Question Bank.

Subject: Software Engineering

UNIT-1: INTRODUCTION

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

i)Difference between software engineering and system engineering

->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

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

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? L1,2

->Generic products

- Stand-alone systems that are marketed and sold to any customer who wishes t0buy 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.

- Customized products
  - 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

5M

3. List and Explain the essential attributes of good software? L1,2

->

4M

4. Define Software Engineering? And Explain the four fundamental activities that are common to all software process? L1,2

->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

- Software specification, where customers and engineers define the software that is to be produced and the constraints on its operation.
- Software development, where the software is designed and programmed.
- Software validation, where the software is checked to ensure that it is what the customer requires.
- Software evolution, where the software is modified to reflect changing customer and market requirements.

6M

5.List and explain Issues that affect many different type of software? 6M

->Heterogeneity

- Increasingly, systems are required to operate as distributed systems across networks that include different types of computer and mobile devices.

Business and social change

- Business and society are changing incredibly quickly as emerging economies develop and new technologies become available. They need to be able to change their existing software and to rapidly develop new software.

Security and trust

- As software is intertwined with all aspects of our lives, it is essential that we can trust that software.

5. Define Software Engineering, List and explain essential attributes of goodsoftware? 6M

-> 1. **Functionality:** A good software must be able to do what it was designed to do. The software requirements must guide the design and implementation of the software.

2. **Usability:** The software must be usable; the users must not find it difficult to figure out how a good software works. A good software is user-centered and user-friendly.

3. **Efficiency:** Efficiency means that perform it's operations with minimal time and processing power. A good software uses the least amount of processing power and memory needed to achieve the desired result.

4. **Maintainability:** A good software must evolve with changing requirements.

5. **Security:** A good software must be secure. It should not cause physical or economic damage in the event of a system failure. Unauthorized users must not be allowed access to the system.

6. **Reliability:** A reliable system will rarely fail, and even when it does fail, there are recovery mechanisms in the software to recover from the failure with minimal losses.

6. List and explain any five Software Engineering(ACM/IEEE) Code of Ethics and Professional Practices? L1,2 8M

-> 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.

7. List and Explain the issues related to software engineering ethics? L1,2

Confidentiality

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

Competence

iii) Engineers should not misrepresent their level of competence. They should not knowingly accept work which is outwith their competence.

Intellectual property rights

iv) 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

v) 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).

8. Define software engineering. Explain any five FAQ of software engineering?L1,2 8M

->

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-1: SOFTWARE PROCESSES

1. What is software process model? Explain the types of software process model? 6M

-> A software process model is an abstraction of the software development process. The models specify the stages and order of a process. So, think of this as a representation of the **order of activities** of the process and the **sequence** in which they are performed.

1>Waterfall Model

The waterfall model is a **sequential, plan driven-process** where you must plan and schedule all your activities before starting the project. Each activity in the waterfall model is represented as a separate phase arranged in linear order.

It has the following phases:

- Requirements
- Design
- Implementation
- Testing
- Deployment
- Maintenance

The waterfall model has a rigid structure, so it should be used in cases where the requirements are understood completely and unlikely to radically change.

2>V Model

The V model (Verification and Validation model) is an extension of the waterfall model. All the requirements are gathered at the start and cannot be changed. You have a corresponding testing activity for each stage. For every phase in the development cycle, there is an **associated testing phase.**

The V model is highly disciplined, easy to understand, and makes project management easier. But it isn't good for complex projects or projects that have unclear or changing requirements. This makes the V model a good choice for software where downtimes and failures are unacceptable.

3>Incremental Model

The incremental model divides the system's functionality into **small increments** that are delivered one after the other in quick succession. The most important functionality is implemented in the initial increments.

The subsequent increments expand on the previous ones until everything has been updated and implemented.

Incremental development is based on developing an initial implementation, exposing it to user feedback, and evolving it through new versions. The process' activities are interwoven by feedback.

It is efficient as the developers only focus on what is important and bugs are fixed as they arise, but you need a **clear and complete definition** of the whole system before you start.

The incremental model is great for projects that have loosely-coupled parts and projects with complete and clear requirements.

4>Iterative Model

The iterative development model develops a system through **building small portions** of all the features. This helps to meet initial scope quickly and release it for feedback.

In the iterative model, you start off by implementing a small set of the software requirements. These are then **enhanced iteratively** in the evolving versions until the system is completed. This process model starts with part of the software, which is then implemented and reviewed to identify further requirements.

This makes it easy to identify and **fix any functional or design flaws**. It also makes it easier to manage risk and change requirements.

The deadline and budget may change throughout the development process, especially for large complex projects. The iterative model is a good choice for large software that can be **easily broken down into modules**.

5>RAD Model

The Rapid Application Development (RAD model) is based on iterative development and prototyping with **little planning involved**. You develop functional modules in parallel for faster product delivery. It involves the following phases:

1. Business modeling
2. Data modeling
3. Process modeling
4. Application generation
5. Testing and turnover

But it can be complex to manage. Therefore, the RAD model is great for systems that need to be produced in a short time and have known requirements.

2. Compare agile methodology with waterfall methodology.? L4 6M

- -> Agile is an incremental and iterative approach; Waterfall is a linear and sequential approach.
- Agile separates a project into sprints; Waterfall divides a project into phases.
- Agile helps complete many small projects; Waterfall helps complete one single project.
- Agile introduces a product mindset with a focus on customer satisfaction; Waterfall introduces a project mindset with a focus on successful project delivery.
- Requirements are prepared everyday in Agile, while requirements are prepared once at the start in Waterfall.
- Agile allows requirement changes at any time; Waterfall avoids scope changes once the project starts.
- Testing is performed concurrently with development in Agile; testing phase comes only after the build phase in Waterfall.
- Test teams in Agile can take part in requirements change; test teams in Waterfall do not get involved in requirements change
- 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.? L4 8M

->

INCREMENTAL MODEL	**WATERFALL MODEL**
Detailed documentation is not strictly Required in Incremental model.	Detailed documentation is strictly Required in Waterfall model.
Returning to previous stage/phase is allowed in waterfall model.	Returning to previous stage/phase is never advice in waterfall model.
Testing is done after every iteration of phase in incremental model.	Testing is done after completion of all coding phase in waterfall model.
Low risk in incremental model.	More risk in waterfall model.
Incremental model can't handle large project.	Waterfall model can't handle large project.
Early stage planning(not Extremely) is necessary in Incremental model.	Early stage planning(Extremely) is necessary in Waterfall model.
Delivery time is short.	Delivery time is not short.
There are multiple development cycles take place in incremental model.	There is only one development cycles in waterfall model.
Cost of incremental model is Low.	Cost of Waterfall model is Low.
Flexibility to change in incremental model is Easy.	Flexibility to change in waterfall model is difficult.
Overlapping of phases is possible in waterfall model.	Overlapping of phases is not possible in waterfall model.
Large team is not required in waterfall model.	Large team is require in waterfall model.

In the olden days, Waterfall model was used to develop enterprise applications like Customer Relationship Management (CRM) systems, Human Resource Management Systems (HRMS), Supply Chain Management Systems, Inventory Management Systems, Point of Sales (POS) systems for Retail chains etc.

4. Explain Reuse oriented developmental model with neat diagram? Also discuss the benefits of this model as compared to waterfall model? L2 10M

5. With a neat diagram explain waterfall model? Explain the problems involved in waterfall model.? L2

   The waterfall model is a **sequential, plan driven-process** where you must plan and schedule all your activities before starting the project. Each activity in the waterfall model is represented as a separate phase arranged in linear order.

   10M

http://tryqa.com/what-is-waterfall-model-advantages-disadvantages-and-when-to-use-it/

6. With a neat diagram explain Incremental development model also mention the benefits and problems involved in Incremental development model.? L2 10M

   The incremental model divides the system's functionality into **small increments** that are delivered one after the other in quick succession. The most important functionality is implemented in the initial increments.

   The subsequent increments expand on the previous ones until everything has been updated and implemented.

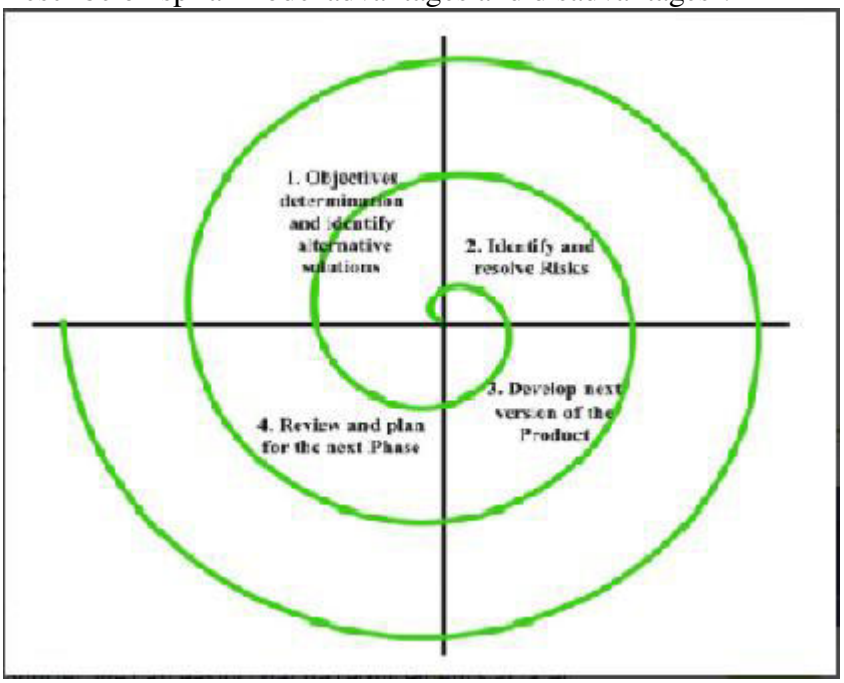
   http://tryqa.com/what-is-incremental-model-advantages-disadvantages-and-when-to-use-it/

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. ? L2 8M

-> Boehm's spiral model is an adaptable model. It can support both change avoidance and change tolerance. Because, this model assumes that changes are a result of project risks and includes explicit risk management activities to reduce these risks by each loop in the spiral. Each loop in the spiral is split into four sectors.

1. Objective setting: Specific objectives for the phase are identified.

8. Describe on spiral model advantages and disadvantages ?L2



-> **Advantages of Spiral Model:**

1. Software is produced early in the software life cycle.
2. Risk handling is one of important advantages of the Spiral model, it is best development model to follow due to the risk analysis and risk handling at every phase.
3. Flexibility in requirements. In this model, we can easily change requirements at later phases and can be incorporated accurately. Also, additional Functionality can be added at a later date.
4. It is good for large and complex projects.
5. It is good for customer satisfaction. We can involve customers in the development of products at early phase of the software development. Also, software is produced early in the software life cycle.
6. Strong approval and documentation control.
7. It is suitable for high risk projects, where business needs may be unstable. A highly customized product can be developed using this.

Disadvantages of Spiral Model:

1. It is not suitable for small projects as it is expensive.
2. It is much more complex than other SDLC models. Process is complex.
3. Too much dependable on Risk Analysis and requires highly specific expertise.
4. Difficulty in time management. As the number of phases is unknown at the start of the project, so time estimation is very difficult.
5. Spiral may go on indefinitely.
6. End of the project may not be known early.
7. It is not suitable for low risk projects.
8. May be hard to define objective, verifiable milestones. Large numbers of intermediate stages require excessive documentation.

5M

9. What is prototype? Explain the process of prototype development with diagram? Mention the benefits of using prototype? L2 10M

https://www.guru99.com/software-engineering-prototyping-model.html#:~:text=Prototyping%20Model%20is%20a%20software,are%20not%20known%20in%20detail.

10. What are the two approaches that may be used to reduce the cost of rework?L2 5M

-> The following are proven approaches to reducing rework:

1. 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.
2. 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 behavior. Some modeling 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.
3. 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? L2 8M

-> **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.

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.

12. Explain why systems developed as prototypes should not normally be used as production systems? L2 7M

- ->Prototypes should be discarded after development as they are not a good basis for a production system:
  - It may be impossible to tune the system to meet non-functional requirements;
  - Prototypes are normally undocumented;
  - The prototype structure is usually degraded through rapid change;
  - The prototype probably will not meet normal organisational quality standards.