#### INF3705 – 2021 Nov Exam MCQ Notes

[2021 Assignment 1 Memo (own) 2](#_Toc87897482)

[2021 Assignment 1 Memo (other student) 7](#_Toc87897483)

[2021 Assignment 2 Memo (own) 11](#_Toc87897484)

[2021 Assignment 2 Memo (other student) 16](#_Toc87897485)

[2021 Practice Assessment 20](#_Toc87897486)

[2020 Assignment 1 Memo 29](#_Toc87897487)

[2020 Assignment 2 Memo 32](#_Toc87897488)

[Random MCQ Notes 35](#_Toc87897489)

[MCQs Per Chapter - 2020 35](#_Toc87897490)

[Chapter 2 35](#_Toc87897491)

[Chapter 3 37](#_Toc87897492)

[Chapter 4 38](#_Toc87897493)

[Chapter 5 39](#_Toc87897494)

[Chapter 8 41](#_Toc87897495)

[Chapter 10 42](#_Toc87897496)

[Chapter 17 45](#_Toc87897497)

[Chapter 18 47](#_Toc87897498)

[Chapter 19 49](#_Toc87897499)

[Telegram Upload “More extras” 50](#_Toc87897500)

[Chapter 8 Additional MyUNISA Questions 51](#_Toc87897501)

[Notes based on common pages asked in MCQs. 52](#_Toc87897502)

[Chapter 1 – Introduction (17) 52](#_Toc87897503)

[Key Points 52](#_Toc87897504)

[Other Notes 53](#_Toc87897505)

[Chapter 2 – Software processes (43) 54](#_Toc87897506)

[Key Points 54](#_Toc87897507)

[Other Notes 54](#_Toc87897508)

[Chapter 3 – Agile software development (72) 55](#_Toc87897509)

[Key Points 55](#_Toc87897510)

[Other Notes 55](#_Toc87897511)

[Chapter 4 – Requirements engineering (101) 57](#_Toc87897512)

[Key Points 57](#_Toc87897513)

[Other Notes 57](#_Toc87897514)

[Chapter 5 – System modeling (138) 58](#_Toc87897515)

[Key Points 58](#_Toc87897516)

[Other Notes 59](#_Toc87897517)

[Chapter 6 – Architectural design (167) 60](#_Toc87897518)

[Key Points 60](#_Toc87897519)

[Other Notes 60](#_Toc87897520)

[Chapter 7 – Design and implementation (196) 61](#_Toc87897521)

[Key Points 61](#_Toc87897522)

[Other Notes 61](#_Toc87897523)

[Chapter 8 – Software testing (226) 62](#_Toc87897524)

[Key Points 62](#_Toc87897525)

[Other notes 62](#_Toc87897526)

[Chapter 9 – Software evolution (255) 63](#_Toc87897527)

[Key Points 63](#_Toc87897528)

[Other Notes 64](#_Toc87897529)

[Chapter 10 – Dependable systems (285) 65](#_Toc87897530)

[Key Points 65](#_Toc87897531)

[Other Notes 65](#_Toc87897532)

[Chapter 11 – Reliability engineering (306) 66](#_Toc87897533)

[Key Points 66](#_Toc87897534)

[Other Notes 66](#_Toc87897535)

[Chapter 12 – Safety engineering (339) 67](#_Toc87897536)

[Key Points 67](#_Toc87897537)

[Other Notes 68](#_Toc87897538)

[Chapter 15 – Software reuse (437) 68](#_Toc87897539)

[Key Points 68](#_Toc87897540)

[Other Notes 68](#_Toc87897541)

[Chapter 18 – Service-oriented software engineering (520) 69](#_Toc87897542)

[Key Points 69](#_Toc87897543)

[Other Notes 69](#_Toc87897544)

[Chapter 19 – Systems engineering (551) 70](#_Toc87897545)

[Key Points 70](#_Toc87897546)

[Other Notes 70](#_Toc87897547)

# 2021 Assignment 1 Memo (own)

Part 1 of 9 - chapter 1 Questions

Question 1 of 25

Which of the following doesn't form part of the key challenges facing software engineering?

A. Coping with increasing diversity

B. Demand for reduced delivery times

C. demanding trustworthy software

D. Focus of Computer science theory and fundamentals

Page 20.

Question 2 of 25

Software engineering is an engineering discipline that is concerned with?

A. All aspects of software development

B. Only software design and development

C. Agility software development

D. System development life cycle

Page 20.

Part 2 of 9 - Chapter 2 Questions

Question 3 of 25

Which one of the following is a major advantage of incremental development over the waterfall model?

A. the cost of software re-engineering

B. stand-alone application systems

C. the cost of implementing requirements

D. web services that are developed according to the service standards

Page 50.

Question 4 of 25

Incremental development has three major advantages over waterfall model. Which of the following doesn't form part of the major advantage ?

A. The early delivery and development of useful software

B. The easy way to get customer feedback

C. The cost of implements

D. The system structure that tends to degrades new increments

Page 51 – this is mentioned as a disadvantage.

Question 5 of 25

The stages of the waterfall model directly reflect the fundamental software activities such as?

A. integration and configuration

B. Integration and system testing

C. Requirement analysis and definition

D. Incremental development

Pages 47-48.

Part 3 of 9 - Chapter 3 Questions

Question 6 of 25

A fundamental requirement of scaling agile method is to?

A. allow customer involvement

B. keep customer involved

C. embrace change

D. integrate them with plan driven approaches

Page 91.

Question 7 of 25

Agile approach to software development consists of design and implementation to be?

A. ventral activities to the software design phase

B. central activities in the software process

C. part of the final products

D. part of the development process

Page 74.

Part 4 of 9 - chapter 4 Questions

Question 8 of 25

When a standard format is used for specifying functional requirements, the following information should be included?

A. A description of its inputs and the origins of those inputs

B. A description of software ethnography

C. A description of requirement elicitation

D. A description of standard formats

Page 123.

Question 9 of 25

Which of the following is not part of the non-functional requirements?

A. Robustness

B. Efficiency requirements

C. Product requirements

D. Usability requirements

Figure 4.3 on page 108.

Question 10 of 25

The aims of requirement elicitation process are to?

A. understanding the work that stakeholders do

B. understanding the requirements discovery process

C. understanding the work that system developers do

D. understanding requirement classification

Page 112.

Part 5 of 9 - chapter 5 Questions

Question 11 of 25

Model of the new system are used during requirement engineering to?

A. Help clarify what the existing system does

B. help to explain the proposed requirements to other system stakeholder

C. help to keep he focus on the stakeholder’s discussion

D. help with external perspective

Page 139.

Question 12 of 25

Which of the following doesn’t form part of the four fundamental view?

A. Stakeholder view

B. A process view

C. A development view

D. A logic view

Page 174.

Question 13 of 25

Which of the following doesn't form part of the UML diagram?

A. Activity diagrams

B. Use case diagrams

C. development diagrams

D. Sequence diagrams

Page 141.

Part 6 of 9 - chapter 6 Questions

Question 14 of 25

A software architecture is a description of how a software system is .............?

A. organised

B. Implemented

C. Designed

D. Developed

Page 184 / 192.

Question 15 of 25

Architecture patterns are a means of ............... knowledge about generic system architectures.

A. Recycling

B. Reusing

C. Redeveloping

D. reproducing

Page 192.

Question 16 of 25

To develop a system design from concept to detailed, object-oriental design, you need to

A. Develop some systems artefacts

B. Develop design model

C. Develop a blueprint

D. Develop a system context model

Page 198.

Part 7 of 9 - chapter 7 Questions

Question 17 of 25

The process of object oriented design includes activities to?

A. to implement system architecture

B. to design a system architecture

C. to educate people about open source

D. to documents the entire system architecture

Page 222.

Question 18 of 25

Which of the following is not part of software re-used at different levels?

A. Concept level

B. Component level

C. Abstraction level

D. Object level

Page 213-214.

Question 19 of 25

Which of the following doesn’t form part of the Gang of four definition of four essential element of design patterns?

A. A provisional standard way of accessing the elements

B. A name that is meaning reference to the patterns

C. A solution description of the part of the design solutions

D. A description of the problem area that explain when the patterns may the application

Page 210.

Part 8 of 9 - chapter 8 Questions

Question 20 of 25

Typically a commercial software system has to go through three stages of testing. Which one doesn’t form part of the stages?

A. release testing

B. user testing

C. development testing

D. specification testing

Page 226.

Question 21 of 25

Test-first development approach is an approach to development whereby tests are written before the code to be.........?

A. tested

B. Debugging

C. executed

D. run

Page 252.

Question 22 of 25

Development testing includes units testing in which you test.......... objects and methods?

A. Collective

B. Developed

C. Designed

D. Individual

Page 252.

Part 9 of 9 - chapter 9 Questions

Question 23 of 25

Re-factoring, making small programs change that preserve.........., can be thought of preventative maintenance?

A. availability

B. Malfunctionality

C. Functionality

D. maintainability

Page 280.

Question 24 of 25

Software reengineering is concerned with restructuring and ................... Software in order to make it easier to understand and change?

A. Re-developing

B. Re-integrating

C. Redocumenting

D. Re-designing

Page 280.

Question 25 of 25

Software engineering is a special process with requirements design, implementation and ……… going on throughout the lifetime of the system?

A. Iteration

B. Evaluations

C. Testing

D. Modelling

Page 256 – spiral – not special.

# 2021 Assignment 1 Memo (other student)

Question 1 of 25

Which of the following forms part of four fundamental activities which are common to all software processes?

A. Software dependency

B. Software acceptancy

C. Software validation

D. Software Efficiency

Page 23.

Question 2 of 25

Software engineering is an engineering discipline that is concerned with?

A. System development life cycle

B. Agility software development

C. Only software design and development

D. All aspects of software development

Page 20.

Question 3 of 25

The stages of the waterfall model directly reflect the fundamental software activities such as?

A. integration and configuration

B. Requirement analysis and definition

C. Integration and system testing

D. Incremental development

Pages 47-48.

Question 4 of 25

Which one is an example of software tools that provides process that is supported by automating some process activities?

A. the collection of objects that are developed as a component

B. The program debugging through the provision of information about an executing program

C. web services that are developed according to services standards

D. The stand application

Page 53.

Question 5 of 25

A prototype is an early version of a software system that is used to?

A. implement concepts

B. replicates concepts

C. demonstrate concepts

D. finalize concepts

Page 62.

Question 6 of 25

Which one of the following doesn’t form part of the principles of agile methods?

A. customer involvement

B. enhancement change

C. customer satisfactory

D. Maintain Simplicity

Table at tope of page 76 / 91.

Question 7 of 25

Which of the following is not the key featuring of testing in XP?

A. Collective ownership

B. Test-first development

C. sustainable pace

D. Embrace change

Page 81. See table at top of page 78 as well.

Question 8 of 25

The aims of requirement elicitation process are to?

A. understanding requirement classification

B. understanding the work that stakeholders do

C. understanding the requirements discovery process

D. understanding the work that system developers do

Page 112.

Question 9 of 25

When a standard format is used for specifying functional requirements, the following information should be included?

A. A description of requirement elicitation

B. A description of its inputs and the origins of those inputs

C. A description of software ethnography

D. A description of standard formats

Page 123.

Question 10 of 25

Which of the following form part of the fundamental approaches to the requirement elicitation?

A. interview

B. case study

C. discussions

D. survey

Page 115.

Question 11 of 25

Model of the new system are used during requirement engineering to?

A. Help clarify what the existing system does

B. help with external perspective

C. help to keep he focus on the stakeholder’s discussion

D. help to explain the proposed requirements to other system stakeholder

Page 139.

Question 12 of 25

Which of the following doesn't form part of the UML diagram?

A. Activity diagrams

B. Use case diagrams

C. development diagrams

D. Sequence diagrams

Page 141.

Question 13 of 25

A model-driven engineering is an approach to..........?

A. Systems analysis

B. Enterprise Architecture

C. Systems architecture

D. Software development

Page 163.

Question 14 of 25

Which of the following doesn’t form part of the four fundamental view?

A. A process view

B. A development view

C. A logic view

D. Stakeholder view

Page 174.

Question 15 of 25

Architectures may be documented from several different perspective or .............?

A. Ideology

B. Development

C. View

D. Analysis

Page 192.

Question 16 of 25

A software architecture is a description of how a software system is .............?

A. Developed

B. organised

C. Designed

D. Implemented

Page 184 / 192.

Question 17 of 25

The process of object oriented design includes activities to?

A. to implement system architecture

B. to educate people about open source

C. to design a system architecture

D. to documents the entire system architecture

Page 222.

Question 18 of 25

Which of the following is not part of software re-used at different levels?

A. Abstraction level

B. Concept level

C. Object level

D. Component level

Page 213-214.

Question 19 of 25

Software design and implementation are ................ activities.

A. Same

B. Interdependent

C. Interleaved

D. Different

Page 222.

Question 20 of 25

Some of the guidelines for interface testing are?

A. message passing interface

B. procedural interface

C. share memory interfaces

D. use stress testing

Page 239.

Question 21 of 25

Test-first development approach is an approach to development whereby tests are written before the code to be.........?

A. run

B. executed

C. tested

D. Debugging

Page 252.

Question 22 of 25

Scenario testing is useful because it replicates the ............. use of the system?

A. Experimental

B. Typical

C. Testing

D. Practical

Page 252.

Question 23 of 25

Re-factoring, making small programs change that preserve.........., can be thought of preventative maintenance?

A. Functionality

B. Malfunctionality

C. availability

D. maintainability

Page 280.

Question 24 of 25

Software reengineering is concerned with restructuring and ................... Software in order to make it easier to understand and change?

A. Re-designing

B. Re-developing

C. Re-integrating

D. Redocumenting

Page 280.

Question 25 of 25

Software engineering is a special process with requirements design, implementation and ……… going on throughout the lifetime of the system?

A. Modelling

B. Testing

C. Evaluations

D. Iteration

Page 256 – spiral – not special.

# 2021 Assignment 2 Memo (own)

Question 1 of 25

Which of the following below reason is true about the dependability of the systems been more important that their functionality?

A. system failure costs may be enormous

B. many years maintenance usually degrades the system structure, making it increasingly

C. The data processed by the system may be maintained in different files

D. the program style and the usage conventions are inconsistent.

Page 286.

Question 2 of 25

Which one of the following is not part of the fire principles dimensions to dependability?

A. safety informally

B. availability informally

C. Maintainability

D. reliability informally

Page 288.

Question 3 of 25

The dependability of a computer system is a system property that reflects the user's degree of ........ in the system.

A. Trust

B. Maintainability

C. Workability

D. Functionality

Page 288.

Question 4 of 25

System dependability is important because....... of the critical system can lead to large economic losses, serious information loss, physical damage or threats to human life...

A. Success

B. Sustainability

C. Failure

D. running

Page 303.

Part 2 of 6 - Multiple choice questions

Question 5 of 25

Availability is the probability of failure-free operation over a specified time, in a given environment, for a specific purpose?

True

False

Page 309. Reliability: The probability of failure-free operation over a specified time, in a given environment, for a specific purpose.

Question 6 of 25

Negotiating the test results is one of the stages in the acceptance testing process.

True

False

Confused – False is given as the right answer, but pages 250-251.

Question 7 of 25

Re-processing is a process of making improvement to a program to slow down degradation.

True

False

Page 278. Refactoring is the process of making improvements to a program to slow down degradation through change.

Question 8 of 25

Refactoring which is making program changes to preserve a system functionality is a form of Maintenance to repair software faults?

True

False

Best seems to be page 280, which suggests true.

Question 9 of 25

Testing is part of a broader process of Software verification and validation.

True

False

Page 710.

Part 3 of 6 - chapter 12 Questions

Question 10 of 25

The starting point for generating functional safety requirements is usually domain knowledge, safety standards, and …………………..?

A. regulations

B. procedures

C. process

D. reactive

Page 344.

Question 11 of 25

A hazard is a system state that could be lead to…………………..?

A. system failure

B. system crush

C. accident

D. damage

Page 342.

Question 12 of 25

Hazard analysis is the process of discovering the root causes of hazard in a ………………….system?

A. safety-critical

B. agile critical

C. error-free

D. risk-free

Page 349.

Question 13 of 25

Safety-critical systems are systems whose failures can lead to human.......?

A. Success

B. Advancement

C. Delay

D. Injury

Page 287 / 368.

Part 4 of 6 - chapter 15 Questions

Question 14 of 25

Reused-based software engineering is an approach to development that tries to……… reuse of existing software?

A. minimum

B. re-arrange

C. redefine

D. maximum

Page 438.

Question 15 of 25

Which one of the following falls amongst the benefits of software re-use?

A. lack of tool support

B. “not-invented-here”’ syndrome

C. reduced process risk

D. increase maintenance costs

Table on 439.

Question 16 of 25

systems may be developed by configuring a single, generic application systems or by integrating two or more ......... systems

A. Agile

B. User

C. Application

D. redundant

Page 461.

Question 17 of 25

Which of the following doesn’t form part or key factors that one should consider when planning re-use?

A. Object and function re-use software component that implement a single function

B. The expected software lifetime

C. The background, skills and experience of the development

D. The development schedule for the software

Page 441-442.

Part 5 of 6 - chapter 18 Questions

Question 18 of 25

A graphical workflow language, such as BPMN, may be used to describe a business process and the........used in the process?

A. technique

B. Approach

C. Style

D. Service

Page 548.

Question 19 of 25

The fundamental elements in a RESTful architecture is a …………………..?

A. attributes

B. outputs

C. resources

D. inputs

Page 548.

Question 20 of 25

REST is an architectural style based on transferring representations of resources from a server to………….?

A. central database

B. client

C. cloud

D. relational database

Page 530.

Question 21 of 25

XML-based service description include definitions of XML……………………?

A. syntax

B. special characters

C. namespaces

D. elements

Page 528.

Part 6 of 6 - chapter 19 Questions

Question 22 of 25

System development is a complex process in which the elements that are part of the system are developed or purchased and then integrated to create the……… systems?

A. agile

B. final

C. partial

D. computer-based

Page 570.

Question 23 of 25

System development processes include requirements specification, design, construction, integrating and......?

A. developing

B. documenting

C. Testing

D. error-identifying

Page 577.

Question 24 of 25

When a system is put into use, the operational processes and the system itself inevitably change to reflect changes to the business requirements and the ........... environment.

A. Desktop

B. Systems

C. User

D. Application

Page 577.

Question 25 of 25

Which one of the following doesn’t form part of the four overlapping stage in the lifetime of larger complex systems?

A. maintenance

B. development

C. operations

D. procurement

Page 553-554.

# 2021 Assignment 2 Memo (other student)

Question 1 of 25

The dependability of a computer system is a property of the system that reflects its………….?

A. resilience

B. trustworthiness

C. reliability

D. vulnerability

Page 288.

Question 2 of 25

Which one of the following is not part of the fire principles dimensions to dependability?

A. Maintainability

B. availability informally

C. reliability informally

D. safety informally

Page 288.

Question 3 of 25

System dependability is important because....... of the critical system can lead to large economic losses, serious information loss, physical damage or threats to human life...

A. running

B. Sustainability

C. Failure

D. Success

Page 303.

Question 4 of 25

Which of the following below reason is true about the dependability of the systems been more important that their functionality?

A. many years maintenance usually degrades the system structure, making it increasingly

B. the program style and the usage conventions are inconsistent.

C. system failure costs may be enormous

D. The data processed by the system may be maintained in different files

Page 286.

Question 5 of 25

The aim of Software validation is to ensure that the software meets the customer needs?

True

False

Page 228.

Question 6 of 25

Testing is part of a broader process of Software verification and validation.

True

False

Page 710.

Question 7 of 25

A backup mechanism exploits on reusable components in its design?

True

False (Object oriented design exploits this)

Question 8 of 25

Re-processing is a process of making improvements to a program to slow down degradation.

True

False (Refactoring is a process…)

Page 278.

Question 9 of 25

Probability of failure on demand (POFOD), Rate of occurrence of failures (ROCOF), Availability (AVAIL) are three metrics used to specify reliability and availability of a system?

True

False

Page 313.

Question 10 of 25

The starting point for generating functional safety requirements is usually domain knowledge, safety standards, and …………………..?

A. reactive

B. regulations

C. procedures

D. process

Page 344.

Question 11 of 25

Safety-critical system are systems in which it is essential that the system operations is always……….?

A. safe

B. secure

C. hazard

D. critical

Page 341.

Question 12 of 25

Hazard analysis is the process of discovering the root causes of hazard in a ………………….system?

A. agile critical

B. error-free

C. safety-critical

D. risk-free

Page 349.

Question 13 of 25

Safety-critical systems are systems whose failures can lead to human.......?

A. Advancement

B. Success

C. Delay

D. Injury

Page 368.

Question 14 of 25

systems may be developed by configuring a single, generic application systems or by integrating two or more ......... systems

A. User

B. Application

C. Agile

D. redundant

Page 461.

Question 15 of 25

Reused-based software engineering is an approach to development that tries to……… reuse of existing software?

A. minimum

B. re-arrange

C. redefine

D. maximum

Page 438.

Question 16 of 25

Which one of the following falls amongst the benefits of software re-use?

A. reduced process risk

B. lack of tool support

C. “not-invented-here”’ syndrome

D. increase maintenance costs

Table on 439.

Question 17 of 25

Which of the following doesn’t form part or key factors that one should consider when planning re-use?

A. The background, skills and experience of the development

B. The expected software lifetime

C. The development schedule for the software

D. Object and function re-use software component that implement a single function

Page 441-442.

Question 18 of 25

Service-oriented system have loosely couple architecture where service binding may change during systems…………….?

A. testing

B. maintenance

C. execution

D. development

Page 522.

Question 19 of 25

REST is an architectural style based on transferring representations of resources from a server to………….?

A. cloud

B. relational database

C. central database

D. client

Page 530.

Question 20 of 25

The fundamental elements in a RESTful architecture is a …………………..?

A. resources

B. attributes

C. inputs

D. outputs

Page 548.

Question 21 of 25

A graphical workflow language, such as BPMN, may be used to describe a business process and the........used in the process?

A. Style

B. Service

C. Approach

D. technique

Page 548.

Question 22 of 25

When a system is put into use, the operational processes and the system itself inevitably change to reflect changes to the business requirements and the ........... environment.

A. Desktop

B. Application

C. User

D. Systems

Page 577.

Question 23 of 25

System development processes include requirements specification, design, construction, integrating and......?

A. Testing

B. documenting

C. developing

D. error-identifying

Page 577.

Question 24 of 25

System development is a complex process in which the elements that are part of the system are developed or purchased and then integrated to create the……… systems?

A. final

B. computer-based

C. partial

D. agile

Page 570.

Question 25 of 25

Which one of the following doesn’t form part of the four overlapping stage in the lifetime of larger complex systems?

A. development

B. maintenance

C. procurement

D. operations

Page 553-554.

# 2021 Practice Assessment

Question 1 of 60

Design phase is followed by

A. Testing

B. Maintenance

C. Coding

D. None of the above.

Question 2 of 60

Where is the prototyping model of software development well suited?

A. When requirements are well defined.

B. For projects with large development teams.

C. When a customer cannot define requirements clearly.

D. None of the above.

Question 3 of 60

What is the meaning of requirement elicitation in software engineering?

A. Gathering of requirement.

B. Understanding of requirement.

C. Getting the requirements from client.

D. All of the above.

Question 4 of 60

If requirements are easily understandable and defined then which model is best suited?

A. Spiral model

B. Waterfall model

C. Prototyping model

D. None of the above

Question 5 of 60

Select the option that suits the Manifesto for Agile Software Development

A. Individuals and interactions

B. Working software

C. Customer collaboration

D. All of the above

Page 76.

Question 6 of 60

How is plan driven development different from agile development ?

A. Outputs are decided through a process of negotiation during the software development process

B. Specification, design, implementation and testing are interleaved

C. Iteration occurs within activities

D. All of the above

Question 7 of 60

Which of the following does not apply to agility to a software process?

A. Uses incremental product delivery strategy

B. Only essential work products are produced

C. Eliminate the use of project planning and testing

D. All of the above

Question 8 of 60

Which three framework activities are present in Adaptive Software Development (ASD) ?

A. analysis, design, coding

B. requirements gathering, adaptive cycle planning, iterative development

C. speculation, collaboration, learning

D. All of the above

Question 9 of 60

When is a Sprint Retrospective ceremony performed?

A. Whenever the team suggests

B. At the end of each Sprint

C. Whenever needed

D. All of the above

Question 10 of 60

In the elicitation process, the developers discuss with the client and end users and know their expectations for the software.

A. Organizing requirements

B. Requirement gathering

C. Negotiation & discussion

D. All of the above

Question 11 of 60

The computer-based system can have a profound effect on the design that is chosen and also the implementation approach will be applied.

A. Behavioral elements

B. Flow-oriented elements

C. Scenario-based elements

D. All of the above

Question 12 of 60

Which model in system modelling depicts the dynamic behaviour of the system ?

A. Context Model

B. Behavioral Model

C. Data Model

D. Object Model

Page 139.

Question 13 of 60

The UML supports event-based modeling using \_\_\_\_\_\_\_\_\_\_\_\_ diagrams.

A. Deployment

B. Collaboration

C. State chart

D. All of the above

Question 14 of 60

Which of the following statement is incorrect regarding the Class-responsibility-collaborator (CRC) modeling ?

A. All use-case scenarios (and corresponding use-case diagrams) are organized into categories in CRC modelling

B. The review leader reads the use-case deliberately

C. Only developers in the review (of the CRC model) are given a subset of the CRC model index cards

D. All of the above

Question 15 of 60

A \_\_\_\_\_\_\_\_ view shows the system hardware and how software components are distributed across the processors in the system.

A. physical

B. logical

C. process

D. All of above

Page 174.

Question 16 of 60

Software design is a creative activity in which you identify software components and their relationships, based on a .........................

A. Customer’s requirements

B. Task requirement

C. Resource requirement

Page 197.

Question 17 of 60

Implementation is the process of realizing the design as a ....................

A. Portfolio

B. Program

C. Programmes

Page 197.

Question 18 of 60

There are a variety of different object-oriented design processes that depend on the organization using the process. Common activities in these processes include:

A. Identify monetary plan

B. Identify old system and unpack it

C. Identify the principal system objects

Page 198.

Question 19 of 60

At this level, you don’t reuse software directly but use knowledge of successful abstractions in the design of your software.

A. The abstraction level

B. The extraction level

C. The distraction level

Page 213.

Question 20 of 60

An approach to software development in which the source code of a software system is published and volunteers are invited to participate in the development process.

A. Closed source pre developemt

B. Open source development

C. Open software foundation task

Page 219.

Question 21 of 60

It is intended to show that a program does what it is intended to do and to discover program defects before it is put into use.

A. Program testing

B. Portfolio testing

C. Programme testing

Question 22 of 60

To demonstrate to the developer and the customer that the software meets its requirements. This means that there should be at least one test for every requirement in the requirements document.

A. Generic software product

B. Custom software

C. Interface design

Page 227.

Question 23 of 60

To demonstrate to the developer and the customer that the software meets its requirements.It means that there should be tests for all of the system features, plus combinations of these features, that will be incorporated in the product release.

A. Generic software product

B. Custume software

C. interface design

Page 227.

Question 24 of 60

To demonstrate to the developer and the system customer that the software meets its requirements

A. Defect testing

B. Verification testing design

C. Validation testing

Question 25 of 60

Development testing includes all testing activities that are carried out by the team developing the system,testing where some or all of the components in a system are integrated and the system is tested as a whole.

A. Sytem testing

B. Components testsing

C. Unit testing

Page 232.

Question 26 of 60

At this stage, the software remains useful but the only changes made are those required to keep it operational i.e. bug fixes and changes to reflect changes in the software’s environment. No new functionality is added.

A. Evolution

B. Servicing

C. Phase out

Page 258.

Question 27 of 60

One of Lehman’s laws states : A program that is used in a real-world environment must necessarily change, or else become progressively less useful in that environment.

A. Large program evolution

B. Continuing change

C. Increasing complexity

Page 271.

Question 28 of 60

Concerned with assessing which parts of the system may cause problems and have high maintenance costs

A. Maintenance prediction

B. Contractual responsibility

C. Team stability

Page 274.

Question 29 of 60

It is a continuous process of improvement throughout the development and evolution process. It is intended to avoid the structure and code degradation that increases the costs and difficulties of maintaining a system.

A. Refactoring

B. Reengineering

C. Offshore

Page 279.

Question 30 of 60

‘Bad smells’ in program code : It occur when the same group of data items (fields in classes, parameters in methods) re-occur in several places in a program. These can often be replaced with an object that encapsulates all of the data.

A. Data clumps

B. Speculative generality

C. Duplicate code

Page 279.

Question 31 of 60

Where is the prototyping model of software development well suited?

A. When requirements are well defined.

B. For projects with large development teams.

C. When a customer cannot define requirements clearly.

D. All of the above

Question 32 of 60

Which of the following is/are Project Estimation Technique?

A. Heuristic Estimation Technique.

B. Heuristic Estimation Technique.

C. Analytical Estimation Technique.

D. All of the above.

Question 33 of 60

What kind of investments does organization have in order to make most of by the software architects in their design?

A. Libraries

B. Standards and guidelines

C. Software tools

D. All of the above.

Question 34 of 60

Look at comparable systems that have been developed elsewhere (if any) and assess whether or not the proposed system could be implemented using current hardware and software technologies.

A. System structure development

B. System vision document

C. Feasibility study

D. Problem understanding

Page 565.

Question 35 of 60

An erroneous system state that can lead to system behaviour that is unexpected by system users. The value of transmission time is set incorrectly (to 24.XX rather than 00.XX) when the faulty code is executed.

A. System error

B. System failure

C. Human error

D. System fault

Page 307.

Question 36 of 60

A characteristic of a software system that can lead to a system error. The fault is the inclusion of the code to add 1 hour to the time of the last transmission, without a check if the time is greater than or equal to 23.00.

A. System error

B. System failure

C. Human error

D. System fault

Page 307.

Question 37 of 60

The process should be understandable by people apart from process participants, who can check that process standards are being followed and make suggestions for process improvement.

A. Auditable

B. Standardized

C. Robust

D. Diverse

Table at the top of page 298.

Question 38 of 60

CMM model in Software Engineering is a technique of \_\_\_\_\_\_.

A. Develop the software.

B. Improve the software process.

C. Improve the testing process.

D. All of the above.

Question 39 of 60

Which of the following does not apply to agility to a software process?

A. Uses incremental product delivery strategy

B. Only essential work products are produced

C. Eliminate the use of project planning and testing

D. All of the above.

Table top page 78.

Question 40 of 60

How is plan driven development different from agile development?

A. Outputs are decided through a process of negotiation during the software development process

B. Specification, design, implementation and testing are interleaved

C. Iteration occurs within activities

D. All of the above

Question 41 of 60

Which three framework activities are present in Adaptive Software Development (ASD)?

A. Analysis, design, coding

B. Requirements gathering, adaptive cycle planning, iterative development

C. Speculation, collaboration, learning

D. All of the above.

Question 42 of 60

Which one of the following is not a step of requirement engineering?

A. Elicitation

B. Design

C. Analysis

D. Documentation

Question 43 of 60

In which elicitation process the developers discuss with the client and end users and know their expectations from the software?

A. Requirement gathering

B. Organizing requirements

C. Negotiation & discussion

D. Documentation

Question 44 of 60

Which document is created by system analyst after the requirements are collected from various stakeholders?

A. Software requirement specification

B. Software requirement validation

C. Feasibility study

D. Requirement Gathering

Question 45 of 60

Which documentation works as a key tool for software designer, developer and their test team to carry out their respective tasks?

A. Requirement documentation

B. User documentation

C. Software design documentation

D. Technical documentation

Question 46 of 60

“Consider a system where, a heat sensor detects an intrusion and alerts the security company.” What kind of a requirement the system is providing?

A. Functional

B. Non-Functional

C. Known Requirement

D. None of the above

Question 47 of 60

The requirements that result from requirements analysis are typically expressed from one of three perspectives or views. What is that perspective or view?

A. Developer

B. User

C. Non-Functional

D. Physical

Question 48 of 60

Which model in system modelling depicts the dynamic behaviour of the system?

A. Context Model

B. Behavioural Model

C. Data Model

D. Object Model

Page 139.

Question 49 of 60

Which perspective in system modelling shows the system or data architecture?

A. Structural perspective

B. Behavioural perspective

C. External perspective

D. All of the above.

Page 139.

The UML supports event-based modeling using \_\_\_\_\_\_\_\_\_\_\_\_ diagrams.

a) Deployment

b) Collaboration

c) State chart

d) All of the mentioned

Question 51 of 60

What are the categories in which quality attributes are divided in?

A. Development Attributes

B. Operational Attributes

C. Functional Attributes

D. Development & Operational Attributes

Question 52 of 60

Which of these comes under development attribute?

A. Maintainability

B. Reusability

C. Performance

D. Maintainability & Reusability

Question 53 of 60

What kind of investments does organization have in order to make most of by the software architects in their design?

A. Libraries

B. Standards and guidelines

C. Software tools

D. All of the above.

Question 54 of 60

Look at comparable systems that have been developed elsewhere (if any) and assess whether or not the proposed system could be implemented using current hardware and software technologies.

A. System structure development

B. System vision document

C. Feasibility study

D. Problem understanding

Reset Selection

Page 565.

Question 55 of 60

During this stage, the abstract workflow design is transformed to an executable program and the service interface is defined. You can use a conventional programming language, such as Java or a workflow language, such as WS-BPEL.

A. Refine work flow

B. Test completed service

C. Create work flow program

D. Formulate outline workflow

Page 543.

Question 56 of 60

An approach to V & V that examines the source code of a system, looking for errors and anomalies. It allows all parts of a program to be checked, not just those parts that are exercised by system tests.

A. Model checking

B. Safety and dependability

C. Static analysis

D. Change management

Some information on page 359-360.

Question 57 of 60

An erroneous system state that can lead to system behaviour that is unexpected by system users. The value of transmission time is set incorrectly (to 24.XX rather than 00.XX) when the faulty code is executed.

A. System error

B. System failure

C. Human error

D. System fault

Page 307.

Question 58 of 60

A characteristic of a software system that can lead to a system error. The fault is the inclusion of the code to add 1 hour to the time of the last transmission, without a check if the time is greater than or equal to 23.00.

A. System error

B. System failure

C. Human error

D. System fault

Page 307.

Question 59 of 60

The process should be understandable by people apart from process participants, who can check that process standards are being followed and make suggestions for process improvement.

A. Auditable

B. Standardized

C. Robust

D. Diverse

Table at the top of page 298.

Question 60 of 60

Why is it necessary to introduce some methods and documentation from plan-based approaches when scaling agile methods to larger projects that are developed by distributed development teams?

A. Project planning is often essential when developing software with larger teams.

B. Requirements analysis and documentation is not important to decide how to distribute the work across teams.

C. Design documentation especially interface specifications are not important so that teams can develop independently.

D. Risk management may not be required because it is not necessary to ensure that all of the teams understand the risks faced.

# 2020 Assignment 1 Memo

1.

Software engineering is an engineering discipline that is concerned with?

a. only software design and development

b. system development life cycle

c. agility software development

d. all aspects of software development

2.

Which of the following doesn’t form part of the fundament ideas of software development?

a. Software process

b. Software dependability and security

c. Software re-use

d. Software architecture

3.

Which of the following doesn’t form part of the fundamental activities of software engineering?

a. Software customization

b. Software specification

c. Software development

d. Software validation and evolution

4.

The stages of the waterfall model directly reflect the fundamental software activities such as?

a. Incremental development

b. integration and configuration

c. Requirement analysis and definition

d. Integration and system testing

5.

A prototype is an early version of a software system that is used to?

a. demonstrate concepts

b. finalize concepts

c. replicates concepts

d. implement concepts

6.

Which one of the following is a major advantage of incremental development over the waterfall model?

a. the cost of implementing requirements

b. stand-alone application systems

c. the cost of software re-engineering

d. web services that are developed according to the service standards

7.

Agile approach to software development consists of design and implementation to be?

a. part of the final products

b. part of the development process

c. central activities in the software process

d. ventral activities to the software design phase

8.

Which one of the following doesn’t form part of the principles of agile methods?

a. customer satisfactory

b. customer involvement

c. embrace change

d. Maintain Simplicity

9.

A fundamental requirement of scaling agile method is to?

a. allow customer involvement

b. embrace change

c. keep customer involved

d. integrate them with plan driven approaches

10.

When a standard format is used for specifying functional requirements, the following information should be included?

a. A description of its inputs and the origins of those inputs

b. A description of standard formats

c. A description of software ethnography

d. A description of requirement elicitation

11.

The aims of requirement elicitation process are to?

a. understanding the requirements discovery process

b. understanding requirement classification

c. understanding the work that stakeholders do

d. understanding the work that system developers do

12.

Which of the following form part of the fundamental approaches to the requirement elicitation?

a. interview

b. survey

c. case study

d. discussions

13.

Requirement validation is the process of checking that?

a. Validation processes are aligned

b. the system and user requirements are outlined

c. requirements defines the system that customers wants

d. the system is capable of doing what it was meant to do

14.

Model of the existing system are used during requirements engineering to?

a. to assist discussing design proposals

b. to document the system for implementation

c. to generate a complete or partial system implementation

d. to clarify what existing system does

15.

Model of the new system are used during requirement engineering to?

a. Help clarify what the existing system does

b. help to keep he focus on the stakeholder’s discussion

c. help to explain the proposed requirements to other system stakeholder

d. help with external perspective

16.

Which of the following doesn’t form part of the four fundamental view?

a. A logic view

b. A process view

c. Stakeholder view

d. A development view

17.

A software architecture can be design at which level of abstraction?

a. Architecture in larger

b. Architecture in a holistic approach

c. Architecture at a framework level

d. Architecture at conceptual level

18.

To develop a system design from concept to detailed, object-oriental design, you need to

a. Develop design model

b. Develop a blueprint

c. Develop some systems artefacts

d. Develop a system context model

19.

Which of the following is not part of software re-used at different levels?

a. Abstraction level

b. object level

c. component level

d. Concept level

20.

The process of object oriented design includes activities to?

a. to implement system architecture

b. to design a system architecture

c. to documents the entire system architecture

d. to educate people about open source

21.

Which of the following doesn’t form part of the Gang of four definition of four essential element of design patterns?

a. a name that is meaning reference to the patterns

b. a description of the problem area that explain when the patterns may the application

c. a solution description of the part of the design solutions

d. a provisional standard way of accessing the elements

22.

When one uses the UML to develop a design, he /she should develop two kinds of the design model which are

a. structured and dynamic model

b. static and dynamic model

c. static and subsystem model

d. structural and sequence model

23.

Typically a commercial software system has to go through three stages of testing. Which one doesn’t form part of the stages?

a. specification testing

b. development testing

c. release testing

d. user testing

24.

Interface errors are one of the most common form of error in complex system (Luz 1993). Which of the following doesn’t form part of the three classes of errors?

a. Interface misuse

b. logic and system error

c. interface misunderstanding

d. timing error

25

Some of the guidelines for interface testing are?

a. share memory interfaces

b. procedural interface

c. message passing interface

d. use stress testing

# 2020 Assignment 2 Memo

1.

Software engineering is a special process with requirements design, implementation and ……… going on throughout the lifetime of the system?

a. evaluations

b. iteration

c. testing

d. modelling

2.

Which one of the following doesn’t form part of change request that arises with operational systems?

a. if a series fault is detected, has to be repaired to allow normal operations.

b. if change to the system operating environment have unexpected effects that disrupts normal operations.

c. if development changes has used agile approach but operating systems prefers a plan- based approach.

d. if there are unanticipated changes to the business running the systems, such as the emergence of new competitors.

3.

Legacy systems are older systems that rely on language and ..…….. that are no longer used for new system development?

a. technology

b. techniques

c. agile systems

d. models

4.

Which one of the following doesn’t form part of the reason why it is expensive and risky to replace legacy systems?

a. low quality, low business values keeping these systems in operations will be expensive

b. the system documentation is often inadequate and out date

c. The system may have been optimized for space utilization or execution speed so that it

runs effectively.

d. The data processed by the system may be maintained in different files that have incompatible structures.

5.

Which of the following below reason is true about the dependability of the systems been more important that their functionality?

a. system failure costs may be enormous

b. the program style and the usage conventions are inconsistent.

c. many years maintenance usually degrades the system structure, making it increasingly

difficult to understand

d. The data processed by the system may be maintained in different files

6.

Which one of the following is not part of the fire principles dimensions to dependability?

a. availability informally, the availability of a system is the probability that it will be up and running and able to deliver useful services to users at any given time.

b. reliability informally, the reliability of a system is the probability, over a given period of time that the system will correctly deliver services as expected by the user.

c. safety informally, the security of a system is judged by how likely it is that he system will

cause damage to people or its environment.

d. Maintainability, as the system are used, new equipment emerge, and it is important to maintain the value of a system.

7.

The dependability of a computer system is a property of the system that reflects its………….?

a. vulnerability

b. trustworthiness

c. resilience

d. reliability

8.

A sociotechnical systems are so complex that it is impossible to understand them as a………….

a. holistic approach

b. whole

c. stack

d. layers

9.

Safety-critical system are systems in which it is essential that the system operations is always……….?

a. secure

b. critical

c. safe

d. hazard

10.

A hazard is a system state that could be lead to…………………..?

a. accident

b. system failure

c. system crush

d. damage

11.

The starting point for generating functional safety requirements is usually domain knowledge, safety standards, and …………………..?

a. reactive

b. process

c. regulations

d. procedures

12.

Hazard analysis is the process of discovering the root causes of hazard in a ………………….system?

a. agile critical

b. safety-critical

c. risk-free

d. error-free

13.

Reused-based software engineering is an approach to development that tries to……… reuse of existing software?

a. re-arrange

b. redefine

c. maximum

d. minimum

14.

Which of the following doesn’t form part or key factors that one should consider when planning re-use?

a. The development schedule for the software

b. The expected software lifetime

c. Object and function re-use software component that implement a single function

d. The background, skills and experience of the development

15.

Which one of the following falls amongst the benefits of software re-use?

a. reduced process risk

b. lack of tool support

c. increase maintenance costs

d. “not-invented-here”’ syndrome

16.

Which one of the following forms part of approaches that support software re-use?

a. increase maintenance cost

b. legacy system wrapping

c. increased dependability

d. accelerated development

17.

Service-oriented system have loosely couple architecture where service binding may change during systems…………….?

a. development

b. testing

c. execution

d. maintenance

18.

The UDDI (Universal Description, Discovery and integration) discovery standard defines the components of a service specification intended to help potential uses discover the existing of…………..?

a. software

b. application

c. service

d. protocol

19.

XML-based service description include definitions of XML……………………?

a. namespaces

b. special characters

c. syntax

d. elements

20.

The fundamental elements in a RESTful architecture is a …………………..?

a. attributes

b. resources

c. inputs

d. outputs

21.

REST is an architectural style based on transferring representations of resources from a server to………….?

a. cloud

b. central database

c. relational database

d. client

22.

Services engineering is the process of developing services for re-use is …………….. ……………. Application?

a. service design

b. service candidate

c. service implement

d. service oriented

23.

System engineering includes everything to do with procuring, specifying, development, deploying, operating and maintaining both technical and………. System?

a. critical

b. technical

c. sociotechnical

d. computer-based

24.

Which one of the following doesn’t form part of the four overlapping stage in the lifetime of larger complex systems?

a. development

b. procurement

c. operations

d. maintenance

25.

System development is a complex process in which the elements that are part of the system are developed or purchased and then integrated to create the……… systems?

a. partial

b. final

c. agile

d. computer-based

# Random MCQ Notes

## MCQs Per Chapter - 2020

### Chapter 2

1: Which of the following are recognized process flow types?

a. Concurrent process flow

b. Iterative process flow

c. Linear process flow

d. Spiral process flow

e. both a & c

2: Software processes can be constructed out of pre-existing software patterns to best meet the needs of a software project.

a. True

b. False

3: Which of these are standards for assessing software processes?

a. SEI

b. SPICE

c. ISO 9000

d. ISO 9001

E. both b & d

4: The waterfall model of software development is

a. A reasonable approach when requirements are well defined.

b. A good approach when a working program is required quickly.

c. The best approach to use for projects with large development teams.

d. An old fashioned model that is rarely used any more.

5: The incremental model of software development is

a. A reasonable approach when requirements are well defined.

b. A good approach when a working core product is required quickly.

c. The best approach to use for projects with large development teams.

d. A revolutionary model that is not used for commercial products.

6: Evolutionary software process models

a. Are iterative in nature

b. Can easily accommodate product requirements changes

c. Do not generally produce throwaway systems

d. All of the above

7: The prototyping model of software development is

a. A reasonable approach when requirements are well defined.

b. A useful approach when a customer cannot define requirements clearly.

c. The best approach to use for projects with large development teams.

d. A risky model that rarely produces a meaningful product.

8: The spiral model of software development

a. Ends with the delivery of the software product

b. Is more chaotic than the incremental model

c. Includes project risks evaluation during each iteration

d. All of the above

9: The concurrent development model is

a. Another name for concurrent engineering.

b. Defines events that trigger engineering activity state transitions.

c. Only used for development of parallel or distributed systems.

d. Used whenever a large number of change requests are anticipated.

10: The component-based development model is

a. Only appropriate for computer hardware design.

b. Not able to support the development of reusable components.

c. Dependent on object technologies for support.

d. Not cost effective by known quantifiable software metrics.

11: The formal methods model of software development makes use of mathematical methods to

a. Define the specification for computer-based systems

b. Develop defect free computer-based systems

c. Verify the correctness of computer-based systems

d. All of the above

12: Which of these is not one of the phase names defined by the Unified Process model for software development?

a. Inception phase

b. Elaboration phase

c. Construction phase

d. Validation phase

13: Which of these is not a characteristic of Personal Software Process?

a. Emphasizes personal measurement of work product

b. Practitioner requires careful supervision by the project manager

c. Individual practitioner is responsible for estimating and scheduling

d. Practitioner is empowered to control quality of software work products

14: Which of these are objectives of Team Software Process?

a. Accelerate software process improvement

b. Allow better time management by highly trained professionals

c. Build self-directed software teams

d. Show managers how to reduce costs and sustain quality

e. Both b c

15: Process technology tools allow software organizations to compress schedules by skipping unimportant activities.

a. True

b. False

16: It is generally accepted that one cannot have weak software processes and create high quality end products.

a. True

b. False

### Chapter 3

1: Agility is nothing more than the ability of a project team to respond rapidly to change.

a. True

b. False

2: Which of the following is not necessary to apply agility to a software process?

a. Eliminate the use of project planning and testing

b. Only essential work products are produced

c. Process allows team to streamline tasks

d. Uses incremental product delivery strategy

3: How do you create agile processes to manage unpredictability?

a. Requirements gathering must be conducted very carefully

b. Risk analysis must be conducted before planning takes place

c. Software increments must be delivered in short time periods

d. Software processes must adapt to changes incrementally

e. Both c (correction) and d

4: In agile software processes the highest priorities is to satisfy the customer through early and continuous delivery of valuable software.

a. True

b. False

5: Which of the following traits need to exist among the members of an agile software team?

a. Competence

b. Decision-making ability

c. Mutual trust and respect

d. All of the above

6: In agile development it is more important to build software that meets the customers’ needs today than worry about features that might be needed in the future.

a. True

b. False

7: What are the four framework activities found in the Extreme Programming (XP) process model?

a. analysis, design, coding, testing

b. planning, analysis, design, coding

c. planning, analysis, coding, testing

d. planning, design, coding, testing

8: All agile process models conform to a greater or lesser degree to the principles stated in the "Manifesto for Agile Software Development".

a. True

b. False

9: What are the three framework activities for the Adaptive Software Development (ASD) process model?

a. analysis, design, coding

b. feasibility study, functional model iteration, implementation

c. requirements gathering, adaptive cycle planning, iterative development

d. speculation, collaboration, learning

10: Which is not one of the key questions that is answered by each team member at each daily Scrum meeting?

a. What did you do since the last meeting?

b. What obstacles are you encountering?

c. What is the cause of the problems you are encountering?

d. What do you plan to accomplish be the next team meeting?

11: The Dynamic Systems Development Method (DSDM) suggests a philosophy that is based on the Pareto principle (80% of the application can be delivered in 20% of the time required to build the complete application).

a. True

b. False

12: In Feature Driven Development (FDD) a client-valued feature is a client-valued function that can be delivered in two weeks or less.

a. True

b. False

13: Agile Modeling (AM) provides guidance to practitioner during which of these software tasks?

a. Analysis

b. Design

c. Coding

d. Testing

e. both a and b

14: Agile Unified Process uses the classic UP phased activities (inception, elaboration, construction, transition) to help the team visualize the overall process flow.

a. True

b. False

### Chapter 4

1: Software engineering principles have about a three year half-life.

a. True

b. False

2: Which of the following is not one of core principles of software engineering practice?

a. All design should be as simple as possible, but no simpler

b. A software system exists only to provide value to its users.

c. Pareto principle (20% of any product requires 80% of the effort)

d. Remember that you produce others will consume

3: Every communication activity should have a facilitator to make sure that the customer is not allowed to dominate the proceedings.

a. True

b. False

4: The agile view of iterative customer communication and collaboration is applicable to all software engineering practice.

a. True

b. False

5: One reason to involve everyone on the software team in the planning activity is to

a. adjust the granularity of the plan

b. control feature creep

c. get all team members to "sign up" to the plan

d. understand the problem scope

6: Project plans should not be changed once they are adopted by a team.

a. True

b. False

7: Requirements models depict software in which three domains?

a. architecture, interface, component

b. cost, risk, schedule

c. information, function, behavior

d. None of the above

8: The design model should be traceable to the requirements model?

a. True

b. False

9: Teams using agile software practices do not generally create models.

a. True

b. False

10: Which of the following is not one of the principles of good coding?

a. Create unit tests before you begin coding

b. Create a visual layout that aids understanding

c. Refractor the code after you complete the first coding pass

d. Write self-documenting code, not program documentation

11: A successful test I ones that discovers at least one as-yet undiscovered error.

a. True

b. False

12: Which of the following are valid reasons for collecting customer feedback concerning delivered software?

a. Allows developers to make changes to the delivered increment

b. Delivery schedule can be revised to reflect changes

c. Developers can identify changes to incorporate into next increment

d. All of the above

### Chapter 5

1: Requirements engineering is a generic process that does not vary from one software project to another.

a. True

b. False

2: During project inception the intent of the of the tasks are to determine

a. basic problem understanding

b. nature of the solution needed

c. people who want a solution

d. none of the above

e. a, b and c

3: Three things that make requirements elicitation difficult are problems of

a. budgeting

b. scope

c. understanding

d. volatility

e. b, c and d

4: A stakeholder is anyone who will purchase the completed software system under development.

a. True

b. False

5: It is relatively common for different customers to propose conflicting requirements, each arguing that his or her version is the right one.

a. True

b. False

6: Which of the following is not one of the context-free questions that would be used during project inception?

a. What will be the economic benefit from a good solution?

b. Who is behind the request for work?

c. Who will pay for the work?

d. Who will use the solution?

7: In collaborative requirements gathering the facilitator

a. arranges the meeting place

b. can not be a customer

c. controls the meeting

d. must be an outsider

8: Which of the following is not one of the requirement classifications used in Quality Function Deployment (QFD)?

a. exciting

b. expected

c. mandatory

d. normal

9: The work products produced during requirement elicitation will vary depending on the

a. size of the budget

b. size of the product being built

c. software process being used

d. stakeholders needs

10: Developers and customers create use-cases to help the software team understand how different classes of end-users will use functions.

a. True

b. False

11: Use-case actors are always people, never system devices.

a. True

b. False

12: The result of the requirements engineering task is an analysis model that defines which of the following problem domain(s)?

a. information

b. functional

c. behavioral

d. all of the above

13: Analysis patterns facilitate the transformation of the analysis model into a design model by suggesting reliable solutions to common problems.

a. True

b. False

14: In win-win negotiation, the customer’s needs are met even though the developer’s need may not be.

a. True

b. False

15: In requirements validation the requirements model is reviewed to ensure its technical feasibility.

a. True

b. False

### Chapter 8

1: Which of the following are areas of concern in the design model?

a. architecture

b. data

c. interfaces

d. project scope

e. a,b and c

2: The importance of software design can be summarized in a single word

a. accuracy

b. complexity

c. efficiency

d. quality

3: Which of these are characteristics of a good design?

a. exhibits strong coupling between its modules

b. implements all requirements in the analysis model

c. includes test cases for all components

d. provides a complete picture of the software

e. both b and d

4: Which of the following is not a characteristic common to all design methods?

a. configuration management

b. functional component representation

c. quality assessment guidelines

d. refinement heuristics

5: What types of abstraction are used in software design?

a. control

b. data

c. environmental

d. procedural

e. a, b, and d

6: Which of the following can be used to represent the architectural design of a piece of software?

a. Dynamic models

b. Functional models

c. Structural models

d. All of the above

7: Design patterns are not applicable to the design of object-oriented software?

a. True

b. False

8: Since modularity is an important design goal it is not possible to have too many modules in a proposed design.

a. True

b. False

9: Information hiding makes program maintenance easier by hiding data and procedure from unaffected parts of the program.

a. True

b. False

10: Cohesion is a qualitative indication of the degree to which a module

a. can be written more compactly.

b. focuses on just one thing.

c. is able to complete its function in a timely manner.

d. is connected to other modules and the outside world.

11: Coupling is a qualitative indication of the degree to which a module

a. can be written more compactly.

b. focuses on just one thing.

c. is able to complete its function in a timely manner.

d. is connected to other modules and the outside world.

12: When using structured design methodologies the process of stepwise refinement is unnecessary.

a. True

b. False

13: Software designs are refactored to allow the creation of software that is easier to integrate, easier to test, and easier to maintain.

a. True

b. False

14: Which of the following is not one of the five design class types

a. Business domain classes

b. Entity classes

c. Process classes

d. User interface classes

15: Which design model elements are used to depict a model of information represented from the user’s view?

a. Architectural design elements

b. Component-level design elements

c. Data design elements

d. Interface design elements

16: Which design is equivalent to the floor plan of a house?

a. Architectural design

b. Component-level design

c. Data design

d. Interface design

17: Which design model is equivalent to the detailed drawings of the access points and external utilities for a house?

a. Architectural design

b. Component-level design

c. Data design

d. Interface design

18: Which design model is equivalent to a set of detailed drawings for each room in a house?

a. Architectural design

b. Component-level design

c. Data design

d. Interface design

19: The deployment design elements specify the build order for the software components.

a. True

b. False

### Chapter 10

1: In the most general sense a component is a modular building block for computer software.

a. True

b. False

2: In the context of object-oriented software engineering a component contains

a. attributes and operations

b. instances of each class

c. roles for each actor (device or user)

d. set of collaborating classes

3: In traditional software engineering modules must serve in which of the following roles?

a. Control component

b. Infrastructure component

c. Problem domain component

d. All of the above

4: Software engineers always need to create components from scratch in order to meet customer expectations fully.

a. True

b. False

5: Which of the following is not one of the four principles used to guide component-level design?

a. Dependency Inversion Principle

b. Interface Segregation Principle

c. Open-Closed Principle

d. Parsimonious Complexity Principle

6: The use of stereotypes can help identify the nature of components at the detailed design level.

a. True

b. False

7: Classes and components that exhibit functional, layer, or communicational cohesion are relatively easy to implement, test, and maintain.

a. True

b. False

8: Software coupling is a sign of poor architectural design and can always be avoided in every system.

a. True

b. False

9: In component design elaboration requires which of the following elements to be describe in detail?

a. Algorithms

b. Attributes

c. Interfaces

d. Operations

e. b, c and d

10: In component-level design persistent data sources refer to

a. Component libraries

b. Databases

c. Files

d. All of the above

e. both b and c

11: WebApp content design at the component level focuses on content objects and the manner in which they interact.

a. True

b. False

12: A WebApp functional architecture describes the key functional components and how they interact with each other.

a. True

b. False

13: Which of these constructs is used in structured programming?

a. branching

b. condition

c. repetition

d. sequence

e. b, c and d

14: Which of these is a graphical notation for depicting procedural detail?

a. box diagram

b. decision table

c. ER diagram

d. flowchart

15: A decision table should be used

a. to document all conditional statements

b. to guide the development of the project management plan

c. only when building an expert system

d. when a complex set of conditions and actions appears in a component

16: A program design language (PDL) is often a

a. combination of programming constructs and narrative text

b. legitimate programming language in its own right

c. machine readable software development language

d. useful way to represent software architecture

17: In component-based software engineering, the development team examines the requirements to see which are amenable to composition, rather than construction, before beginning detailed design tasks.

a. True

b. False

18: Which of the following is not one of the major activities of domain engineering?

a. analysis

b. construction

c. dissemination

d. validation

19: Which of the following factors would not be considered during component qualification?

a. application programming interface (API)

b. development and integration tools required

c. exception handling

d. testing equipment required

20: Which is the following is a technique used for component wrapping?

a. black-box wrapping

b. clear-box wrapping

c. gray-box wrapping

d. white-box wrapping

21: Which of the following is not one of the issues that form a basis for design for reuse?

a. object-oriented programming

b. program templates

c. standard data

d. standard interface protocols

22: In a reuse environment, library queries are often characterized using the \_\_\_\_\_\_\_\_ element of the 3C Model.

a. concept

b. content

c. context

d. all of the above

### Chapter 17

1: In software quality assurance work there is no difference between software verification and software validation.

a. True

b. False

2: The best reason for using Independent software test teams is that

a. software developers do not need to do any testing

b. strangers will test the software mercilessly

c. testers do not get involved with the project until testing begins

d. the conflicts of interest between developers and testers is reduced

3: What is the normal order of activities in which traditional software testing is organized?

a. integration testing

b. system testing

c. unit testing

d. validation testing

e. c, a, d and b

4: By collecting software metrics and making use of existing software reliability models it is possible to develop meaningful guidelines for determining when software testing is done.

a. True

b. False

5: Which of the following strategic issues needs to be addressed in a successful software testing process?

a. conduct formal technical reviews prior to testing

b. specify requirements in a quantifiable manner

c. use independent test teams

d. wait till code is written prior to writing the test plan

e. both a and b

6: Which of the following need to be assessed during unit testing?

a. algorithmic performance

b. code stability

c. error handling

d. execution paths

e. both a and b

7: Units and stubs are not needed for unit testing because the modules are tested independently of one another.

a. True

b. False

8: Top-down integration testing has as it’s major advantage(s) that

a. low level modules never need testing

b. major decision points are tested early

c. no drivers need to be written

d. no stubs need to be written

e. both b and c

9: Bottom-up integration testing has as it’s major advantage(s) that

a. major decision points are tested early

b. no drivers need to be written

c. no stubs need to be written

d. regression testing is not required

10: Regression testing should be a normal part of integration testing because as a new module is added to the system new

a. control logic is invoked

b. data flow paths are established

c. drivers require testing

d. all of the above

e. both and a and b

11: Smoke testing might best be described as

a. bulletproofing shrink-wrapped software

b. rolling integration testing

c. testing that hides implementation errors

d. unit testing for small programs

12: When testing object-oriented software it is important to test each class operation separately as part of the unit testing process.

a. True

b. False

13: The OO testing integration strategy involves testing

a. groups of classes that collaborate or communicate in some way

b. single operations as they are added to the evolving class implementation

c. operator programs derived from use-case scenarios

d. none of the above

14: Since many WebApps evolve continuously, the testing process must be ongoing as well.

a. True

b. False

15: The focus of validation testing is to uncover places that s user will be able to observe failure of the software to conform to its requirements.

a. True

b. False

16: Software validation is achieved through a series of tests performed by the user once the software is deployed in his or her work environment.

a. True

b. False

17: Configuration reviews are not needed if regression testing has been rigorously applied during software integration.

a. True

b. False

18: Acceptance tests are normally conducted by the

a. developer

b. end users

c. test team

d. systems engineers

19: Recovery testing is a system test that forces the software to fail in a variety of ways and verifies that software is able to continue execution without interruption.

a. True

b. False

20: Security testing attempts to verify that protection mechanisms built into a system protect it from improper penetration.

a. True

b. False

21: Stress testing examines the pressures placed on the user during system use in extreme environments.

a. True

b. False

22: Performance testing is only important for real-time or embedded systems.

a. True

b. False

23: Debugging is not testing, but always occurs as a consequence of testing.

a. True

b. False

24: Which of the following is an approach to debugging?

a. backtracking

b. brute force

c. cause elimination

d. code restructuring

e. a, b, and c

### Chapter 18

1: With thorough testing it is possible to remove all defects from a program prior to delivery to the customer.

a. True

b. False

2: Which of the following are characteristics of testable software?

a. observability

b. simplicity

c. stability

d. all of the above

3: The testing technique that requires devising test cases to demonstrate that each program function is operational is called

a. black-box testing

b. glass-box testing

c. grey-box testing

d. white-box testing

4: The testing technique that requires devising test cases to exercise the internal logic of a software module is called

a. behavioral testing

b. black-box testing

c. grey-box testing

d. white-box testing

5: What types of errors are missed by black-box testing and can be uncovered by white-box testing?

a. behavioral errors

b. logic errors

c. performance errors

d. typographical errors

e. both b and d

6: Program flow graphs are identical to program flowcharts.

a. True

b. False

7: The cyclomatic complexity metric provides the designer with information regarding the number of

a. cycles in the program

b. errors in the program

c. independent logic paths in the program

d. statements in the program

8: The cyclomatic complexity of a program can be computed directly from a PDL representation of an algorithm without drawing a program flow graph.

a. True

b. False

9: Condition testing is a control structure testing technique where the criteria used to design test cases is that they

a. rely on basis path testing

b. exercise the logical conditions in a program module

c. select test paths based on the locations and uses of variables

d. focus on testing the validity of loop constructs

10: Data flow testing is a control structure testing technique where the criteria used to design test cases is that they

a. rely on basis path testing

b. exercise the logical conditions in a program module

c. select test paths based on the locations and uses of variables

d. focus on testing the validity of loop constructs

11: Loop testing is a control structure testing technique where the criteria used to design test cases is that they

a. rely basis path testing

b. exercise the logical conditions in a program module

c. select test paths based on the locations and uses of variables

d. focus on testing the validity of loop constructs

12: Black-box testing attempts to find errors in which of the following categories

a. incorrect or missing functions

b. interface errors

c. performance errors

d. none of the above

e. a, b and c

13: Graph-based testing methods can only be used for object-oriented systems

a. True

b. False

14: Equivalence testing divides the input domain into classes of data from which test cases can be derived to reduce the total number of test cases that must be developed.

a. True

b. False

15: Boundary value analysis can only be used to do white-box testing.

a. True

b. False

16: Orthogonal array testing enables the test designer to maximize the coverage of the test cases devised for relatively small input domains.

a. True

b. False

17: Test derived from behavioral class models should be based on the

a. data flow diagram

b. object-relation diagram

c. state transition diagram

d. use-case diagram

18: Client/server architectures cannot be properly tested because network load is highly variable.

a. True

b. False

19: Real-time applications add a new and potentially difficult element to the testing mix

a. performance

b. reliability

c. security

d. time

### Chapter 19

1: It is not possible to test object-oriented software without including error discovery techniques applied to the system OOA and OOD models.

a. True

b. False

2: The correctness of the OOA and OOD model is accomplished using formal technical reviews by the software quality assurance team.

a. True

b. False

3: The consistency of object-oriented models may be judged by reviewing the CRC card model.

a. True

b. False

4: Test case design for OO software is driven by the algorithmic detail of the individual operations.

a. True

b. False

5: Integration testing of object-oriented software can be accomplished by which of the following testing strategies?

a. Cluster testing

b. Glass-box testing

c. Thread-based testing

d. Use-based testing

e. a, c, and d

6: Validation of object-oriented software focuses on user visible actions and outputs from the system.

a. True

b. False

7: Encapsulation of attributes and operations inside objects makes it easy to obtain object state information during testing.

a. True

b. False

8: Use-cases can provide useful input into the design of black-box and state-based tests of OO software.

a. True

b. False

9: Fault-based testing is best reserved for

a. conventional software testing

b. operations and classes that are critical or suspect

c. use-case validation

d. white-box testing of operator algorithms

10: Testing OO class operations is made more difficult by

a. encapsulation

b. inheritance

c. polymorphism

d. both b and c

11: Scenario-based testing

a. concentrates on actor and software interaction

b. misses errors in specifications

c. misses errors in subsystem interactions

d. both a and b

12: Deep structure testing is not design to

a. object behaviors

b. communication mechanisms

c. exercise object dependencies

d. exercise structure observable by the user

13: Random order tests are conducted to exercise different class instance life histories.

a. True

b. False

14: Which of these techniques is not useful for partition testing at the class level

a. attribute-based partitioning

b. category-based partitioning

c. equivalence class partitioning

d. state-based partitioning

15: Multiple class testing is too complex to be tested using random test cases.

a. True

b. False

## Telegram Upload “More extras”

Which of the following is not a diagram studied in Requirement Analysis ?

a. Use Cases

b. Entity Relationship Diagram

c. State Transition Diagram

d. Activity Diagram

How many feasibility studies is conducted in Requirement Analysis?

a. Two

b. Three

c. Four

d. Five

How many phases are there in Requirement Analysis ?

a. Three

b. Four

c. Five

d. Six

\_\_\_\_\_\_\_\_\_ and \_\_\_\_\_\_\_\_\_ are the two issues of Requirement Analysis.

a. Performance, Design

b. Stakeholder, Developer

c. Functional, Non-Functional

d. Basic, Advanced

Coad and Yourdon suggested \_\_\_\_\_\_\_ selection characteristics that should be used as an analyst considers each potential object for inclusion in the requirement analysis model.

a. Three

b. Four

c. Five

d. Six

Which of the following property does not correspond to a good Software Requirements Specification (SRS)?

a. Verifiable

b. Ambiguous

c. Complete

d. Traceable

Which of the following property of SRS is depicted by the statement : “Conformity to a standard is maintained” ?

a. Correct

b. Complete

c. Consistent

d. Modifiable

## Chapter 8 Additional MyUNISA Questions

1.A tester is executing a test to evaluate, and it complies with the user requirement for a certain field be populated by using a dropdown box containing a list of values, at that time tester is performing \_\_\_\_\_\_\_\_\_\_.

a. White-box Testing

b. Black-box Testing

c. Load Testing

d. Regression Testing

2. Which of the following is non-functional testing for an e-commerce website?

a) People can buy goods

b) people can return faulty goods

c) security of system during transaction

d) 1000 people can log into system at same time.

3.A program with high cyclomatic complexity is almost likely to be:

a) Difficult to test

b) Small

c) Difficult to write

d) Large

4. Regression testing is triggered by changes of the source code, whereas ...... regression testing is triggered by specification changes.

a) progressive, Corrective

b) Corrective, progressive

c) incremental, un-incremental

d) None of above

5. The extent to which the software can control to operate correctly despite the introduction of invalid input is called as

a) Portability

b) Fault tolerance

c) Robustness

d) Reliability

6. Which of the following is likely to benefit most from the use of test tools providing test capture and replay facilities?

a) Regression testing

b) Integration testing

c) System testing

d) User acceptance testing

7. Analyse the following highly simplified procedure:

Ask: “What type of ticket do you require, single or return?”

IF the customer wants ‘return’

Ask: “What rate, Standard or Cheap-day?”

IF the customer replies ‘Cheap-day’

Say: “That will be R11:20”

ELSE

Say: “That will be R19:50”

ENDIF

ELSE

Say: “That will be R9:75”

ENDIF

Now decide the minimum number of tests that are needed to ensure that all

the questions have been asked, all combinations have occurred and all

replies given.

a) 3

b) 4

c) 5

d) 6

8. Which of the following is the best source of Expected Outcomes for User Acceptance Test scripts?

a) Actual results

b) Program specification

c) User requirements

d) System specification

9. What is the main difference between a walkthrough and an inspection?

a) An inspection is led by the author, whilst a walkthrough is led by a trained moderator.

b) An inspection has a trained leader, whilst a walkthrough has no leader.

c) Authors are not present during inspections, whilst they are during walkthroughs.

d) A walkthrough is led by the author, whilst an inspection is led by a trained moderator.

10. A Person who documents all the issues, problems, and open points that were identified during a formal review.

a) Moderator.

b) Scribe

c) Author

d) Manager

# Notes based on common pages asked in MCQs.

## Chapter 1 – Introduction (17)

### Key Points

1. Software engineering is an engineering discipline that is concerned with all aspects of software production.
2. Software is not just a program or programs but also includes all electronic documentation that is needed by system users, quality assurance staff, and developers. Essential software product attributes are maintainability, dependability and security, efficiency, and acceptability.
3. The software process includes all of the activities involved in software development. The high-level activities of specification, development, validation, and evolution are part of all software processes.
4. There are many different types of system, and each requires appropriate software engineering tools and techniques for their development. Few, if any, specific design and implementation techniques are applicable to all kinds of system.
5. The fundamental ideas of software engineering are applicable to all types of software system. These fundamentals include managed software processes, software dependability and security, requirements engineering, and software reuse.
6. Software engineers have responsibilities to the engineering profession and society. They should not simply be concerned with technical issues but should be aware of the ethical issues that affect their work.
7. Professional societies publish codes of conduct that embed ethical and professional standards. These set out the standards of behavior expected of their members.

### Other Notes

#### Frequently asked questions about software engineering (20)

|  |  |
| --- | --- |
| Frequently asked questions about software engineering | |
| 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*** from initial conception to operation and maintenance. |
| 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. |
| What are the key challenges facing software engineering? | Coping with ***increasing diversity, demands for reduced delivery times and developing trustworthy software***. |
| What are the costs of software engineering? | Roughly 60% of software costs are development costs, 40% are testing costs. For custom software, evolution costs often exceed development costs. |
| What are the best software engineering techniques and methods? | While all software projects have to be professionally managed and developed, different techniques are appropriate for different types of system. For example, games should always be developed using a series of prototypes whereas safety critical control systems require a complete and analyzable specification to be developed. There are no methods and techniques that are good for everything. |
| What differences has the Internet made to software engineering? | Not only has the Internet led to the development of massive, highly distributed, service-based systems, it has also supported the creation of an “app” industry for mobile devices which has changed the economics of software. |

#### Four fundamental activities are common to all software processes. (23)

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

## Chapter 2 – Software processes (43)

### Key Points

1. Software processes are the activities involved in producing a software system. Software process models are abstract representations of these processes.
2. General process models describe the organization of software processes. Examples of these general models include the waterfall model, incremental development, and reusable component configuration and integration.
3. Requirements engineering is the process of developing a software specification. Specifications are intended to communicate the system needs of the customer to the system developers.
4. Design and implementation processes are concerned with transforming a requirements specification into an executable software system.
5. Software validation is the process of checking that the system conforms to its specification and that it meets the real needs of the users of the system.
6. Software evolution takes place when you change existing software systems to meet new requirements. Changes are continuous, and the software must evolve to remain useful.
7. Processes should include activities to cope with change. This may involve a prototyping phase that helps avoid poor decisions on requirements and design. Processes may be structured for iterative development and delivery so that changes may be made without disrupting the system as a whole.
8. Process improvement is the process of improving existing software processes to improve software quality, lower development costs, or reduce development time. It is a cyclic process involving process measurement, analysis, and change.

### Other Notes

#### The stages of the waterfall model directly reflect the fundamental software development activities: (47-48)

1. ***Requirements 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. It establishes 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 that 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.

#### Benefits and disadvantages of incremental development over waterfall (50)

Incremental development has three ***major advantages*** over the waterfall model:

1. The ***cost of implementing requirements*** changes is reduced. The amount of analysis and documentation that has to be redone is significantly less than is required with the waterfall model.
2. 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. Customers find it difficult to judge progress from software design documents.
3. Early delivery and deployment of useful software to the customer is possible, even if all of the functionality has not been included. Customers are able to use and gain value from the software earlier than is possible with a waterfall process.

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

1. 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.
2. ***System structure tends to degrade as new increments are added***. Regular change leads to messy code as new functionality is added in whatever way is possible. It becomes increasingly difficult and costly to add new features to a system. To reduce structural degradation and general code messiness, agile methods suggest that you should regularly refactor (improve and restructure) the software.

#### Software tools provide process support by automating some process activities and by providing information about the software that is being developed. For example: (53)

1. The development of graphical system models as part of the requirements specification or the software design
2. The generation of code from these graphical models
3. The generation of user interfaces from a graphical interface description that is created interactively by the user
4. ***Program debugging through the provision of information about an executing program***
5. The automated translation of programs written using an old version of a programming language to a more recent version

#### A prototype is an early version of a software system that is used to demonstrate concepts, try out design options, and find out more about the problem and its possible solutions. (62)

## Chapter 3 – Agile software development (72)

### Key Points

1. Agile methods are iterative development methods that focus on reducing process overheads and documentation and on incremental software delivery. They involve customer representatives directly in the development process.
2. The decision on whether to use an agile or a plan-driven approach to development should depend on the type of software being developed, the capabilities of the development team, and the culture of the company developing the system. In practice, a mix of agile and plan-based techniques may be used.
3. Agile development practices include requirements expressed as user stories, pair programming, refactoring, continuous integration, and test-first development.
4. Scrum is an agile method that provides a framework for organizing agile projects. It is centered around a set of sprints, which are fixed time periods when a system increment is developed. Planning is based on prioritizing a backlog of work and selecting the highest priority tasks for a sprint.
5. To scale agile methods, some plan-based practices have to be integrated with agile practice. These include up-front requirements, multiple customer representatives, more documentation, common tooling across project teams, and the alignment of releases across teams.

### Other Notes

#### Agile approaches to software development consider design and implementation to be the central activities in the software process. They incorporate other activities, such as requirements elicitation and testing, into design and implementation. (74)

#### This <agile> manifesto states: (76)

*We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:*

*Individuals and interactions over processes and tools*

*Working software over comprehensive documentation*

*Customer collaboration over contract negotiation*

*Responding to change over following a plan*

*That is, while there is value in the items on the right, we value the items on the*

*left more.*

#### Extreme programming practices (78)

|  |  |
| --- | --- |
| Extreme programming practices | |
| Collective ownership | The pairs of developers work on all areas of the system, so that no islands of expertise develop and all the developers take responsibility for all of the code. Anyone can change anything. |
| Continuous integration | As soon as the work on a task is complete, it is integrated into the whole system. After any such integration, all the unit tests in the system must pass. |
| Incremental planning | Requirements are recorded on “story cards,” and the stories to be included in a release are determined by the time available and their relative priority. The developers break these stories into development “tasks.” |
| On-site customer | A representative of the end-user of the system (the Customer) should be available full time for the use of the XP team. In an extreme programming process, the customer is a member of the development team and is responsible for bringing system requirements to the team for implementation. |
| Pair programming | Developers work in pairs, checking each other's work and providing the support to always do a good job. |
| Refactoring | All developers are expected to refactor the code continuously as soon as potential code improvements are found. This keeps the code simple and maintainable. |
| Simple design | Enough design is carried out to meet the current requirements and no more. |
| Small releases | The minimal useful set of functionality that provides business value is developed first. Releases of the system are frequent and incrementally add functionality to the first release. |
| Sustainable pace | Large amounts of overtime are not considered acceptable, as the net effect is often to reduce code quality and medium-term productivity. |
| ***Test first development*** | An automated unit test framework is used to write tests for a new piece of functionality before that functionality itself is implemented. |

#### Agile principles and organizational practice (91)

|  |  |
| --- | --- |
| Agile principles and organizational practice | |
| Customer involvement | This depends on having a customer who is willing and able to spend time with the development team and who can represent all system stakeholders. Often, customer representatives have other demands on their time and cannot play a full part in the software development. Where there are external stakeholders, such as regulators, it is difficult to represent their views to the agile team. |
| Embrace change | Prioritizing changes can be extremely difficult, especially in systems for which there are many stakeholders. Typically, each stakeholder gives different priorities to different changes. |
| Incremental delivery | Rapid iterations and short-term planning for development does not always fit in with the longer-term planning cycles of business planning and marketing. Marketing managers may need to know product features several months in advance to prepare an effective marketing campaign. |
| Maintain simplicity | Under pressure from delivery schedules, team members may not have time to carry out desirable system simplifications. |
| People, not process | Individual team members may not have suitable personalities for the intense involvement that is typical of agile methods and therefore may not interact well with other team members. |

#### A fundamental requirement of scaling agile methods is to integrate them with plan driven approaches. (91)

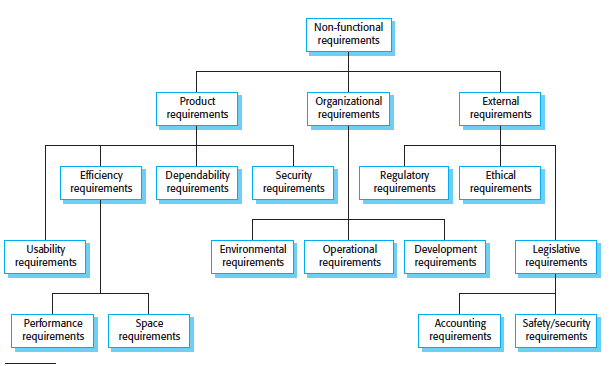
## Chapter 4 – Requirements engineering (101)

### Key Points

1. Requirements for a software system set out what the system should do and define constraints on its operation and implementation.
2. Functional requirements are statements of the services that the system must provide or are descriptions of how some computations must be carried out.
3. Non-functional requirements often constrain the system being developed and the development process being used. These might be product requirements, organizational requirements, or external requirements. They often relate to the emergent properties of the system and therefore apply to the system as a whole.
4. The requirements engineering process includes requirements elicitation, requirements specification, requirements validation, and requirements management.
5. Requirements elicitation is an iterative process that can be represented as a spiral of activities— requirements discovery, requirements classification and organization, requirements negotiation, and requirements documentation.
6. Requirements specification is the process of formally documenting the user and system requirements and creating a software requirements document.
7. The software requirements document is an agreed statement of the system requirements. It should be organized so that both system customers and software developers can use it.
8. Requirements validation is the process of checking the requirements for validity, consistency, completeness, realism, and verifiability.
9. Business, organizational, and technical changes inevitably lead to changes to the requirements for a software system. Requirements management is the process of managing and controlling these changes.

### Other Notes

#### Types of non-functional requirements (108)



#### The aims of the requirements elicitation process are to understand the work that stakeholders do and how they might use a new system to help support that work. (112)

#### There are two fundamental approaches to requirements elicitation: (115)

1. Interviewing, where you talk to people about what they do.
2. Observation or ethnography, where you watch people doing their job to see what artifacts they use, how they use them, and so on.

#### Standard format is used for specifying functional requirements (123)

When a standard format is used for specifying functional requirements, the following

information should be included:

1. A description of the function or entity being specified.
2. ***A description of its inputs and the origin of these inputs.***
3. A description of its outputs and the destination of these outputs.
4. Information about the information needed for the computation or other entities in the system that are required (the “requires” part).
5. A description of the action to be taken.
6. If a functional approach is used, a precondition setting out what must be true before the function is called, and a postcondition specifying what is true after the function is called.
7. A description of the side effects (if any) of the operation.

## Chapter 5 – System modeling (138)

### Key Points

1. A model is an abstract view of a system that deliberately ignores some system details. Complementary system models can be developed to show the system’s context, interactions, structure, and behavior.
2. Context models show how a system that is being modeled is positioned in an environment with other systems and processes. They help define the boundaries of the system to be developed.
3. Use case diagrams and sequence diagrams are used to describe the interactions between users and systems in the system being designed. Use cases describe interactions between a system and external actors; sequence diagrams add more information to these by showing interactions between system objects.
4. Structural models show the organization and architecture of a system. Class diagrams are used to define the static structure of classes in a system and their associations.
5. Behavioral models are used to describe the dynamic behavior of an executing system. This behavior can be modeled from the perspective of the data processed by the system or by the events that stimulate responses from a system.
6. Activity diagrams may be used to model the processing of data, where each activity represents one process step.
7. State diagrams are used to model a system’s behavior in response to internal or external events.
8. Model-driven engineering is an approach to software development in which a system is represented as a set of models that can be automatically transformed to executable code.

### Other Notes

#### You may develop models of both the existing system and the system to be developed: (139)

1. ***Models of the existing system are used during requirements engineering***. They help clarify what the existing system does, and they can be used to focus a stakeholder discussion on its strengths and weaknesses.
2. ***Models of the new system are used during requirements engineering to help explain the proposed requirements to other system stakeholders***. Engineers use these models to discuss design proposals and to document the system for implementation. If you use a model-driven engineering process (Brambilla, Cabot, and Wimmer 2012), you can generate a complete or partial system implementation from system models.

#### You may develop different models to represent the system from different perspectives. (139)

For example:

1. An external perspective, where you model the context or environment of the system.
2. An interaction perspective, where you model the interactions between a system and its environment, or between the components of a system.
3. A structural perspective, where you model the organization of a system or the structure of the data processed by the system.
4. A behavioral perspective, where you model the dynamic behavior of the system and how it responds to events.

#### I therefore concentrate on these five UML diagram types here: (140-141)

1. ***Activity diagrams***, which show the activities involved in a process or in data processing.
2. ***Use case diagrams***, which show the interactions between a system and its environment.
3. ***Sequence diagrams***, which show interactions between actors and the system and between system components.
4. ***Class diagrams***, which show the object classes in the system and the associations between these classes.
5. ***State diagrams***, which show how the system reacts to internal and external events.

#### He suggests the following views: (174)

1. ***A logical view***, which shows the key abstractions in the system as objects or object classes. It should be possible to relate the system requirements to entities in this logical view.
2. ***A process view***, which shows how, at runtime, the system is composed of interacting processes. This view is useful for making judgments about non-functional system characteristics such as performance and availability.
3. ***A development view***, which shows how the software is decomposed for development; that is, it shows the breakdown of the software into components that are implemented by a single developer or development team. This view is useful for software managers and programmers.
4. ***A physical view***, which shows the system hardware and how software components are distributed across the processors in the system. This view is useful for systems engineers planning a system deployment.

## Chapter 6 – Architectural design (167)

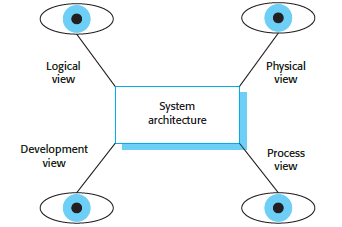
### Key Points

1. A software architecture is a description of how a software system is organized. Properties of a system such as performance, security, and availability are influenced by the architecture used.
2. Architectural design decisions include decisions on the type of application, the distribution of the system, the architectural styles to be used, and the ways in which the architecture should be documented and evaluated.
3. Architectures may be documented from several different perspectives or views. Possible views include a conceptual view, a logical view, a process view, a development view, and a physical view.
4. Architectural patterns are a means of reusing knowledge about generic system architectures. They describe the architecture, explain when it may be used, and point out its advantages and disadvantages.
5. Commonly used Architectural patterns include model-view-controller, layered architecture, repository, client–server, and pipe and filter.
6. Generic models of application systems architectures help us understand the operation of applications, compare applications of the same type, validate application system designs, and assess large-scale components for reuse.
7. Transaction processing systems are interactive systems that allow information in a database to be remotely accessed and modified by a number of users. Information systems and resource management systems are examples of transaction processing systems.
8. Language processing systems are used to translate texts from one language into another and to carry out the instructions specified in the input language. They include a translator and an abstract machine that executes the generated language.

### Other Notes

#### He suggests the following views: (174)

1. ***A logical view***, which shows the key abstractions in the system as objects or object classes. It should be possible to relate the system requirements to entities in this logical view.
2. ***A process view***, which shows how, at runtime, the system is composed of interacting processes. This view is useful for making judgments about non-functional system characteristics such as performance and availability.
3. ***A development view***, which shows how the software is decomposed for development; that is, it shows the breakdown of the software into components that are implemented by a single developer or development team. This view is useful for software managers and programmers.
4. ***A physical view***, which shows the system hardware and how software components are distributed across the processors in the system. This view is useful for systems engineers planning a system deployment.



## Chapter 7 – Design and implementation (196)

### Key Points

1. Software design and implementation are interleaved activities. The level of detail in the design depends on the type of system being developed and whether you are using a plan-driven or agile approach.
2. The process of object-oriented design includes activities to design the system architecture, identify objects in the system, describe the design using different object models, and document the component interfaces.
3. A range of different models may be produced during an object-oriented design process. These include static models (class models, generalization models, association models) and dynamic models (sequence models, state machine models).
4. Component interfaces must be defined precisely so that other objects can use them. A UML interface stereotype may be used to define interfaces.
5. When developing software, you should always consider the possibility of reusing existing software, either as components, services, or complete systems.
6. Configuration management is the process of managing changes to an evolving software system. It is essential when a team of people is cooperating to develop software.
7. Most software development is host-target development. You use an IDE on a host machine to develop the software, which is transferred to a target machine for execution.
8. Open-source development involves making the source code of a system publicly available. This means that many people can propose changes and improvements to the software.

### Other Notes

#### Software design is a creative activity in which you identify software components and their relationships, based on a customer’s requirements. (197)

#### Implementation is the process of realizing the design as a program. (197)

#### To develop a system design from concept to detailed, object-oriented design, you need to: (198)

1. Understand and define the context and the external interactions with the system.
2. Design the system architecture.
3. ***Identify the principal objects in the system.***
4. ***Develop design models.***
5. Specify interfaces.

#### Software reuse is possible at a number of different levels (213):

1. ***The abstraction level*** At this level, you don’t reuse software directly but rather use knowledge of successful abstractions in the design of your software. Design patterns and architectural patterns (covered in Chapter 6) are ways of representing abstract knowledge for reuse.
2. ***The object level*** At this level, you directly reuse objects from a library rather than writing the code yourself. To implement this type of reuse, you have to find appropriate libraries and discover if the objects and methods offer the functionality that you need. For example, if you need to process email messages in a Java program, you may use objects and methods from a JavaMail library.
3. ***The component level*** Components are collections of objects and object classes that operate together to provide related functions and services. You often have to adapt and extend the component by adding some code of your own. An example of component-level reuse is where you build your user interface using a framework. This is a set of general object classes that implement event handling, display management, etc. You add connections to the data to be displayed and write code to define specific display details such as screen layout and colors.
4. ***The system level*** At this level, you reuse entire application systems. This function usually involves some kind of configuration of these systems. This may be done by adding and modifying code (if you are reusing a software product line) or by using the system’s own configuration interface. Most commercial systems are now built in this way where generic application systems systems are adapted and reused. Sometimes this approach may involve integrating several application systems to create a new system.

#### The Gang of Four defined the four essential elements of design patterns in their book on patterns: (210)

1. A name that is a meaningful reference to the pattern.
2. A description of the problem area that explains when the pattern may be applied.
3. A solution description of the parts of the design solution, their relationships and their responsibilities. This is not a concrete design description. It is a template for a design solution that can be instantiated in different ways. This is often expressed graphically and shows the relationships between the objects and object classes in the solution.
4. A statement of the consequences—the results and trade-offs—of applying the pattern. This can help designers understand whether or not a pattern can be used in a particular situation.

#### Open-source development is an approach to software development in which the source code of a software system is published and volunteers are invited to participate in the development process (219).

## Chapter 8 – Software testing (226)

### Key Points

1. Testing can only show the presence of errors in a program. It cannot show that there are no remaining faults.
2. Development testing is the responsibility of the software development team. A separate team should be responsible for testing a system before it is released to customers. In the user testing process, customers or system users provide test data and check that tests are successful.
3. Development testing includes unit testing in which you test individual objects and methods; component testing in which you test related groups of objects; and system testing in which you test partial or complete systems.
4. When testing software, you should try to “break” the software by using experience and guidelines to choose types of test cases that have been effective in discovering defects in other systems.
5. Wherever possible, you should write automated tests. The tests are embedded in a program that can be run every time a change is made to a system.
6. Test-first development is an approach to development whereby tests are written before the code to be tested. Small code changes are made, and the code is refactored until all tests execute successfully.
7. Scenario testing is useful because it replicates the practical use of the system. It involves inventing a typical usage scenario and using this to derive test cases.
8. Acceptance testing is a user testing process in which the aim is to decide if the software is good enough to be deployed and used in its planned operational environment.

### Other notes

#### Know about three distinct types of testing—component testing, system testing, and release testing (226)

#### When you test software, you are trying to do two things: (227)

1. Demonstrate to the developer and the customer that the software meets its requirements. For ***custom software***, this means that there should be at least one test for every requirement in the requirements document. For ***generic software products***, it means that there should be tests for all of the system features that will be included in the product release. You may also test combinations of features to check for unwanted interactions between them.
2. Find inputs or input sequences where the behavior of the software is incorrect, undesirable, or does not conform to its specification. These are caused by defects (bugs) in the software. When you test software to find defects, you are trying to root out undesirable system behavior such as system crashes, unwanted interactions with other systems, incorrect computations, and data corruption.

#### The aim of software validation is to ensure that the software meets the customer’s expectations. (228)

#### There are three stages of development testing: (232)

1. Unit testing, where individual program units or object classes are tested. Unit testing should focus on testing the functionality of objects or methods.
2. Component testing, where several individual units are integrated to create composite components. Component testing should focus on testing the component interfaces that provide access to the component functions.
3. System testing, where some or all of the components in a system are integrated and the system is tested as a whole. System testing should focus on testing component interactions.

#### Some general guidelines for interface testing are: (239)

1. Examine the code to be tested and identify each call to an external component. Design a set of tests in which the values of the parameters to the external components are at the extreme ends of their ranges. These extreme values are most likely to reveal interface inconsistencies.
2. Where pointers are passed across an interface, always test the interface with null pointer parameters.
3. Where a component is called through a procedural interface, design tests that deliberately cause the component to fail. Differing failure assumptions are one of the most common specification misunderstandings.
4. Use stress testing in message passing systems. This means that you should design tests that generate many more messages than are likely to occur in practice. This is an effective way of revealing timing problems.
5. Where several components interact through shared memory, design tests that vary the order in which these components are activated. These tests may reveal implicit assumptions made by the programmer about the order in which the shared data is produced and consumed.

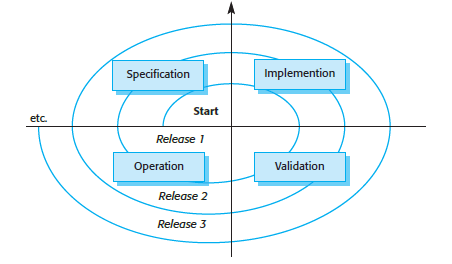
## Chapter 9 – Software evolution (255)

### Key Points

1. Software development and evolution can be thought of as an integrated, iterative process that can be represented using a spiral model.
2. For custom systems, the costs of software maintenance usually exceed the software development costs.
3. The process of software evolution is driven by requests for changes and includes change impact analysis, release planning, and change implementation.
4. Legacy systems are older software systems, developed using obsolete software and hardware technologies, that remain useful for a business.
5. It is often cheaper and less risky to maintain a legacy system than to develop a replacement system using modern technology.
6. The business value of a legacy system and the quality of the application software and its environment should be assessed to determine whether a system should be replaced, transformed, or maintained.
7. There are three types of software maintenance, namely, bug fixing, modifying software to work in a new environment, and implementing new or changed requirements.
8. Software reengineering is concerned with restructuring and redocumenting software to make it easier to understand and change.
9. Refactoring, making small program changes that preserve functionality, can be thought of as preventative maintenance.

### Other Notes

#### Software engineering is therefore a spiral process with requirements, design, implementation, and testing going on throughout the lifetime of the system (256).



#### During the servicing phase, the software is still useful, but only small tactical changes are made to it. During this stage, the company is usually considering how the software can be replaced. (258)

#### This led to so-called Lehman’s Laws, which are said to apply to all large-scale software systems. The most important of these laws are: (271)

1. A program must continually change if it is to remain useful.
2. As an evolving program changes, its structure is degraded.
3. Over a program’s lifetime, the rate of change is roughly constant and independent of the resources available.
4. The incremental change in each release of a system is roughly constant.
5. New functionality must be added to systems to increase user satisfaction.

#### Maintenance prediction is concerned with trying to assess the changes that may be required in a software system and with identifying those parts of the system that are likely to be the most expensive to change. (274)

#### Refactoring is the process of making improvements to a program to slow down degradation through change. (278)

#### Refactoring is a continuous process of improvement throughout the development and evolution process. It is intended to avoid the structure and code degradation that increases the costs and difficulties of maintaining a system. (279)

#### Examples of bad smells that can be improved through refactoring include: (279)

1. Duplicate code The same or very similar code may be included at different places in a program. This can be removed and implemented as a single method or function that is called as required.
2. Long methods If a method is too long, it should be redesigned as a number of shorter methods.
3. Switch (case) statements These often involve duplication, where the switch depends on the type of a value. The switch statements may be scattered around a program. In object-oriented languages, you can often use polymorphism to achieve the same thing.
4. Data clumping Data clumps occur ***when the same group of data items (fields in classes, parameters in methods) reoccurs in several places in a program. These can often be replaced with an object that encapsulates all of the data.***
5. Speculative generality This occurs when developers include generality in a program in case it is required in the future. This can often simply be removed.

## Chapter 10 – Dependable systems (285)

### Key Points

1. System dependability is important because failure of critical computer systems can lead to large economic losses, serious information loss, physical damage or threats to human life.
2. The dependability of a computer system is a system property that reflects the user’s degree of trust in the system. The most important dimensions of dependability are availability, reliability, safety, security, and resilience.
3. Sociotechnical systems include computer hardware, software, and people, and are situated within an organization. They are designed to support organizational or business goals and objectives.
4. The use of a dependable, repeatable process is essential if faults in a system are to be minimized. The process should include verification and validation activities at all stages, from requirements definition through to system implementation.
5. The use of redundancy and diversity in hardware, software processes, and software systems is essential to the development of dependable systems.
6. Formal methods, where a formal model of a system is used as a basis for development, help reduce the number of specification and implementation errors in a system. However, formal methods have had a limited take-up in industry because of concerns about the cost-effectiveness of this approach.

### Other Notes

#### The dependability of systems is usually more important than their detailed functionality for the following reasons: (286)

1. System failures affect a large number of people Many systems include functionality that is rarely used. If this functionality were left out of the system, only a small number of users would be affected. System failures that affect the availability of a system potentially affect all users of the system. Unavailable systems may mean that normal business is impossible.
2. Users often reject systems that are unreliable, unsafe, or insecure If users find that a system is unreliable or insecure, they will refuse to use it. Furthermore, they may also refuse to buy or use other products from the company that produced the unreliable system. They do not want a repetition of their bad experience with an undependable system.
3. ***System failure costs may be enormous*** For some applications, such as a reactor control system or an aircraft navigation system, the cost of system failure is orders of magnitude greater than the cost of the control system. Failures in systems that control critical infrastructure such as the power network have widespread economic consequences.
4. Undependable systems may cause information loss Data is very expensive to collect and maintain; it is usually worth much more than the computer system on which it is processed. The cost of recovering lost or corrupt data is usually very high.

#### There are five principal dimensions to dependability, as I have shown in Figure 10.1. (288)

1. Availability Informally, the availability of a system is the probability that it will be up and running and able to deliver useful services to users at any given time.
2. Reliability Informally, the reliability of a system is the probability, over a given period of time, that the system will correctly deliver services as expected by the user.
3. Safety Informally, the safety of a system is a judgment of how likely it is that the system will cause damage to people or its environment.
4. Security Informally, the security of a system is a judgment of how likely it is that the system can resist accidental or deliberate intrusions.
5. Resilience Informally, the resilience of a system is a judgment of how well that system can maintain the continuity of its critical services in the presence of disruptive events, such as equipment failure and cyberattacks. Resilience is a more recent addition to the set of dependability properties that were originally suggested by Laprie.

#### The dependability of a computer system is a property of the system that reflects its trustworthiness. Trustworthiness here essentially means the degree of confidence a user has that the system will operate as they expect and that the system will not “fail” in normal use. (288)

#### Attributes of dependable processes (298)

|  |  |
| --- | --- |
| Attributes of dependable processes | |
| Auditable | The process should be understandable by people apart from process participants, who can check that process standards are being followed and make suggestions for process improvement. |
| Diverse | The process should include redundant and diverse verification and validation activities. |
| Documentable | The process should have a defined process model that sets out the activities in the process and the documentation that is to be produced during these activities. |
| Robust | The process should be able to recover from failures of individual process activities. |
| Standardized | A comprehensive set of software development standards covering software production and documentation should be available. |

## Chapter 11 – Reliability engineering (306)

### Key Points

1. Software reliability can be achieved by avoiding the introduction of faults, by detecting and removing faults before system deployment, and by including fault-tolerance facilities that allow the system to remain operational after a fault has caused a system failure.
2. Reliability requirements can be defined quantitatively in the system requirements specification. Reliability metrics include probability of failure on demand (POFOD), rate of occurrence of failure (ROCOF), and availability (AVAIL).
3. Functional reliability requirements are requirements for system functionality, such as checking and redundancy requirements, which help the system meet its non-functional reliability requirements.
4. Dependable system architectures are system architectures that are designed for fault tolerance. A number of architectural styles support fault tolerance, including protection systems, self-monitoring architectures, and N-version programming.
5. Software diversity is difficult to achieve because it is practically impossible to ensure that each version of the software is truly independent.
6. Dependable programming relies on including redundancy in a program as checks on the validity of inputs and the values of program variables.
7. Statistical testing is used to estimate software reliability. It relies on testing the system with test data that matches an operational profile, which reflects the distribution of inputs to the software when it is in use.

### Other Notes

#### Brian Randell, a pioneer researcher in software reliability, defined a fault–error–failure model (Randell 2000) based on the notion that human errors cause faults; faults lead to errors, and errors lead to system failures. He defined these terms precisely: (307)

1. Human error or mistake Human behavior that results in the introduction of faults into a system. For example, in the wilderness weather system, a programmer might decide that the way to compute the time for the next transmission is to add 1 hour to the current time. This works except when the transmission time is between 23.00 and midnight (midnight is 00.00 in the 24-hour clock).
2. System fault A characteristic of a software system that can lead to a system error. The fault in the above example is the inclusion of code to add 1 to a variable called Transmission\_time, without a check to see if the value of Transmission\_time is greater than or equal to 23.00.
3. System error An erroneous system state during execution that can lead to system behavior that is unexpected by system users. In this example, the value of the variable Transmission\_time is set incorrectly to 24.XX rather than 00.XX when the faulty code is executed.
4. System failure An event that occurs at some point in time when the system does not deliver a service as expected by its users. In this case, no weather data is transmitted because the time is invalid.

#### More precise definitions of availability and reliability are: (309)

1. Reliability The probability of failure-free operation over a specified time, in a given environment, for a specific purpose.
2. Availability The probability that a system, at a point in time, will be operational and able to deliver the requested services.

#### Three metrics may be used to specify reliability and availability: (313)

1. Probability of failure on demand (POFOD) If you use this metric, you define the probability that a demand for service from a system will result in a system failure. So, POFOD = 0.001 means that there is a 1/1000 chance that a failure will occur when a demand is made.
2. Rate of occurrence of failures (ROCOF) This metric sets out the probable number of system failures that are likely to be observed relative to a certain time period (e.g., an hour), or to the number of system executions. In the example above, the ROCOF is 1/1000. The reciprocal of ROCOF is the mean time to failure (MTTF), which is sometimes used as a reliability metric. MTTF is the average number of time units between observed system failures. A ROCOF of two failures per hour implies that the mean time to failure is 30 minutes.
3. Availability (AVAIL) AVAIL is the probability that a system will be operational when a demand is made for service. Therefore, an availability of 0.9999 means that, on average, the system will be available for 99.99% of the operating time. Figure 11.4 shows what different levels of availability mean in practice.

#### As I explained in Chapter 8, reviews and inspections are used alongside program testing as part of the general process of software verification and validation. (710 – chapter 24?)

## Chapter 12 – Safety engineering (339)

### Key Points

1. Safety-critical systems are systems whose failure can lead to human injury or death.
2. A hazard-driven approach may be used to understand the safety requirements for safety-critical systems. You identify potential hazards and decompose them (using methods such as fault tree analysis) to discover their root causes. You then specify requirements to avoid or recover from these problems.
3. It is important to have a well-defined, certified process for safety-critical systems development. The process should include the identification and monitoring of potential hazards.
4. Static analysis is an approach to V & V that examines the source code (or other representation) of a system, looking for errors and anomalies. It allows all parts of a program to be checked, not just those parts that are exercised by system tests.
5. Model checking is a formal approach to static analysis that exhaustively checks all states in a system for potential errors.
6. Safety and dependability cases collect all of the evidence that demonstrates a system is safe and dependable. Safety cases are required when an external regulator must certify the system before it is used.

### Other Notes

#### Safety-critical systems are systems in which it is essential that system operation is always safe. (341)

#### A hazard is a system state that could lead to an accident. (342)

#### The starting point for generating functional safety requirements is usually domain knowledge, safety standards, and regulations. (344)

#### Hazard analysis is the process of discovering the root causes of hazards in a safety-critical system. (349)

## Chapter 15 – Software reuse (437)

### Key Points

1. There are many different ways to reuse software. These range from the reuse of classes and methods in libraries to the reuse of complete application systems.
2. The advantages of software reuse are lower costs, faster software development, and lower risks. System dependability is increased. Specialists can be used more effectively by concentrating their expertise on the design of reusable components.
3. Application frameworks are collections of concrete and abstract objects that are designed for reuse through specialization and the addition of new objects. They usually incorporate good design practice through design patterns.
4. Software product lines are related applications that are developed from one or more base applications. A generic system is adapted and specialized to meet specific requirements for functionality, target platform, or operational configuration.
5. Application system reuse is concerned with the reuse of large-scale, off-the-shelf systems. These provide a lot of functionality, and their reuse can radically reduce costs and development time. Systems may be developed by configuring a single, generic application system or by integrating two or more application systems.
6. Potential problems with application system reuse include lack of control over functionality, performance, and system evolution; the need for support from external vendors; and difficulties in ensuring that systems can interoperate.

### Other Notes

#### Reuse-based software engineering is an approach to development that tries to maximize the reuse of existing software. (438)

#### Benefits of software reuse (439)

|  |  |
| --- | --- |
| Benefits of software reuse | |
| Accelerated development | Bringing a system to market as early as possible is often more important than overall development costs. Reusing software can speed up system production because both development and validation time may be reduced. |
| Effective use of specialists | Instead of doing the same work over and over again, application specialists can develop reusable software that encapsulates their knowledge. |
| Increased dependability | Reused software, which has been tried and tested in working systems, should be more dependable than new software. Its design and implementation faults should have been found and fixed. |
| Lower development costs | Development costs are proportional to the size of the software being developed. Reusing software means that fewer lines of code have to be written. |
| Reduced process risk | The cost of existing software is already known, while the costs of development are always a matter of judgment. This is an important factor for project management because it reduces the margin of error in project cost estimation. This is especially true when large software components such as subsystems are reused. |
| Standards compliance | Some standards, such as user interface standards, can be implemented as a set of reusable components. For example, if menus in a user interface are implemented using reusable components, all applications present the same menu formats to users. The use of standard user interfaces improves dependability because users make fewer mistakes when presented with a familiar interface. |

#### Key factors that you should consider when planning reuse are: (441-442)

1. The development schedule for the software
2. The expected software lifetime
3. The background, skills and experience of the development team
4. The criticality of the software and its non-functional requirements
5. The application domain
6. The platform on which the system will run

## Chapter 18 – Service-oriented software engineering (520)

### Key Points

1. Service-oriented architecture is an approach to software engineering where reusable, standardized services are the basic building blocks for application systems.
2. Services may be implemented within a service-oriented architecture using a set of XML-based web service standards. These include standards for service communication, interface definition, and service enactment in workflows.
3. Alternatively, a RESTful architecture may be used, which is based on resources and standard operations on these resources. A RESTful approach uses the http and https protocols for service communication and maps operations on the standard http verbs POST, GET, PUT, and DELETE.
4. Services may be classified as utility services that provide a general-purpose functionality, business services that implement part of a business process, or coordination services that coordinate the execution of other services.
5. The service engineering process involves identifying candidate services for implementation, defining the service interface, and implementing, testing, and deploying the service.
6. The development of software using services is based on the idea that programs are created by composing and configuring services to create new composite services and systems.
7. Graphical workflow languages, such as BPMN, may be used to describe a business process and the services used in that process. These languages can describe interactions between the organizations that are involved.

### Other Notes

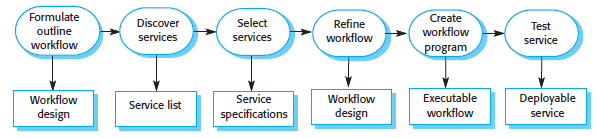
#### Service-oriented systems have loosely coupled architectures where service bindings may change during system execution. (522)

#### XML-based service descriptions include definitions of XML namespaces. (528)

#### REST is an architectural style based on transferring representations of resources from a server to a client. (530)

#### I have shown the six key stages in the process of system construction by composition in (543)

1. Formulate outline workflow In this initial stage of service design, you use the requirements for the composite service as a basis for creating an “ideal” service design. You should create a fairly abstract design at this stage, with the intention of adding details once you know more about available services.
2. Discover services During this stage of the process, you look for existing services to include in the composition. Most service reuse is within enterprises, so this may involve searching local service catalogs. Alternatively, you may search the services offered by trusted service providers, such as Oracle and Microsoft.
3. Select possible services From the set of possible service candidates that you have discovered, you then select possible services that can implement workflow activities. Your selection criteria will obviously include the functionality of the services offered. They may also include the cost of the services and the quality of service (responsiveness, availability, etc.) offered.
4. Refine workflow On the basis of information about the services that you have selected, you then refine the workflow. This involves adding detail to the abstract description and perhaps adding or removing workflow activities. You may then repeat the service discovery and selection stages. Once a stable set of services has been chosen and the final workflow design established, you move on to the next stage in the process.
5. Create workflow program During this stage, the abstract workflow design is transformed to an executable program and the service interface is defined. You can implement workflow programs using a programming language, such as Java or C#, or by using a workflow language, such as BPMN (explained below). This stage may also involve the creation of web-based user interfaces to allow the new service to be accessed from a web browser.
6. Test completed service or application The process of testing the completed, composite service is more complex than component testing in situations where external services are used.



## Chapter 19 – Systems engineering (551)

### Key Points

1. Systems engineering is concerned with all aspects of specifying, buying, designing, and testing complex sociotechnical systems.
2. Sociotechnical systems include computer hardware, software, and people, and are situated within an organization. They are designed to support organizational or business goals and objectives.
3. The emergent properties of a system are characteristics of the system as a whole rather than of its component parts. They include properties such as performance, reliability, usability, safety, and security.
4. The fundamental systems engineering processes are conceptual systems design, system procurement, system development, and system operation.
5. Conceptual systems design is a key activity where high-level system requirements and a vision of the operational system is developed.
6. System procurement covers all of the activities involved in deciding what system to buy and who should supply that system. Different procurement processes are used for off-the-shelf application systems, configurable COTS systems, and custom systems.
7. System development processes include requirements specification, design, construction, integration, and testing.
8. When a system is put into use, the operational processes and the system itself inevitably change to reflect changes to the business requirements and the system’s environment.

### Other Notes

#### System development is a complex process in which the elements that are part of the system are developed or purchased and then integrated to create the final system. (570)

#### There are four overlapping stages (Figure 19.1) in the lifetime of large, complex systems: (553-554)

1. Conceptual design This initial systems engineering activity develops the concept of the type of system that is required. It sets out, in nontechnical language, the purpose of the system, why it is needed, and the high-level features that users might expect to see in the system. It may also describe broad constraints, such as the need for interoperability with other systems. These limit the freedom of systems engineers in designing and developing the system.
2. Procurement or acquisition During this stage, the conceptual design is further developed so that information is available to make decisions about the contract for the system development. This may involve making decisions about the distribution of functionality across hardware, software, and operational processes. You also make decisions about which hardware and software has to be acquired, which suppliers should develop the system, and the terms and conditions of the supply contract.
3. Development During this stage, the system is developed. Development processes include requirements definition, system design, hardware and software engineering, system integration, and testing. Operational processes are defined, and the training courses for system users are designed.
4. Operation At this stage, the system is deployed, users are trained, and the system is brought into use. The planned operational processes usually then have to change to reflect the real working environment where the system is used. Over time, the system evolves as new requirements are identified. Eventually, the system declines in value, and it is decommissioned and replaced.

#### In a feasibility study, you should look at comparable systems that have been developed elsewhere and technological issues (e.g., use of mobile devices) that may affect use of the system. Then you need to assess whether or not the system could be implemented using current hardware and software technologies. (565)