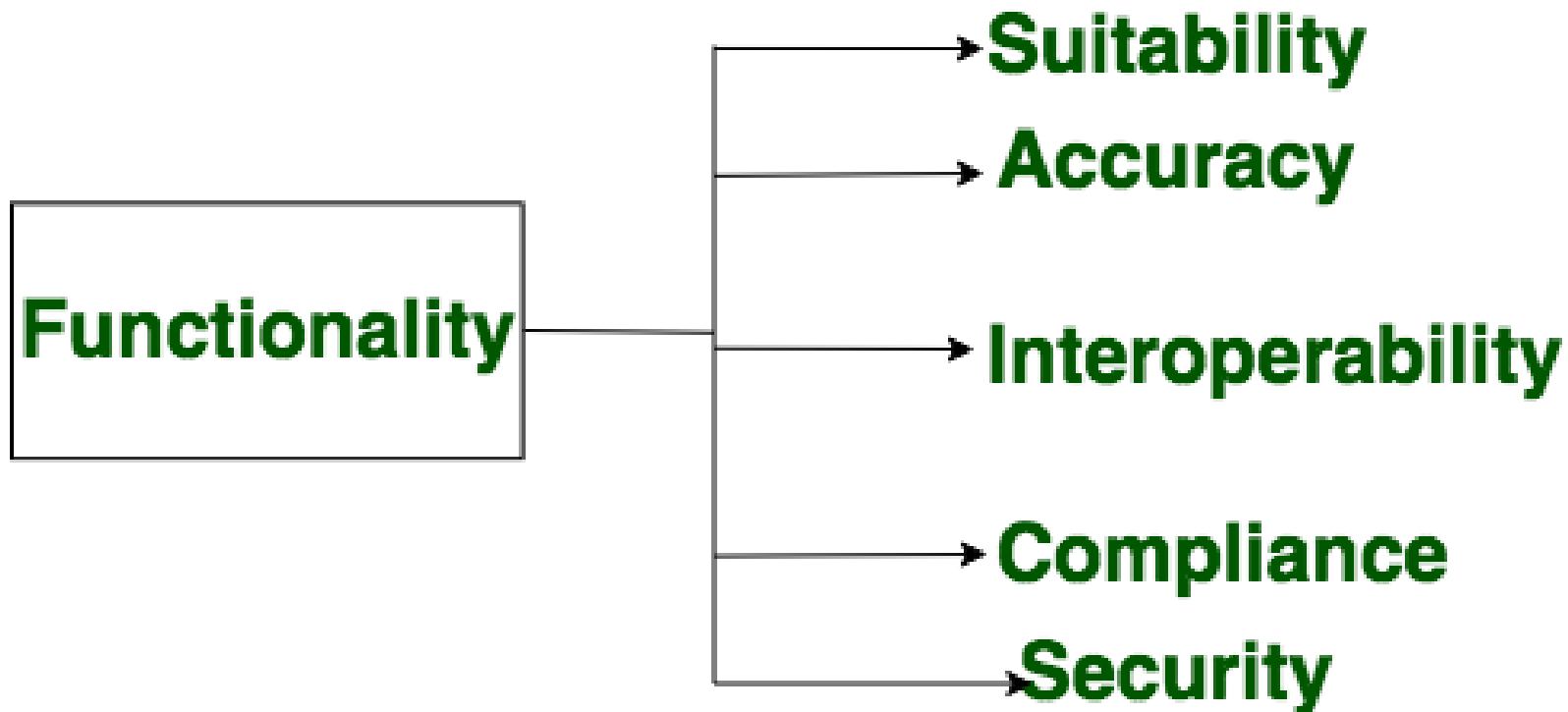
The application of a systematic ,disciplined, quantifiable approach to the development ,operation and maintenance of software, in order to obtain economically software that is reliable and works efficiently on real machines

Software Characteristics



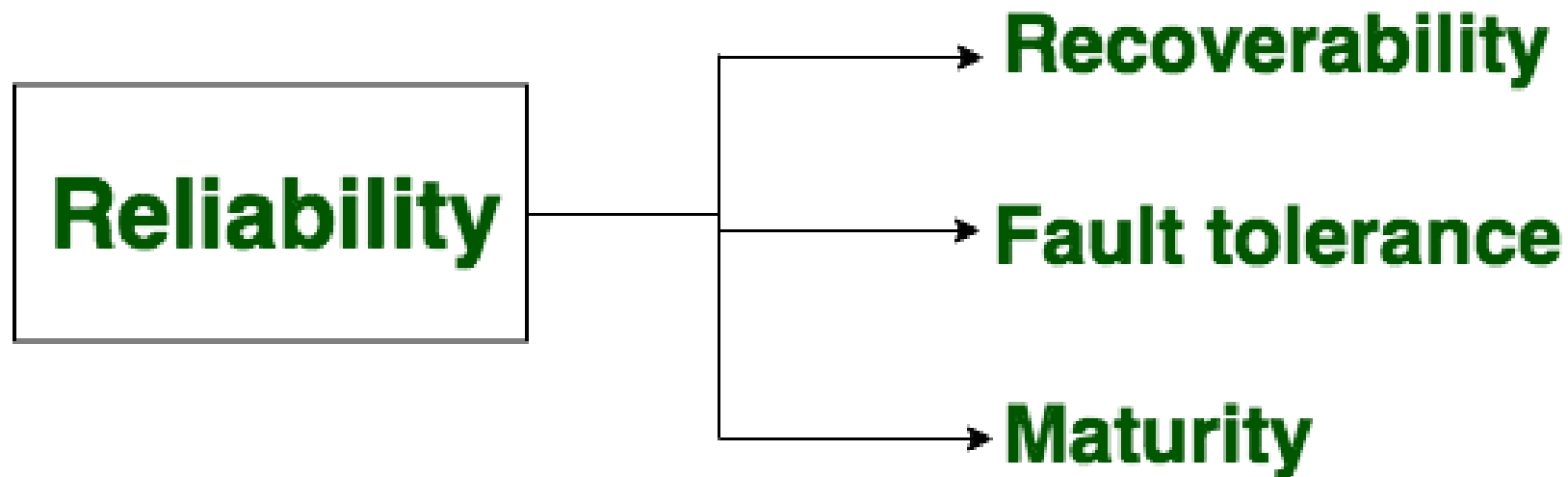
Functionality

- It refers to the degree of performance of the software against its intended purpose.
- Required functions are:



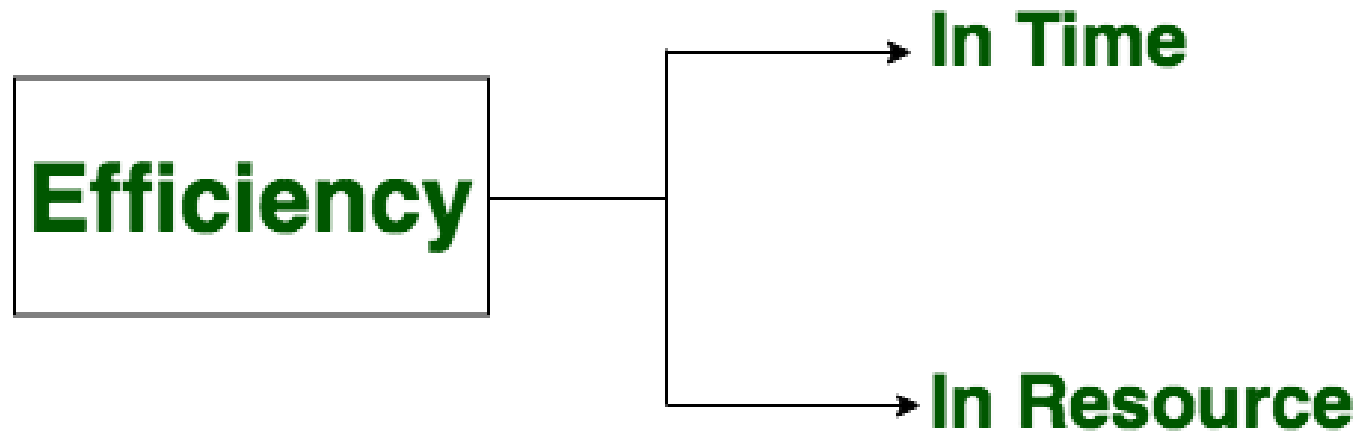
Reliability

- A set of attributes that bears on the capability of software to maintain its level of performance under the given condition for a stated period of time.
- Required functions are:



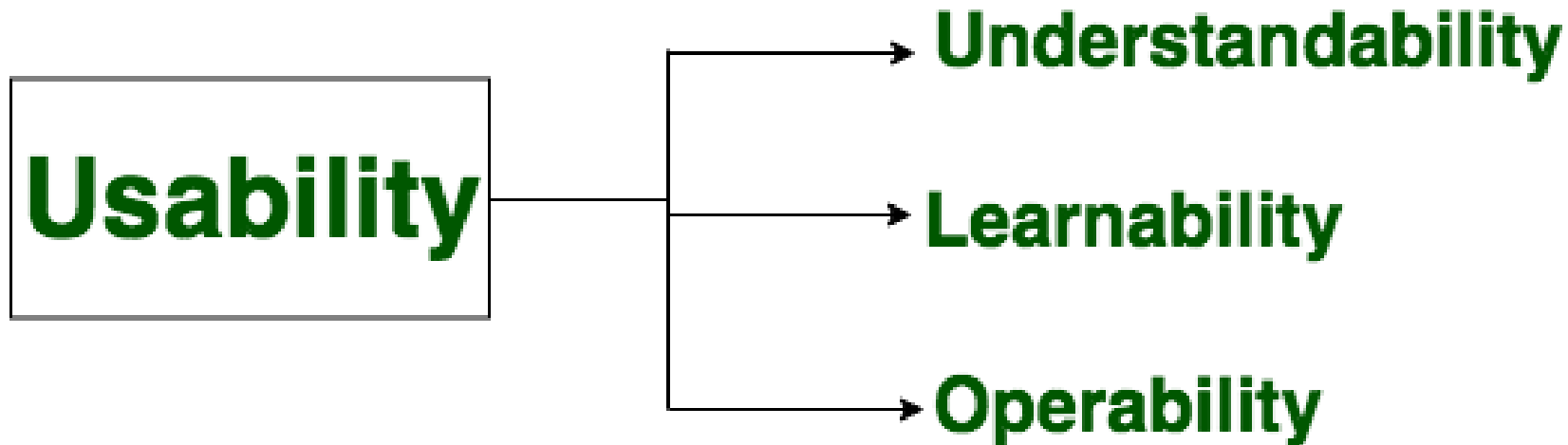
Efficiency

- It refers to the ability of the software to use system resources in the most effective and efficient manner. The software should make effective use of storage space and executive command as per desired timing requirements.
- Required functions are:



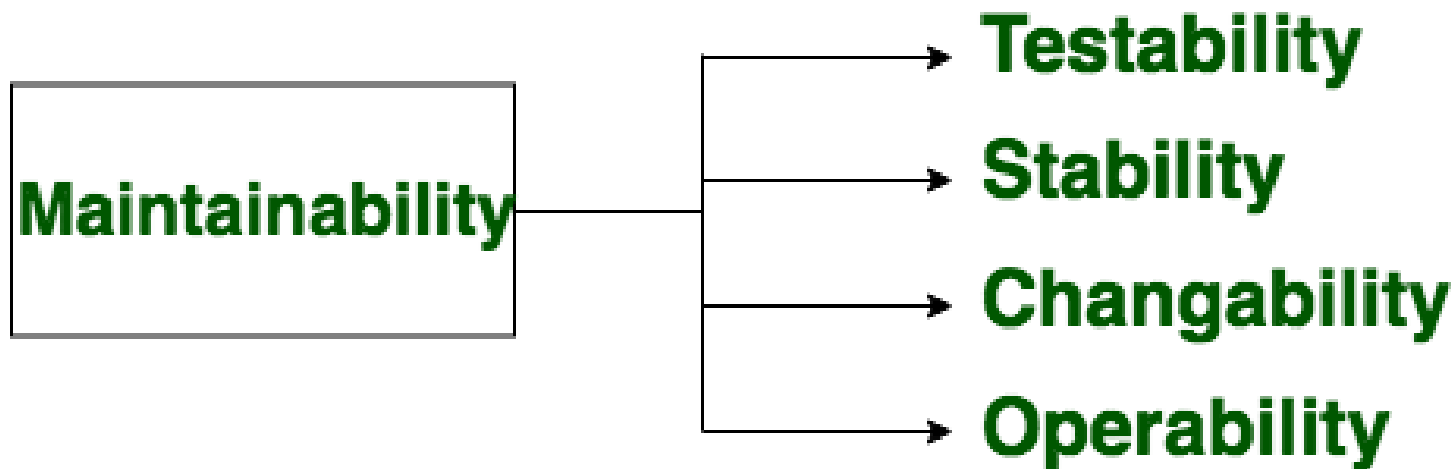
Usability

- It refers to the extent to which the software can be used with ease. the amount of effort or time required to learn how to use the software.
- Required functions are:



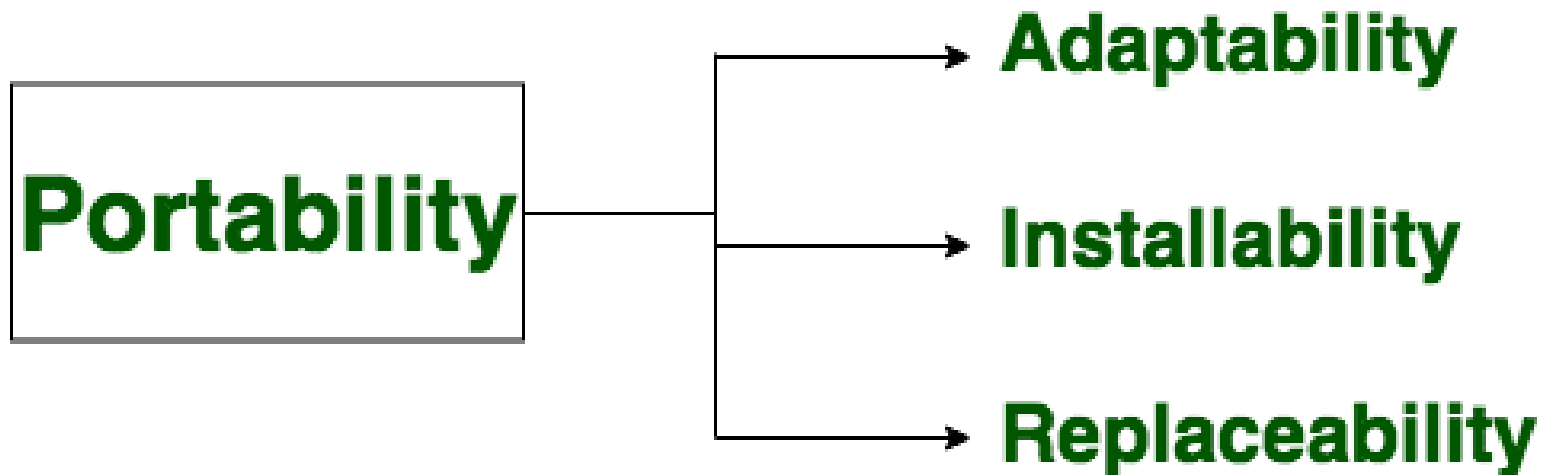
Maintainability

- It refers to the ease with which the modifications can be made in a software system to extend its functionality, improve its performance, or correct errors.
- Required functions are:



Portability

- A set of attributes that bears on the ability of software to be transferred from one environment to another, without or minimum changes.
- Required functions are:



Software is developed or engineered; it is not manufactured in the classical sense

Apart from above mention qualities of software, there are various characteristics of software in software engineering:

- Although some similarities exist between software development and hardware manufacturing, few activities are fundamentally different.
- In both activities, high quality is achieved through good design, but the manufacturing phase for hardware can introduce quality problems than software.

The software doesn't “wear out.”

- Hardware components suffer from the growing effects of many other environmental factors. Stated simply, the hardware begins to wear out.
- Software is not susceptible to the environmental maladies that cause hardware to wear out.
- When a hardware component wears out, it is replaced by a spare part.
- There are no software spare parts.
- Every software failure indicates an error in design or in the process through which design was translated into machine-executable code. Therefore, the software maintenance tasks that accommodate requests for change involve considerably more complexity than hardware maintenance. However, the implication is clear—the software doesn't wear out. But it does deteriorate.

The software continues to be custom-built

- A software part should be planned and carried out with the goal that it tends to be reused in various projects.
- Current reusable segments encapsulate the two information and the preparation that is applied to the information, empowering the programmer to make new applications from reusable parts.
- In the hardware world, component reuse is a natural part of the engineering process.

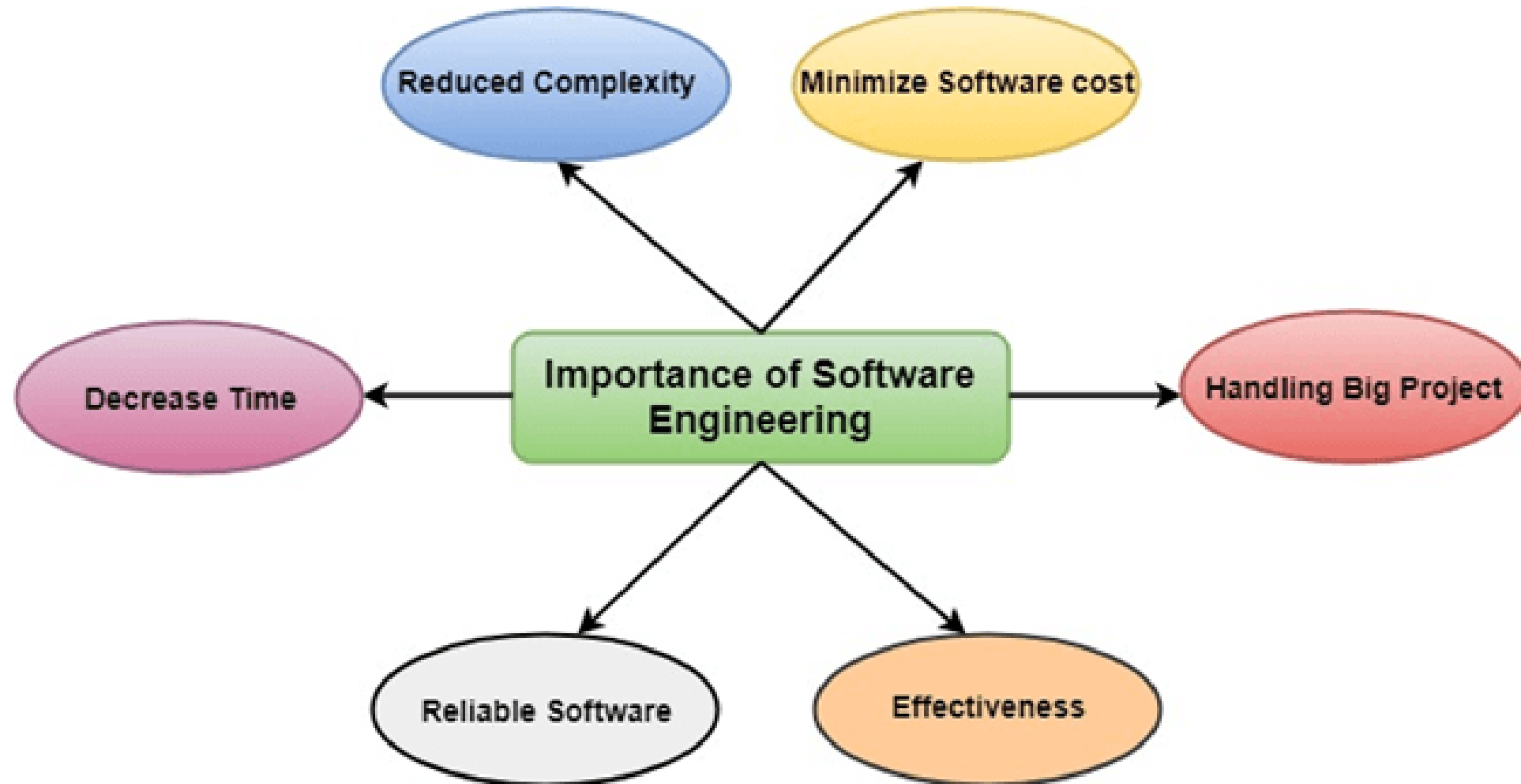
Definitions of software engineering

- Software engineering is defined as a process of analyzing user requirements and then designing, building, and testing software application which will satisfy those requirements.
- Software Engineering: (1) The application of a **systematic, disciplined, quantifiable approach** to the **development, operation, and maintenance** of software;
- Boehm defines software engineering, which involves, ‘the practical application of scientific knowledge to the creative design and building of computer programs. It also includes associated documentation needed for developing, operating, and maintaining them.’

What was the Software Crisis?

- It was in the late 1960s when many software projects failed.
- Many software became over budget. Output was an unreliable software which is expensive to maintain.
- Larger software was difficult and quite expensive to maintain.
- Lots of software not able to satisfy the growing requirements of the customer.
- Complexities of software projects increased whenever its hardware capability increased.
- Demand for new software increased faster compared with the ability to generate new software.

Importance of Software Engineering



Importance of Software Engineering

- **Reduces complexity:** Big software is always complicated and challenging to progress. Software engineering has a great solution to reduce the complication of any project. Software engineering divides big problems into various small issues. And then start solving each small issue one by one. All these small problems are solved independently to each other.
- **To minimize software cost:** Software needs a lot of hardwork and software engineers are highly paid experts. A lot of manpower is required to develop software with a large number of codes. But in software engineering, programmers project everything and decrease all those things that are not needed. In turn, the cost for software productions becomes less as compared to any software that does not use software engineering method.

Importance of Software Engineering

- **To decrease time:** Anything that is not made according to the project always wastes time. And if you are making great software, then you may need to run many codes to get the definitive running code. This is a very time-consuming procedure, and if it is not well handled, then this can take a lot of time. So if you are making your software according to the software engineering method, then it will decrease a lot of time.
- **Handling big projects:** Big projects are not done in a couple of days, and they need lots of patience, planning, and management. And to invest six and seven months of any company, it requires heaps of planning, direction, testing, and maintenance. No one can say that he has given four months of a company to the task, and the project is still in its first stage. Because the company has provided many resources to the plan and it should be completed. So to handle a big project without any problem, the company has to go for a software engineering method.

Importance of Software Engineering

- **Reliable software:** Software should be secure, means if you have delivered the software, then it should work for at least its given time or subscription. And if any bugs come in the software, the company is responsible for solving all these bugs. Because in software engineering, testing and maintenance are given, so there is no worry of its reliability.
- **Effectiveness:** Effectiveness comes if anything has made according to the standards. Software standards are the big target of companies to make it more effective. So Software becomes more effective in the act with the help of software engineering.

Software Development Team Roles and Responsibilities

- PROJECT MANAGER
- PRODUCT OWNER
- TECH LEAD
- FULL-STACK DEVELOPER
- FRONT-END DEVELOPER
- BACK-END DEVELOPER
- QA LEAD
- QA ENGINEER
- TESTER
- UI DESIGNER
- UX DESIGNER

Software Development Team Roles and Responsibilities

- **PROJECT MANAGER**

Works on a higher level of abstraction and is responsible for a budget, risk, schedule and contract management. It's highly possible that project managers don't well-know the product which they are creating. They use different methodologies and are focused mainly on controlling the project.

A PM will ensure that there will be no changes incompatible with specifications.

- **PRODUCT OWNER**

Such a person is close to that of an analyst and concentrates on a product and its features. Their primary role is to take care of business value maximisation (suitability of the product to use). Product owners are focused on constant changes because they are using agile methodology.

Software Development Team Roles and Responsibilities

- **TEAM LEAD**

Typically, such a role is given to one of the developers, not necessarily the best or most experienced. This person should have leadership qualities which allow for maintaining communication between remote development teams and for example – clients.

Team leads ensure that the team has sufficient performance levels and is also responsible for conflict prevention and resolution.

- **TECH LEAD**

Some time ago we would use the term ‘architect’ instead of tech lead. Nowadays, ‘analyst’ fits better. But who is this? It is, in fact, the ‘lead example’ when it comes to technical knowledge of team members.

Tech leads can solve problems not linked with the development process itself, i.e. issues connected with the integration with external providers, or those regarding hardware.

Software Development Team Roles and Responsibilities

- **FULL-STACK DEVELOPER**

It's a programmer not specialised in one specific area. Because of this, a full stack developer may appear less-advanced than a front-end or back-end developer. However, thanks to a broad range of knowledge and skills, they can implement a solution on every architectural layer of the system (view, business logic, database).

In less complex, demanding projects where costs play a huge role, a full stack developer is indispensable.

- **FRONT-END DEVELOPER**

Front-end developers are the point of contact between the end client (user) and the delivered business solution (a system undergoing implementation). Such a person must, on the one hand, provide the best possible application reception by the user (interface responsiveness, content clarity). On the other hand, they're responsible for proper communication with the business logic layer.

Software Development Team Roles and Responsibilities

- **BACK-END DEVELOPER**

It's a programmer focused mainly on developing business logic and data layers. And because elements are crucial to system functioning; its correctness depends on the quality of back-end developer's work.

What's more, in a case of larger, more complex systems, such developer's capabilities of building data queries (i.e. SQL) and its optimisation are of enormous value.

- **QA LEAD**

Quality Assurance Lead is responsible for building the QA team and its management. One of their most crucial responsibilities is to ensure a proper relationship level between persons accountable for areas of the system under construction. In the end, it's a QA manager who is primarily responsible for end users' reactions to the brand-new system.

QA Lead, similarly, to a Team Lead, is a person accountable for proper relations between QA team members, mitigation of conflicts, praising good work, and motivating to further self-development.

Software Development Team Roles and Responsibilities

- **QA ENGINEER**

It's a person responsible for preparing tools that allow for automating processes which verify software quality. Such a form of software testing makes it possible to check if regression errors occur. In other words, to check if work on new features didn't cause errors in already existing and functioning system areas.

It's a huge benefit as it lowers staff costs (duplicative testing performed by testers) and shortens stability verification time.

- **TESTER**

Testers fulfil an otherwise necessary role in the QA team. They are responsible mainly for conducting manual tests, that is system verification by using it analogically, as opposed to how end users will operate it. Apart from a most-likely standard approach to using particular features, a tester must propose alternative paths, boundary conditions, and exceptions which will help to eliminate as many existing errors as possible.

What is Feasibility Study?

- As the name implies, a feasibility analysis is used to determine the viability of an idea, such as ensuring a project is legally and technically feasible as well as economically justifiable. It tells us whether a project is worth the investment—in some cases, a project may not be doable. There can be many reasons for this, including requiring too many resources, which not only prevents those resources from performing other tasks but also may cost more than an organization would earn back by taking on a project that isn't profitable.
- A well-designed study should offer a historical background of the business or project, such as a description of the product or service, accounting statements, details of operations and management, marketing research and policies, financial data, legal requirements, and tax obligations. Generally, such studies precede technical development and project implementation.

Types of Feasibility Study

- A feasibility analysis evaluates the project's potential for success; therefore, perceived objectivity is an essential factor in the credibility of the study for potential investors and lending institutions. There are five types of feasibility study—separate areas that a feasibility study examines, described below.
 1. Technical Feasibility
 2. Economic Feasibility
 3. Legal Feasibility
 4. Operational Feasibility
 5. Scheduling Feasibility

1. Technical Feasibility

- This assessment focuses on the technical resources available to the organization. It helps organizations determine whether the technical resources meet capacity and whether the technical team is capable of converting the ideas into working systems. Technical feasibility also involves the evaluation of the hardware, software, and other technical requirements of the proposed system.

2. Economic Feasibility

- This assessment typically involves a cost/ benefits analysis of the project, helping organizations determine the viability, cost, and benefits associated with a project before financial resources are allocated. It also serves as an independent project assessment and enhances project credibility—helping decision-makers determine the positive economic benefits to the organization that the proposed project will provide.

3. Legal Feasibility

- This assessment investigates whether any aspect of the proposed project conflicts with legal requirements like zoning laws, data protection acts or social media laws. Let's say an organization wants to construct a new office building in a specific location. A feasibility study might reveal the organization's ideal location isn't zoned for that type of business. That organization has just saved considerable time and effort by learning that their project was not feasible right from the beginning.

4. Operational Feasibility

This assessment involves undertaking a study to analyze and determine whether—and how well—the organization's needs can be met by completing the project. Operational feasibility studies also examine how a project plan satisfies the requirements identified in the requirements analysis phase of system development.

5. Scheduling Feasibility

- This assessment is the most important for project success; after all, a project will fail if not completed on time. In scheduling feasibility, an organization estimates how much time the project will take to complete.

Software Myths

- Affect managers, customers (and other non-technical stakeholders) and practitioners
- Are believable because they often have elements of truth, but ...
- Invariably lead to bad decisions, therefore ...
- Insist on reality as you navigate your way through software engineering

Management Myths

- **Myth:** The members of an organization can acquire all-the information, they require from a manual, which contains standards, procedures, and principles.
- **Reality:**
 1. Standards are often incomplete, inadaptable, and outdated.
 2. Developers are often **unaware** of all the established standards.
 3. Developers rarely follow all the known standards because not all the standards tend to decrease the delivery time of software while maintaining its quality.
- **Myth:** If the project is behind schedule, increasing the number of programmers can reduce the time gap.
- **Reality:**
 1. Adding more manpower to project, that is behind schedule, further delays the project.
 2. New workers take longer to learn about project compared to those already working on the project.

User Myths

- **Myth:** If the project is outsourced to a third party, the management can relax and let the other firm develop software for them.
- **Reality:** Outsourcing software to a third party does not help the organization, which is incompetent in managing and controlling the software project internally. The organization invariably suffers when it out sources the software project.
- **Myth:** Brief requirement stated in the initial process is enough to start development; detailed requirements can be added at the later stages.
- **Reality:**
 - 1.Starting development with incomplete and ambiguous requirements often lead to software failure. Instead, a complete and formal description of requirements is essential before starting development.
 - 2.Adding requirements at a later stage often requires repeating the entire development process.

User Myths

- **Myth:** Software is flexible; hence software requirement changes can be added during any phase of the development process.
- **Reality:** Incorporating change requests earlier in the development process costs lesser than those that occurs at later stages. This is because incorporating changes later may require **redesigning** and **extra resources**.

Developer Myths

- **Myth:** Software development is considered complete when the code is delivered.
- **Reality:** 50% to 70% of all the efforts are exhausted after the software is delivered to the user.
- **Myth:** The success of a software project depends on the quality of the product produced.
- **Reality:** The quality of programs is not the only factor that makes the project successful instead the documentation and software configuration also play a crucial role.
- **Myth:** Software engineering requires unnecessary documentation, which slows down the project.
- **Reality:** Software engineering is about creating quality at every level of the software project. Proper documentation enhances quality which results in reducing the amount of rework.

Developer Myths

- **Myth:** The only product that is delivered after the completion of a project is the working program(s).
- **Reality:** The deliverables of a successful project includes not only the working program but also the documentation to guide the users for using the software.
- **Myth:** Software quality can be assessed only after the program is executed.
- **Reality:** The quality of software can be measured during any phase of development process by applying some quality assurance mechanism. One such mechanism is formal technical review that can be effectively used during each phase of development to uncover certain errors.

The Software Process

- A process is a collection of activities, actions and tasks that are performed when some work product is to be created.
- Framework is a Standard way to build and deploy applications. Software Process Framework is the foundation of complete software engineering process. Software process framework includes set of all umbrella activities. It also includes number of framework activities that are applicable to all software projects.

Definition of Software Process

- A **framework** for the activities, actions, and tasks that are required to build high-quality software.
- Software Process defines the approach that is taken as software is engineered.

Software Process Framework

A generic process framework encompasses five activities which are given below one by one:

Communication:

In this activity, heavy communication with customers and other stakeholders, as well as requirement gathering is done.

Planning:

In this activity, we discuss the technical related tasks, work schedule, risks, required resources, etc.

Modeling:

Modeling is about building representations of things in the ‘real world’. In modeling activity, a product’s model is created in order to better understand the requirements.

Software Process Framework

Construction:

In software engineering, construction is the application of set of procedures that are needed to assemble the product. In this activity, we generate the code and test the product in order to make better product.

Deployment:

In this activity, a complete or non-complete product or software is represented to the customers to evaluate and give feedback. On the basis of their feedback, we modify the product for supply of better product.

A Generic Process Model

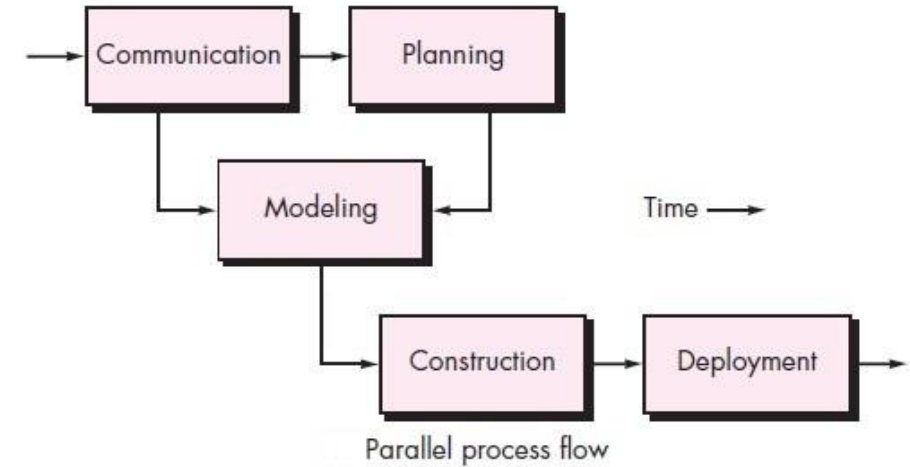
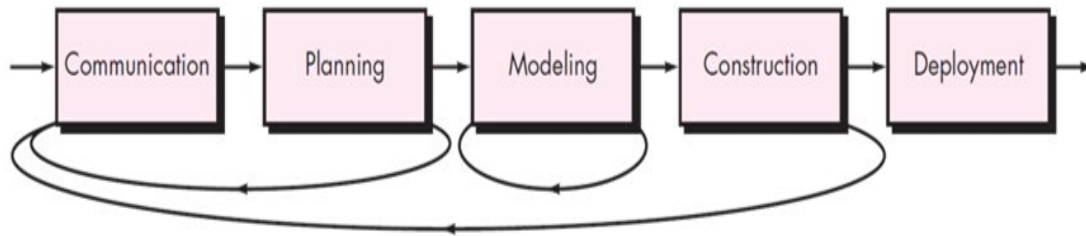
- A generic process framework for software engineering defines **five framework activities-communication, planning, modeling, construction, and deployment.**
- In addition, a **set of umbrella activities-** project tracking and control, risk management, quality assurance, configuration management, technical reviews, and others are applied throughout the process.
- Next question is: how the framework activities and the actions and tasks that occur within each activity are organized **with respect to sequence and time?**

Process Flow

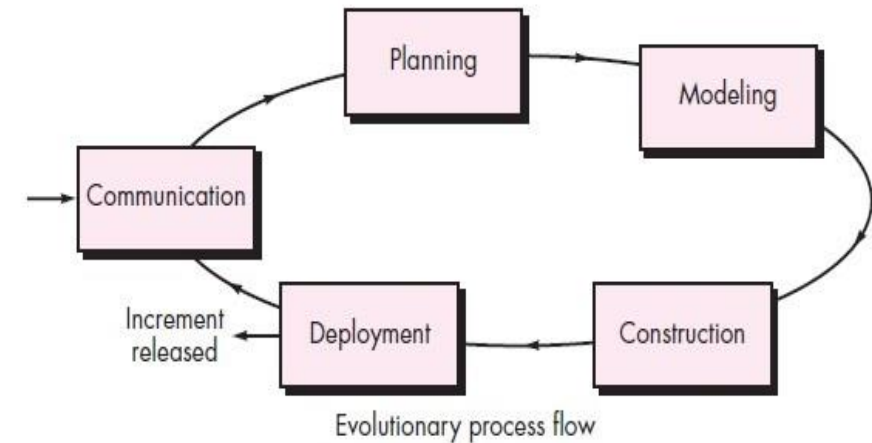
- Linear process flow



- Iterative process flow



Parallel process flow



Evolutionary process flow

Process Flow

- Linear process flow executes each of the five activities in sequence.
- An iterative process flow repeats one or more of the activities before proceeding to the next.
- An evolutionary process flow executes the activities in a circular manner. Each circuit leads to a more complete version of the software.
- A parallel process flow executes one or more activities in parallel with other activities (modeling for one aspect of the software in parallel with construction of another aspect of the software).

Adapting a Process Model

Depends on :

- overall flow of **activities, actions, and tasks** and the interdependencies among them
- the degree to which actions and tasks are defined within each framework activity
- the degree to which work products are identified and required
- the manner which quality assurance activities are applied
- the manner in which project tracking and control activities are applied
- the overall degree of detail and rigor with which the process is described
- the degree to which customer and other stakeholders are involved with the project
- the level of autonomy given to the software team
- the degree to which team organization and roles are prescribed

Prescriptive Models & Specialized Process Models

WATERFALL MODEL

INCREMENTAL PROCESS MODEL

- The Incremental Model
- The RAD Model

EVOLUTIONARY PROCESS MODELS

- Prototyping.
- The Spiral model
- The Concurrent Development Model

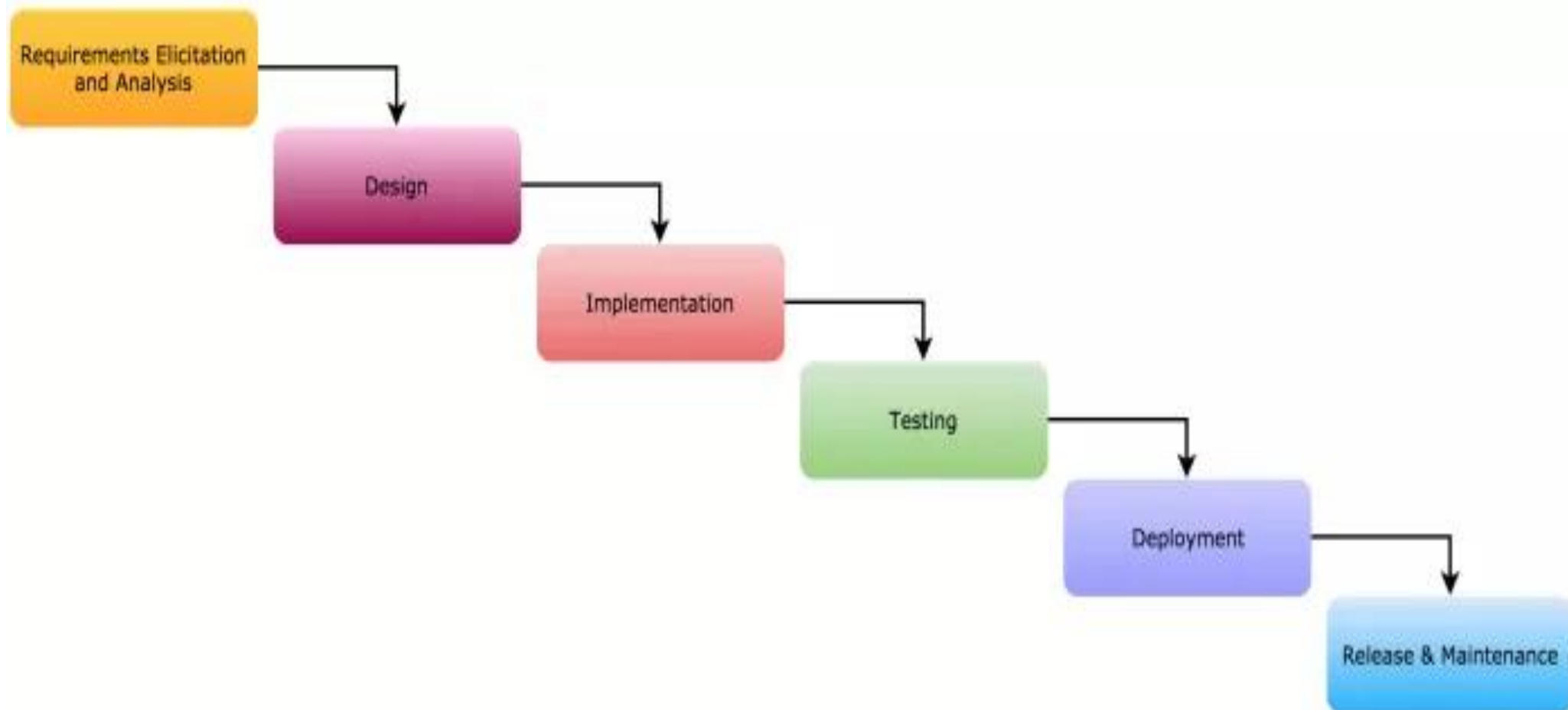
SPECIALIZED PROCESS MODELS.

- Component-based Development
- The Formal Methods Model
- Aspect-Oriented Software Development

THE UNIFIED PROCESS.

AGILE PROCESS

The Waterfall Model



The Waterfall Model

- 1) **Requirement Gathering and analysis:** Here requirements of the system or project that is to be developed are captured from client and all these requirements are documented in a requirement specification document to prepare a SRS. This SRS is verified by the client and after acceptance next phase begins.
- 2) **System Design:** Here a design of the system is prepared. Plan system hardware and software requirements. It helps in defining the overall system architecture.
- 3) **Implementation:** Once the design of a system is prepared, the system is broken into small programs called units, which are integrated in the next phase. In this phase the software is coded and simple unit testing is performed.

The Waterfall Model

4)**Testing:** After testing all the units are integrated into a system and system testing is performed. This is used to find any bugs or failure in the system or to verify that system is built as per the requirements of client.

5)**Deployment:** Once the testing phase is done, the product or software is deployed or installed in the customer environment.

6)**Maintenance:** Maintenance is required to fix the issues and to release improved versions to enhance the product.

The Waterfall Model

Advantages:

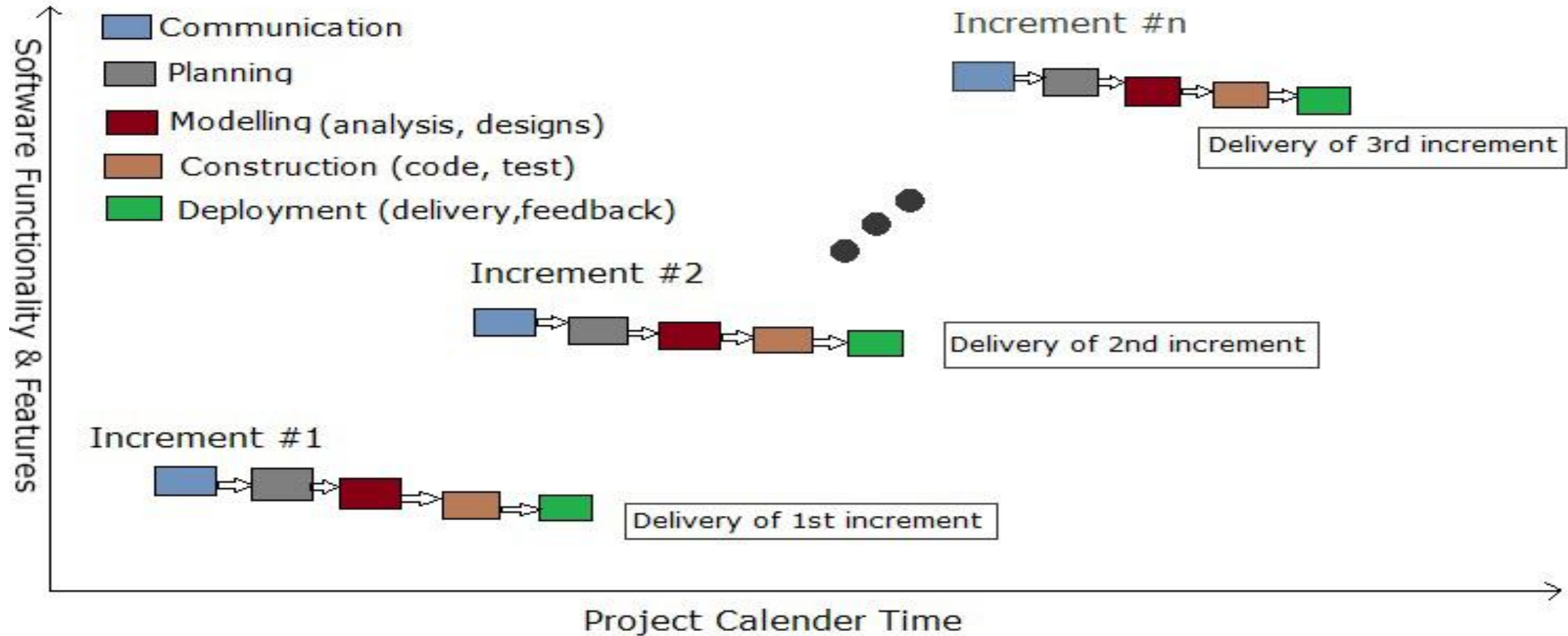
- Model is simple, easy to understand.
- Waterfall model is useful for smaller projects and it gives an appropriate result.
- System Requirements are documented and understood by projects team members
- Each phase works independently and do not overlap the other phases.
- Customer interaction is only at in the beginning and at the last phase of the project.

The Waterfall Model

Problems/Disadvantages:

- This model is not desirable for complex and bigger project where requirement changes frequently and risk factor is higher.
- This model also not good for ongoing projects.
- Customer interaction is less and it cannot adopt the changes in system requirements.
- This model requires more documentation which is not suitable for big projects.
- Customer feedback only at the end of the product.
- Small change makes lot of problems in the development

The Incremental Model



Incremental model

- The incremental model applies the waterfall model incrementally.
- The series of releases is referred to as “**increments**”, each increment providing more functionality to the customers.
- The product is defined as finished only when it satisfies all of its requirements.
- It involves both development and maintenance.
- After **the first increment**, a **core product** is delivered, which can already be used by the customer. Based on customer feedback, a plan is developed for the next increments, and modifications are made accordingly.
- The incremental philosophy is also used in the **agile process model**.

Incremental model: Characteristics

Characteristics of Incremental Model

1. System is broken down into many **mini development projects**.
2. Partial systems are built to produce the final system.
3. First tackled **highest priority requirements**.
4. The requirement of a portion is frozen **once the incremented portion is developed**.

Tasks involved:

Communication: helps to understand the objective.

Planning: required as many people (software teams) work on the same project but different function at same time.

Modeling: involves business modeling, data modeling, and process modeling.

Construction: this involves the reuse software components and automatic code.

Deployment: integration of all the increments

Advantages & Disadvantages

Advantages of Incremental model:

- i) Generates working software quickly and early during the software life cycle.
- ii) This model is more flexible – less costly to change scope and requirements.
- iii) It is easier to test and debug during a smaller iteration.
- iv) In this model customer can respond to each built.
- v) Lowers initial delivery cost.

Disadvantages of Incremental model:

- i) Needs good planning and design.
- ii) Needs a clear and complete definition of the whole system before it can be broken down and built incrementally.
- iii) Total cost is higher than other traditional model

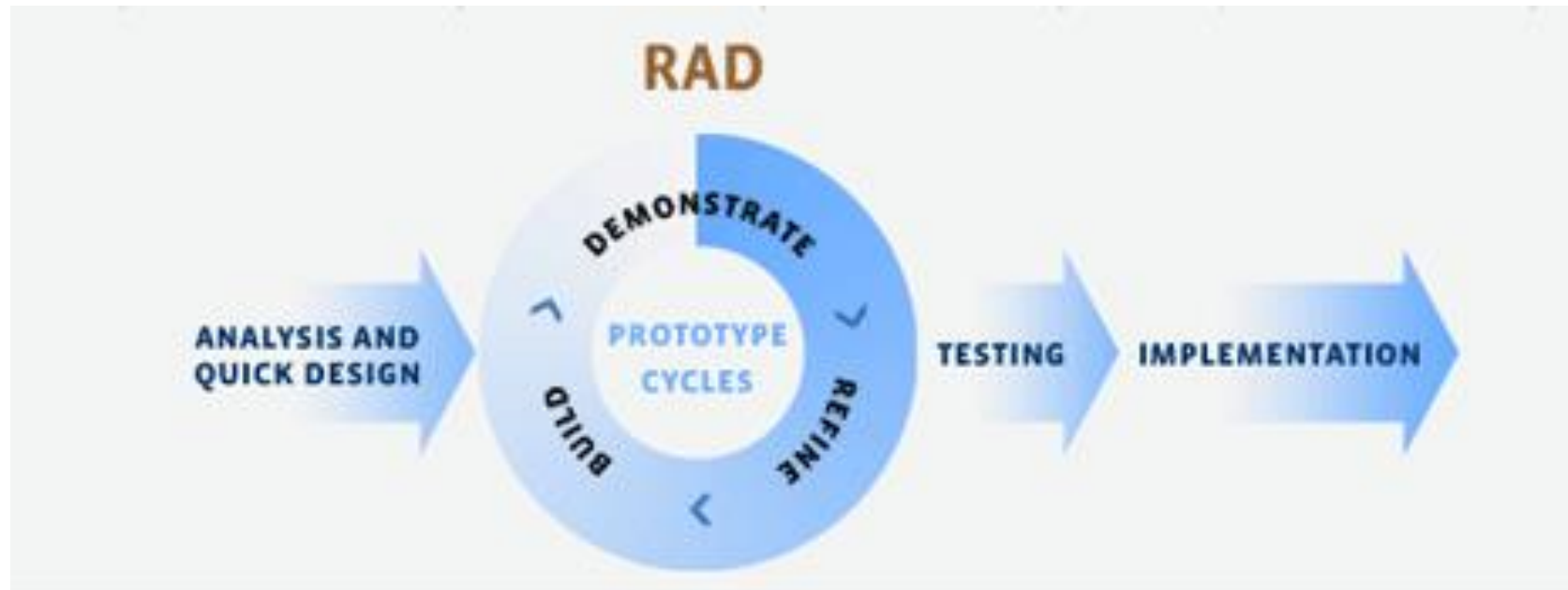
When to use Incremental Model ?

- When the requirements of the complete system are clearly defined and understood.
- Major requirements are defined; however, some details can evolve with time.
- There is a need to get a product to the market early.
- A new technology is being used
- Resources with needed skill set are not available
- There are some high risk features and goals.

RAD Model

- RAD is a **Rapid Application Development model**.
- Using the RAD model, software product is developed in a **short period** of time.
- The initial activity starts with the **communication** between customer and developer.
- **Planning** depends upon the initial requirements and then the requirements are divided into groups.
- Planning is more important to work together on different modules
- Rapid application development emphasizes working software and user feedback over strict planning and requirements recording.

RAD Model



RAD Model

1. Figure out the requirements : Identify why the software or app is built and what the project is supposed to accomplish.

Involve users, developers, and designers to discuss the purpose of the system and a estimated project timeline. The budget is a strong constraint.

2. Build prototypes: Team will start working on building functional models right away. The engineers and designers will create and improve upon working prototypes

3. Get user feedback :

RAD calls for ongoing collaboration between your team and users in order to create a high-quality system. The users will be the ones providing feedback to improve your prototypes.

A general model of the design process

4. Do it again : Repeat steps two and three until you feel like your project is done or until you have all working parts assembled together to meet a client's requirements.

5. Test, test, test :

Run the system through different scenarios and make sure it accomplishes the system's goal.

RAD Model

Advantages of RAD Model

- The process of application development and delivery are **fast**.
- This model is **flexible**, if any **changes** are required.
- Reviews are taken from the clients at the starting of the development hence there are lesser chances to miss the requirements.
- Always having something to show your client means you can get their **feedback** and quickly implement any changes

Disadvantages of RAD Model

- The **feedback** from the user is required at **every development phase**.
- This model is not a good choice for **long term and large projects**.

When is RAD useful?

- When a **quick delivery** of a product is needed for a customer.
- When there are **going to be changes** made to the prototype throughout the process before the final product is completed.
- When there are plenty of **knowledgeable developers and engineers** on hand and the customer must also remain committed to the process and the schedule.
- When either of these two components is not available, the RAD formula can fail.

Evolutionary Process Models

- This model divides the development cycle **so that users are able to get access** to the product at the end of each cycle.
- Evolutionary models are **iterative type models**.
- They allow to develop more complete versions of the software.
- The users provide **feedback on the product** for planning stage of the next cycle and the development team responds, often by changing the product, plans or process.

Following are the evolutionary process models.

1. The prototyping model
2. The spiral model
3. Concurrent development model

Evolutionary Models: Prototyping

- Prototype is defined as first or preliminary form using which other forms are copied or derived.
- Prototype model is a set of **general objectives** for **software**.
- It does **not identify** the requirements like detailed input, output.
- It is software working model of limited functionality.
- Working programs are quickly produced.

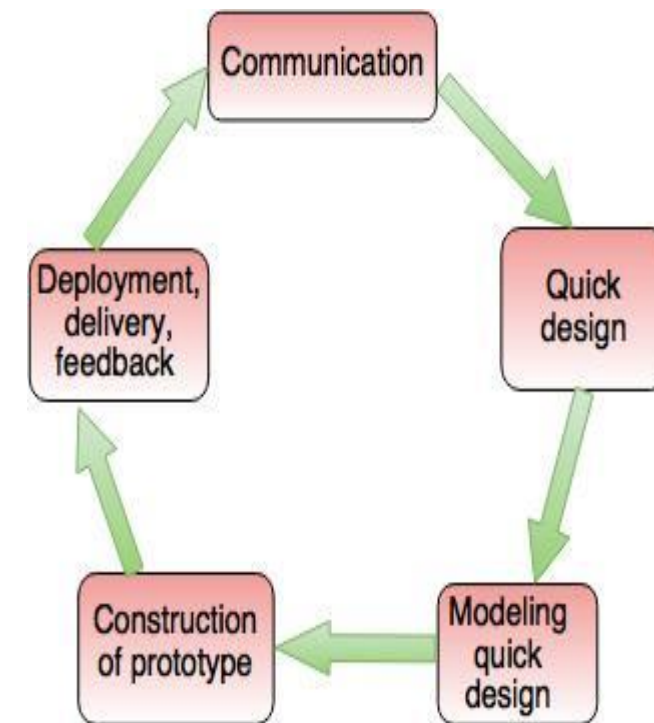


Fig. - The Prototyping Model

Prototyping Model

1. Communication: Developer and customer meet and discuss the overall objectives

2. Quick design

- Quick design is implemented when requirements are known.
- It includes only the important aspects like input and output format of the software and focusses on those aspects which are visible to the user rather than the detailed plan.
- It helps to construct a prototype.

3. Modeling quick design

- This phase gives the clear idea about the development of software
- It allows the developer to better understand the exact requirements.

4. Construction of prototype

- The prototype is evaluated by the customer itself.

5. Deployment, delivery, feedback

- If the user is not satisfied then it refines according to the requirements of the user.
- The process of refining the prototype is repeated until all the requirements of users are met.
- When the users are satisfied with the developed prototype then the system is developed on the basis of final prototype.

Prototyping Model

Advantages of Prototyping Model

- Prototype model need not know the detailed input, output, processes, adaptability of operating system and full machine interaction.
- In the development process of this model **users are actively involved**.
- The development process is the best platform to understand the system by the user.
- Errors are detected much earlier.
- Gives quick user feedback for better solutions.
- It identifies the missing functionality easily and identifies confusing or difficult functions.

Disadvantages of Prototyping Model:

- The client involvement is more and it is not always considered by the developer.
- It is a slow process because it takes more time for development.
- Many changes can disturb the rhythm of the development team.
- It is a thrown away prototype when the users are confused with it.

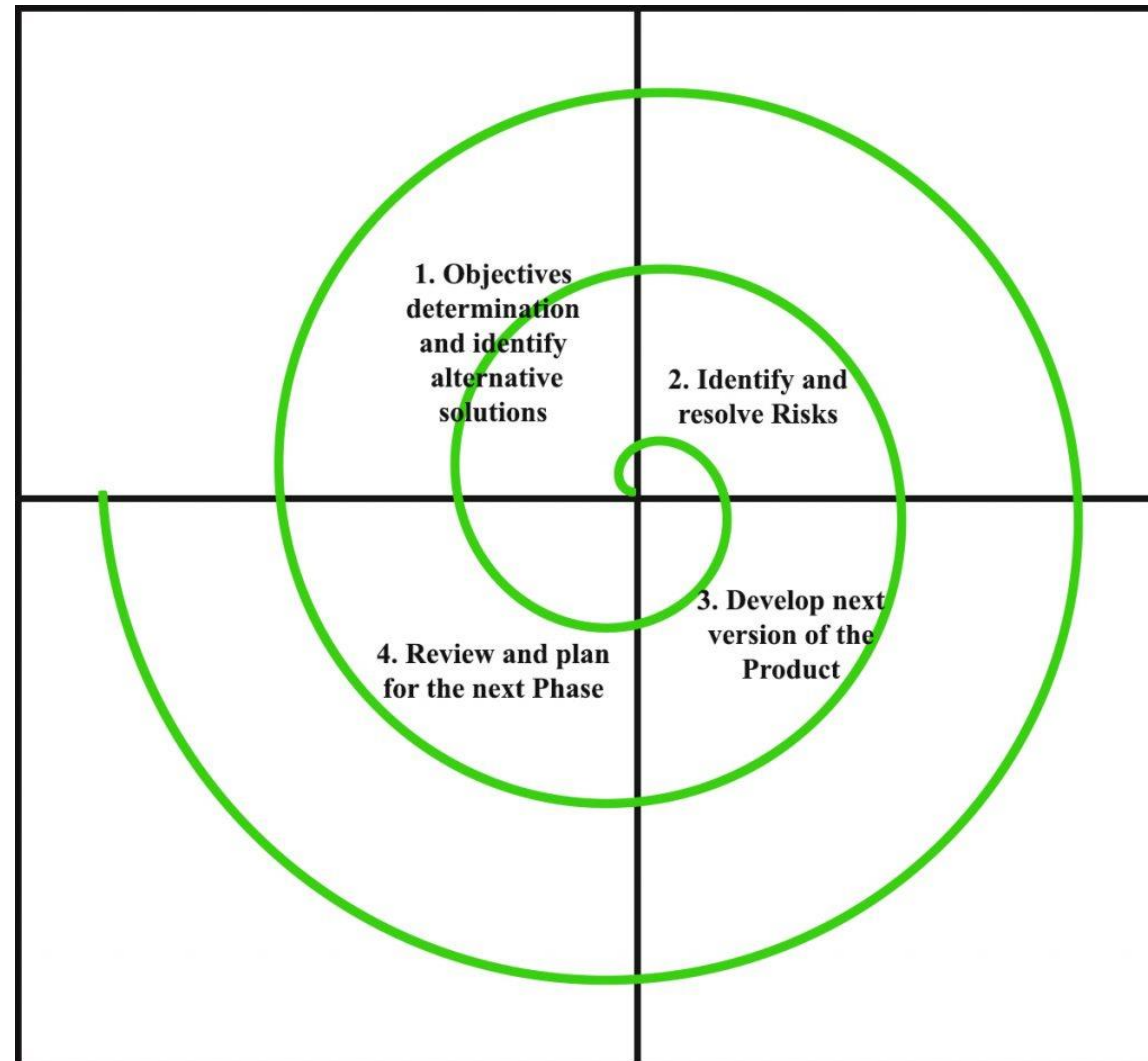
Prototype vs Incremental Model

- **Prototype Model:** Instead of freezing the requirements before a design or coding can proceed, a throwaway prototype is built to understand the requirements.
- **Incremental model :** the whole requirement is divided into various builds. Multiple development cycles take place here, making the life cycle a “multi-waterfall” cycle. Cycles are divided up into smaller, more easily managed modules.

Spiral Model

- Spiral model is a risk driven process model -which means that the overall success of a project highly depends on the risks analysis phase.
- Requires skills and expertise.
- It is used for generating the large software projects.
- In spiral model, an alternate solution is provided if the risk is found in the risk analysis, then **alternate solutions** are suggested and implemented.
- It is a combination of prototype and sequential model or waterfall model.
- In one iteration all activities are done, for large project's the output is small.
- Turns out costlier

Spiral Model



Stages in Spiral Model

Each phase of the Spiral Model is divided into four quadrants as shown in the above figure. The functions of these four quadrants are discussed below-

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.

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 are identified and the risks are resolved using the best possible strategy. At the end of this quadrant, the Prototype is built for the best possible solution.

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.

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.

Spiral model

- **Advantages of Spiral Model**
- It reduces high amount of risk.
- It is good for large and critical projects.
- It gives strong approval and documentation control.
- In spiral model, the software is produced early in the life cycle process.

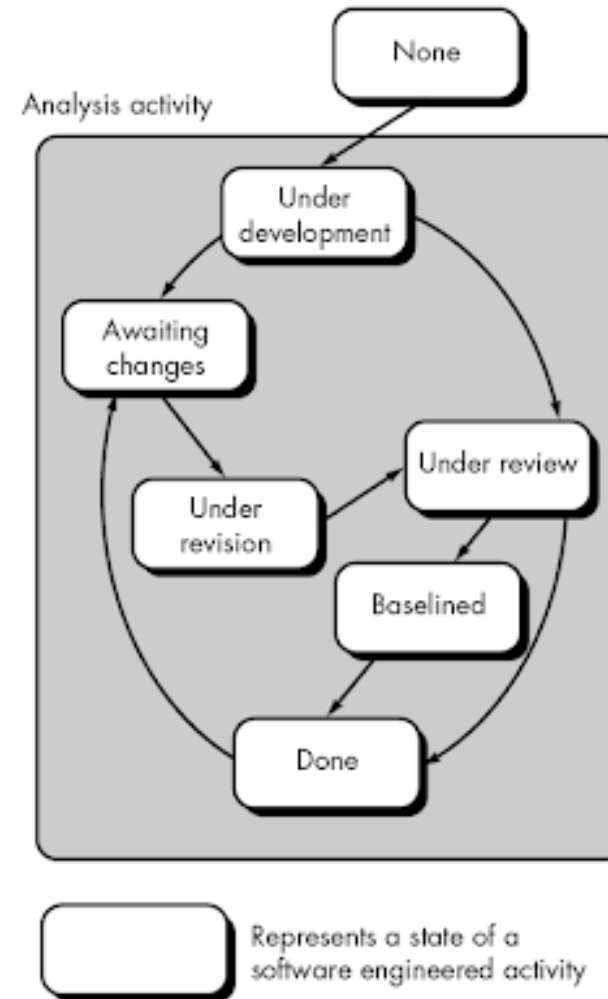
Disadvantages of Spiral Model

- It can be costly to develop a software model.
- It is not used for small projects.

Concurrent Development Model

- The *concurrent development model*, sometimes called *concurrent engineering*, can be represented schematically as a series of framework activities, Software engineering actions of tasks, and their associated states.
- The concurrent model is often more appropriate for system engineering projects where different engineering teams are involved and is more complex

Concurrent Development Model



Concurrent Development Model

- The activity—analysis—may be in any one of the states noted at any given time. Similarly, other activities (e.g., design or customer communication) can be represented in an analogous manner. All activities exist concurrently but reside in different states. For example, early in a project the customer communication activity (not shown in the figure) has completed its first iteration and exists in the awaiting changes state. The analysis activity (which existed in the none state while initial customer communication was completed) now makes a transition into the under development state. If, however, the customer indicates that changes in requirements must be made, the analysis activity moves from the under development state into the awaiting changes state.

concurrent development model

Advantages of the concurrent development model

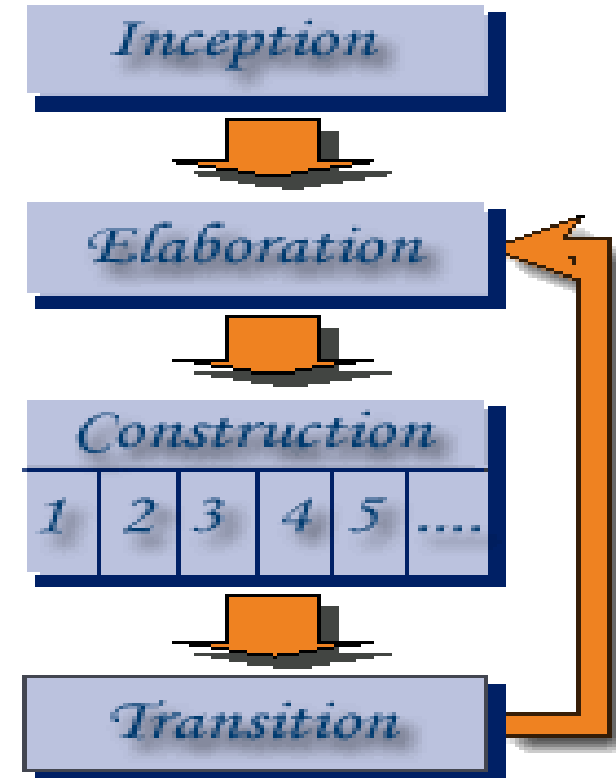
- This model is applicable to all types of software development processes.
- It is easy for understanding and use.
- It gives immediate feedback from testing.
- It provides an accurate picture of the current state of a project.

Disadvantages of the concurrent development model

- It needs better communication between the team members. This may not be achieved all the time.
- It requires to remember the status of the different activities.

The Unified Process

- Unified Process is based on the enlargement and refinement of a system through multiple iterations, with cyclic feedback and adaptation.
- The system is developed incrementally over time, iteration by iteration, and thus this approach is also known as iterative and incremental software development.
- The iterations are spread over four phases where each phase consists of one or more iterations



Inception

- **Inception** —the first and the shortest phase in the project. It is used to prepare basis for the project, including preparation of business case, establishing project scope and setting boundaries, outlining key requirements, and possible architecture solution together with design tradeoffs, identifying risks, and development of initial project plan—schedule with main milestones and cost estimates. If the inception phase lasts for too long, it is like an indicator stating that the project vision and goals are not clear to the stakeholders. With no clear goals and vision the project most likely is doomed to fail. At this scenario it is better to take a pause at the very beginning of the project to refine the vision and goals. Otherwise it could lead to unnecessary make-overs and schedule delays in further phases.

Elaboration

- **Elaboration** —during this phase the project team is expected to capture a majority of system's requirements (e.g., in the form of use cases), to perform identified risk analysis and make a plan of risk management to reduce or eliminate their impact on final schedule and product, to establish design and architecture (e.g., using basic class diagrams, package diagrams, and deployment diagrams), to create a plan (schedule, cost estimates, and achievable milestones) for the next (construction) phase.

Construction

- **Construction** —the longest and largest phase within Unified Process. During this phase, the design of the system is finalized and refined and the system is built using the basis created during elaboration phase. The construction phase is divided into multiple iterations, for each iteration to result in an executable release of the system. The final iteration of construction phase releases fully completed system which is to be deployed during transition phase, and

Transition

- **Transition** —the final project phase which delivers the new system to its end-users. Transition phase includes also data migration from legacy systems and user trainings.

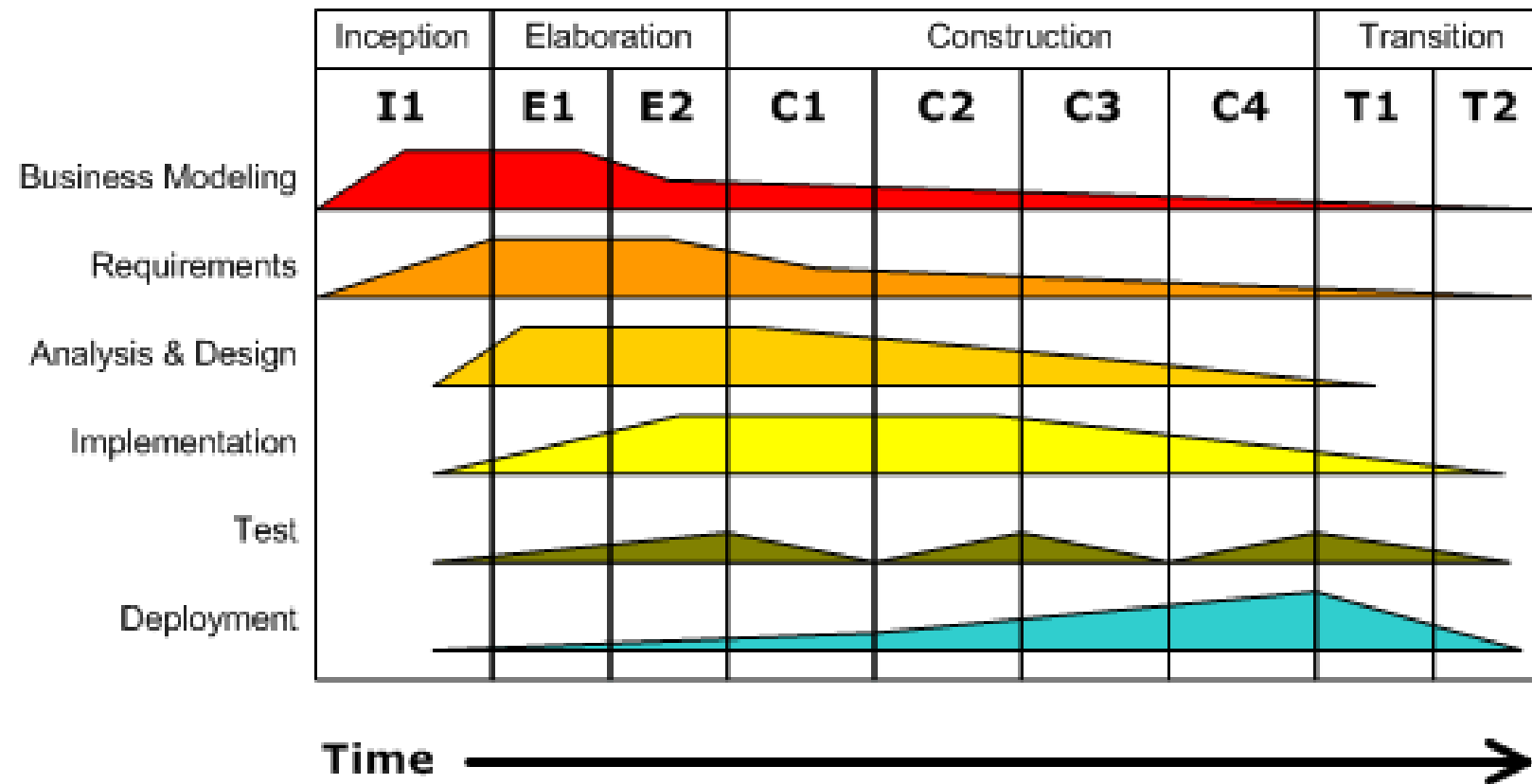
Phases in the Unified Process

- The disciplines and phases of Unified Process are given in Fig. where the phases are columns and the disciplines are rows. It clearly shows that the relative effort across disciplines changes over time from iteration to iteration, e.g., initial iterations apply greater relative effort on requirements and design while the latter—more on testing and deployment.

Phases in the Unified Process (Cont...)

Iterative Development

Business value is delivered incrementally in time-boxed cross-discipline iterations.



Static workflows in the Unified Process

This focuses on activities that take place during the development process – and called workflows

There are six core process workflows identified in the process and three core supporting workflows.

Workflow	Description
Business modelling	The business processes are modelled using business use cases .
Requirements	Actors who interact with the system are identified and use cases are developed to model the system requirements.
Analysis and design	A design model is created and documented using architectural models, component models, object models and sequence models.
Implementation	The components in the system are implemented and structured into implementation sub-systems. Automatic code generation from design models helps accelerate this process.

Static workflows in the Unified Process

Workflow	Description
Testing	Testing is an iterative process that is carried out in conjunction with implementation. System testing follows the completion of the implementation.
Deployment	A product release is created, distributed to users and installed in their workplace.
Configuration and change management	This supporting workflow managed changes to the system
Project management	This supporting workflow manages the system development
Environment	This workflow is concerned with making appropriate software tools available to the software development team.

Unified Process good practice

- **Develop software iteratively** : Plan increments based on customer priorities and deliver **highest priority** increments first.
- **Manage requirements** :Explicitly **document** customer requirements and keep track of changes to these requirements.
- **Use component-based architectures** :Organize the system architecture as a set of **reusable components**.
- **Visually model software** : Use **graphical UML models** to present static and dynamic views of the software.
- **Verify software quality**: Ensure that the software meet's organizational **quality** standards.
- **Control changes to software** :**Manage software changes** using a change management system and configuration management tools.

How to choose an appropriate process model?

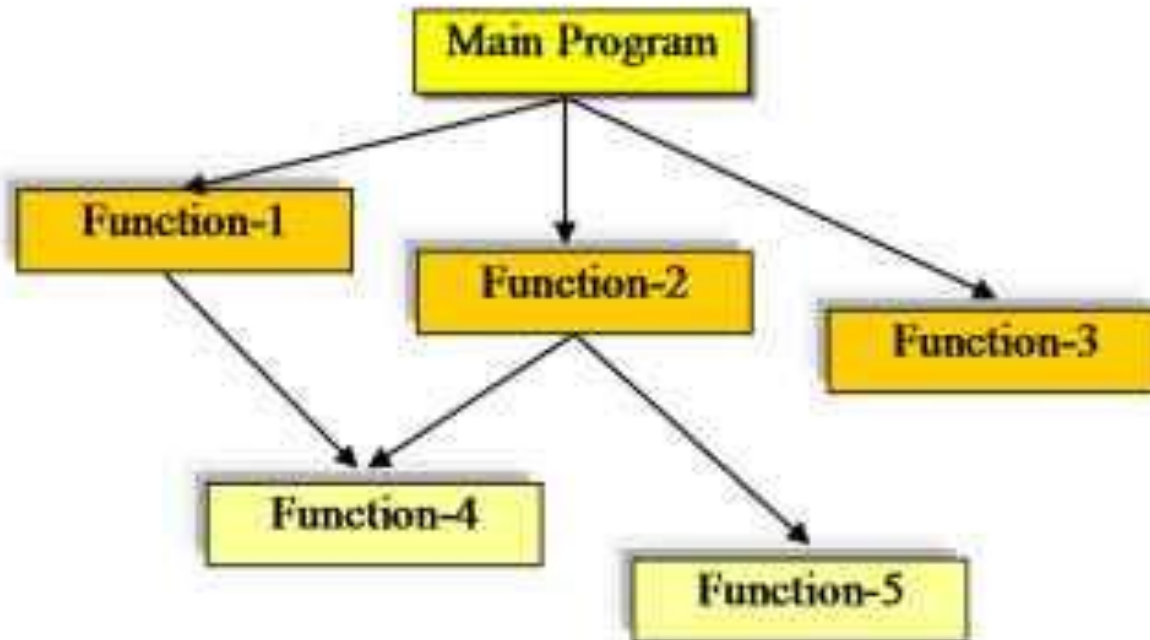
- **Characteristics of the software to be developed:** . for product and embedded development, the Iterative Waterfall model can be preferred. The evolutionary model is suitable to develop an object-oriented project. User interface part of the project is mainly developed through prototyping model.
- **Characteristics of the development team:** If the development team is experienced in developing similar software, then even an embedded software can be developed using the Iterative Waterfall model. If the development team is entirely novice, then even a simple application may require a prototyping model.
- **Risk associated with the project:** If the risks are few and can be anticipated at the start of the project, then prototyping model is useful. If the risks are difficult to determine at the beginning of the project but are likely to increase as the development proceeds, then the spiral model is the best model to use.
- **Characteristics of the customer:** If the customer is not quite familiar with computers, then the requirements are likely to change frequently as it would be difficult to form complete, consistent and unambiguous requirements. Thus, a prototyping model may be necessary to reduce later change requests from the customers.
 - the evolutionary model is useful as the customer can experience a partially working software much and reduces the customer's trauma of getting used to an entirely new system.

Summary

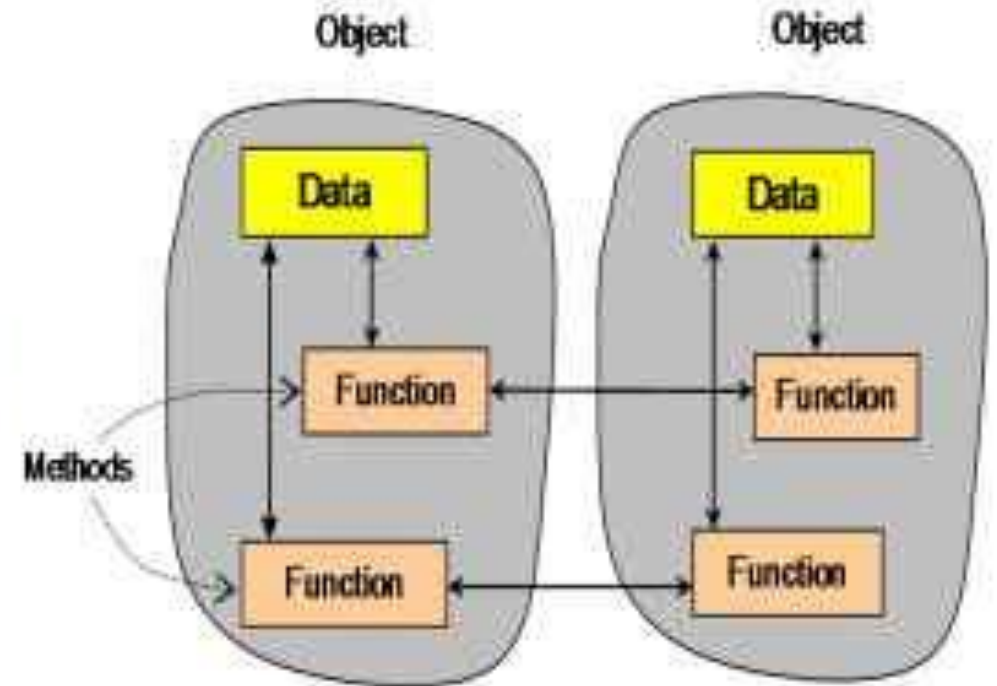
- Process is a means to achieve project objectives of high Quality
- Software process models are abstract representations of these processes
- Process models define generic process, which can form basis of project process
- Process typically has stages, each stage focusing on an identifiable task
- Many models for development process have been proposed
- Waterfall model, Evolutionary development and component-based software Engineering, Iterative process models are some process model.
- The Rational Unified Process is a generic process model that separates activities from phases
- A prototype can be used to give end-users a concrete impression of the system's capabilities

Structured Vs Object Oriented Programming

Procedure-oriented Programming



Object-oriented Programming



Structured Vs Object Oriented Programming

- **Definition**

Structured programming is a programming paradigm which divides the code into modules or function, while OOP is a programming paradigm based on the concept of objects, which contain data in the form of fields known as attributes, and code in the form of procedures known as methods. Thus, this explains the main difference between structured and object oriented programming.

- **Main Focus**

Furthermore, structured programming focuses on dividing the program into a set of functions in which each function works as a subprogram while object oriented programming focuses on representing a program using a set of objects which encapsulates data and object.

- **Modification**

Moreover, it is difficult to modify the structured programs while it is easier to modify the Object Oriented programs.

Structured Vs Object Oriented Programming

- **Communication**

In structured programming, the main method communicates with the functions by calling those functions in the main program whereas, in object oriented programming, the objects communicate with each other by passing messages. Hence, this is an important difference between structured and object oriented programming.

- **Access Specifiers**

- There are no access specifiers in structured programming while there are access specifiers such as private, public and protected in Object Oriented Programming. Thus, this is also an important difference between structured and object oriented programming.

- **Security**

- Besides, data is not secure in structured programming, but it is secure in object oriented programming.

Structured Vs Object Oriented Programming

- Code Reusability

Also, it is difficult to reuse code in structured programming, whereas it is easier to reuse code in object oriented programming.

- Conclusion

Overall, structured and object oriented programming are two major programming paradigms. The main difference between structured and object oriented programming is that structured programming helps to develop a program using a set of modules or functions while object oriented programming helps to construct a program using a set of objects and their interactions.

Sr.No	Comparison Parameter	Procedural Programming	Object-Oriented Programming
1	Definition	Based on the concept of a Procedure call, Procedural Programming is one type of programming paradigm or programming model based on Structured Programming.	Based on the concept of Object, which contains data and code, Object-Oriented Programming is one type of programming paradigm or programming model.
2	Short-Form	Procedural Oriented Programming Model is called as POP.	Object-Oriented Programming Model is called as OOP.
3	Approach	POP follows Top-Down approach.	OOP follows Bottom-Up approach.
4	Access Modifiers	POP languages do not support any kind of access modifiers.	OOP languages supports different kinds of access modifiers. Example: Java supports access modifiers like public, private, and protected.
5	Security	Due to unavailability of any kind of access modes, POP is less secure.	Due to availability of access modifiers classes can contain private variables and methods, hence OOP is more secure.

Sr.No	Comparison Parameter	Procedural Programming	Object-Oriented Programming
6	Main Concept	Main concept of POP is Procedures. Procedures are sequence of actions that need to be performed. Data is generally stored in variables in the form of Methods.	Main concept of OOP is Objects and classes. Data is generally stored in the form of Attributes and in the terms of objects.
7	Importance	Key Difference between both the approaches is the importance, In POP, importance is given to the functions over data. Importance is given to the sequence that need to be followed.	In OOP, importance is given to the data, the way one stores the data and how securely data is accessed. Security and accessibility of data is very important aspect in OOP.
8	Real World Orientation	POP is not inclined with real world orientation because while understanding POP concepts, one cannot relate it with real world examples.	OOP approach itself is derived from looking at real world examples. Hence, OOP concepts can easily be understood using comparison with real world.

Sr.No	Comparison Parameter	Procedural Programming	Object-Oriented Programming
9	Complexity	POP programs is less complex to write because there is no need to define classes or access specifiers. But after writing once, it is hard to read the program or to update the program relative to OOP.	OOP program is relatively complex to write because of availability of classes, Inheritance, access specifiers and other OOP paradigms. But it is easy to update the program or read the program.
10	Data Movement	Data can move freely from one function to another because there is not any access specifiers.	Data cannot move freely in OOP because of access specifiers. But objects can communicate with each other through member functions.
11	Data Sharing	Data can be shared with global scope or local scope.	Data can be shared with global, local, and class level scope.
12	Code Reusability	Code reusability is very limited in POP. It can only be achieved through functions.	Code reusablity is one of the aim of OOP. It can be achieved using the concept of Inheritance.

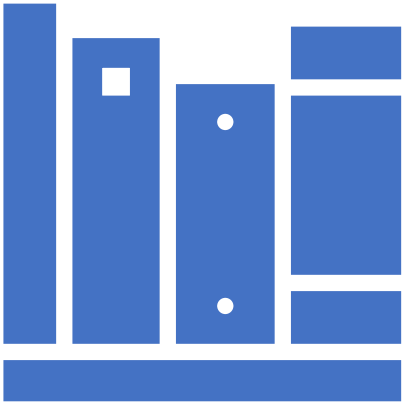
Sr. No	Comparison Parameter	Procedural Programming	Object-Oriented Programming
13	Overloading	The concept of overloading is not present in POP.	The overloading can be achieved in the form of Operator-Overloading and Function-Overloading in OOP.
14	Virtual Function and Virtual Classes	Concept of virtual function or virtual classes is not available in POP.	There are many object-oriented programming systems available that uses virtual functions and virtual classes.
15	Size Of Problem	POP is generally preferred when the size of the problem is very small.	OOP is preferred when one have to implement large tasks and systems.
16	Speed	POP program generally runs faster than OOP.	OOP is relatively slow in execution compared to POP.
16	Programming Languages	POP languages: C, COBOL, Pascal, FORTRAN, BASIC, etc.	OOP Languages: C++, Python, Java, Kotlin, JavaScript, Dart, Solidity, etc.

References

- Roger S Pressman, Software Engineering: A Practitioner's Approach, Mcgraw-Hill, ISBN: 0073375977, Seventh Edition, 2014
- Pankaj Jalote, Software Engineering: A Precise Approach, Wiley India.2010.

Disclaimer:

- a. Information included in this slides came from multiple sources. We have tried our best to cite the sources. Please refer to the [References](#) to learn about the sources, when applicable.
- b. The slides should be used only for academic purposes (e.g., in teaching a class), and should not be used for commercial purposes.



Unit-I End
