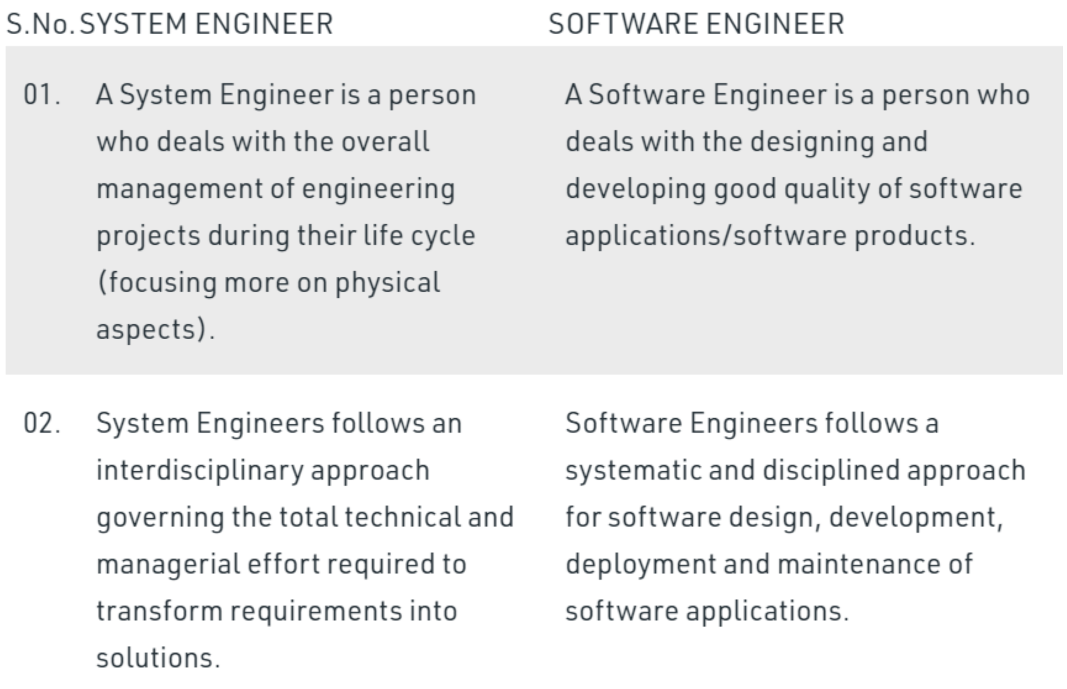
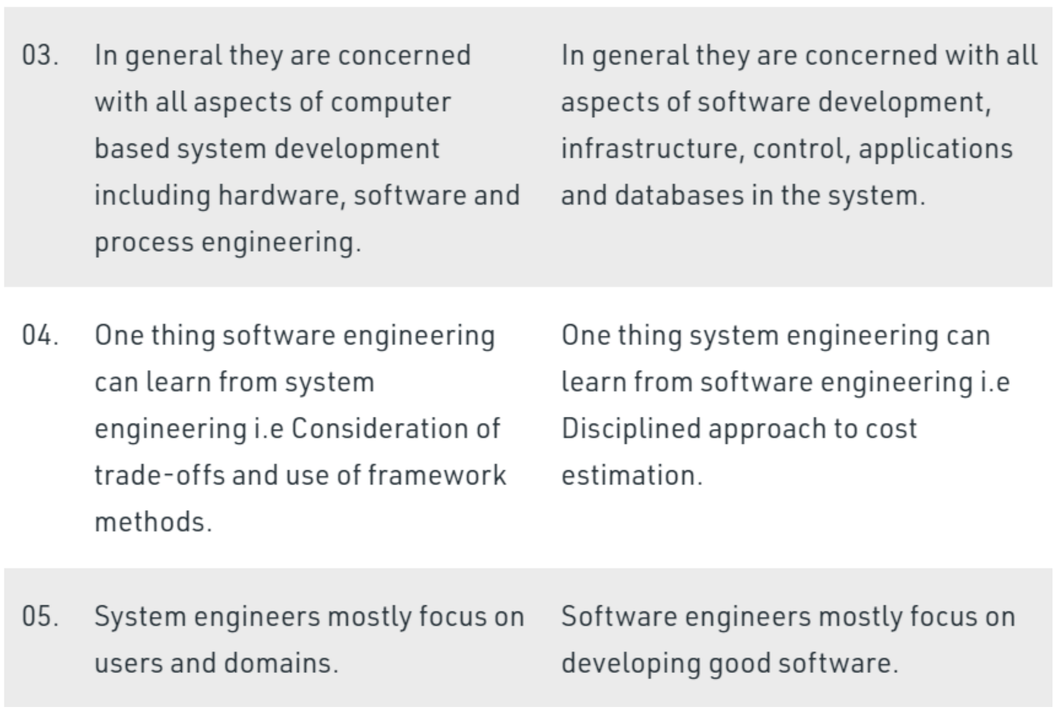
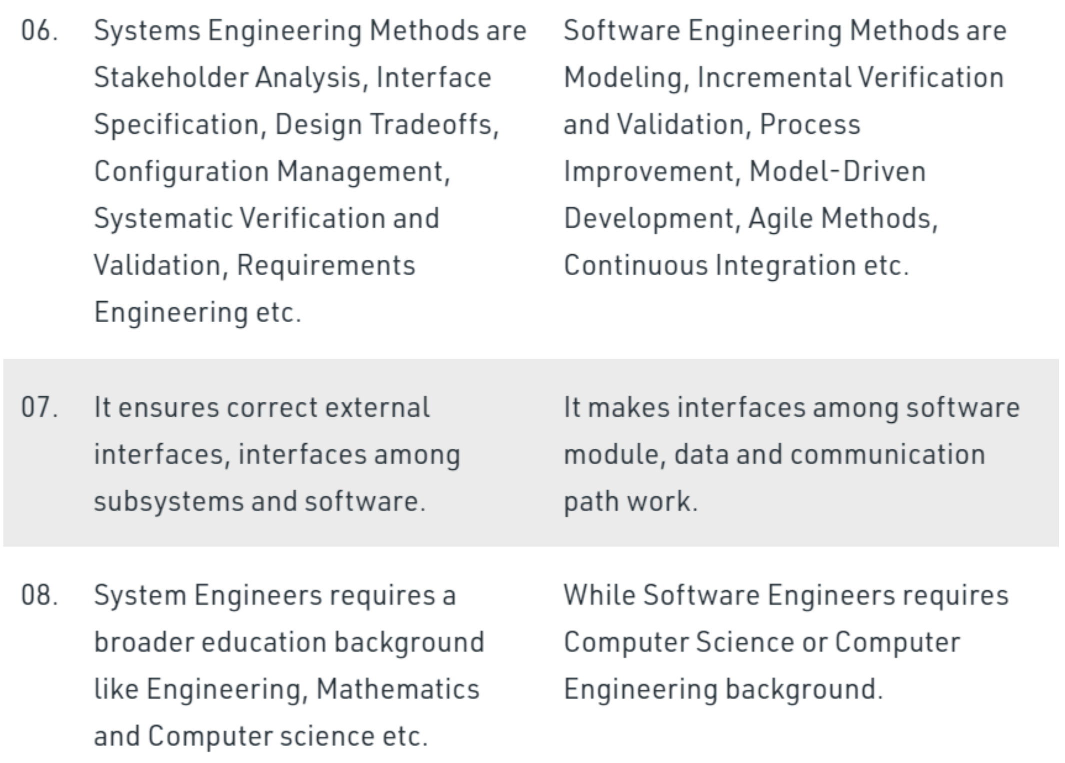
UNIT - 1

1. **Answer following frequently asked questions about software Engineering. L1 6M**

   i) Difference between software engineering and system engineering

S.No.	SYSTEM ENGINEER	SOFTWARE ENGINEER
01.	A System Engineer is a person who deals with the overall management of engineering projects during their life cycle (focusing more on physical aspects).	A Software Engineer is a person who deals with the designing and developing good quality of software applications/software products.
02.	System Engineers follows an interdisciplinary approach governing the total technical and managerial effort required to transform requirements into solutions.	Software Engineers follows a systematic and disciplined approach for software design, development, deployment and maintenance of software applications.
03.	In general they are concerned with all aspects of computer based system development including hardware, software and process engineering.	In general they are concerned with all aspects of software development, infrastructure, control, applications and databases in the system.
04.	One thing software engineering can learn from system engineering i.e Consideration of trade-offs and use of framework methods.	One thing system engineering can learn from software engineering i.e Disciplined approach to cost estimation.
05.	System engineers mostly focus on users and domains.	Software engineers mostly focus on developing good software.



06.	Systems Engineering Methods are Stakeholder Analysis, Interface Specification, Design Tradeoffs, Configuration Management, Systematic Verification and Validation, Requirements Engineering etc.	Software Engineering Methods are Modeling, Incremental Verification and Validation, Process Improvement, Model-Driven Development, Agile Methods, Continuous Integration etc.
07.	It ensures correct external interfaces, interfaces among subsystems and software.	It makes interfaces among software module, data and communication path work.
08.	System Engineers requires a broader education background like Engineering, Mathematics and Computer science etc.	While Software Engineers requires Computer Science or Computer Engineering background.

–

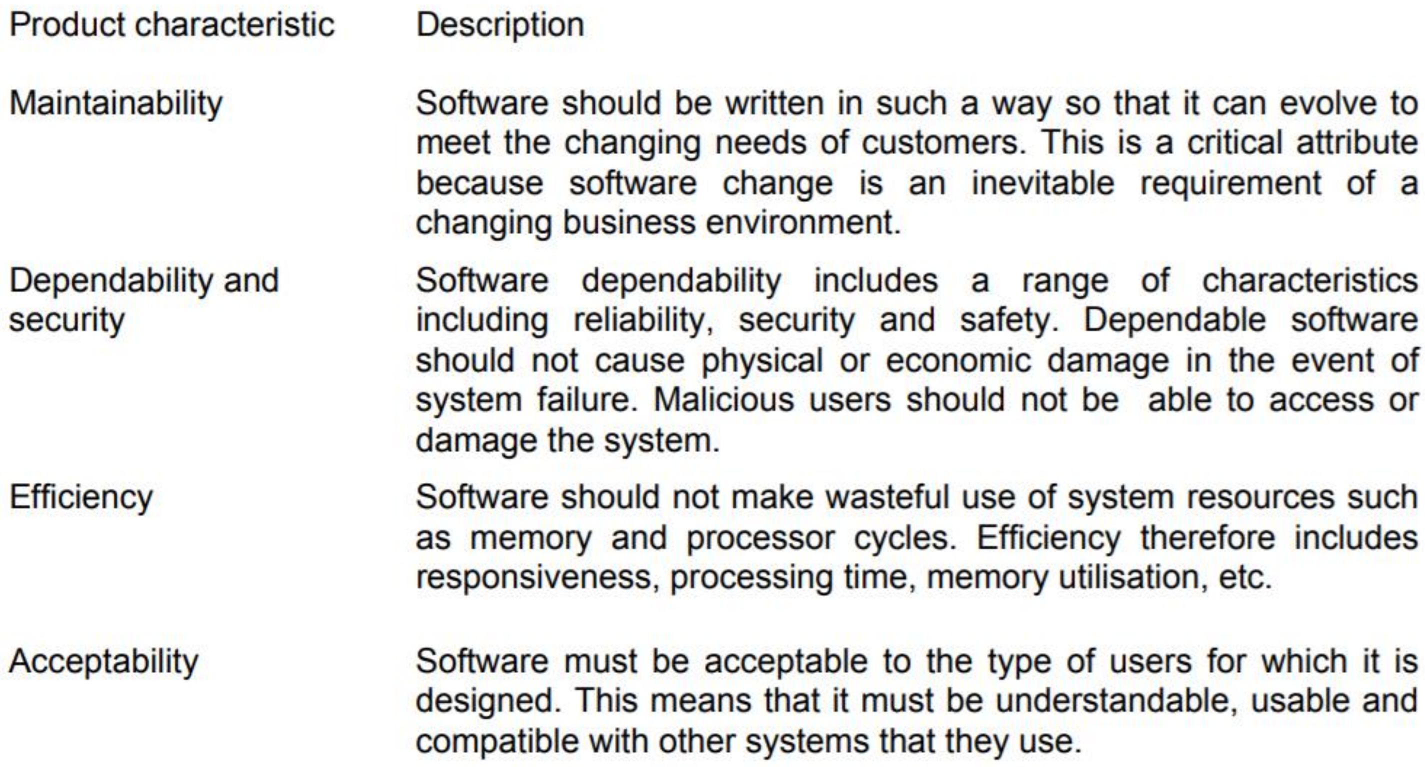
ii)What are the key challenges facing software engineering
Coping with increasing diversity, demands for reduced delivery times and developing trustworthy software.

iii) what are the fundamental software engineering activities.
Software specification, software development, software validation and software evolution.

2. **Define and explain the difference between Generic and Customized product with example?**

Generic Products	Customized Products
– Stand-alone systems that are marketed and sold to any customer who wishes to buy them. – The specification of what the software should do is owned by the software developer and decisions on software change are made by the developer. – Examples – PC software such as graphics programs, project management tools; CAD software; software for specific markets such as appointments systems for dentists.	– Software that is commissioned by a specific customer to meet their own needs. – The specification of what the software should do is owned by the customer for the software and they make decisions on software changes that are required. – Examples – embedded control systems, air traffic control software, traffic monitoring systems.

3. List and explain the essential attributes of good software?



Product characteristic	Description
Maintainability	Software should be written in such a way so that it can evolve to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of a changing business environment.
Dependability and security	Software dependability includes a range of characteristics including reliability, security and safety. Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system.
Efficiency	Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilisation, etc.
Acceptability	Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable and compatible with other systems that they use.

4. Define Software Engineering? And explain the four fundamental activities that are common to all software process?

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

1. **Software specification:** This is written description of the scope of the software; functionalities, constraints and limitations are clearly defined here.

2. **Software Design and Implementation:** The entails all the processes involved in developing a software based on a given specification.

3. **Software validation:** The software must be tested to ensure that the requirements in the software specification document are met. The developed software must perform the functions required by the customer.

4. **Software evolution:** Software evolution requires that a given software can be modified to meet changing requirements in the future.

5. **List and explain Issues that affect many different types of software?**

   1. Heterogeneity:

   Systems are increasingly essential to operate as distributed systems across networks, which include various types of computers and mobile devices. Software can be installed on cell phones, as well as on general-purpose computers.

   For older operating programs written in various programming languages, you also need to implement new applications.

   The goal here is to create strategies that are versatile enough to deal with this heterogeneity and construct dependable applications.

   2. Business and social change:

   As emerging economies grow and new innovations become available, business and society are evolving extremely rapidly. They need to be in a position to update their current software and create new software quickly.

   Many conventional software development approaches are time consuming, so it sometimes takes longer than expected to produce new systems. They need to change, so that the time taking for the program to provide value to its clients is decreased.

   3. Security and trust:

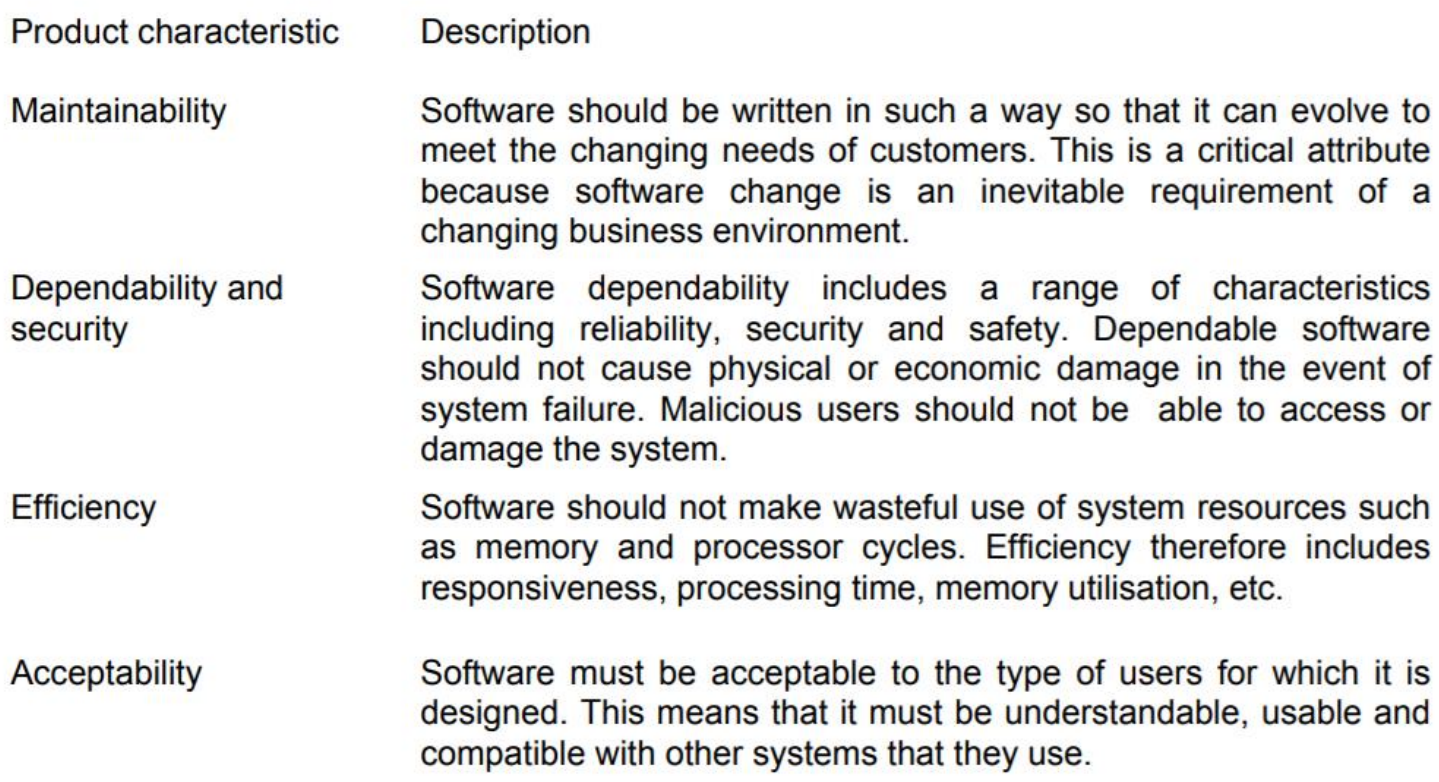
   Since software is intertwined with all facets of our lives, we need to be able to trust the software. This is especially true for remote software systems, which accessed via a web page or web service interface. We must ensure that unauthorized users are unable to target our applications, and that information is protected and secured.

   4. Scales

   The software must be built over a broad range of scales, from very tiny embedded systems in portable or wearable devices through the Internet sizes, a cloud-based framework that serves global population.

6. **Define Software Engineering, List and explain essential attributes of good software?**

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



Product characteristic	Description
Maintainability	Software should be written in such a way so that it can evolve to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of a changing business environment.
Dependability and security	Software dependability includes a range of characteristics including reliability, security and safety. Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system.
Efficiency	Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilisation, etc.
Acceptability	Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable and compatible with other systems that they use.

7. List and explain any five Software Engineering (ACM/IEEE) Code of Ethics and Professional Practices?

- PUBLIC - Software engineers shall act consistently with the public interest.
- 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.
- PRODUCT - Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.
- JUDGMENT - Software engineers shall maintain integrity and independence in their professional judgment.
- MANAGEMENT - Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.
- PROFESSION - Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.
- COLLEAGUES - Software engineers shall be fair to and supportive of their colleagues.
- 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.

8. **List and explain the issues related to software engineering ethics?**

   • Confidentiality – Engineers should normally respect the confidentiality of their employers or clients irrespective of whether or not a formal confidentiality agreement has been signed.

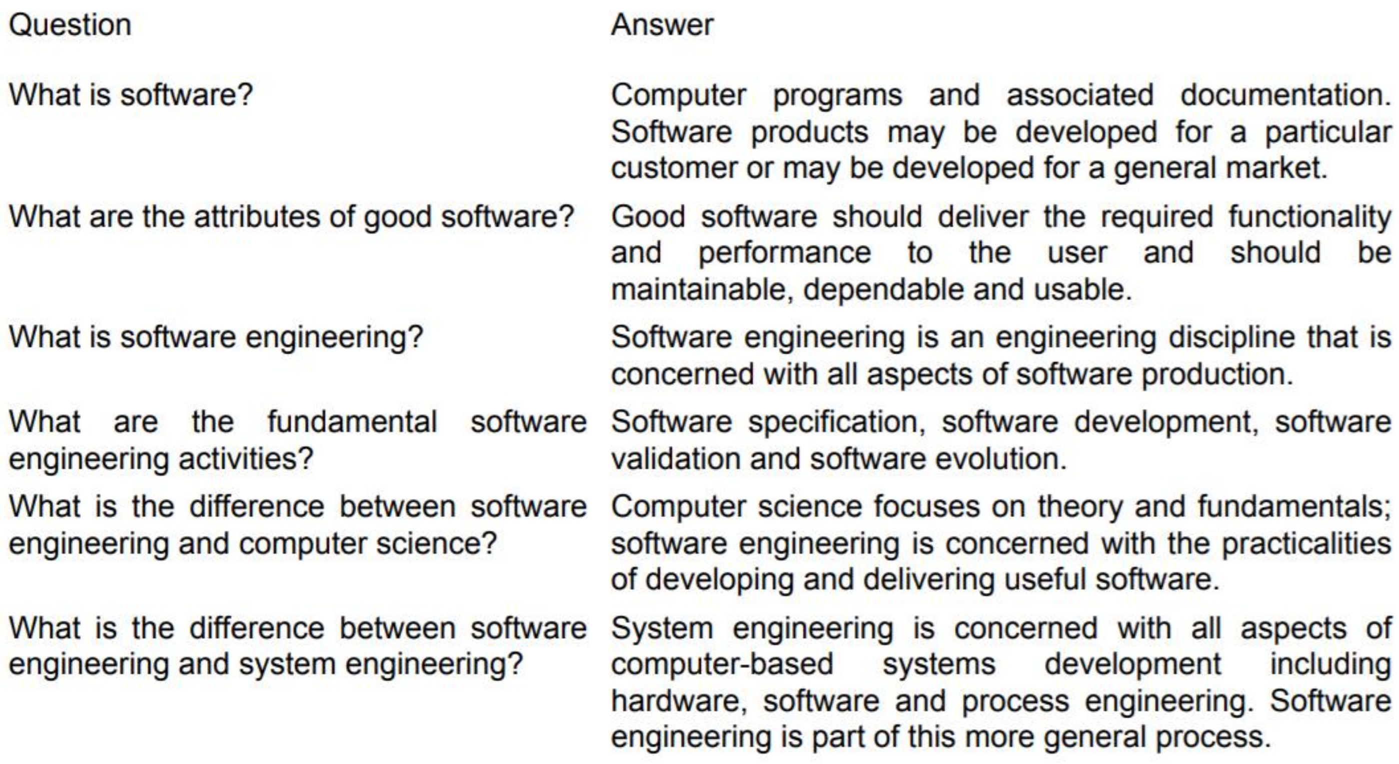
   • Competence – Engineers should not misrepresent their level of competence. They should not knowingly accept work which is out with their competence.

   • Intellectual property rights – Engineers should be aware of local laws governing the use of intellectual property such as patents, copyright, etc. They should be careful to ensure that the intellectual property of employers and clients is protected.

   • Computer misuse – Software engineers should not use their technical skills to misuse other people's computers. Computer misuse ranges from relatively trivial (game playing on an employer's machine, say) to extremely serious (dissemination of viruses).

9. **Define software engineering. Explain any five FAQ of software engineering?**

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



Question	Answer
What is software?	Computer programs and associated documentation. 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 software engineering?	Software engineering is an engineering discipline that is concerned with all aspects of software production.
What are the fundamental software engineering activities?	Software specification, software development, software validation and software evolution.
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.

UNIT – 2

1. What is software process model? Explain the types of software process model.

A software process model is an abstract representation of a process. It presents a description of a process from some particular perspective.

- **The waterfall model** This takes the fundamental process activities of specification, development, validation, and evolution and represents them as separate process phases such as requirements specification, software design, implementation, testing, and so on.

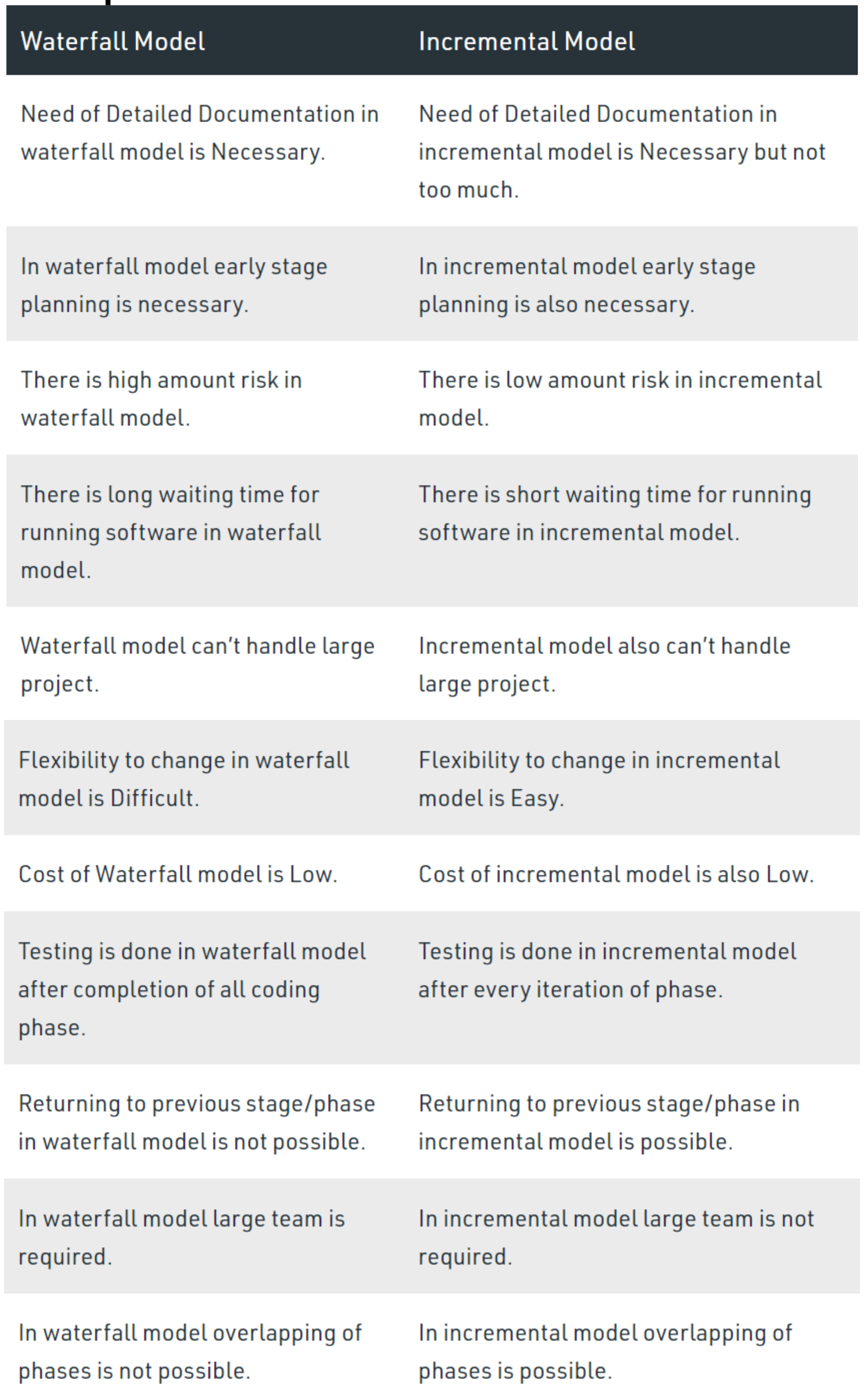
- **Incremental development** This approach interleaves the activities of specification, development, and validation. The system is developed as a series of versions (increments), with each version adding functionality to the previous version.

- **Reuse-oriented software engineering** This approach is based on the existence of a significant number of reusable components. The system development process focuses on integrating these components into a system rather than developing them from scratch.

2. Compare agile methodology with waterfall methodology.

i. Agile is an incremental and iterative approach; Waterfall is a linear and sequential approach.
ii. Agile separates a project into sprints; Waterfall divides a project into phases.
iii. Agile helps complete many small projects; Waterfall helps complete one single project.
iv. Agile introduces a product mindset with a focus on customer satisfaction; Waterfall introduces a project mindset with a focus on successful project delivery.
v. Requirements are prepared every day in Agile, while requirements are prepared once at the start in Waterfall.
vi. Agile allows requirement changes at any time; Waterfall avoids scope changes once the project starts.
vii. Testing is performed concurrently with development in Agile; testing phase comes only after the build phase in Waterfall.
viii. Test teams in Agile can take part in requirements change; test teams in Waterfall do not get involved in requirements change
ix. Agile enables the entire team to manage the project without a dedicated project manager; Waterfall requires a project manager who plays an essential role in every phase.

3. Differentiate between Waterfall model and Incremental development with relevant example.



Waterfall Model	Incremental Model
Need of Detailed Documentation in waterfall model is Necessary.	Need of Detailed Documentation in incremental model is Necessary but not too much.
In waterfall model early stage planning is necessary.	In incremental model early stage planning is also necessary.
There is high amount risk in waterfall model.	There is low amount risk in incremental model.
There is long waiting time for running software in waterfall model.	There is short waiting time for running software in incremental model.
Waterfall model can't handle large project.	Incremental model also can't handle large project.
Flexibility to change in waterfall model is Difficult.	Flexibility to change in incremental model is Easy.
Cost of Waterfall model is Low.	Cost of incremental model is also Low.
Testing is done in waterfall model after completion of all coding phase.	Testing is done in incremental model after every iteration of phase.
Returning to previous stage/phase in waterfall model is not possible.	Returning to previous stage/phase in incremental model is possible.
In waterfall model large team is required.	In incremental model large team is not required.
In waterfall model overlapping of phases is not possible.	In incremental model overlapping of phases is possible.

4. Explain Reuse oriented developmental model with neat diagram? Also discuss the benefits of this model as compared to waterfall model.

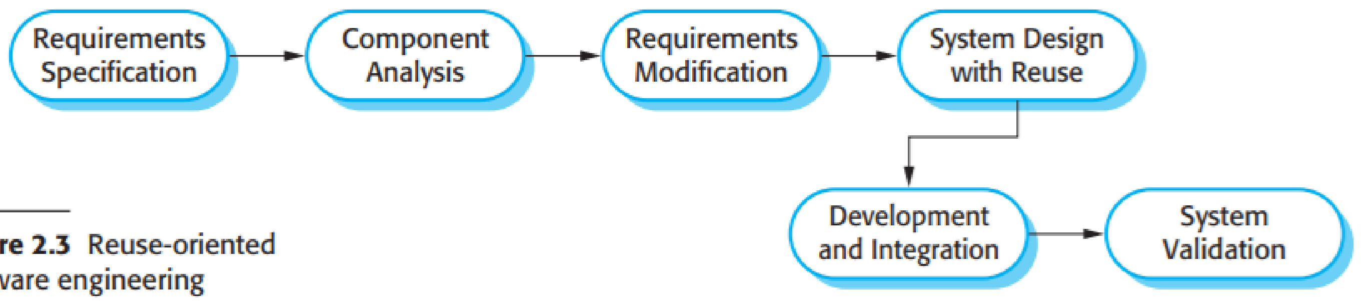

Requirements Specification
Component Analysis
Requirements Modification
System Design with Reuse
Development and Integration
System Validation

Figure 2.3 Reuse-oriented software engineering

- Based on systematic reuse where systems are integrated from existing components or COTS (Commercial-off-the-shelf) systems.
    - Component analysis Given the requirements specification; a search is made for components to implement that specification.
    - Requirements modification During this stage, the requirements are analysed using information about the components that have been discovered. They are then modified to reflect the available components.
    - System design with reuse During this phase, the framework of the system is designed or an existing framework is reused.
    - Development and integration Software that cannot be externally procured is developed, and the components and COTS systems are integrated to create the new system.
- There are three types of software component that may be used in a reuse-oriented process:
    - Web services.
    - Collections of objects.
    - Stand-alone software systems.
- Reuse is now the standard approach for building many types of business system.

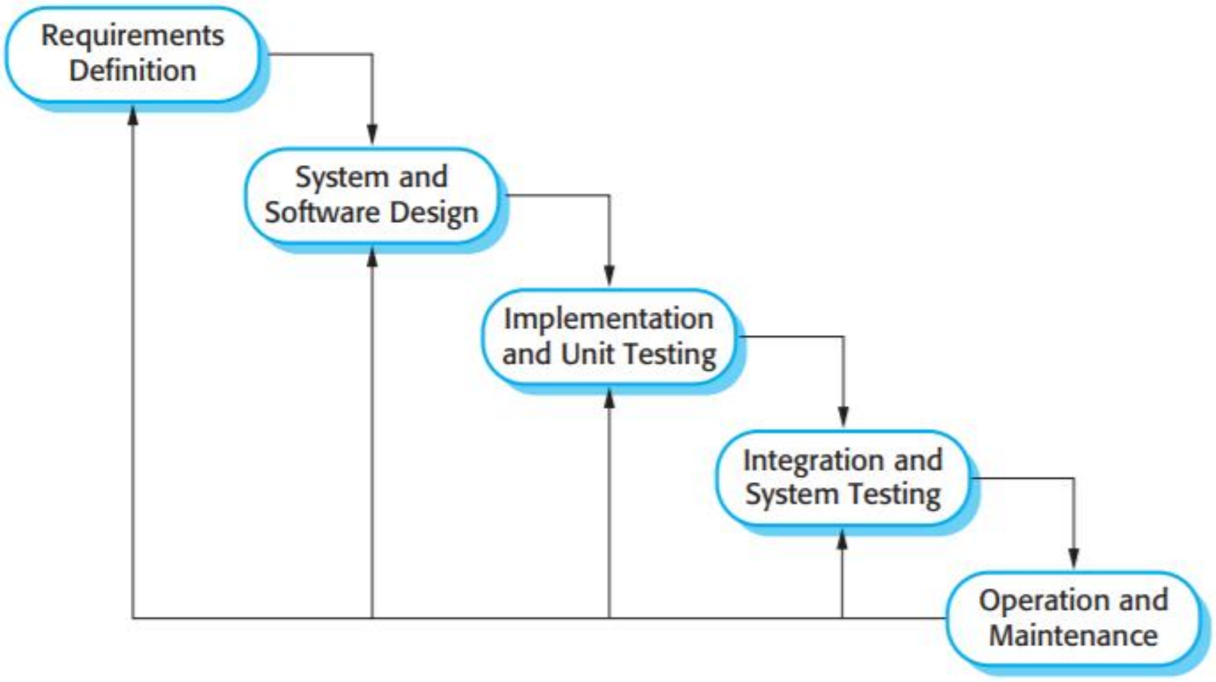
Compared to the waterfall model, incremental development has three important benefits:

i. There is a reduced cost of dealing with changing requirements since the amount of analysis and documentation activities that has to be redone is much less than is required when using the waterfall model.

ii. It is much easier to get customer feedback on the work done. In waterfall-like processes (at least, theoretically) customers first meet the system when it is fully developed and tested and is deployed. If something is not acceptable than it is a real problem (that is the reason why waterfall should only be applied when requirements are well-understood). In iterative development, customers regularly receive a (not fully functional) version of the system that can be commented which provides valuable feedback.

iii. More rapid delivery and deployment is possible so customers can get value out of the software earlier than is possible with the waterfall process. In incremental development it is allowed (and advised) to deliver useful functionalities even in early stages of the development.

5. With a neat diagram explain waterfall model? Explain the problems involved in waterfall model.


Requirements Definition
System and Software Design
Implementation and Unit Testing
Integration and System Testing
Operation and Maintenance

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

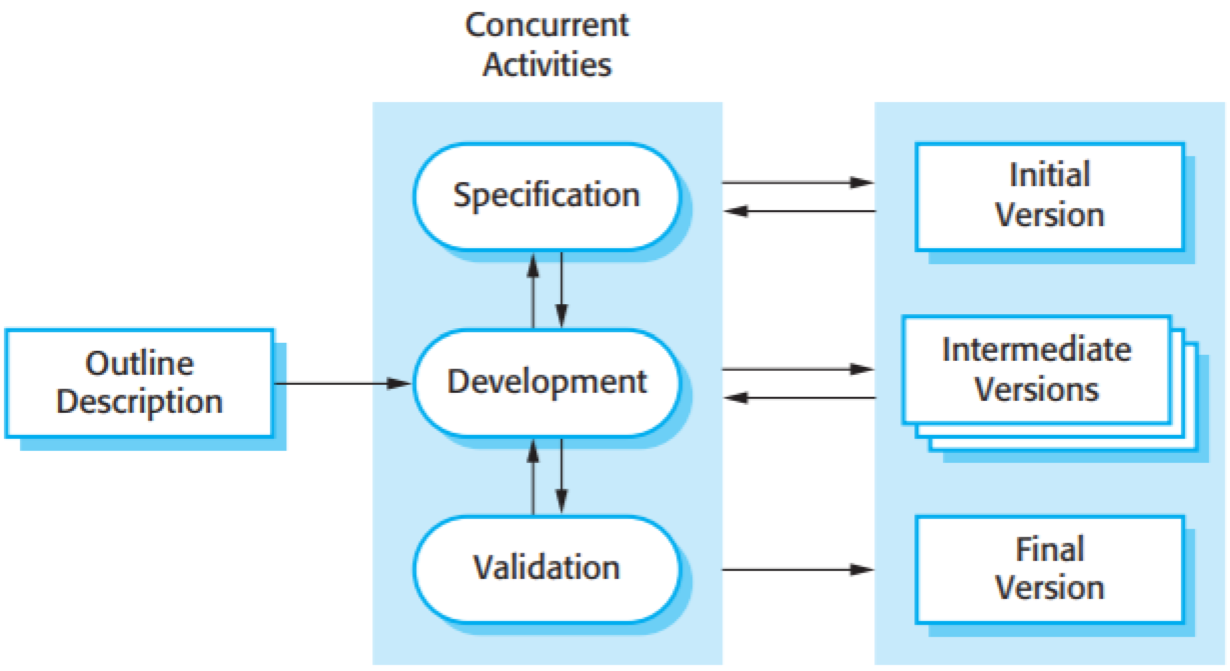
1. Requirement's analysis and definition
   The system's services, constraints, and goals are established by consultation with system users. They are then defined in detail and serve as a system specification.
2. System and software design
   The systems design process allocates the requirements to either hardware or software systems by establishing an overall system architecture. Software design involves identifying and describing the fundamental software system abstractions and their relationships.
3. Implementation and unit testing During this stage
   The software design is realized as a set of programs or program units. Unit testing involves verifying that each unit meets its specification.
4. Integration and system testing
   The individual program units or programs are integrated and tested as a complete system to ensure that the software requirements have been met. After testing, the software system is delivered to the customer.
5. Operation and maintenance Normally
   This is the longest life cycle phase. The system is installed and put into practical use. Maintenance involves correcting errors which were not

discovered in earlier stages of the life cycle, improving the implementation of system units and enhancing the system's services as new requirements are discovered.

Problems Involved in Waterfall Model:

• Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements.
  – Therefore, this model is only appropriate when the requirements are well-understood and changes will be fairly limited during the design process.
  – Few business systems have stable requirements.
• The waterfall model is mostly used for large systems engineering projects where a system is developed at several sites.
  – In those circumstances, the plan-driven nature of the waterfall model helps coordinate the work.

6. With a neat diagram explain Incremental development model also mention the benefits and problems involved in Incremental development model.



- Incremental development is based on the idea of developing an initial implementation, exposing this to user comment and evolving it through several versions until an adequate system has been developed.
- Specification, development, and validation activities are interleaved rather than separate, with rapid feedback across activities.
- It is a fundamental part of agile approaches, is better than a waterfall approach for most business, e-commerce, and personal systems.
- By developing the software incrementally, it is cheaper and easier to make changes in the software as it is being developed.

- Each increment or version of the system incorporates some of the functionality that is needed by the customer.
- The early increments of the system include the most important or most urgently required functionality. This means that the customer can evaluate the system at a relatively early stage in the development to see if it delivers what is required. If not, then only the current increment has to be changed and, possibly, new functionality defined for later increments.

Incremental development benefits
• The cost of accommodating changing customer requirements is reduced.
– The amount of analysis and documentation that has to be redone is much less than is required with the waterfall model.
• It is easier to get customer feedback on the development work that has been done.
– Customers can comment on demonstrations of the software and see how much has been implemented.
• More rapid delivery and deployment of useful software to the customer is possible.
– Customers are able to use and gain value from the software earlier than is possible with a waterfall process.

Incremental development problems
• The process is not visible.
– Managers need regular deliverables to measure progress. If systems are developed quickly, it is not cost-effective to produce documents that reflect every version of the system.
• System structure tends to degrade as new increments are added.
– Unless time and money is spent on refactoring to improve the software, regular change tends to corrupt its structure. Incorporating further software changes becomes increasingly difficult and costly.

7. **Explain why Boehm's spiral model is an adaptable model that can support both change avoidance and change tolerance activities. In practice, this model has not been widely used. Suggest why this might be the case.**

   The spiral model combines change avoidance with change tolerance. It assumes that changes are a result of project risks and includes explicit risk management activities to reduce these risks.
   Each loop in the spiral is split into four sectors:

   a. **Objective setting**
   Specific objectives for that phase of the project are defined. Constraints on the process and the product are identified and a detailed management plan is drawn up. Project risks are identified. Alternative strategies, depending on these risks, may be planned.

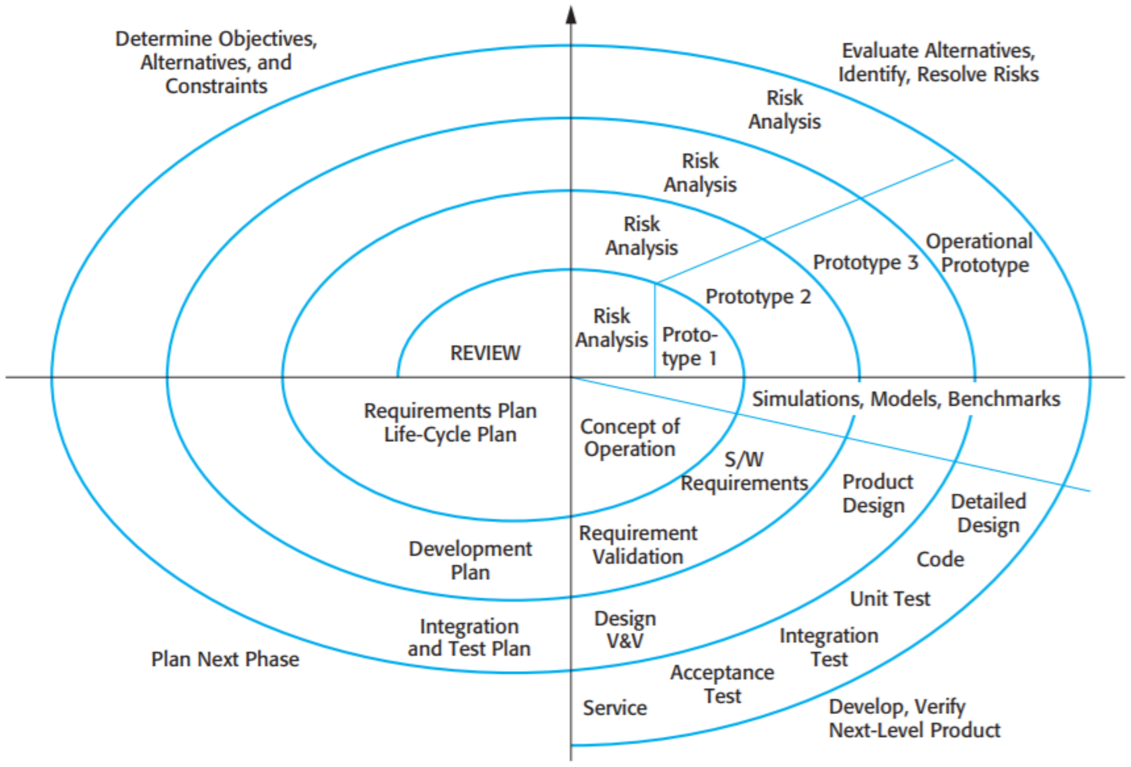
   b. **Risk assessment and reduction**
   For each of the identified project risks, a detailed analysis is carried out. Steps are taken to reduce the risk. For example, if there is a risk that the requirements are inappropriate, a prototype system may be developed.

c. Development and validation

After risk evaluation, a development model for the system is chosen. For example, throwaway prototyping may be the best development approach if user interface risks are dominant. If safety risks are the main consideration, development based on formal transformations may be the most appropriate process, and so on. If the main identified risk is sub-system integration, the waterfall model may be the best development model to use.

d. Planning

The project is reviewed and a decision made whether to continue with a further loop of the spiral. If it is decided to continue, plans are drawn up for the next phase of the project.



-It is not suitable for small projects as it is expensive. It is much more complex than other SDLC models. Process is complex. Too much dependable on Risk Analysis and requires highly specific expertise.

-Spiral Model is not suitable for small projects as it is expensive. Too much dependability on Risk Analysis: The successful completion of the project is very much dependent on Risk Analysis. Without very highly experienced experts, it is going to be a failure to develop a project using this model.

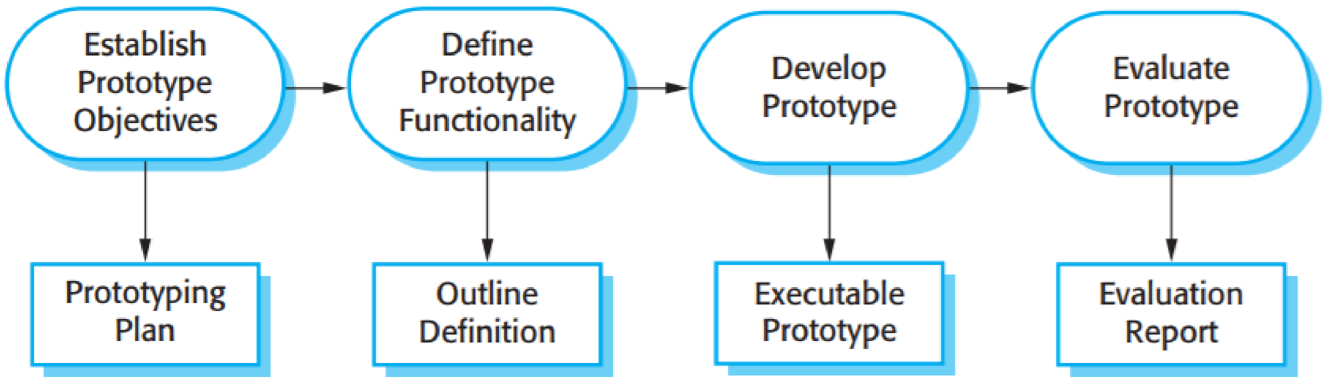
8. Describe on spiral model advantages and disadvantages.

- It is not suitable for small projects as it is expensive.
- It is much more complex than other SDLC models. Process is complex.
- Too much dependable on Risk Analysis and requires highly specific expertise.
- Difficulty in time management. As the number of phases is unknown at the start of the project, so time estimation is very difficult.
- Spiral may go on indefinitely.

- End of the project may not be known early.
- It is not suitable for low-risk projects.
- May be hard to define objective, verifiable milestones. Large numbers of intermediate stages require excessive documentation.

9. What is prototype? Explain the process of prototype development with diagram? Mention the benefits of using prototype.

A prototype is an initial version of a system used to demonstrate concepts and try out design options.



- The objectives of prototyping should be made explicit from the start of the process.
- The same prototype cannot meet all objectives. If the objectives are left unstated, management or end-users may misunderstand the function of the prototype.
- Consequently, they may not get the benefits that they expected from the prototype development.
- The next stage in the process is to decide what to put into and, perhaps more importantly, what to leave out of the prototype system.
- To reduce prototyping costs and accelerate the delivery schedule, you may leave some functionality out of the prototype.
- You may decide to relax non-functional requirements such as response time and memory utilization.
- The final stage of the process is prototype evaluation. Provision must be made during this stage for user training and the prototype objectives should be used to derive a plan for evaluation.
- Users need time to become comfortable with a new system and to settle into a normal pattern of usage. Once they are using the system normally, they then discover requirements errors and omissions.

10. What are the two approaches that may be used to reduce the cost of rework?

- Integrate test design into the requirements process. Ambiguity in defining requirements is the major cause of rework. Deriving test cases from the functional specification is an activity that quickly surfaces these ambiguities. If writing and validation of a specification includes writing of a complete suite of test cases, then the development team is much more likely to get it right the first time.
- Apply model-based technology to functional specifications and testing. These technologies provide comprehensive descriptions of functionality and thus reduce

misunderstandings between the business and the development teams as to expected application behaviour. Some modelling techniques automatically design complete test suites. This means that most defects are caught in the early phases of testing when they are still relatively cheap to fix.

- Invest in test-driven development processes such as agile or incremental development. A major advantage of these approaches is that they focus on reducing functional ambiguities and discovering defects quickly, making rework less expensive.

11. Explain why incremental development is the most effective approach for developing business software systems. Why is this model less appropriate for real-time systems engineering?

• **Incremental development** is established on the indication of developing the primary execution, showing this to customer/user proxy and changing it over some versions until the tolerable system has been developed. This incremental development is a major part of agile approaches and reflects the way that we resolve software system problems.

Generally speaking, developing business software systems are more difficult, software exhaustive and changes happens regularly when business processes or targets are changed. In these cases using incremental development model to achieve business software systems requirements makes more sense.

So, the incremental development is better approach for developing business software systems.

Real-time systems accuracy depends both on an input response and the time taken to produce the output.

• Generally real-time systems contain many hardware modules which are cannot be incremental and not easy to alter.

• Moreover, real-time systems are typically protection critical; it must be built on a good planned procedure.

• In Incremental development model process is not visible and system structure inclines to reduce as new increments are added. Money and time is expended on it to improve the software, regular change towards to corrupt its system structure.

So, incremental development model is less appropriate for real-time systems engineering.

12. Explain why systems developed as prototypes should not normally be used as production systems.

- A general problem with prototyping is that the prototype may not necessarily be used in the same way as the final system.
- The tester of the prototype may not be typical of system users.
- The training time during prototype evaluation may be insufficient. If the prototype is slow, the evaluators may adjust their way of working and avoid those system features that have slow response times.
- When provided with better response in the final system, they may use it in a different way.
- Developers are sometimes pressured by managers to deliver throwaway prototypes, particularly when there are delays in delivering the final version of the software. However, this is usually unwise:
  - It may be impossible to tune the prototype to meet non-functional requirements, such as performance, security, robustness, and reliability requirements, which were ignored during prototype development.
  - Rapid change during development inevitably means that the prototype is undocumented. The only design specification is the prototype code. This is not good enough for long-term maintenance.
  - The changes made during prototype development will probably have degraded the system structure. The system will be difficult and expensive to maintain.
  - Organizational quality standards are normally relaxed for prototype development.

NOTE: Not every answer can be correct.