# Introduction

Software is:

(1) instructions (computer programs) that when executed provide desired features, function, and performance;

(2) data structures that enable the programs to adequately

manipulate information and

(3) descriptive information in both hard copy and

virtual forms that describes the operation and use of the programs.

Software is a

logical rather than a physical system element.

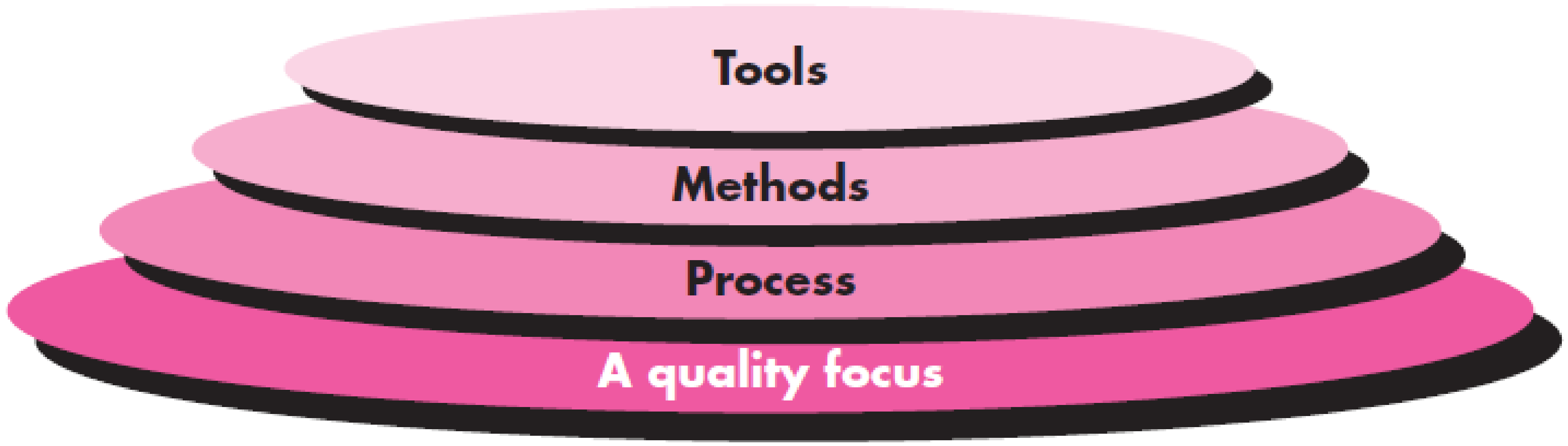
## Continue…

- Therefore software has characteristics.
- software is engineered
- software doesn’t wear out
- software is complex

**Engineering** on the other hand, is all about developing products, using well-defined, scientific principles and methods.

## Continue...

Software engineering is a layered technology.



## Process…

- The foundation for software engineering is the *process* layer. The software engineering process is the glue that holds the technology layers together and enables rational and timely development of computer software.
- Process defines a framework that must be established for effective delivery of software engineering technology.
- The software process forms the basis for management control of software projects and establishes the context in which technical methods are applied, work products(models, documents, data, reports, forms, etc.) are produced, milestones are established, quality is ensured, and change is properly managed.

# Methods and Tools…

Software engineering *methods* provide the technical how-to build software. Methods encompass a broad array of tasks that include communication, requirements analysis, design modeling, program construction, testing, and support.

- Software engineering methods rely on a set of basic principles that govern each area of the technology
- Software engineering *tools* provide automated or semi-automated support for the

process and the methods. When tools are integrated so that information created by

one tool can be used by another.

## Terminologies

- Actor

  An actor represents the role of a user that interacts with the system.

  Some of the e.g. of the actors used in L.M.S are admin, data entry operator, student, faculty etc.

- A use case describes who(any user) does what(interaction) with the system, for what goal, without considering the internal details of the system. A complete set of use cases explains the various ways to use the system.
- The use case model depicts actors, use cases and the relationship between them.
- A use case scenario is an instance of a use case or a complete path through the use case. The basic flow is one scenario and every alternative path gives another scenario.

## Continue…

- System and Subsystems

A system is an organized and arranged structure as a whole that consists of interrelated and well-defined procedures, processes and methods, All systems consists of inputs, outputs, feedback and boundaries

The system may consist of several subsystems. Subsystems are a way of reducing complexity of the system. For e.g. in a company accounts, sales, marketing etc. are different subsystem. In object oriented analysis, objects may be grouped together to form a subsystem.

# Continue…

- Class

  A class is a template that consists of attributes and operations.
- Responsibilities are attributes and operations included in a class.
- Collaborations are the other classes that a class calls in order to achieve the functionality.
- Measures, Metrics and Measurement
- A measure provides a quantitative indication of the amount, dimension, capacity or size of the some attributes of a product or process.
- Measurement is the act of determining a measure.
- The metric is a quantitative measure of the degree to which a product or process possesses a given attributes.

## Continue…

- Software quality and reliability
- Software reliability is defined as "the probability of failure free operation for a specified time in a specified environment"
- Software quality determines how good the software designed is and how good the software conforms to that design
- Some practitioners also feel that quality and reliability are the same thing, but that is not true. Reliability is just one part of quality.
- To produce good quality product, a software tester must verify and validate throughout the software development process.

## Continue..

- Quality Assurance and Quality Control
- The purpose of QA activity is to enforce standards and techniques to improve the development process and prevent the previous bugs from ever occurring. A good QA activity enforces good software engineering practices which help to produce good quality software. The QA group monitors and guides throughout the software development life cycle. This is a defect prevention technique and concentrates on the process of the software development. E.g are reviews, audits etc.
- Quality control attempts to build a software and test it thoroughly.

## Continue…

- If failures are experienced, remove the causes of failures and ensure the correctness of removal.
- It concentrates on specific products rather than processes as in the case of QA.
- This is a defect-detection and correction activity which is usually done after the completion of the software development.
- E.g. software testing.

# Continue…

- Verification and validation
- As per IEEE " It is the process of evaluating the system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of the phase. "
- We apply verification activities from the early phase of the software development and check/review the documents generated after the completion of each phase.
- Hence it is the process of reviewing the requirement document, design document, source code and other related documents of the project.
- This is the manual testing and involves only looking at the documents in order to ensure what comes out is that we expected to get

## Continue..

- Validation:
- As per IEEE “It is the process of evaluating a system or component during or at the end of the development process to determine whether it satisfies the specified requirements.”
- It requires the actual execution of the program. It is a dynamic testing and requires computer for execution of the program.
- Hence testing = Verification + Validation
- Both are essential and complementary activities of software testing.

## Continue..

- Fault, Error, Bug and Failure
- All these terms are used interchangeably, although the terms error, mistake and defect are synonyms in software testing terminology. When we make an error during coding, we call this ‘bug’. Hence, error/mistake/defect in coding is called a bug.
- A fault is the representation of an error, where the representation is the mode of expression, such as data flow diagram, entity-relationship (ER) diagrams, source code, use cases, etc. If the fault is in the source code, we call it a bug.
- A failure is the result of execution of a fault and is dynamic in nature. When the expected output does not match with the observed output, we experience a failure. The program has to execute for a failure to occur. A fault may lead to many failures. A particular fault may cause different failures, depending on the inputs to the program.

## Continue..

- States and events
- A state is an abstract situation in the life cycle of an entity that occurs in response to occurrence of some event.
- An event is an input(a message or method call).
- Due to the occurrence of some event, the system transits from one state to the other.

## Traditional Approach and Object Oriented Approach

- The system is viewed as Collection of processes/objects
- DFD, ER, data dictionary used to describe the system/UML
- Reusable source code not produced/produced
- Follows Top down/bottom up
- Non-iterative/highly iterative

## Advantages and limitation

- All methods have their own advantages and disadvantages.
- The OOA method lacks the design of interfaces and notation.
- The OOD method is stronger in design but weaker in analysis of the system.
- The OMT method is stronger in analysis part but weaker in design part.
- The OOSE is stronger in behavioral areas as compared to the other areas.

## Object oriented Modeling

- Object oriented modeling is a way of constructing visual models based on real world objects.
- Modeling helps in understanding the problems, developing proper documents and producing well-designed programs.
- The models must depict the various views of the system and these models must be verified to check whether they capture the customer’s requirements.
- After they represent the required details, these models may be transformed into source code.
- Most popular methodologies were OOD, OMT and OOSE.
- All these methods are combined into the Unified Modeling Language.
- Thus the UML represents the combination of the notations used by Booch

## Continue..

Rambaugh and Jacobson

- Thus, the UML represents the combination of the notations used by Booch, Rumbaugh and Jacobson.
- The best concepts and processes were extracted from all the methodologies till date and combined into UML.
- The UML was adopted by Object Management Group in nov. 1997

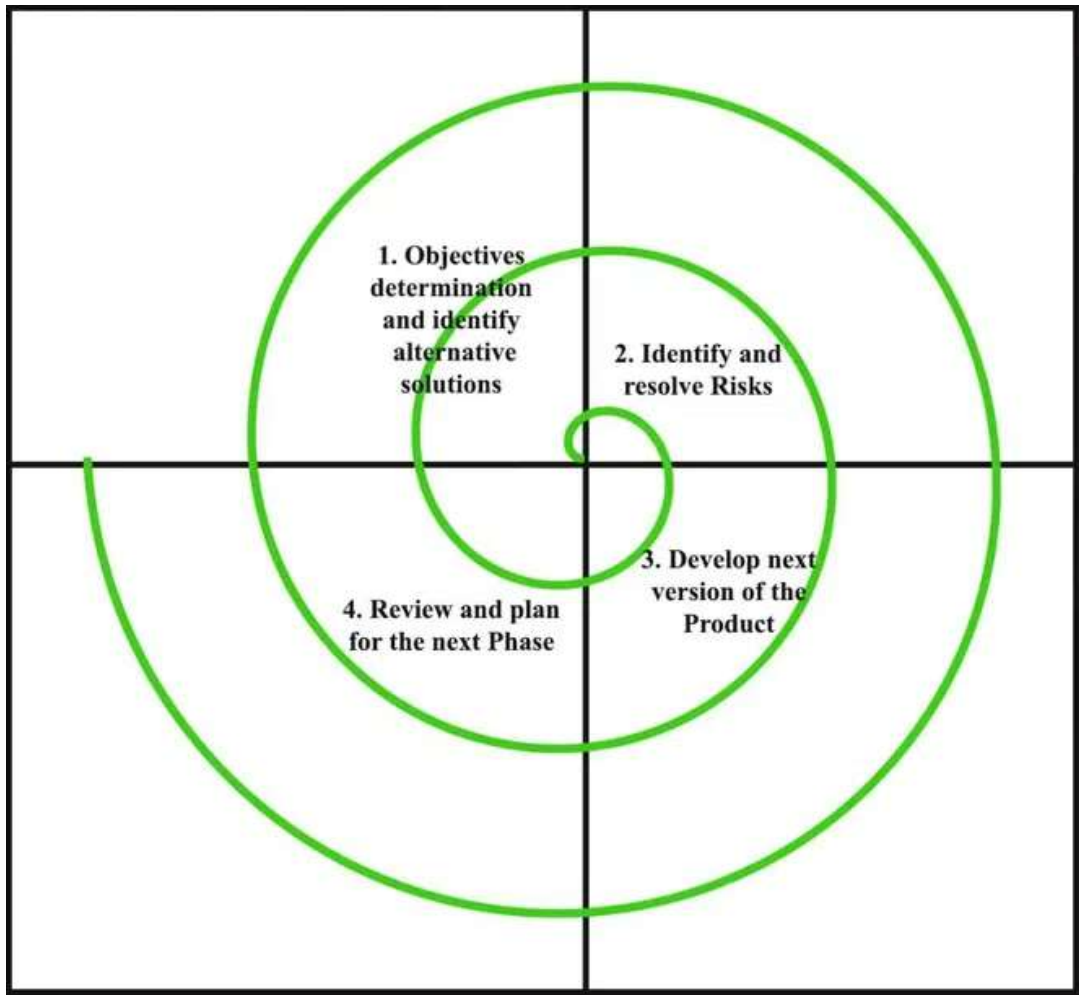
## Spiral Model

- **The Spiral Model** is one of the most important Software Development Life Cycle models, which provides support for **Risk Handling**. In its diagrammatic representation, it looks like a spiral with many loops. The exact number of loops of the spiral is unknown and can vary from project to project. Each loop of the spiral is called a **Phase of the software development process.**
- The exact number of phases needed to develop the product can be varied by the project manager depending upon the project risks. As the project manager dynamically determines the number of phases, the project manager has an important role to develop a product using the spiral model.
- The Spiral Model is a **Software Development Life Cycle (SDLC)** model that provides a systematic and iterative approach to software development. It is based on the idea of a spiral, with each iteration of the spiral representing a complete software development cycle, from requirements gathering and analysis to design, implementation, testing, and maintenance.

## Process of spiral model

- **Planning:** The first phase of the Spiral Model is the planning phase, where the scope of the project is determined and a plan is created for the next iteration of the spiral.
- **Risk Analysis:** In the risk analysis phase, the risks associated with the project are identified and evaluated.
- **Engineering:** In the engineering phase, the software is developed based on the requirements gathered in the previous iteration.
- **Evaluation:** In the evaluation phase, the software is evaluated to determine if it meets the customer's requirements and if it is of high quality.
- **Planning:** The next iteration of the spiral begins with a new planning phase, based on the results of the evaluation.
- The Spiral Model is often used for complex and large software development projects, as it allows for a more flexible and adaptable approach to software development. It is also well-suited to projects with significant uncertainty or high levels of risk.
- The Radius of the spiral at any point represents the expenses(cost) of the project so far, and the angular dimension represents the progress made so far in the current phase.

## Continue…


1. Objectives determination and identify alternative solutions
2. Identify and resolve Risks
3. Develop next version of the Product
4. Review and plan for the next Phase

## Specialized Process Model

Component-Based Development(CBD)
The component Based Development Model has the characteristics of a spiral model,
hence is evolutionary and iterative in nature. In this model, applications are built from pre-packaged software
components that are available from different vendors. Components are modular products with well-defined funct
The modelling and construction stages are begun with identifying candidate components suitable for the project
Steps involved in this approach are implemented in an evolutionary fashion:

1.Components suitable for the application domain are selected.
2.Component integration issues are catered to.
3.Software architecture is designed based on the selected components.
4.Components are integrated into the architecture.
5.Testing of the functionality of components is done.

Software can be re-used in this manner, cost and time is reduced.

## Continue…

- Formal Methods Model(FMM)
- In the Formal Methods Model, mathematical methods are applied in the process of developing software. It Uses Formal Specification Language (FSL) to define each system characteristics. FSL defines the syntax, notations for representing system specifications, several objects and relations to define the system in detail.
- The careful mathematical analysis that is done in FMM results in a defect-free system. It also helps to identify and correct ambiguity and inconsistency easily. This method is time-consuming and expensive in nature. Also, knowledge of formal methods is necessary for developers and is a challenge.