

## CS 422 Software Engineering Principles

Study Questions (Ch1-10 Sommerville)

Including some miscellaneous (Miscel) materials covered in lecture or homework (HW)

1. (Miscel) What is the difference between fault avoidance and fault tolerance?

**Fault avoidance** (a process oriented concept) seeks to prevent faults from being introduced into the software. **Fault tolerance** (a product oriented concept) accepts faults in a limited capacity and masks their manifestation (i.e., failures).

2. (HW Article) What is the difference between black box testing and white box testing?

In **black-box** testing, the specification supplies the organizing information for systematic testing. What is often called functional testing isolates a collection of actions (functions) a program should perform and requires test data to exercise each function. Interestingly, functional testing directly tries what is expected of the software. You can plan tests as soon as you have the specification, which is a plus. **White-box testing** and structural testing are essentially the same thing. They seek to exercise the code by looking at the structure of the code (e.g., tests force the execution of each statement).

3. (HW Article) Explain the terms structural, random testing and equivalence partitioning.

**Structural testing** seeks to exercise the code by looking at the structure of the code and its tests force the execution of each statement.

**Random testing** requires many more orders of magnitude more test points than current practice. Its based on the premise that uniformly distributed state space based testing should be much better than partition testing at establishing confidence in apparently defect-free software

Equivalence partitioning divides the input space into “equivalent” pieces that uniformly cover (from a statistical basis) all possible classes of paths.

4. (Miscel but see ¶24.4 in Sommerville) What is *Cleanroom* software development?

Each part of the development process is handled by a distinct groups of developers (in a customer supplier relationship).

5. (Miscel) What are the major components of a context diagram (give an example if you need to)?

It must define the top-level environment and interfaces of the system under development. According to the Email I sent out, the components of the context diagram are: (1) Your project is in the center and the bubbles surrounding it are the context. (2) External (to your part of the project) entities such as users or other components (show the dependencies [i.e., inputs and outputs]); and (3) Human interfaces (if applicable).

6. (Miscel but see Chap 22) Define the terms (a) Verification and (b) Validation.

Verification - is the product built right?

Validation - is it the right product?

7. (Miscel Video HW) Define what is a process (name three things and what are they).

Activities (things that are done); Products (inputs and outputs); Sequencing (relationships among activities and products).

8. (Miscel) What are standards and why are they important in software engineering?

Standards are a recommended practice that must be unambiguous and precise. They describe recommended approaches for the particular phase or artifact which is underway. Actually, the existence of a standard does not imply that there are no other ways to produce, test, measure, purchase, market or provide goods or services related to the scope of said standard.

9. (Miscel) What are the three sections of a progress report. Why are progress reports important in software engineering?

There are three basic sections in the progress report that we are using in this class: (1) Accomplishments, (2) Plans, and (3) Problems / Issues. The report is important as a tool used by the project participants and management to gauge the status of work relative to the established schedule. It is also an important communication tool. Project

10. (Chap 1) Fill in the blanks of the following sentences:

Software Engineering is concerned with theories, methods/practices and tools which are needed to develop the software for computer systems (e.g., aerospace, avionics, telecommunications, government, health care, etc.). Different from other engineering disciplines because it is not constrained by materials governed by physical laws or by manufacturing processes. Software Engineers model parts of the real/physical world in software. The goal is to produce practical software solutions in a cost/efficient effective way. Products that are reliable, robust, useable/dependable/efficient, flexible and maintainable.

11. (Chap 1) What are the four major partitions in the Spiral model of the Software Life Cycle?  
Chap 1, Pg.14

12. (Chap 1) Name three strategies that are used for risk resolution? Chap 1, Pg. 16

13. (Chap 1) Describe the five Process models reviewed in Sommerville and their relation to process visibility? Pg. 18, Fig 1.10

14. (Chap 2 + Miscel) What are the three major facets (influences) in system reliability engineering? Pg. 40

15. (Chap 2) Name the seven steps in System Engineering, and identify the step that uses the 'Big Bang' approach. Fig. 2.5, pg. 33

16. (Chap 3) In project management, what are five examples of outputs from project planning?  
Fig. 3.1
17. (Chap 3) What is an important output from developing a activity network? Critical path, Pg. 54
18. (Chap 3) Define what is a project milestones. Pg. 58, key pts.
19. (Chap 3) What is the critical distinction between a project milestone and a deliverable? Pg. 58  
key pts.
20. (Miscel) Name two subsections of the paragraph named “Specific Requirements” in an SRS  
based on the IEEE Standard.  
The paragraph section named *Specific Requirements* should have all the following sub-  
paragraphs (according to IEEE Std. 830-1993): External Interfaces / Functions / Performance  
Requirements / Logical Database Requirements / Standards Compliance / Software System  
Attributes / (Reliability, Availability, Maintainability, Portability). Any two is correct.
21. (Chap 3) Well-managed projects sometimes fail. Badly managed projects  
inevitably fail.
22. (Chap 3) Project management is a \_\_\_\_\_ **continuous activity** from initial  
concept through to system delivery.
23. (Chap 3) The project plan contains the following sections (i.e., is structured accordingly)  
*Introduction, Project organization, \_\_\_\_\_ Risk analysis, Hardware and  
software resource requirements, Work breakdown, Project schedule, Monitoring and reporting  
mechanisms.*
24. (Chap 3) The scheduling problem is fraught with pit falls. Here are some basic rules of  
guidance: Productivity is not \_\_\_\_\_ **proportional** to the number of people working  
on a task; Adding people to a late project makes it \_\_\_\_\_ **later** because of communication  
overheads; The unexpected \_\_\_\_\_ **always** happens.

25. (Chap 3) In your own words, describe the term critical path based on the activity network described in Chapter 3 of Sommerville.

A path through the activity network shows the dependanceis among tasks and deliverables and their expected duration. The critical path shows longest path in terms of duration. Any task on the critical path that is delayed will cause the whole project to be delayed.

26. (Chap 4/5) Name the two main characteristics and/or purposes for the requirements traceability matrix.

The RTM lists the identifier (an ID number) of all the top level requirement and any derived requirements (derived from the top level). For each individual requirement the method of how the requirement will be verified (as being satisfied) is given (analysis, demonstration, inspection or by analogy). The RTM also gives the location in the design where this requirement is addressed. Its purpose is to provide a concise way to track each individual requirement through the entire process (i.e., up to acceptance testing) into operations and maintenance phase.

27. (Chap 4) Briefly describe what is meant by requirements validation (Page 76).

28. (Chap 4) Briefly describe what is meant by requirements evolution (Page 76).

29. (Chap 5) What is the essential difference between viewpoint-oriented analysis and method-based analysis (Page 97).

30. (Chap 6) A model is an abstracct view of a system which \_\_\_\_\_.  
Complementary system models can be developed that present \_\_\_\_\_ information about the system (Page 114).

31. (Chap 6) Give a brief definition of the following three types of abstract system models (Page 114):

a. Data-flow models

b. Semantic models

c. Object models

32. (Chap 7) Explain the differences (at least three) between Requirements *Definition* and Requirements *Specification*.

**Requirements definition** Customer-oriented descriptions of the system's functions and constraints on its operation

**Requirements specification** Precise and detailed descriptions of the system's functionality and constraints. Intended to communicate what is required to system developers and serve as the basis of a contract for the system development

33. (Chap 7) Explain what is meant by non-functional requirements (give at least three examples).

**Define system properties and constraints** e.g. reliability, response time and storage requirements. Constraints are I/O device capability, system representations, etc. (Process requirements may also be specified mandating a particular CASE system, programming language or development method). Three classifications of non-functional requirements are product, external and organizational

34. (Chap 8) Name and define all the different kinds of SW Prototyping?

Evolutionary and Throw-away

The objective of *evolutionary prototyping* is to **deliver a working system to end-users**. The development starts with those requirements which are *best* understood. The objective of *throw-away prototyping* is to **validate or derive the system requirements**. The prototyping process starts with those requirements which are *poorly* understood

35. What is *Transformational* development (Chap9)?

Formal (provably correct) transformations are applied to successive intermediate representation forms starting with a formal specification and (hopefully) ending with an executable program.

36. What is a functional abstraction (Chap9)?

The formal specification is expressed as input and output predicates (pre and post conditions). *Predicates are logical expressions which are always either true or false. There is no need to be concerned with global state (assuming no side-effects)*

37. Formal specifications are expressed in a mathematical notation with precisely defined (three terms) (Chap9)?

Vocabulary, syntax and semantics

38. Name the phases and ordering of specification and design (Chap9).

Requirements definition, requirements specification, architectural design, software specification and high-level design (respectively).

39. What is system modeling versus architectural design (Chap9)?

Architectural design is essential to structure a specification whereas system modeling is capturing the facets (aspects) of the system into a form that is amenable to analysis and verification and validation.

40. What classes of software are difficult to specify using current techniques (Chap9)?  
Formal specification techniques are *not cost-effective for the development of interactive systems*
41. What are the advantages of formal specification (name four) (Chap9)?  
Provide *insights* into the software requirements and the design  
Formal specifications may be *analyzed mathematically* to demonstrate consistency and completeness of the specification (in addition to other things)  
It may be possible to *prove that the implementation corresponds to the specification*  
Formal specifications may be used *to guide the tester of the component in identifying appropriate test cases*  
Formal specifications may be “*processed*” using software tools.  
It may be possible to *animate* the specification to provide a software prototype
42. Give the seven myths of formal methods (Chap9).  
Perfect software results from formal methods  
Formal methods means program proving  
Formal methods can only be justified for safety-critical systems.  
Formal methods are for mathematicians  
Formal methods increase development costs  
Clients cannot understand formal specifications  
Formal methods have only been used for trivial systems
43. According to the author Sommerville, how should the development costs profile change going from a development process that does not use formal specification to one that does (Chap9)?  
The initial investment in specification will be higher but the costs associated with design and implementation and validation will be lower (when using formal methods for an appropriate application).
44. Describe the idea of pre- and post-condition when used in a software specification (Chap9).  
Set out the pre-conditions  
*A statement about the function parameters stating what is invariably true before the function is executed*  
Set out the post-conditions  
*A statement about the function parameters stating what is invariably true after the function has executed*
45. What is a predicate and how is it used in the context of software specifications (Chap9)?  
*Predicates are logical expressions which are always either true or false*

Predicate operators include the usual logical operators and quantifiers such as **for-all** ( ) and **exists** ( )

46. (Chap9).What is the basic difference between algebraic and model based approaches to formal specification to software systems? Give an example of a language from both domains which operates in the sequential and the parallel worlds of software systems (if you like)

Algebraic approach (Sequential Larch, Concurrent Lotos)

The system is described in terms of interface operations and their relationships

Model-based approach (Sequential Z or VDM, Concurrent CSP or Petri nets)

A model of the system acts as a specification.

This model is constructed using well-understood mathematical entities such as sets and sequences

47. Formal specification forces an analysis of the system \_\_\_\_\_ **requirements** at an early stage (Chap9).

48. Formal specifications are precise and \_\_\_\_\_ **unambiguous** (Chap9).

49. Formal specifications are most applicable in the development of \_\_\_\_\_ **safety - critical** systems because of their inherent high cost (Chap9).

50. Functions can be specified by setting out pre and post conditions for the function. However, this approach does not scale up to \_\_\_\_\_ **large** or medium sized systems (Chap9).