Review Questions on Software Engineering

Chapter 1: Introduction

1. What are the essential attributes of good software?

Maintainability, dependability and security, efficiency and acceptability

Run Effectively and Efficientlly, Easy to be Extended, Easy to be Understood

2. What are the two fundamental types of software product?

Generic products that are designed to meet the needs of many different customers.

Customised products designed to meet the specific needs of a single customer.

3. What is software engineering?

An engineering discipline concerned with all aspects of software production from specification to system maintenance.

4. What are the four fundamental activities in software processes?

Software specification, software development, software validation and software evolution.

5. What is the distinction between computer science and software engineering?

Computer science is concerned with theories and methods of computers and software systems; software engineering is concerned with the practice of software production.

6. What are the 3 general issues that affect many different types of software?

Heterogeneity. Software may have to execute on several different types of system.
Business and social change, which drives requirements for software change.
Security and trust – our software systems have to be secure against external and internal threats so that we can trust those systems.

7. List 5 different types of software application.

Any 5 from stand-alone products, interactive transaction-based systems, embedded control systems, batch processing systems, entertainment systems, systems for modelling and simulation, data collection systems, systems of systems.

8. What software engineering fundamentals apply to all types of software systems?

a. Systems should be developed using a managed and understood development process.
b. Dependability and performance are key system characteristics
c. Understanding and managing the software specification and requirements are important.
d. Effective use should be made of available resources.

9. What are three key characteristics of the engineering of web-based software engineering?

Software reuse is the principal approach for constructing web-based systems, requirements for those systems cannot be completely specified in advance, User interfaces are constrained by the capabilities of web browsers.

10. What is a software engineering code of ethics?

A set of principles that set out, in a general way, standards of expected behaviour for professional software engineers.

Chapter 2: Software processes

1. What are the fundamental activities that are common to all software processes?

Software specification
Software design and implementation
Software validation
Software evolution

2. List the 3 generic process models that are used in software engineering?

The waterfall model
Incremental development
Reuse-oriented software engineering

3. Why are iterations usually limited when the waterfall model is used?

The waterfall model is a document-driven model with documents produced at the end of each phase. Because of the cost of producing and approving documents, iterations and costly and involve significant rework. Hence they are limited.

4. What are the three benefits of incremental development, compared to the waterfall model?

(a) The cost of accommodating changes to customer requirements is reduced.

(b) It is easier to get customer feedback on development work that has been done.

(c) More rapid delivery and deployment of useful software to the customer is possible.

5. What are the development stages in reuse-based development?

Component analysis
Requirements modification
System design with reuse
Development and integration

6. What are the principal requirements engineering activities?

Feasibility study
Requirements elicitation and analysis
Requirements specification
Requirements validation

7. Why is it increasingly irrelevant to distinguish between software development and evolution?

Few software systems are now completely new and a more realistic model of software development is of an iterative process that lasts for the lifetime of the software.

8. What are the advantages of using incremental development and delivery?

Early delivery of critical functionality to the customer
Early increments serve as prototypes to explore requirements
Lower risk of overall project failure
More extensive testing of critical customer functionality

9. What are the 4 sectors in each loop in Boehm's spiral model?

Objective setting
Risk assessment and reduction
Development and validation
Planning

10. What are the six fundamental best practices in the RUP?

Develop software iteratively
Manage requirements
Use component-based architectures
Visually model software
Verify software quality
Control changes to software

Chapter 3: Agile Software Development

1. What are the shared characteristics of different approaches to rapid software development?

The processes of specification, design and implementation are inter-leaved.

The system is developed and delivered as a series of versions.

User interfaces are often developed using an interactive development system that supports rapid UI development.

2. For what types of system are agile approaches to development particularly likely to be successful?

Small and medium-sized software product development.

Custom software development in an organization where there is a clear commitment from customers to become involved in the development process.

3. List the 5 principles of agile methods.

Customer involvement,

Incremental delivery,

People not process,

Embrace change,

Maintain simplicity.

4. List 4 questions that should be asked when deciding whether or not to adopt an agile method of software development.

Any 4 from those below. Others are also possible (see Ch 3)

Is an incremental delivery strategy realistic?

What type of system is being developed?

What is the expected system lifetime?

How is the development team organized?

Is the system subject to external regulation?

How large is the system that is being developed?

5. What are three important characteristics of extreme programming?

Requirements expressed as scenarios,

Pair programming,

Test-first development.

6. What is test-first development?

When a system feature is identified, the tests of the code implementing that feature are written before the code. Tests are automated and all tests are run when a new increment is added to the system.

7. What are the possible problems of test-first development?

Programmers may take short-cuts when developing tests so that the system tests are incomplete.

Some tests can be difficult to write incrementally.

It is difficult to estimate the completeness of a test set.

8. Briefly describe the advantages of pair programming.

It supports the idea of common ownership and responsibility for the code.

It serves as an informal code review process.

It helps support refactoring.

9. What is a Scrum sprint?

A short (3-4 weeks) planning unit in which work to be done is assessed, features are selected for development, the software is implemented and delivered to system stakeholders.

10. What are the barriers to introducing agile methods into large companies?

Project managers may be reluctant to accept the risks of a new approach.

The established quality procedures in large companies may be incompatible with the informal approach to documentation in agile methods.

The existing teams may not have the high level of skills to make use of agile methods.

There may be cultural resistance if there is a long history of plan-driven development in the company.

Chapter 4: Requirements Engineering

1. What are user requirements and system requirements?

User requirements are statements in a language that is understandable to a user of what services the system should provide and the constraints under which it operates.

System requirements are more detailed descriptions of the system services and constraint, written for developers of the system.

2. What is the distinction between functional and non-functional requirements?

Functional requirements define what the system should do. Non- functional requirements are not directly concerned with specific system functions but specify required system properties or place constraints on the system or its development process.

3. List 3 types of non-functional requirement?

Product requirements, that specify or constrain the software's behaviour.

Organisational requirements, are general requirements derived from policies and procedures in the customer's organization.

External requirements, which cover all requirements derived from factors external to the system and its development process.

4. What is the software requirements document?

The official document that defines the requirements that should be implemented by the system developers.

5. What is the distinction between the terms' shall' and 'should' in a user requirements document, which is written in natural language?

'Shall' normally indicates a mandatory requirement

'Should' indicates a desirable but not essential requirement.

6. What are the main advantages of using a standard format to specify requirements?

All requirements have the same format so are easier to read,

The definition of form fields mean that writers are less likely to forget to include information

Some automated processing is possible.

7. What are the principal stages of the requirements engineering process?

1. Requirements elicitation and analysis

2. Requirements specification

3. Requirements validation

8. Give 5 reasons why eliciting requirements is difficult?

1. Stakeholders don't know what they want

2. Stakeholders use their own language that requirements engineers may not understand.

3. Stakeholder requirements may conflict

4. Political factors may influence the system requirements

5. The business environment may change during elicitation.

9. What should be included in a scenario.

1. A description of what's expected when the scenario starts

2. A description of the normal flow of events

3. A description of what can go wrong and how to handle it

4. Information about concurrent activities

5. A description of the system state when the scenario finishes

10. What is a use-case?

A use-case identifies a typical interaction with a system and the actors (human or computer) involved in that interaction.

11. What is ethnography and how is it used in requirements elicitation?

Ethnography is an observational technique where an analyst spends a period of time observing work and noting how the participants carry out their tasks. It is particularly useful in identifying essential cooperation in work processes.

12. What checks should be applied during requirements validation?

1. *Validity checks*
2. *Consistency checks*
3. *Completeness checks*
4. *Realism checks*
5. *The verifiability of the requirements should be assessed.*

13. List three requirements validation techniques?

1. *Requirements reviews*
2. *Prototyping*
3. *Test-case generation*

14. What is requirements management?

The process of managing changes to requirements during requirements specification and after the system has gone into use.

15. What are the stages in the requirements change management process?

a. *Problem analysis and change specification*
b. *Change analysis and costing*
c. *Change implementation*

Chapter 5: System modeling

1. What perspectives may be used for system modelling?

An external perspective

An interaction perspective

A behavioural perspective

A structural perspective.

2. What UML diagram types may be used to represent the essential features of a system?

Activity diagrams

Use case diagrams

Sequence diagrams

Class diagrams

State diagrams

3. What is described in a context model?

The immediate external environment of the system defining the system's context and the dependencies that a system has on its environment. The context model shows what is outside of the system boundary.

4. How are activity diagrams used in describing the context of use of a system?

Activity diagrams may be used to describe the business processes in which the system is used and the other systems which are also used in these processes.

5. What are the principal components of a textual use-case description?

The actors involved

A description of the interactions

The data that is exchanged

The stimulus that triggers the use case

The response of the system

Comments and other information

6. What is illustrated in a UML sequence diagram?

A sequence of interactions between the actors in a system's environment and the objects in the system itself. The sequence of interactions describes the implementation of a system feature or function.

7. How is generalization used to simplify the models of a system with many similar objects?

Assuming that the similar objects have attributes and methods in common, these common attributes and methods are associated with a 'super-class' which generalizes all of the objects sharing these attributes/methods. The specific object classes only declare the attributes/methods specific to that class and they inherit the general attributes/methods from the super-class.

8. What is the basic assumption that underlies event-driven modelling?

That the system can be represented as a model with a finite number of discrete states and external and internal events trigger a transition from one state to another.

9. What are the claimed benefits of model-driven engineering?

Engineers can work at a high level of abstraction without concern for implementation details.

Errors are reduced and the design and implementation process is speeded up.

By using powerful generation tools, implementations of the same system can be automatically generated for different platforms.

10. What are the three types of abstract system model that are recommended by the MDA method?

A computation-independent model (CIM)

A platform-independent model (PIM)

One or more platform-specific models (PSMs)

Chapter 6: Architectural design

1. What are the advantage of explicitly designing and documenting a software architecture?

It improves stakeholder communications

It encourages a detailed analysis of the system

It helps with large-scale reuse.

2. What are the two ways in which an architectural model of a system may be used?

As a means of facilitating discussion about the most appropriate architecture for a system.

As a means of documenting the architecture of an existing or an intended system.

3. List 4 fundamental questions that should be addressed in architectural design?

Is there a generic application architecture that can be used?

How will the system be distributed?

What architectural style or styles are appropriate?

How should the system be structured?

What control strategy should be used?

4. What are the fundamental architectural views proposed in Krutchen 's 4+ 1 model?

A logical view that shows the key abstractions of the system.

A process view that shows the interacting processes in the system

A development view that shows how the system is decomposed for development

A physical view that shows the distribution of software on the system hardware

5. What is an architectural pattern?

A stylized abstract description of good practice in architectural design that has been tried and tested in different systems and environments. The pattern should include information on when it is and is not appropriate to use that architectural design.

6. What is the fundamental characteristic of a repository architecture?

All shared data is held in a central database that can be accessed by all sub-systems.

7. What is the most important advantage of a client-server architecture?

This is a distributed architecture so that it is possible to provide services on different computers. It is easy to add a new server or upgrade existing servers without disrupting the system

8. Briefly describe pipe and filter architecture?

A system is decomposed into a set of functional transformations that consume inputs and produce outputs. Data flows from one function to another (the pipeline) and is transformed as it passes through the sequence.

9. What are transaction-processing applications?

Database-centered applications that process user requests for information and update the information in the database. They are organized so that transactions cannot interfere with each other and the integrity of the database is maintained.

10. What are the principal functions of the 4 layers in a generic information system architecture?

(1) User interface

(2) User communications, authentication and authorization

(3) Information retrieval and modification

(4) Database and transaction management

Chapter 7: Design and implementation

1. What are the 5 key activities in an object-oriented design process?

   Understand and define the context and use of the system.

   Design the system architecture

   Identify the principal objects in the system

   Develop design models

   Specify object interfaces

2. What do you understand by the system context and interaction model?

   The system context is a static model of the other systems in the environment of the system being designed.

   An interaction model is a dynamic model that describes how the system being designed interacts with its environment.

3. Briefly describe 3 approaches that may be used to identify object classes?

   Grammatical analysis identifying nouns and verbs.

   Identify tangible things in the application domain.

   Use scenario-based analysis.

4. Why is it important to specify the interfaces of components that are being developed by a software engineering team?

Interfaces have to be specified so that objects and sub-systems can be designed in parallel. Once an interface has been specified, the developers of other parts of the system may assume that the interface will be implemented.

5. What do Gamma et al. suggest are the four essential elements of a design pattern?

A meaningful name

A description of the problem and when the pattern can be applied

A solution description, which shows the components in the solution and their relationships.

A statement of the consequences of applying the pattern.

6. How do design patterns contribute to reuse?

Patterns and pattern languages are ways to describe best practices, good designs and capture experience in a way that is possible for others to reuse.

7. What are the 4 levels at which software reuse is possible?

The abstraction level where knowledge of successful abstractions is reused.

The object level where objects classes and methods from libraries are reused.

The component level where collections of objects are reused.

The system level where entire application systems are reused.

8. What are the principal aims of software configuration management?

To support system integration so that all developers can access the project code and documents in a controlled way, find out what components have been changed and compile and link components to create a system.

9. What are essential tools in a software development platform?

An integrated compiler and syntax-directed editing systems

A language debugger

Graphical editing tools for UML models

Testing tools that can automatically run program tests

Project support tools for code control

10. Briefly describe the idea of open-source development.

In an open-source development, the source code of a software system is made publicly available and volunteers participate in the further development of the system. Any contributor to an open source development may fix bugs and add new features to a system.

Chapter 8: Testing

1. What is the distinction between validation and verification?

Validation: Are we building the right product?

Verification: Are we building the product right?

2. What are the advantages of inspections over testing?

Inspections can discover many errors. In testing, one error may mask another.

Incomplete versions of a system can be inspected.

Inspections can consider broader quality attributes as well as program defects.

3. Briefly describe the three principal stages of testing for a commercial software system

Development testing, where the system is tested to discover bugs and defects

Release testing where the system is tested to check that it meets its requirements

User testing where the system is tested in the user's environment.

4. What tests should be included in object class testing?

Tests for all operations in isolation,

Tests that set and access all object attributes,

Tests that force the object into all possible states.

5. What guidelines does Whittaker suggest for defect testing?

Chose inputs that force all error messages to be generated,

Design inputs that might cause buffers to overflow,

Repeat the same input numerous times,

Force invalid outputs to be generated

Force computation results to be too large or too small.

6. What is an equivalence partition? Give an example.

A class of inputs or outputs where it is reasonable to expect that the system will behave the same way for all members of the class. For example, all strings with less than 256 characters.

7. What are the three important classes of interface errors?

Interface misuse,

Interface misunderstanding,

Timing errors.

8. What should be the principal concerns of system testing?

Testing the interactions between the components and objects that make up the system.

Testing reusable components and systems to check that they work as expected when integrated into the system.

9. Briefly summarize the test-driven development process

a. Identify increment of functionality required

b. Design tests for this functionality and implement as executable programs.

c. Run test along with other implemented tests. The test will fail.

d. Implement the functionality and re-run the test. Iterate until the test works.

e. Move on to implement the next chunk of functionality

10. What is scenario testing?

Scenario testing is an approach to release testing where you write a story describing how a system may be used and design tests based on the sequence of events in the scenario.

11. What is stress testing and why is it useful?

Increasing the load on a system beyond its design limits to check how the system performs in this situation. This tests the failure behaviour of the system and may reveal defects that are only apparent when the system has to handle a heavy load.

12. What are the three types of user testing?

Alpha testing, where users work with the development team to test the software as it is being developed.

Beta testing where the software is released to selected users for testing before the formal system release

Acceptance testing, where customers test a system to check that it is ready for deployment.

Chapter 9: Evolution

1. Why is software evolution important?

Organisations are completely dependent on their software systems and they are critical business assets. They must invest in evolution so that these systems remain useful and maintain their value.

2. What are the stages in the system evolution process and what triggers that process?

The process is triggered by change requests. Process stages are:

Impact analysis

Release planning

Change implementation

System release

3. Why might it sometimes be necessary to bypass the normal change management system and make urgent changes to a system?

To repair a serious system fault.

To change the system to cope with unexpected changes in the system's operating environment.

To cope with unexpected business change.

4. What are Lehman’s Laws and how were they derived?

Lehman’s ‘laws’ are a set of hypothesis which, it is claimed, set out invariants for system change. .They were derived from studies of the growth and evolution of a number of large software systems.

5. What are the three different types of software maintenance and how is effort distributed across these maintenance types?

Maintenance to repair software faults (17%),

Maintenance to adapt the software to a different environment (18%),

Maintenance to add to or modify the system’s functionality (65%).

6. What factors should be assessed to understand the relationship between a system and its environment?

The number and complexity of system interfaces,

The number of inherently volatile system requirements,

The business processes in which the system is used.

7. What process metrics might be used to assess maintainability?

Number of requests for corrective maintenance,

Average time required for impact analysis,

Average time taken to implement a change request,

Number of outstanding change requests.

8. What are the principal systems re-engineering activities?

Source code translation,
Reverse engineering,
Program structure improvement,
Program modularisation,
Data re-engineering

9. What are the strategic options for legacy system evolution?

Scrap the system completely,
Leave the system unchanged and continue maintenance,
Re-engineer the system to improve maintainability,
Replace all or part of the system with a new system.

10. List four important factors used to assess applications for evolution.

Any four from:
Understandability, Documentation, Data, Performance,
Programming language, Configuration management, Test data,
Personnel skills

Chapter 10: Sociotechnical systems

1. What is the difference between the business process layer and the organizational layer in the sociotechnical systems stack.

   The business process layer is concerned with the specific business processes that are used to support business functions. The organizational layer is concerned with more general strategic issues such as business rules and compliance, organizational policies, etc.

2. What is the difference between a technical and a sociotechnical system?

   Technical systems include hardware and software components but not procedures and processes. Sociotechnical systems are self-aware and include defined operational processes and procedures. People are an inherent part of sociotechnical systems.

3. What are emergent properties?

   System properties that only become apparent when all of the system components have been integrated. In other words, properties that are characteristic of the system as a whole.

4. What are three influences on the reliability of a system?

Hardware reliability

Software reliability

Operator reliability

5. Why are sociotechnical systems non-deterministic?

Partly because they include people whose behaviour may change from day to day and partly because changes to the hardware, software and data in these systems is so frequent that the consequences of these changes are impossible to ascertain.

6. What is a wicked problem?

A problem that is so complex with so many related entities that no definitive problem specification can be produced. The true nature of the problem only emerges when the solution is developed.

7. What are the three principal stages of systems engineering?

1. Procurement
2. Development
3. Operation

8. What are the main drivers for system procurement decisions?

1. *The state of other organizational systems.*
2. *The need to comply with external regulations.*
3. *External competition*
4. *Business re-organization*
5. *Available budget.*

9. Why are plan-driven (rather than agile) processes used in systems engineering?

Because different parts of the system are developed by different groups at the same time and plans are needed to coordinate their activities.

10. What are latent conditions and active failures?

Latent conditions are vulnerabilities and weaknesses in a system that, at some stage, may contribute to system failure.

Active failures are some operational event or human error that triggers a sequence of events that could lead to system failure.

Chapter 11: Dependability and Security

1. Give three reasons why a system's dependability is more important than its detailed functionality.

System failures affect a large number of people, Users may reject systems that are unreliable, unsafe or insecure, System failure costs may be very high, Undependable systems may cause information loss.

2. What are the four principal dependability properties?

Reliability, availability, safety and security

3. List two other system properties that are sometimes considered to be dependability properties.

Any two from:

Repairability, maintainability, survivability, error tolerance.

4. Briefly define what availability means?

Availability is the ability of a system to deliver services when requested OR

The probability that a system will be up and running and able to deliver useful services to users at any given time.

5. Explain how a relatively unreliable system can provide a high level of availability.

Reliability is concerned with the correct delivery of system services. Availability is concerned with the system's operational state. A system can be available but unreliable so long as system failures can be quickly detected and corrected before they affect the normal usage of the system.

6. Explain the difference between a system fault and a system failure.

A fault is an internal system condition that can lead to an erroneous system state. A failure is an externally observed deviation from expected system behaviour.

7. What is the most important difference between the two classes of safety-critical system?

In a primary safety-critical system, a failure can lead directly to an accident. In a secondary safety critical system, a failure can lead to the introduction of faults into another system, whose failure can lead to an accident.

8. What is the distinction between a hazard and an accident?

A hazard is a condition of the system that has the potential to cause an accident. An accident is an unplanned event or sequence of events that results in human death or injury or other damage to the system's environment

9. What are the three principal threats to the security of a system?

Threats to the confidentiality of a system and its data.

Threats to the integrity of a system and its data.

Threats to the availability of a system and its data.

10. What are the three controls may be put in place to enhance system security?

Vulnerability avoidance

Attack detection and neutralization

Exposure limitation and recovery