Software systems are abstract and intangible. They are not constrained by the properties of materials, governed by physical laws.

software Computer programs and associated documentation.

Software validation or, more generally, verification and validation (V&V) is intended to show that a system both conforms to its specification and that it meets the expectations of the system customer.

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

Boehm’s spiral model:A risk-driven software process framework was proposed by Boehm (1988).

software engineering is an engineering discipline that is concerned with all aspects of software production.

waterfall model is consistent with other engineering process models and docu-mentation is produced at each phase.

Incremental development: Incremental development is based on the idea of developing an initial implementa tion

Software specification or requirements engineering is the process of understanding and defining what services are required from the system and identifying the con-straints on the system’s operation and development.

Software evolution:The flexibility of software systems is one of the main reasons why more and more software is being incorporated in large, complex systems .

Reuse-oriented software engineering:In the majority of software projects, there is some software reuse.

a reuse-oriented advantage :amount of software to be developed and so reducing cost and risks.

Process activities: Real software processes are interleaved sequences of technical, collaborative, and managerial activities

The functional requirements: for a system describe what the system should do.

Non-functional requirements, as the name suggests, are requirements that are not directly concerned with the specific services delivered by the system to its users.

Domain requirements are derived from the application domain of the system rather than from the specific needs of system users.

Prototyping:A prototype is an initial version of a software system that is used to demonstrate concepts,

The problem with domain requirements is that software engineers may not understand the characteristics of the domain in which the system operates

Requirements specification is the process of writing down the user and system requirements in a requirements document.

Customer involvement: Customers should be closely involved throughout the development process.

Incremental delivery: The software is developed in increments with the customer specifying the requirements to be included in each increment.

User requirements are statements, in a natural language plus diagrams, of what services the system is expected to provide to system users and the constraints under which it must operate.

System requirements are more detailed descriptions of the software system’s

functions, services, and operational constraints.

People not process: The skills of the development team should be recognized and exploited.

Embrace change: Expect the system requirements to change and so design the system to accommodate these changes.

Maintain simplicity :Focus on simplicity in both the software being developed and in the development process.

Software engineering is criticized as inadequate for software failures are a consequence of two factors:

1.Increasing demands

2. Low expectations

Software products may be developed for a particular customer or may be developed for a general market. What are the attributes of good software?

Good software should deliver the required functionality and performance to the user and should be maintainable, dependable, and usable.

What is the difference between software engineering and computer science?

Computer science focuses on theory and fundamentals;

software engineering is concerned with the practicalities of developing and delivering useful software.

What is the difference between software engineering and system engineering?

System engineering is concerned with all aspects of computer-based systems development including hardware, software, and process engineering.

Software engineering is part of this more general process.

What are the key challenges facing software engineering?

Coping with increasing diversity, demands for reduced delivery times, and developing trustworthy software.

What are the costs of software engineering?

Roughly 60% of software costs are development costs; 40% are testing costs. For custom software, evolution costs often exceed development costs.

What are the best software engineering techniques and methods?

While all software projects have to be professionally managed and developed, different techniques are appropriate for different types of system. For example, games should always be developed using a series of prototypes whereas safety critical control systems require a complete and analyzable specification to be developed. You can’t, therefore, say that one method is better than another.

What differences has the Web made to software engineering?

The Web has led to the availability of software services and the possibility of developing highly distributed service-based systems. Web-based systems development has led to important advances in programming languages and software reuse.

There are two kinds of software products:

1. *Generic products*

2. *Customized products*

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

1. Engineering discipline
2. *All aspects of software production*

Software engineering is important for two reasons:

1. More and more, individuals and society rely on advanced software systems.
2. It is usually cheaper

Different types of systems need different development processes. , the specification and the program are usually developed together.

1. Computer science is concerned with the theories and methods that underlie computers and software systems, whereas software engineering is concerned with the practical problems of producing software.
2. System engineering is concerned with all aspects of the development and evolution of complex systems

there are three general issues that affect many different types of software:

1. Heterogeneity Increasingly,

2. Business and social change Business

3. Security and trust

There are many different types of application including:

1. Stand-alone applications 2. Interactive transaction-based applications

3. Embedded control systems 4. Batch processing

5. Entertainment systems 6. Systems for modeling and simulation

7. Data collection systems 8. Systems of systems

there are software engineering fundamentals that apply to all types of software system:

1. They should be developed using a managed and understood development process.

2. Dependability and performance are important for all types of systems.

3. Understanding and managing the software specification and requirements

4. You should make as effective use as possible of existing resources.

However, there are areas where standards of acceptable behavior are not bound by laws but by the more tenuous notion of professional responsibility. Some of these are:

1. Confidentiality

2. Competence

3. Intellectual property rights

4. Computer misuse

Software Engineering Code of Ethics and Professional Practice:

1. PUBLIC — Software engineers shall act consistently with the public interest.

2. CLIENT AND EMPLOYER — Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.

3. PRODUCT — Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.

4. JUDGMENT — Software engineers shall maintain integrity and independence in their professional judgment.

5. MANAGEMENT — Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.

6. PROFESSION — Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.

7. COLLEAGUES — Software engineers shall be fair to and supportive of their colleagues.

8. SELF — Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

There are many different software processes but all must include four activities that are fundamental to software engineering: What are the fundamental software engineering activities?

1. Software specification 2. Software design 3. Software validation 4. Software evolution

process descriptions may also include:

1. Products 2. Roles 3. Pre- and post-conditions

The process models that I cover here are:

1. The waterfall model 2. Incremental development 3. Reuse-oriented software engineering

The principal stages of the waterfall model directly reflect the fundamental devel Opment activities:

1. Requirements analysis 2. System and software design

3. Implementation and unit testing 4. Integration and system testing

5. Operation and maintenance .

Incremental development has three important benefits, compared to the waterfall model:

1. The cost of accommodating changing customer requirements is reduced.

2. It is easier to get customer feedback on the development work that has been done

3. More rapid delivery and deployment of useful software

From a management perspective, the incremental approach has two problems:

1. The process is not visible 2. System structure tends to degrade as new increments are added

processes, the intermediate stages in a reuse-oriented process are different. These stages are:

1. Component analysis 2. Requirements modification

3. System design 4. Development and integration

There are three types of software component that may be used in a reuse-oriented process:

1. Web services 2. Collections of objects 3. Stand-alone software systems

Software development tools:

■ The development of graphical system models as part of the requirements specification or the software design

■ The generation of code from these graphical models

■ The generation of user interfaces from a graphical interface description that is created interactively by the user

■ Program debugging through the provision of information about an executing program

■ The automated translation of programs written using an old version of a programming language to a more

The requirements engineering process (Figure 2.4) There are four main activities in the requirements engineering process:

1. Feasibility study 2. Requirements elicitation and analysis .

3. Requirements specification 4. Requirements validation

shows four activities that may be part of the design process for information systems:

1. Architectural design 2. Interface design 3. Component design 4. Database design

The stages in the testing process are:

1. Development testing. 2. System testing . 3. Acceptance testing

There are two related approaches that may be used to reduce the costs of rework:

1. Change avoidance 2. Change tolerance

section, I discuss two ways of coping with change and changing system requirements. These are:

1. System prototyping 2. Incremental delivery

A software prototype can be used in a software development process to help anticipate changes that may be required:

1. In the requirements engineering process 2. In the system design process

Incremental delivery has a number of advantages:

1. Customers can use the early increments as prototypes and gain experience

2. Customers do not have to wait until the entire

3. The process maintains the benefits of incremental

4. As the highest-priority services

However, there are problems with incremental delivery:

1. Most systems require 2. Iterative development 3. The essence of iterative processes

Each loop in the spiral is split into four sectors:

1. Objective setting. 2. Risk assessment and reduction 3. Development and validation 4. Planning

In contrast, the RUP is normally described from three perspectives:

1. A dynamic perspective 2. A static perspective 3. A practice perspective

unlike the waterfall model where phases are equated with process activities, the phases in the RUP are more closely related to business rather than technical concerns. Figure 2.11 shows the phases in the RUP. These are:

1. Inception 2. Elaboration 3. Construction 4. Transition

Although there are many approaches to rapid software development, they share some fundamental characteristics:

1. The processes of specification, design, and implementation are interleaved.

2. The system is developed in a series of versions

3. System user interfaces are often developed using an interactive development

The implementation of these requirements may be diffused throughout the system. There are two reasons for this:

1.Non-functional requirements 2. A single non-functional requirement

Two types of Non-functional

1. *Product requirements*  2. *Organizational requirements*  3. *External requirements*