

18CSC206J SOFTWARE ENGINEERING AND PROJECT MANAGEMENT
UNIT-I

INTRODUCTION TO SOFTWARE ENGINEERING AND SOFTWARE PROJECT MANAGEMENT

1 MARKS

S.NO	QUESTION	LEVEL	CLO	PG. NO
1.	Define Software. a) Software is set of programs b) Software is documentation and configuration of data c) Software is set of programs, documentation & configuration of data d) none of the mentioned	1	1	6
2.	“Software engineers should not use their technical skills to <i>misuse</i> other people’s computers. “Here the term <i>misuse</i> refers to a) Unauthorized access to computer material b) Unauthorized modification of computer material c) Dissemination of viruses or other malware d) All of the mentioned	1	1	12
3.	The below one that does not account for software failure. a) Increasing Demand b) Low expectation c) Increasing Supply d) Less reliable and expensive	1	1	11
4.	Select the one that does not affect different types of software as a whole? a) Heterogeneity b) Flexibility c) Business and social change d) Security	1	1	70
5.	Build & Fix Model is suitable for programming exercises of _____ LOC (Line of Code). a) 100-200 b) 200-400 c) 400-1000 d) above 1000	1	1	88
6.	Identify one of the following models is not suitable for accommodating any change. a) Build & Fix Model b) Prototyping Model c) RAD Model d) Waterfall Model	1	1	28

7.	Identify the one is not the types of prototype of Prototyping Model. a) Horizontal Prototype b) Vertical Prototype c) Diagonal Prototype d) Domain Prototype	1	1	30
8.	Identify the model can be selected if user is involved in all the phases of SDLC. a) Waterfall Model b) Prototyping Model c) RAD Model d) both Prototyping Model & RAD Model	1	1	34
9.	SDLC stands for a) Software Development Life Cycle b) System Development Life cycle c) Software Design Life Cycle d) System Design Life Cycle	1	1	36
10.	List the major drawback of using RAD Model. a) Highly specialized & skilled developers/designers are required b) Increases reusability of components c) Encourages customer/client feedback d) Increases reusability of components, Highly specialized & skilled developers/designers are required	1	1	32
11.	RAD Model has a) 2 phases b) 3 phase c) 5 phases d) 6 phases	1	1	32
12.	Identify the one of the following is not an Evolutionary Process Model. a) WINWIN Spiral Model b) Incremental Model c) Concurrent Development Model d) All of the mentioned	1	1	34
13.	The Incremental Model is a result of combination of elements of which two models. a) Build & FIX Model & Waterfall Model b) Linear Model & RAD Model c) Linear Model & Prototyping Model d) Waterfall Model & RAD Model	1	1	36
14.	Choose the major advantage of using Incremental Model. a) Customer can respond to each increment b) Easier to test and debug	1	1	36

c) It is used when there is a need to get a product to the market early d) Easier to test and debug & It is used when there is a need to get a product to the market early			
15. The spiral model was originally proposed by a) IBM b) Barry Boehm c) Pressman d) Royce	1	1	36
16. The spiral model has two dimensions namely _____ and _____ [CLO-1] [T1 Page no:37] a) diagonal, angular b) radial, perpendicular c) radial, angular d) diagonal, perpendicular	1	1	37
17. Recognize WINWIN Spiral Model different from Spiral Model. a) It defines tasks required to define resources, timelines, and other project related information b) It defines a set of negotiation activities at the beginning of each pass around the spiral c) It defines tasks required to assess both technical and management risks d) It defines tasks required to construct, test, install, and provide user support	1	1	38
18. Identify the disadvantage of Spiral Model. a) Doesn't work well for smaller projects b) High amount of risk analysis c) Strong approval and documentation control d) Additional Functionality can be added at a later date	1	1	39
19. If you were to create client/server applications, which model would you go for. a) WINWIN Spiral Model b) Spiral Model c) Concurrent Model d) Incremental Mode	1	1	41
20. Selection of a model is based on a) Requirements b) Development team & Users c) Project type and associated risk d) All of the mentioned	1	1	56
21. Identify two models don't allow defining requirements early in the cycle. a) Waterfall & RAD	1	1	44

b) Prototyping & Spiral			
c) Prototyping & RAD			
d) Waterfall & Spiral			
22. Identify the following life cycle model can be chosen if the development team has less experience on similar projects.	1	1	42
a) Spiral			
b) Waterfall			
c) RAD			
d) Iterative Enhancement Model			
23. If you were a lead developer of a software company and you are asked to submit a project/product within a stipulated time-frame with no cost barriers, which model would you select.	1	1	32
a) Waterfall			
b) Spiral			
c) RAD			
d) Incremental			
24. Identify the two of the following models will not be able to give the desired outcome if user's participation is not involved.	1	1	30
a) Waterfall & Spiral			
b) RAD & Spiral			
c) RAD & Waterfall			
d) RAD & Prototyping			
25. A company is developing an advance version of their current software available in the market, what model approach would they prefer.	1	1	44
a) RAD			
b) Iterative Enhancement			
c) Both RAD & Iterative Enhancement			
d) Spiral			
26. Spiral Model has high reliability requirements.	1	1	46
a) True			
b) False			
27. Choose one of the following is not a software process quality.	1	1	57
a) Productivity			
b) Portability			
c) Timeliness			
d) Visibility			
28. _____ & _____ are two kinds of software products.	1	1	57
[CLO-1] [T1 Page no: 57]			
a) CAD, CAM			
b) Firmware, Embedded			
c) Generic, Customized			

d) None of the mentioned			
29. Identify one of the following is not an application of embedded software product.	1	1	58
a) keypad control of a security system b) pattern recognition game playing c) digital function of dashboard display in a car d) none of the mentioned			
30. Purpose of process is to deliver software a) in time b) with acceptable quality c) that is cost efficient d) both in time & with acceptable quality	1	1	61
31. The work associated with software engineering can be categorized into three generic phases, regardless of application area, project size, or complexity namely the_____ phase which focuses on <i>what</i> , the_____ phase which focuses on <i>how</i> and the_____ phase which focuses on <i>change</i> . i. support ii. Development iii. Definition a) 1, 2, 3 b) 2, 1, 3 c) 3, 2, 1 d) 3, 1, 2	1	1	68
32. Identify the following activities of a Generic Process framework provides a feedback report. a) Communication b) Planning c) Modeling & Construction d) Deployment	1	1	57
33. Identify one of the following is not an Umbrella Activity that complements the five process framework activities and help team manage and control progress, quality, change, and risk. a) Reusability management b) Risk management c) Measurement d) User Reviews	1	1	23
34. Four types of change are encountered during the support phase. Identify one of the following is not one that falls into such category. a) Translation b) Correction c) Adaptation	1	1	22

d) Prevention			
35. Choose an internal software quality from given below:	1	1	95
a) scalability			
b) usability			
c) reusability			
d) reliability			
36. RUP stands for _____ created by a division of _____	1	1	57
a) Rational Unified Program, IBM			
b) Rational Unified Process, Infosys			
c) Rational Unified Process, Microsoft			
d) Rational Unified Process, IBM			
37. Name the phase of the RUP is used to establish a business case for the system.	1	1	51
a) Transition			
b) Elaboration			
c) Construction			
d) Inception			
38. The longer a fault exists in software	1	1	42
a) the more tedious its removal becomes			
b) the more costly it is to detect and correct			
c) the less likely it is to be properly corrected			
d) All of the mentioned			
39. Select the option that suits the Manifesto for Agile Software Development	1	1	59
a) Individuals and interactions			
b) Working software			
c) Customer collaboration			
d) All of the mentioned			
40. Agile Software Development is based on	1	1	59
a) Incremental Development			
b) Iterative Development			
c) Linear Development			
d) Both Incremental and Iterative Development			
41. Identify the following is not an agile method.	1	1	60
a) XP			
b) 4GT			
c) AUP			
d) All of the mentioned			
42. Determine plan driven development different from agile development.	1	1	63
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 mentioned

43. Number of phases is there in Scrum. 1 1 72
 a) Two
b) Three
 c) Four
 d) Scrum is an agile method which means it does not have phases
44. Identify the following does not apply to agility to a software process. 1 1 62
 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 mentioned
45. Determine three framework activities are present in Adaptive Software Development (ASD). 1 1 76
 a) analysis, design, coding
 b) requirements gathering, adaptive cycle planning, iterative development
c) speculation, collaboration, learning
 d) all of the mentioned
46. 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. 1 1 66
a) True
 b) False
47. In XP, as soon as the work on a task is complete, it is integrated into the whole system. 1 1 65
a) True
 b) False
48. User requirements are expressed as _____ in Extreme Programming. 1 1 66
 a) implementation tasks
 b) functionalities
c) scenarios
 d) none of the mentioned
49. 48. Will a customer involved test development and validation in XP. 1 1 66
 a) Yes
 b) No
c) It may vary from Customer to Customer
 d) None of the mentioned
50. List the four framework activities are found in the Extreme Programming (XP). 1 1 66
 a) analysis, design, coding, testing
 b) planning, analysis, design, coding

- c) **planning, design, coding, testing**
- d) planning, analysis, coding, testing

4 MARKS:

1. Define software project?

L1 CLO2

A Software Project is the complete procedure of software development from requirement gathering to testing and maintenance, carried out according to the execution methodologies, in a specified period of time to achieve intended software product.

2. What is the need for software project management?

L2 CLO2

Software is said to be an intangible product. Software development is a kind of all new stream in world business and there's very little experience in building software products. Most software products are tailor made to fit client's requirements. The most important is that the underlying technology changes and advances so frequently and rapidly that experience of one product may not be applied to the other one. All such business and environmental constraints bring risk in software development hence it is essential to manage software projects efficiently.

It is necessary for an organization to deliver quality product, keeping the cost within client's budget constrain and deliver the project as per scheduled. Hence in order, software project management is necessary to incorporate user requirements along with budget and time constraints.

3. What is software project planning?

L1 CLO2

Software project planning is task, which is performed before the production of software actually starts. It is there for the software production but involves no concrete activity that has any direction connection with software production; rather it is a set of multiple processes, which facilitates software production.

4. Mention the uses of prototyping paradigm

L2 CLO2

Prototype methodology is defined as a Software Development model in which a prototype is built, tests, and then reworked when needed until an acceptable prototype is achieved. It also creates a base to produce the final system.

Software prototyping model works best in scenarios where the project's requirement are not known. It is an iterative, trial, and error method which take place between the developer and the client.

5. Mention the disadvantages of water fall model.

L2 CLO2

Once an application is in the **testing** stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.

No working software is produced until late during the life cycle.

High amounts of risk and uncertainty.

Not a good model for complex and object-oriented projects.

Poor model for long and ongoing projects.

Not suitable for the projects where requirements are at a moderate to high risk of changing.

6. Identify the stages involved in a typical project life cycle L2 CLO2
 Requirements analysis, Specification, Design, Coding , Verification & Validation, Implementation installation, Maintenance & Support
7. Discuss the advantages of spiral model L2 CLO2
 The main advantages of spiral model is, it is realistic and typifies most software development products/projects. It combines the best features of most of the earlier models. It strikes a good balance mechanism for early problem identification and correction while not missing out proactive problem prevention.
8. Difference between verification and validation L2 CLO2

Verification

1. **Verification** is a static practice of verifying documents, design, code and program.
2. It does not involve executing the code.
3. It is human based checking of documents and files.
4. **Verification** uses methods like inspections, reviews, walkthroughs, and Desk-checking etc.
5. **Verification** is to check whether the software conforms to specifications.
6. It can catch errors that validation cannot catch. It is low level exercise.
7. Target is requirements specification, application and software architecture, high level, complete design, and database design etc.

Validation

1. **Validation** is a dynamic mech and testing the actual product.
2. It always involves executing the code.
3. It is computer based execution.
4. **Validation** uses methods (functional) testing, gray box testing, (structural) testing etc.
5. **Validation** is to check whether customer expectations and requirements are met.
6. It can catch errors that verification cannot catch. It is High Level Exercise.
7. Target is actual product-a unit, integrated modules, and effective system.

9. When to use RAD model? L2 CLO2
 RAD should be used when there is a need to create a system that can be modularized in 2-3 months of time. It should be used if there's high availability of designers for modeling and the budget is high enough to afford their cost along with the cost of automated code generating tools.

RAD SDLC model should be chosen only if resources with high business knowledge are available and there is a need to produce the system in a short span of time (2-3 months).

10. List out the merits of incremental model.

L1 CLO2

The merits of incremental model are :

The incremental model can be adopted when there are less number of people involved in the project.

Technical risks can be managed with each increment.

For a very small time span, at least core product can be delivered to the customer.

11. Analyze the characteristics which makes software projects different from other project? L2 CLO2

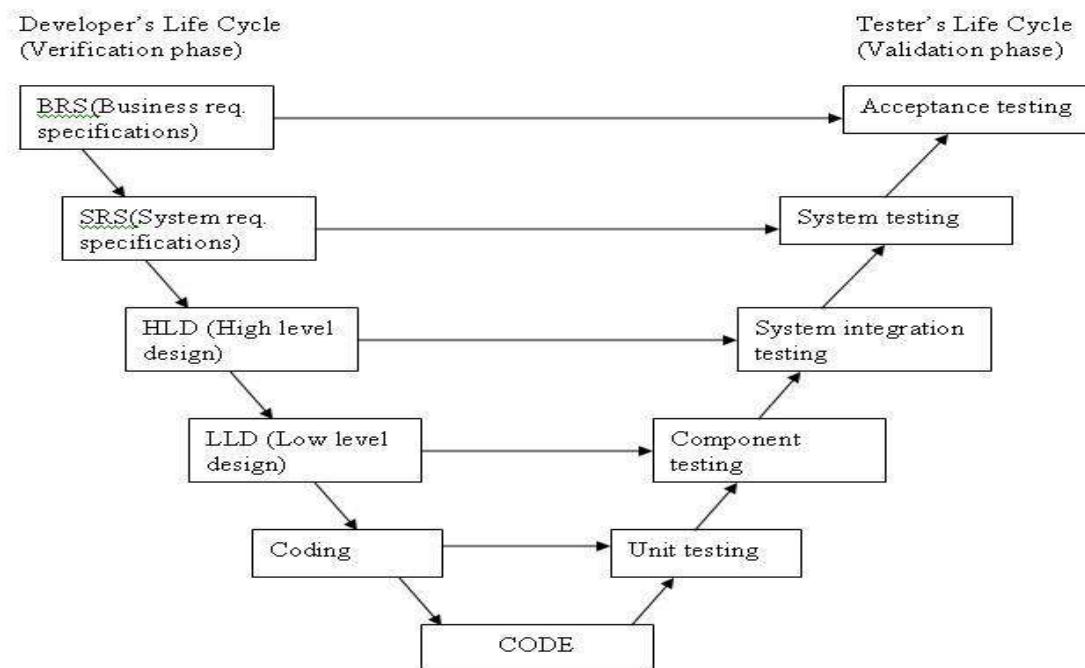
Invisibility: When a physical artifact such as a bridge or road is being constructed the progress being made can actually be seen. With software, progress is not immediately visible. **Complexity:** Per dollar, pound or euro spent, software products contain more complexity than other engineered artifacts.

Flexibility: The ease with which software can be changed is usually seen as one of its strengths. However this means that where the software system interfaces with a physical or organizational system, it is expected that, where necessary, the software will change to accommodate the other components rather than vice versa. This means the software systems are likely to be subject to a high degree of change

12. What is V-model?

L1 CLO2

V- Model means Verification and Validation model. V-Shaped life cycle is a sequential path of execution of processes. Each phase must be completed before the next phase begins.



13. List out the differences between waterfall model and spiral model

L2 CLO2

WATERFALL MODEL**SPIRAL MODEL**

Waterfall model works in sequential method.

In waterfall model errors or risks are identified and rectified after the completion of stages.

Waterfall model is adopted by customers.

Waterfall model is applicable for small project.

In waterfall model requirements and early stage planning is necessary.

Flexibility to change in waterfall model is Difficult.

There is high amount risk in waterfall model.

While spiral model works in evolutionary method.

In spiral model errors or risks are identified and rectified earlier.

While spiral model is adopted by developers.

While Spiral model is used for large project.

While in spiral model requirements and early stage planning is necessary if required.

Flexibility to change in spiral model is not Difficult.

There is low amount risk in spiral model.

14. Explain Scrum?

L2 CLO2

Scrum principles are consistent with the agile manifesto and are used to guide development activities within a process that incorporates the following framework activities: requirements, analysis, design, evolution, and delivery. Within each framework activity, work tasks occur within a process pattern (discussed in the following paragraph) called a sprint. The work conducted within a sprint (the number of sprints required for each framework activity will vary depending on product complexity and size) is adapted to the problem at hand and is defined and often modified in real time by the Scrum team. The overall flow of the Scrum process is illustrated

15. What is requirement engineering?

L1 CLO2

The broad spectrum of tasks and techniques that lead to an understanding of requirements is called requirements engineering. From a software process perspective, requirements engineering is a major software engineering action that begins during the communication activity and continues into the modeling activity

Requirements engineering builds a bridge to design and construction.

Requirements engineering provides the appropriate mechanism for understanding what the customer wants, analyzing need, assessing feasibility, negotiating a reasonable solution, specifying the solution unambiguously, validating the specification, and managing the requirements as they are transformed into an operational system

16. List out the tasks included in requirement engineering

L2 CLO2

inception, elicitation, elaboration, negotiation, specification, validation, and management.

17. Define requirement elicitation

L1 CLO2

Requirements elicitation (also called requirements gathering) combines elements of problem solving, elaboration, negotiation, and specification. In order to encourage a collaborative, team-oriented approach to requirements gathering, stakeholders work together to identify the problem, propose elements of the solution, negotiate different approaches and specify a preliminary set of solution requirements

18. List out the different approaches to collaborative requirements gathering.

L2 CLO2

Meetings are conducted and attended by both software engineers and other stakeholders.

Rules for preparation and participation are established.

An agenda is suggested that is formal enough to cover all important points but informal enough to encourage the free flow of ideas.

A “facilitator” (can be a customer, a developer, or an outsider) controls the meeting.

A “definition mechanism” (can be work sheets, flip charts, or wall stickers or an electronic bulletin board, chat room, or virtual forum) is used.

19. What are the different techniques to estimate the size of a program

L2 CLO2

Two techniques to estimate size of program:-

1. LOC (Line Of Code) -- A line of code is any line of program text that is not a comment or blank line, regardless of the number of statements or fragments of statements on the line. This specifically includes all line containing program header, declarations, executable and nonexecutable statements.

2. Function count – It measures the functionality from the user’s point of view that is on the basis of what the user request and receives in return. Therefore it deals with the functionality being delivered and not with the line of code, source modules, files etc.

20. Define COCOMO model?

L1 CLO2

The Constructive Cost Model (COCOMO) is an algorithmic software cost estimation model developed by Barry W. Boehm. The model uses a basic regression formula with parameters that are derived from historical project data and current as well as future project characteristics. COCOMO consists of a hierarchy of three increasingly detailed and accurate forms. The first level, Basic COCOMO is good for quick, early, rough order of magnitude estimates of software costs, but its accuracy is limited due to its lack of factors to account for difference in project attributes (Cost Drivers). Intermediate COCOMO takes these Cost Drivers into account and Detailed COCOMO additionally accounts for the influence of individual project phases.

21. Outline the umbrella activities of a software process?

L2 CLO2

The umbrella activities of a software process are:

Software project tracking and control.

Risk Management.

Software Quality Assurance.

Formal Technical Reviews

Software Configuration Management

Work product preparation and production

Reusability management, Measurement.

22. Which of the software engineering paradigms would be most effective? Why?

L2 CLO2

Incremental / Spiral model will be most effective.

Reasons :

It combines linear sequential model with iterative nature of prototyping.

Focuses on delivery of product at each increment.

Can be planned to manage technical risks.

23. What is Risk and explain the risk management process.

L2 CLO2

"Risk" is a problem that could cause some loss or threaten the progress of the project, but which has not happened yet. These potential issues might harm cost, schedule or technical success of the project and the quality of our software device, or project team morale.

Risk Management is the system of identifying addressing and eliminating these problems before they can damage the project.

24. Illustrate the types of risks.

L2 CLO2

There are different types of risks which can affect a software project:

Resource risk

Technology risk

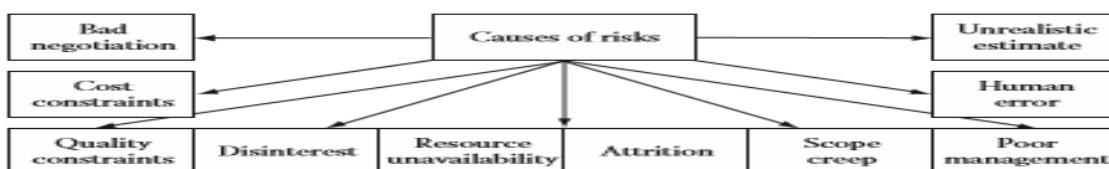
Budget risk

Quality risk

Time risk

25. Find out the causes of risk

L2 CLO2



12 Marks:

1. What are the major differences between system engineering and software engineering? L2 CLO2
State explains the stages that distinguish the two.
2. Explain with two examples of software development projects would be amenable to evolutionary prototyping. Why is evolutionary prototyping suitable in these cases? L2 CLO2
3. Explain Water fall Model. What are the problems that are sometimes encountered when the waterfall model is applied? L2 CLO2
4. Which is more important-the product or process? Justify your answer. L2 CLO2
5. With suitable illustration explain SPIRAL model evolutionary software development. L2 CLO2
6. Explain the Evolutionary and Incremental Model. What are the Advantages and Disadvantages? L2 CLO2
7. Write short notes on System engineering and Computer based System. L2 CLO2
8. Explain System Engineering hierarchy. What are the restraining factors to construct a system model? L2 CLO2
9. Explain Component Based Development model in detail L2 CLO2
10. How do you differentiate software engineering from system engineering? L2 CLO2
11. Explain in detail the following s/w process models with a neat diagram. L2 CLO2
 - i) Evolutionary process model. ii) Incremental Process model.
12. Explain the spiral model? What is the task region in the spiral model? How does the customer wins by getting the system or product that satisfy the majority of the customer's needs and the developer wins by working to realistic and achievable budgets and deadline? L2 CLO2
13. What are the necessities of Life cycle model? Elaborate on the various issues of Software life cycle. L2 CLO2
14. How does system engineering differ from software engineering? Also write brief notes on computer based system and system engineering hierarchy. L2 CLO2
15. Differentiate product engineering and business engineering overview L2 CLO2
16. Explain the process model that combines the element of waterfall and iterative fashion. L2 CLO2
17. Explain briefly about the following (i) business process engineering (ii) product engineering. L2 CLO2
18. Explain briefly about the following (i) Computer based system (ii) System engineering process. L2 CLO2

18CSC206J SOFTWARE ENGINEERING AND PROJECT MANAGEMENT
UNIT-II
SOFTWARE DESIGN

1 MARKS

S.NO	QUESTION	LEVEL	CLO	PG. NO
1.	A program should not have any bugs that inhibit its function is called as ----- -----	1	2	216
	a. Commodity b. Delight c. Firmness d. Analysis			
2.	A program should be suitable for the purposes for which it was intended is called as ----- a. Commodity b. Delight c. Firmness d. Analysis	1	2	216
3.	The experience of using the program should be a pleasurable one is called as -----. a. Commodity b. Delight c. Firmness d. Analysis	1	2	216
4.	The objects and relationships defined in the ----- diagram provide the basis for the data design action. a. CRC diagram b. Activity diagram c. Class diagram d. Usecase diagram	1	2	217
5.	The----- design describes how the software communicates with systems that interoperate with it, and with humans who use it.	1	2	217

- a. Interface
 - b. Collaboration
 - c. Activity
 - d. package
6. Which design is used to transforms structural elements of the software architecture In to a procedural description of software components? 2 2 217
- a. Interface
 - b. Component**
 - c. Activity
 - a. package
7. ----- is assessed by evaluating the feature set and capabilities of the Program. 1 2 220
- a. Functionality**
 - b. Usability
 - c. Reliability
 - d. Performance
8. What is measured by considering processing speed, response time, resource consumption, throughput, and efficiency? 1 2 220
- a. Reliability
 - b. Performance**
 - c. Functionality
 - d. Usability
9. An ----- Abstraction refers to a sequence of instructions that have a specific and limited function. 1 2 223
- a. design
 - b. procedural**
 - c. data
 - d. component
10. A ----- Abstraction is a named collection of data that describes a data object. 1 2 223
- a. design**

- b. procedural
 - c. data
 - d. component
11. ----- represent architecture as an organized collection of program components. 1 2 224
- a. Structural models**
 - b. Framework models
 - c. Dynamic models
 - d. Process models
12. ----- increase the level of design abstraction by attempting to identify repeatable architectural design frameworks that are encountered in similar types of applications. 1 2 224
- a. Structural models
 - b. Framework models**
 - c. Dynamic models
 - d. Process models
13. Which models address the behavioral aspects of the program architecture, indicating how the structure or system configuration may change as a function of external events? 1 2 224
- a. Structural models
 - b. Framework models
 - c. Dynamic models**
 - d. Process models
14. Which models focus on the design of the business or technical process that the system must accommodate? 1 2 224
- a. Structural models
 - b. Framework models
 - c. Dynamic models
 - d. Process models**
15. ----- can be used to represent the functional hierarchy of a system. 1 2 224
- a. Structural models
 - b. Functional models**

- c. Dynamic models
 d. Process models
16. Which is an indication of the relative functional strength of a module? 1 2 227
- a. **Cohesion**
 b. Coupling
 c. Elaboration
 d. refactoring
17. ----- is an indication of the relative interdependence among modules. 1 2 227
- a. Cohesion
b. Coupling
 c. Elaboration
 d. refactoring
18. Which is a reorganization technique that simplifies the design (or code) of a component without changing its function or behavior? 1 2 229
- a. a. Cohesion
 b. Coupling
 c. Elaboration
d. refactoring
19. Which class is used to represent data stores that will persist beyond the execution of the software? 1 2 230
- a. System classes
b. Persistent classes
 c. Business domain classes
 d. User interface classes
20. The ----- dimension indicates the evolution of the design model as design tasks are executed as part of the software process. 1 2 233
- a. Component
 b. Abstraction
c. Process
 d. deployment

21. The ----- dimension represents the level of detail as each element of the analysis model is transformed into a design equivalent and then refined iteratively. 1 2 233
- a. Component
 - b. Abstraction**
 - c. Process
 - d. deployment
22. ----- Elements indicate how software functionality and subsystems will be allocated within the physical computing environment that will support the software. 1 2 233
- a. Component level design
 - b. Deployment-level design**
 - c. Architectural Design Elements
 - d. Interface Design Elements
23. Which design is used to represents the structure of data and program components that are required to build a computer-based system? 1 2 242
- a. Pattern oriented design
 - b. Web application design
 - c. Architectural design**
 - d. Component level design
24. A data store resides at the center of ----- architecture and is accessed frequently by other components that update, add, delete, or modify data within the store. 1 2 250
- a. Object oriented
 - b. Data center**
 - c. Data flow
 - d. Call & return
25. ----- Architecture is applied when input data are to be transformed through a series of computational or manipulative components into output data. 1 2 250
- a. Object oriented
 - b. Data center
 - c. Data flow**

- d. Call & return
26. The systems that use the target system as part of some higher-level processing scheme is called as ----- 1 2 256
- Superordinate systems**
 - Subordinate systems
 - Peer-level systems
 - Actors
27. The system that are used by the target system and provide data or processing that are necessary to complete target system functionality is known as ----- 1 2 256
- Superordinate systems
 - Subordinate systems**
 - Peer-level systems
 - Actors
28. The information is either produced or consumed by the peers and the target system is called as ----- 1 2 256
- Superordinate systems
 - Subordinate systems
 - Peer-level systems**
 - Actors
29. Which dependencies is used to represent dependence relationships among consumers who use the same resource or producers who produce for the same consumers? 1 2 263
- Sharing**
 - Flow
 - Constrained
 - data
30. Which dependency represents relationships between producers and consumers of resources? 1 2 263
- Sharing
 - Flow**
 - Constrained

- d. data
31. ----- Dependencies represent constraints on the relative flow of control among a set of activities. 1 2 264
- a. Sharing
 - b. Flow
 - c. Constrained**
 - d. data
32. A ----- Component that coordinates the invocation of all other problem domain components. 1 2 279
- a. Problem domain component
 - b. Infrastructure component
 - c. Control component**
 - d. Design component
33. What type of a component implements a complete or partial function that is required by the customer. 1 2 279
- a. Problem domain component**
 - b. Infrastructure component
 - c. Control component
 - d. Design component.
34. Which component is responsible for functions that support the processing required in the problem domain? 1 2 279
- a. Problem domain component
 - b. Infrastructure component**
 - c. Control component
 - d. Design component
35. Which level of cohesion occurs when a component performs a targeted computation and then returns a result? 1 2 287
- a. **Functional**
 - b. Layer
 - c. Communicational
 - d. congestion

36. What type of cohesion occurs when a higher layer accesses the services of a lower layer, but lower layers do not access higher layers? 1 2 287
- a. Functional
 - b. Layer**
 - c. Communicational
 - d. congestion
37. All ----- operations that access the same data are defined within one class. 1 2 287
- a. Functional
 - b. Layer
 - c. Communicational**
 - d. congestion
38. Which coupling Occurs when operation A() invokes operation B() and passes a control flag to B? 1 2 289
- a. Control coupling**
 - b. Layer
 - c. Communicational
 - d. congestion
39. Which coupling occurs when operations pass long strings of data arguments? 1 2 289
- a. Control coupling
 - b. Data**
 - c. Communicational
 - d. congestion
40. Which coupling occurs when one component “surreptitiously modifies data that is internal to another component”? 1 2 289
- a. Control coupling
 - b. Data
 - c. Communicational
 - d. content**
41. The intent of ----- engineering is to identify, construct, catalog, and disseminate a set of software components that have applicability to existing 1 2 303

and future software in a particular application domain.

- a. data
- b. domain**
- c. pattern
- d. structure

42. Which focuses on the profile of the users who will interact with the System? 1 2 320

- a. Interface analysis**
- b. interface design
- c. Interface construction
- d. Interface validation

43. ----- is to define a set of interface objects and actions that enable a user to perform all defined tasks. 1 2 320

- a. Interface analysis
- b. interface design**
- c. Interface construction
- d. Interface validation

44. Which is normally begins with the creation of a prototype that enables usage scenarios to be evaluated? 1 2 320

- a. Interface analysis
- b. interface design
- c. Interface construction**
- d. Interface validation

45. ----- focuses on the ability of the interface, the degree to which the interface is easy to use and easy to learn, and the users' acceptance of the interface as a useful tool in their work. 1 2 320

- a. Interface analysis
- b. interface design
- c. Interface construction
- d. Interface validation**

46. Which patterns focus on the “creation, composition, and representation” of Objects? 1 2 350

- a. **Creational pattern**
 - b. Structural patterns
 - c. Behavioral patterns
 - d. Object pattern
47. ----- focus on problems and solutions associated with how classes and objects are organized and integrated to build a larger structure. 1 2 350
- a. Creational pattern
 - b. Structural patterns**
 - c. Behavioral patterns
 - d. Object pattern
48. Which pattern address problems associated with the assignment of responsibility between objects and the manner in which communication is effected between objects? 1 2 351
- a. Creational pattern
 - b. Structural patterns
 - c. Behavioral patterns**
 - d. Object pattern
49. Which pattern relate to the overall structure of the information space, and the ways in which users will interact with the information. 1 2 362
- a. Information architecture patterns**
 - b. Interaction patterns
 - c. Navigation patterns
 - d. Presentation patterns
50. ----- contribute to the design of the user interface. 1 2 362
- a. Information architecture patterns
 - b. Interaction patterns**
 - c. Navigation patterns
 - d. Presentation patterns

4 MARKS:

1. Write short notes on Software Design. L2 CLO2
 Software design follows the software requirement phase. Based on the

requirements, software is designed in such a way that the features required in the requirements document can be implemented in the software design. Apart from how the features as per functional requirements can be implemented, design also considers factors such as reliability, robustness, security, ease of use, internationalization, localization, and compatibility. All of these are collectively termed as non-functional design requirements.

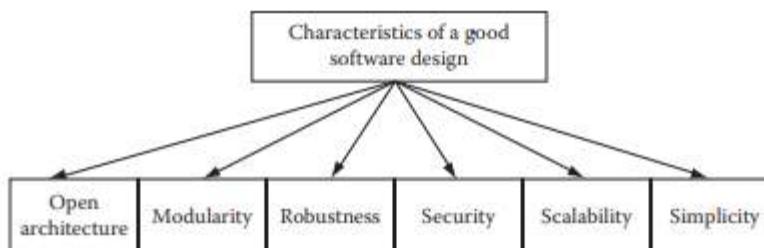
2. What are the important roles of Software Design

L1 CLO2

Software design plays an important role in software development. If the design is good, software will have fewer defects and may be considered reliable. Due to requirement creep as mentioned in a previous section, the design may get unstable, which may lead to a poor quality product. Enterprise software products though may have a lot of features; nevertheless, they need to have open interfaces so that they can be integrated with other software products

3. Draw the Characteristics of a good software design

L1 CLO2



4. What are the types of Software Design?

L1 CLO2

Software design on any project may consist of many work products, which together can be termed the software design for the software product that will be built during the software project. Some examples include prototypes, structural models, object-oriented design, systems analysis, and entity relationship models.

5. Write short notes on Design Standards.

L2 CLO2

If design standards are implemented on a project, then it will help in streamlining activities that are involved during the software design phase. Some industry standards for software design include operator interface standards, test scenarios, safety standards, design constraints, and design tolerances.

6. Difference between Top-Down approach and Bottom-Up Approach.

L2 CLO2

In the top-down approach, the top structure of the product is conceived and designed first. Once the structure is perfected, components that will make the product are designed. Once the major components are designed, the features that make the component are designed.

In the bottom-up approach, first, the minute functions of the software product are structured and designed. Then, the middle-level components are designed, and, finally, the top-level structure is designed. Once some components are designed, they can be shown to the customer, and a buy in can be made for the project.

7. Write short notes on Architectural Design.

L2 CLO2

Architectural design represents the structure of data and program components that are required to build a computer-based system. It considers the architectural style that the system will take, the structure and properties of the components that constitute the

system, and the interrelationships that occur among all architectural components of a system

8. Why is Architecture Design Important? L2 CLO2

- Representations of software architecture are an enabler for communication between all parties (stakeholders) interested in the development of a computer-based system.
- The architecture highlights early design decisions that will have a profound impact on all software engineering work that follows and, as important, on the ultimate success of the system as an operational entity.
- Architecture “constitutes a relatively small, intellectually graspable model of how the system is structured and how its components work together”

9. Compare Data-Centered and Data-Flow architectures. L2 CLO2

A data store (e.g., a file or database) resides at the center of this architecture and is accessed frequently by other components that update, add, delete, or otherwise modify data within the store. Client software accesses a central repository. In some cases the data repository is passive.

Data-Flow architecture is applied when input data are to be transformed through a series of computational or manipulative components into output data. A pipe-and-filter pattern has a set of components, called filters, connected by pipes that transmit data from one component to the next.

10. Distinguish between Object-Oriented and Layered Architectures L2 CLO2

The components of a system encapsulate data and the operations that must be applied to manipulate the data. Communication and coordination between components are accomplished via message passing.

A number of different layers are defined, each accomplishing operations that progressively become closer to the machine instruction set. At the outer layer, components service user interface operations. At the inner layer, components perform operating system interfacing. Intermediate layers provide utility services and application software functions.

11. Define the term Archetypes. L1 CLO2

An archetype is a class or pattern that represents a core abstraction that is critical to the design of an architecture for the target system. In general, a relatively small set of archetypes is required to design even relatively complex systems. The target system architecture is composed of these archetypes, which represent stable elements of the architecture but may be instantiated many different ways based on the behavior of the system

12. How to Refine the Architecture into Components L2 CLO2

The software architecture is refined into components, the structure of the system begins to emerge. You begin with the classes that were described as part of the requirements model. These analysis classes represent entities within the application (business) domain that must be addressed within the software architecture. The architecture must accommodate many infrastructure components that enable application components but have no business connection to the application domain.

13. What is meant by Module Division (Refactoring)? L2 CLO2

Whenever a software product is designed, it is done with good intentions. Care is taken to ensure that the design is extensible, so that when customer needs increase

over time, the product can be extended to take care of those increased needs. one technique is employed, which is known as refactoring. Using refactoring, the internal design of a piece of software code is improved by decreasing coupling among classes of objects and increasing cohesion among classes. Refactoring is very similar to the concept of normalization in relational databases

14. Write short notes on Module Coupling.

L2 CLO2

One area similar to refactoring is coupling between modules. As products mature and more and more lines of code are added to the existing product, coupling between modules tends to increase. This has a profound impact when any changes in code are required. To reduce the chances of product defects, it is necessary to reduce the number of calls among different modules and classes. SOA architecture provides great help here. SOA architecture essentially promotes loose coupling.

15. Define Component-level design

L1 CLO2

Component-level design occurs after the first iteration of architectural design has been completed. At this stage, the overall data and program structure of the software has been established. The intent is to translate the design model into operational software.

16. Difference between Cohesion and Coupling

L2 CLO2

Cohesion as the “single-mindedness” of a component. Within the context of component-level design for object-oriented systems, cohesion implies that a component or class encapsulates only attributes and operations that are closely related to one another and to the class or component itself.

Coupling is a qualitative measure of the degree to which classes are connected to one another. As classes (and components) become more interdependent, coupling increases. An important objective in component-level design is to keep coupling as low as is possible.

17. What are the different types of Coupling?

L1 CLO2

- Content coupling
- Common coupling
- Control coupling
- Stamp coupling
- Data coupling
- Routine call coupling
- Type use coupling
- Inclusion or import coupling
- External coupling

18. What are the steps for conducting Component-Level Design?

L2 CLO2

- Identify all design classes that correspond to the problem domain.
- Identify all design classes that correspond to the infrastructure domain
- Elaborate all design classes that are not acquired as reusable components
- Describe persistent data sources (databases and files) and identify the classes required to manage them
- Develop and elaborate behavioral representations for a class or component

19. Define the term User interface design

L1 CLO2

User interface design creates an effective communication medium between a human and a computer. Following a set of interface design principles, design identifies interface objects and actions and then creates a screen layout that forms the basis for a user interface prototype.

20. What are the steps involved in User-Interface Design? L2 CLO2

1. Using information developed during interface analysis, define interface objects and actions (operations).
2. Define events (user actions) that will cause the state of the user interface to change. Model this behavior.
3. Depict each interface state as it will actually look to the end user.
4. Indicate how the user interprets the state of the system from information provided through the interface.

21. Write short notes on Pattern-based design. L2 CLO2

Pattern-based design creates a new application by finding a set of proven solutions to a clearly delineated set of problems. Each problem and its solution is described by a design pattern that has been cataloged and vetted by other software engineers who have encountered the problem and implemented the solution while designing other applications. Each design pattern provides you with a proven approach to one part of the problem to be solved.

22. What is meant by Design Pattern? L1 CLO2

A design pattern can be characterized as “a three-part rule which expresses a relation between a certain context, a problem, and a solution” .For software design, context allows the reader to understand the environment in which the problem resides and what solution might be appropriate within that environment. A set of requirements, including limitations and constraints, acts as a system of forces that influences how the problem can be interpreted within its context and how the solution can be effectively applied.

23. Define the term WebApps Design. L1 CLO2

Design for WebApps encompasses technical and nontechnical activities that include: establishing the look and feel of the WebApp, creating the aesthetic layout of the user interface, defining the overall architectural structure, developing the content and functionality that reside within the architecture, and planning the navigation that occurs within the WebApp

24. How to design a Pyramid for WebApps? L2 CLO2

The creation of an effective design will typically require a diverse set of skills. Sometimes, for small projects, a single developer may need to be multi-skilled. For larger projects, it may be advisable and/or feasible to draw on the expertise of specialists: Web engineers, graphic designers, content developers, programmers, database specialists, information architects, network engineers, security experts, and testers. Drawing on these diverse skills allows the creation of a model that can be assessed for quality and improved before content and code are generated, tests are conducted, and end-users become involved in large numbers. If analysis is where WebApp quality is established, then design is where the quality is truly embedded.

25. Write about the objectives of WebApps Design. L1 CLO2

The objectives of a WebApp interface are to:

- (1) establish a consistent window into the content and functionality provided by the

interface,

- (2) guide the user through a series of interactions with the WebApp, and
- (3) Organize the navigation options and content available to the user.

26. Write short notes on Design Reuse.

L2 CLO2

A more potent design reuse is becoming available after the advent of the open source paradigm and SOA. In the case of open source, the design reuse is in fact a case of copying existing design and then using it exactly as it is or modifying it to suit your needs. The full interface details are provided by the owner. Using this information, you design your own application.

27. What is meant by Concurrent Engineering in Software Design?

L2 CLO2

Concurrent engineering deals with taking advance information from an earlier stage for a later stage in project, so that both the stages can be performed simultaneously. Though project activities are planned ahead in time, most often there are dependencies between a previous task and the next task in line. So, the latter task cannot start until the previous task finishes. That is why you cannot start developing an application until its design is complete. Moreover, the development will depend on the design.

28. Write Short notes on Design Life-Cycle Management.

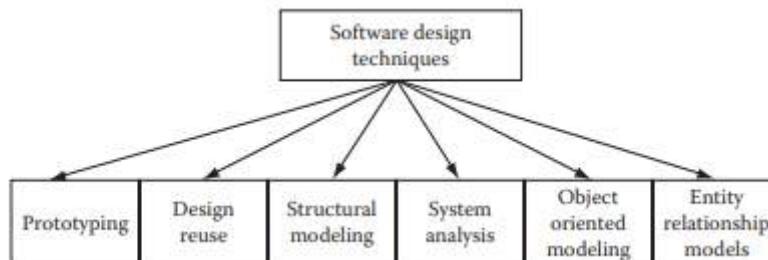
IO CLO2

Software requirements go through design process steps to become a full-fledged software design. At the high level, system analysis is performed. System analysis includes a study of requirements and finding feasibility of converting them into software design. Once the feasibility is done, then the actual software design is made. The software design is in the form of activity diagrams, use cases, prototypes, etc. Once the design process is complete, these design documents are verified and validated through design reviews. Once the design is reviewed and approved, then the design phase is over.

2

29. Draw the diagram for Software design techniques

L1 CLO2



30. Write short notes on Design Activities.

L2 CLO2

Software design activities produce many intermediate documents and work products. These include product architecture description, allocated requirements, product component descriptions, product-related life-cycle process descriptions, key product characteristic descriptions, required physical characteristics and constraints, interface requirements, verification criteria used to ensure that requirements have been achieved, operating environments, modes and states for operations, support, training, manufacturing, disposal, and verifications throughout the life of the product.

12 Marks:

1. Explain how to Translating the requirements model into the design model.	L2	CLO2
2. Describe the Evolution of Software Design in detail.	L2	CLO2
3. Illustrated in detail how the design model can be viewed in two different dimensions.	L3	CLO2
4. Explain the Characteristics of a good software design with a neat diagram.	L2	CLO2
5. Describe the two methods for designing software products or components.	L2	CLO2
6. Describe some of the design characteristics of the software project.	L2	CLO2
7. Discuss different types of software design techniques.	L2	CLO2
8. Explain the refactoring method used to refine the software product.	L2	CLO2
9. What is architecture design why it is important for software design of a project	L2	CLO2
10. Describe in detail about the Data-centered Architecture of a software design.	L2	CLO2
11. Discuss about data flow, Main program/subprogram, and Layered architecture of a software project.	L2	CLO2
12. With neat diagram explain architectural context diagram (ACD) with an example.	L2	CLO2
13. Explain the architecture trade-off analysis method (ATAM) that establishes an iterative evaluation process for software architectures.	L2	CLO2
14. With a step by step process to transform flow characteristics to be mapped into a specific architectural style.	L3	CLO2
15. Examine three important views of what a component is and how it is used as design modeling.	L2	CLO2
16. Illustrate the steps to represent a typical task set for component-level design, when it is applied for an object-oriented system.	L3	CLO2
17. Explain how to design traditional components for a software project design.	L2	CLO2
18. What are the golden rules to form user interface design? Explain.	L2	CLO2
19. Describe user interface design process with a neat diagram.	L2	CLO2
20. Elaborate the different kinds of Interface analysis.	L2	CLO2
21. Discuss User interface design steps of a software project.	L2	CLO2
22. Explain the Interface Design Workflow for WebApps.	L3	CLO2
23. Explain design pattern along with its characteristics and various kinds of patterns	L2	CLO2
24. With a neat diagram explain web apps interface design.	L2	CLO2
25. Describe the web apps architecture of a software project.	L2	CLO2

18CSC206J SOFTWARE ENGINEERING AND PROJECT MANAGEMENT
UNIT-III
SOFTWARE CONSTRUCTION

1 MARKS

S.NO	QUESTIONS	LEVEL	CLO	PG. NO
1.	_____ is an industry strength software product of a large size requires stringent coding standards.	1	3	176
	A.Coding B.Coding Methods C.Coding Framework D.Constructing			
2.	Converting the specifications into software code is totally dependent on the _____ Team.	1	3	176
	A.Coding B.Developing C.Debugging D.Constructing			
3.	_____ increases software code reuse and enhances productivity of developers.	1	3	177
	A.Modularity B.Simplicity C.Clarity D.Reliability			
4.	Standard naming conventions can be used so that the code has _____	1	3	177
	A.Modularity B.Simplicity C.Clarity D.Reliability			

5. Object-oriented programming, abstraction and information hiding can be used to add _____ 1 3 177
- A. Degree of Modularity
 - B.Degree of Simplicity**
 - C.Degree of Clarity
 - D.Degree of Reliability
6. _____ is one of the most important aspects of industry strength software products. 1 3 177
- A.Modularity
 - B.Simplicity
 - C.Clarity
 - D.Reliability**
7. To Ensure safety, the software product must have the error less than _____. 1 3 178
- A.0.00001%**
 - B.0.01%
 - C.0.000001%
 - D.0.001%
8. _____ will ensure a consistent coding production with standard code that will be easy to debug and test.(L2,CLO3),178 1 3 34
- A.Coding
 - B.Coding Methods
 - C.Coding Framework**
 - D.Constructing
9. Which is the most labor intensive phase in software development? (L1,CLO3),178 1 3 36
- A.Software Coding
 - B.Software Developing
 - C.Software Debugging
 - D.Software Constructing**

10. _____ is a powerful tool to eliminate defects and improve software code.(L2,CLO3),179 1 3 32

A. Deskcheck

B. Walkthrough

C. Inspection

D. Code Review

11. _____ is the formal code review initiated by developer.(L2,CLO3),179 1 3 32

A.Deskcheck

B. Walkthrough

C. Inspection

D. Code Review

12. _____ is the final review of the software code.(L2,CLO3),179 1 3 34

A. Deskcheck

B. Walkthrough

C. Inspection

D. Code Review

13. _____ enables programmers to store large pieces of code inside procedures and functions.(L2,CLO3),180 1 3 36

A. Structured programming

B. Object oriented programming

C. Automatic code generation

D. Pair programming

14. Which among the following firm is working to develop automatic code generation system?(L1,CLO3),180 1 3 36

A. Sun Microsystems

B. HP

C. IBM

D. Dell

15. SOA has been evolved recently for the purpose of 1 3 36
_____.(L1,CLO3),181
- A. Software construction
 - B. Inspection
 - C. Code Reuse**
 - D. Code Review
16. Which technique is used in Test driven development?(L2,CLO3),181 1 3 37
- A. SOA
 - B. eXtreme programming**
 - C. Scrum
 - D. Reuse
17. _____ is the quality driven development technique employed in the eXtreme Programming.(L2,CLO3),181 1 3 38
- A. Structured programming
 - B. Object oriented programming
 - C. Automatic code generation
 - D. D. Pair programming**
18. _____ plays an important role in the construction 1 3 39 phase.(L2,CLO3),181
- A. Configuration management**
 - B. Coding
 - C. Coding Methods
 - D. Coding Framework
19. _____ phase is one of the most labor intensive phases in software 1 3 41 development cycle.(L1,CLO3),183
- A. Software construction**
 - B. B. Code Generation
 - C. Automatic Code Generation
 - D. D. Coding

20. Which phase generates the complete source code of the application.(L1,CLO3),183 1 3 56

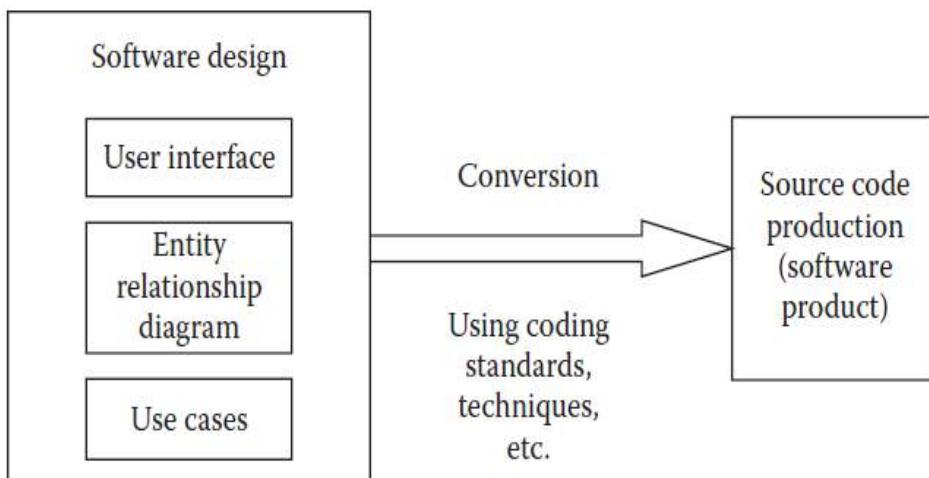
A.Software construction

- B.Code Generation
- C. Automatic Code Generation
- D.Coding

4 MARKS:

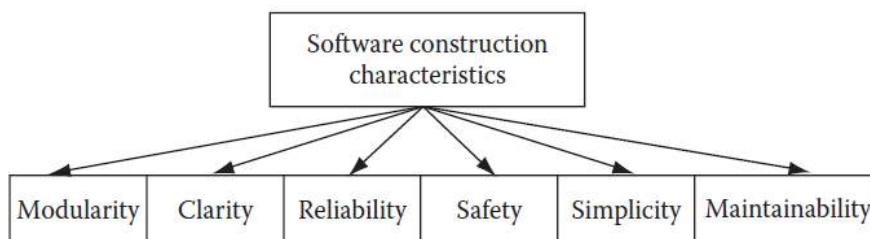
1. Explain Source code production from software design with neat sketch. L2 CLO3

Developers are given software design specifications in the form of use cases, flow diagrams, UI mock ups, etc., and they are supposed to write a code so that the built software matches these specifications. Converting the specifications into software code is totally dependent on the construction team. How well they do it depends on their experience, skills, and the process they follow to do their job. Apart from these facilities, they also need some standards in their coding so that the work is fast as well as has other benefits like maintainability, readability, and reusability.



2. List the coding standards in software construction. L1 CLO3

Some of the coding standards include standards for code modularity, clarity, simplicity, reliability, safety, and maintainability.



3. Elaborate in detail about the Quality Control. L2 CLO3

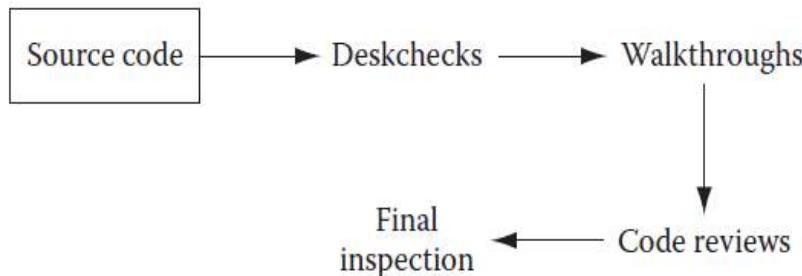
It is estimated that almost 70% of software defects arise from faulty software code. To compound this problem, software construction is the most labor intensive phase in software development. Any construction rework means wasting a lot of effort already put in. Moreover, it is also a fact that it is cheaper to fix any defects found

during construction at the phase level itself. If those defects are allowed to go in software testing, then fixing those defects will become costlier. That is why review of the software code and fixing defects is very important.

4. Categorize the techniques to ensure the quality of the written code.

L2 CLO3

There are some techniques available like deskchecks , walkthroughs, code reviews, inspections, etc. that ensure quality of the written code



5. Elaborate the evolution of different programming techniques in coding methods.

L2 CLO3

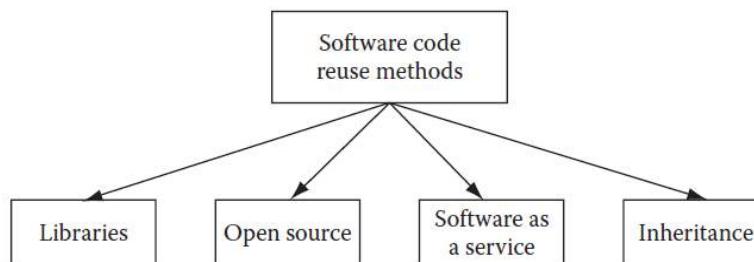
Different programming techniques include,

- Structured Programming
- Object-Oriented Programming
- Automatic Code Generation
- Test-Driven Development
- Pair Programming.

6. Explain in detail about the Code reuse methods.

L2 CLO3

Many techniques have evolved to reduce the labor intensive nature of writing source code. Software code reuse is one such technique. Making a block of source code to create a functionality or general utility library and using it at all places in the source code wherever this kind of functionality or utility is required is an example of code reuse. Code reuse in procedural programming techniques is achieved by creating special functions and utility libraries and using them in the source code. In object-oriented programming, code reuse is done at a more advanced level. The classes Software Construction containing functions and data themselves can not only be reused in the same way as functions and libraries, but the classes can also be modified by way of creating child classes and using them in the source code.



7. Explain the importance of configuration management in construction phase.

L2 CLO3

Configuration management plays an important role in the construction phase. Due to changes in requirements and design, an already developed source code needs to be changed. So it happens that the development team ends up with many versions of a source code during the project. If the version control management is not handled properly, then many developers may start working on a wrong version of source code,

and thus a lot of rework may be needed in the end. There is one more dimension to configuration management for the construction phase. During construction, many software builds are maintained for different versions of the product being developed. These builds can break if a bad piece of code is checked into the build by any developer. When the build is broken, then no other developer can check in his code.

Thus, development is halted until the build is rebuilt with the correct code. Imagine what may happen in the case of distributed teams located at far-flung locations with different time zones and a central build is being maintained. It will be difficult to communicate and manage the build process in such a scenario. In such scenarios, smoke test application can be deployed, which can run whenever a new code is checked-in in the build. If the smoke test fails, that means the build has failed and thus the automated system can e-mail the build information to concerned people. If the build fails, then the developer who had checked-in in the code gets the message and immediately tries to fix the build. Once the build is fixed, then other developers can check-in their code.

8. Discuss about the unit testing in detail.

L2 CLO3

Whenever a developer writes a piece of code, he feels confident that he has written a clean code and that it does not need testing. But most of the time he is wrong. It is because no source code is perfect, especially the first time. Only after some rounds of review it becomes perfect. At the same time, it is very difficult to review one's own code. That is why a quality control measure is taken in form of unit testing to ensure that developers test their codes themselves and only then can submit their code if the code passes the unit tests .

For unit testing, generally developers are comfortable as long as there are no changes required (due to change in design or requirements) in their code. But once some change takes place in the code somewhere, other things change. What would be the impact of that change on other parts of the software product under development? Similarly what impact will it have on their own code if changes take place in other modules being written by other people? Generally, it is one of the most challenging situations in software construction to find the impact of change on other parts of the product under development. Such situations call for unit testing of the written code, and no piece of code should go to build without doing this. A formal and rigid adherence to unit tests should be a must for all source codes being written and no liberty should be allowed.

9. Discuss about the Integration testing in detail.

L2 CLO3

Most software development is done after partitioning the software application under development first and then allocating it to distributed teams. Generally, modules of code are developed first. Later, they need to be integrated with each other to make a complete software application. Modules are integrated with each other through open interfaces. Whether or not the integration is working fine, it must be tested to ensure integration has been achieved. This kind of testing is known as integration testing.

Integration testing has been becoming more and more important, as most software being developed is modular in nature. With the advent of SOA, which is all about loosely coupled software components, integration testing has become even more important.

10. Explain in detail about software construction Artifacts.

L2 CLO3

The software construction phase is one of the most labor intensive phases in software development cycle. This phase generates the complete source code of the application. Apart from source code, documentation is also made so that when any maintenance is required on the built application, the source code could be well

understood, and changing any source code will be easy. Review reports are also generated after reviews are conducted.

11. Summarize about the Pair Programming.

L2 CLO3

Pair programming is a quality driven development technique employed in the eXtreme Programming development model. Here, each development task is assigned to two developers. While one developer writes the code, the other developer sits behind him and guides him through the requirements (functional, nonfunctional). When it is the turn of the other developer to write the code, the first developer sits behind him and guides him on the requirements. So developers take turns for the coding and coaching work. This makes sure that each developer understands the big picture and helps them to write better code with lesser defects.

12. Explain about the Test-Driven Development.

L2 CLO3

This concept is used with iteration-based projects especially with eXtreme Programming technique. Before developers start writing source code, they create test cases and run the tests to see if they run properly and their logic is working. Once it is proved that their logic is perfect, only then they write the source code. So here, tests drive software development, and hence it is appropriately named test-driven development.

13. Explain in detail about Automatic Code Generation.

L2 CLO3

Constructing and generating software code is very labor intensive work. So there has always been fascination about automatic generation of software code. Unfortunately, this is still a dream. Some CASE and modeling tools are available that generate software code. But they are not sophisticated. They are also not complete. Then there are business analyst platforms developed by many ERP software vendors that generate code automatically when analysts configure the product. These analyst platforms are first built using any of the software product development methodologies. The generated code is specific to the platform and runs on the device (hardware and software environment) for which the code is generated.

Generally, any code consists of many construction unit types. Some of these code types include control statements such as loop statements, if statements, etc., and database access, etc. Generating all of the software code required to build a software application is still difficult. But some companies like Sun Microsystems are working to develop such a system.

14. Elaborate Peer reviews.

L2 CLO3

Peer Reviews are employed when a complete review of the source code is not important. Here, the developer sends his piece of code to the designated team members. These team members review the code and send feedback and comments to the developer as suggestions for improvement in the code. The developer reads those feedbacks and may decide to incorporate or to discard those suggestions. So this form of review is totally voluntary. Still, it is a powerful tool to eliminate defects or improve software code.

15. Explain in detail about Reliability.

L2 CLO3

Reliability is one of the most important aspects of industry strength software products. If the software product is not reliable and contains critical defects, then it will not be of much use for end users. Reliability of source code can be increased by sticking to the standard processes for software construction. During reviews, if any defects are found, they can be fixed easily if the source code is neat, simple, and clear.

Reliable source code can be achieved by first designing the software product with future enhancement in consideration as well as by having a solid structure on

which the software product is to be built. When writing pieces of source code based on this structure, there will be little chance of defects entering into the source code. Generally during enhancements, the existing structure is not able to take load of additional source code and thus the structure becomes shaky. If the development team feels that this is the case, then it is far better to restructure the software design and then write a code based on the new structure than to add a spaghetti code on top of a crumbling structure.

12 Marks:

1. Categorize the various coding standards and explain its characteristics with examples. L2 CLO3
2. Classify the different kinds of reviews done at different stages in software code writing. L2 CLO3
3. List the techniques to ensure the quality of written code and discuss them in detail. L2 CLO3
4. Elaborate in detail about Quality control. L2 CLO3
5. Categorize the various coding methods and explain them in detail. L2 CLO3
6. Classify the different kinds of programming techniques and elaborate in detail L2 CLO3
7. Compare and contrast unit testing and integration testing with appropriate scenarios. L2 CLO3
8. Discuss in detail about
 - a. Pair Programming
 - b. Test driven development
 - c. Object oriented programming
9. Explain the following in detail
 - a. Structured Programming
 - b. Automatic code generation
 - c. Software code reuse
10. Explain in detail about
 - a. Configuration management
 - b. software construction Artifacts

18CSC206J SOFTWARE ENGINEERING AND PROJECT MANAGEMENT
UNIT-IV
SOFTWARE TESTING

1 MARKS

S.NO	QUESTIONS	LEVEL	CLO	PG. NO
1.	End result of Software Requirement Analysis is _____. a. Functional and Behavioral b. Architectural and Structural c. Usability and Reliability d. Algorithmic and Data Structure	1	4	R5 153
2.	Which Testing is performed first? a. Black box testing b. White box testing c. Dynamic testing d. Static testing	1	4	R5 194
3.	Verification and Validation uses _____. a. Internal and External resources respectively. b. Internal resources only. c. External resources only. d. External and Internal resources respectively.	1	4	R51 89
4.	Testing beyond normal operational capacity is _____. a. Load testing b. Performance testing c. Stress testing d. Dynamic testing	1	4	R1 479
5.	The expected results of the software is _____. a. Only important in system testing b. Only used in component testing c. Most useful when specified in advance d. Derived from the code	1	4	R1 712
6.	When an expected result is not specified in test case template then _____. a. We cannot run the test. b. It may be difficult to repeat the test. c. It may be difficult to determine if the test has passed or failed. d. We cannot automate the user inputs.	2	4	R5 173
7.	Test cases are created in which phase? a. Test Specification b. Test Planning c. Test Requirement d. Test Configuration	1	4	R1 449

8.	_____ refers to a different set of tasks ensures that the software that has been built is traceable to Customer Requirements.	1	4	R1 450
	a. Verification b. Requirement engineering c. Validation d. None of the above			
9.	Which granularity level of testing checks the behavior of module cooperation?	1	4	R1 459
	a) Unit Testing b) Integration Testing c) Acceptance Testing d) Regression Testing			
10.	Which testing is an integration testing approach that is commonly used when “shrink-wrapped” software products are being developed?	1	4	R1 463
	a) Regression Testing b) Integration testing c) Smoke testing d) Validation testing			
11.	Which of the following is / are not a Iterative Model?	1	4	R1 61
	a. RAD b. Incremental c. V model d. Spiral Model			
12.	Which is not true in case of Unit Testing? [REF 1, PAGE:466] a. It decreases the software development speed.	1	4	R1 466
	b. It can't be expected to catch every error in a program. c. In this tester evaluates if individual units of source code are fit for use. d. It is usually conducted by the development team.			
13.	Focus Testing comes under _____.	1	4	R1 540
	a. Performance Testing b. Acceptance Testing c. Usability Testing d. Component Testing			
14.	Difference between Retesting and Regression Testing is _____	1	4	R5 197
	a. Retesting ensures the original fault has been removed where as regression testing looks for unexpected side-effects.			
	b. Retesting looks for unexpected side-effects where as regression testing ensures the original fault has been removed. c. Retesting is done after faults are fixed where as regression testing is done earlier d. Retesting is done by developers whereas regression testing is done by independent testers			

- | | | | | |
|-----|---|---|---|-----------|
| 15. | First component of the DFD is _____ .
a. Process
b. Flow
c. Entity
d. Level | 1 | 4 | R1
187 |
| 16. | Minimum of four test data are available in _____ .
a. Boundary Value Analysis
b. Equivalence Class Partitioning
c. Both A and B
d. None of these. | 1 | 4 | R1
498 |
| 17. | Which coupling should be avoided in software? [REF 1, PAGE:289]
a. Data coupling
b. Content Coupling
c. Control coupling
d. Stamp coupling. | 1 | 4 | R1
289 |
| 18. | Cyclomatic Complexity cannot be applied in _____ .
a. Re-engineering
b. Risk Management
c. Test Planning
d. Reverse engineering | 1 | 4 | R1
488 |
| 19. | Data classification is done by which type of Decision Tree?
a. Regression Tree
b. Boosted Tree
c. Classification Tree
d. Bagging Tree | 1 | 4 | R1
715 |
| 20. | Which of the following is not a software testing generic characteristics?
a) Different testing techniques are appropriate at different points in time
b) Testing is conducted by the developer of the software or an independent test group
c) Testing and debugging are different activities, but debugging must be accommodated in any testing strategy
d) None of the mentioned | 1 | 4 | R5
335 |

4 MARKS:

1. **What is software testing?** L2 CLO4

In most quality standards documents, software testing is divided into two parts: “validation” and “verification.” While verification implies that the developed software is working as intended by checking the requirement specifications, design, source code, etc., in static mode, validation implies that the software has been validated to be working after running it and checking whether all functionality meets the requirements. Verification techniques are also known as static testing, since the source code is not run to do testing. Each work product including requirement specifications, design, and source code during software development is tested using static methods.

2. How does software testing help in increasing quality of a software product?

L2 CLO4

It is a fact that the exact number of defects in a software product is difficult to find. At best it can be predicted using some defect estimation tools. It is also impossible to detect all defects in a software product. Nevertheless, finding and fixing critical bugs up to an acceptable limit as per expectations is important. If there are more defects in the product after the product enters production, then the project team will be in big trouble. The support costs for a bug ridden product will be too high. So, less than required testing is a certain call for rebuke from stakeholders. So the software testing helps in providing a quality software product which is bug free.

3. What techniques are used for testing software?

L2 CLO4

Verification, Validation, Test Prioritization, Effort Estimation, Test Point Analysis, Defect Tracking

4. List out the Problems with Traditional Development Model with neat diagram.

L2 CLO4

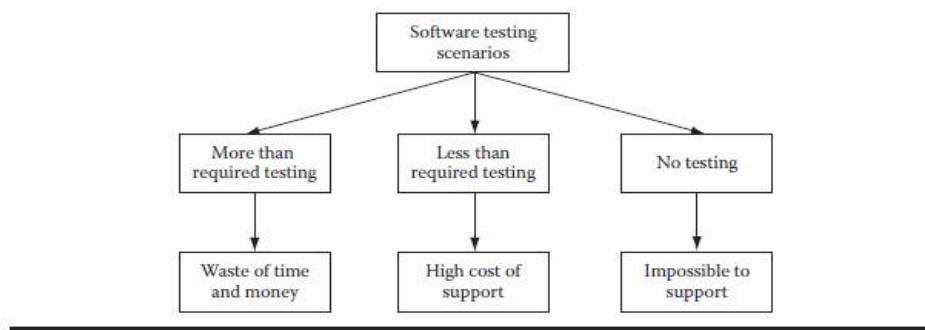


Figure 13.1 Software testing scenarios.

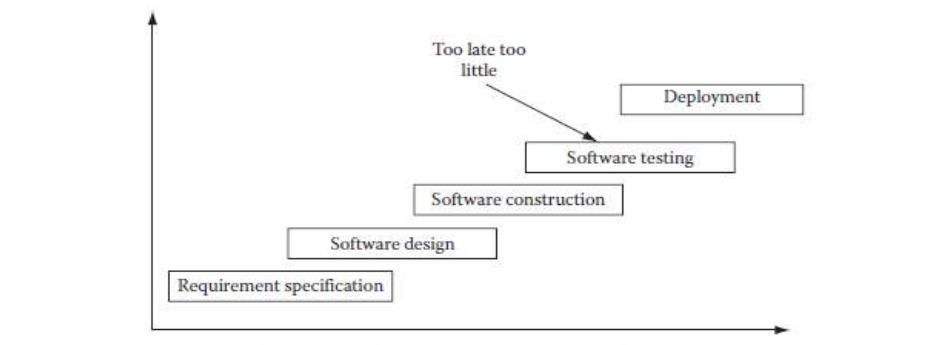


Figure 13.2 Traditional software development model (too little, too late testing).

5. Describe about Verification and Validation.

L2 CLO4

In most quality standards documents, software testing is divided into two parts: “validation” and “verification.” While verification implies that the developed software is working as intended by checking the requirement specifications, design, source code, etc., in static mode, validation implies that the software has been validated to be working after running it and checking whether all functionality meets the requirements.

Verification techniques are also known as static testing, since the source code is not run to do testing. Figure 13.3 shows that each work product including requirement specifications, design, and source code during software development is tested using static methods. The requirement specifications are reviewed for completeness, clarity, design ability, testability, etc. The software design is reviewed for robustness, security,

implementability, scalability, complexity, etc. The source code is reviewed for dead code, unused variables, faulty logic, constructs, etc.

Once the source code is ready to be run as a system, validation testing can be started. Validation testing is also known as dynamic testing as, in this case, the source code is actually run to determine that it is running per specifications. During validation, unit, integration, system, and finally user acceptance testing are performed. Unit testing is done to ensure each unit piece of source code is free from defects. Once unit testing is done, then this piece of code is integrated with the main source code build. But before integrating to the main build, it is strongly advisable to do local integration testing on the developer's own computer. Only when the source code runs smoothly and all integration tests pass, the source code should be integrated with the main build.

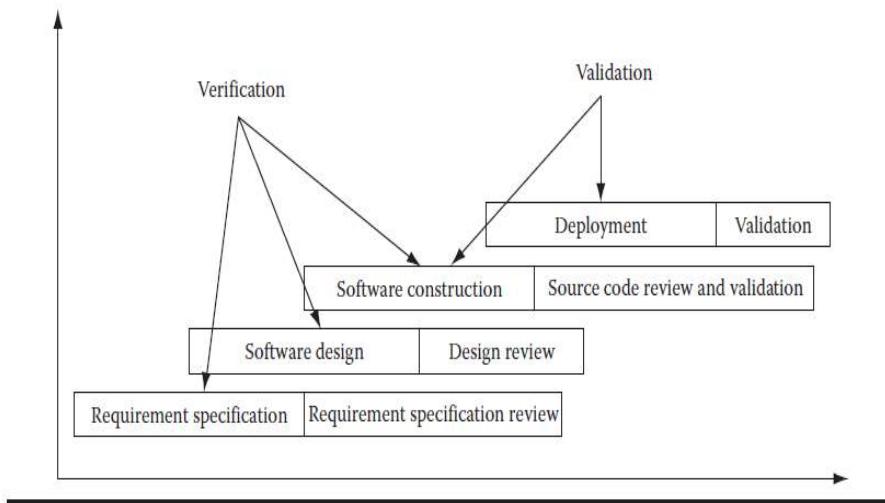


Figure 13.3 Software verification and validation.

6. Write short notes on Risk management in Software testing.

L2 CLO4

The test manager should also do plan for all known risks that could impact the test project. If proper risk mitigation planning is not done, and a mishap occurs, then the test project schedule could be jeopardized, costs could escalate, and/or quality could go down.

Some of the risks that can have severe, adverse impact on a test project include an unrealistic schedule, resource unavailability, skill unavailability, frequent requirement changes, etc. Requirement changes pose a serious threat to testing effort, because for each requirement change, the whole test plan gets changed. The test team has to revise its schedule for additional work as well as to assess impact of the change on the test cases they have to recreate.

For test professional resources, a good alternative resource planning is required. The test manager should, in consultation with human resource manager, keep a line of test professionals who may join in case one is needed on his project.

For scheduling problems, the test manager has to ensure in advance that schedules do not get affected. He has to keep a buffer in the schedule for any eventuality.

To keep a tab on the project budget, the test manager has to ensure that the schedule is not unrealistic and also has to load his test engineers appropriately. If some test engineers are not loaded adequately, then project costs may go higher. For this reason, if any test professionals do not have enough assignments on one project, they

should be assigned work from other projects.

7. State the importance of effort estimation?

L2 CLO4

For making scheduling, resource planning, and budget for a test project, the test manager should make a good effort estimate. Effort estimate should include information such as project size, productivity, and test strategy. While project size and test strategy information comes after consultation with the customer, the productivity figure comes from experience and knowledge of the team members of the project team.

The wideband Delphi technique uses brainstorming sessions to arrive at effort estimate figures after discussing the project details with the project team. This is a good technique because the people who will be assigned the project work will know their own productivity levels and can figure out the size of their assigned project tasks from their own experience. Initial estimates from each team member are then discussed with other team members in an open environment. Each person has his own estimate. These estimates are then unanimously condensed into final estimate figures for each project task.

Effort estimation is one area where no test manager can have a good grasp, at the initial stages of the project. This is because not many details are clear about the project. As the project unfolds, after executing some of its related tasks, things become clearer. At that stage, any test manager can comfortably give an effort estimate for the remaining project tasks.

8. Explain about Test Automation.

L2 CLO4

Most testing tasks are done manually, as they are still difficult to automate. Wherever automation is possible, it can be evaluated. Care should also be taken not to do automation blindly. This is because the initial effort for automation is more than manual testing.

Testing tasks include requirements and design document review, test case scenario creation, test case creation, test case execution, test case management, and defect tracking.

Out of these tasks, test case execution and test case management are the only tasks for which good automation tools are available.

9. Describe about Defect Tracking.

L2 CLO4

Defect tracking is one of the most important activities in a test project [13]. During defect tracking it is ensured that defects are logged and get fixed. All defects and their fixing are tracked carefully (Figure 13.6).

Defect count per hour per day is a common way of measuring performance of a test team. If the testing is done for an in-house software product, traditionally, it used to not be a performance evaluation measurement. What really counted was the number of defects found in production when the software product was deployed and used by end users. But it is too late a performance measurement. What if many of the test team members left before the product was deployed? In fact this is a reality, given the high attrition rate (as much as 20% at many corporations) of software professionals. Once they are gone, there is no point in measuring the performance. Thus, a better measurement would allow for more immediate results. This is achieved by measuring

the defect count per hour per day. Then there is the case of outsourced test projects. If the contract is only for testing up to deployment and not afterward, then measurement does not make sense after the contract has ended.

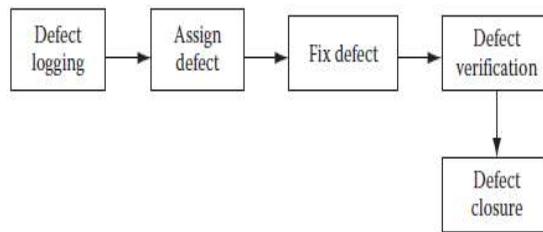


Figure 13.6 Defect life cycle.

A good defect tracking application should be deployed on a central server that is accessible to all test and development teams. Each defect should be logged in such a way that it could be understood by both development and testing teams.

10. Write short notes on Software Testing in Iterative Model. [REF-5. PAGE: 197] L2 CLO4

In an iterative model, each iteration is a short cycle. So the amount of testing in each iteration is also small. Thus, unlike in waterfall model, software testing has a lesser role in the iterative development life cycle.

Generally, software defects tend to increase with the size of software products. Since in iteration mode the software product is small, there will be fewer defects in the product. Although in reality, as the software product grows in size over many iterations, the number of defects per line of software code is bound to increase. In iterative development, regression testing is also a big issue. In each iteration, there will be a large number of regression test cases to run. As the product size increases with iterations, the set of regression test cases also increases. It becomes a liability after a while. Manually running all those regression tests takes a lot of time, which becomes a hindrance for the release schedule. In such cases, the best option is to go for automation of these regression test cases. Automated test cases take much less time (sometimes if the manual running of test cases was taking 5 days, after automation it took only 5 h) to run.

12 Marks:

1. Discuss in detail Test Strategy and Planning. L2 CLO4
2. Explain a case study on Software testing. L3 CLO4
3. (i) List out the Problems with Traditional Development Model with neat diagram L2 CLO4
 (ii) Describe about Verification and Validation. L2
4. (i) Write short notes on Risk management in Software testing. L2 CLO4
 (ii) State the importance of effort estimation. L1
5. (i) Explain about Test point analysis and its components. L2 CLO4
 (ii) Elaborate in detail the Test Automation. L2

- | | | | |
|----|---|----|------|
| 6. | (i) Describe about Defect Tracking. | L2 | CLO4 |
| | (ii) Write short notes on Software Testing in Iterative Model. | L2 | |
| 7. | Elaborate in detail Test Project Monitoring and Control with neat sketch. | L2 | CLO4 |
| 8. | Describe in detail the techniques used for testing software. | L2 | CLO4 |

1. Which of these are incorrect characteristics of Legacy System?
 - a) High maintenance cost
 - b) Clear Software
 - c) Obsolete support software
 - d) obsolete hardware

ANSWER: b) Clear Software

2. The Identify the key Components of Software Maintenance Framework

- a) User requirements
- b) Opportunity
- c) Productivity
- d) Revenue

ANSWER: a) User requirements

3. The Identify the key Factor not affecting Software Maintenance

- a) Relationship of Software product and Environment
- b) Relationship of Software product and User
- c) Relationship of Software product and Software Maintenance team
- d) Relationship of Software product and product implementation

ANSWER: d) Relationship of Software product and product implementation

4. maintenance action addresses errors and faults in your software systems

- a) Preventive
- b) Corrective
- c) Perfective
- d) Adaptive

ANSWER: b) Corrective

5. maintenance action have predicts defects in the software that will affect your customers in the future

- a) Preventive
- b) Corrective
- c) Perfective
- d) Adaptive

ANSWER: d) Adaptive

6. Perfective Maintenance accounts to of all the maintenance activity

- a) 75%
- b) 60%
- c) 50%
- d) 40%

ANSWER: c) 50%

7. The changes you make to prevent the occurrence of errors in the future is

- a) Preventive
- b) Corrective
- c) Perfective
- d) Adaptive

ANSWER: a) Preventive

8. Lehman's laws of software evolution increase in its life time with one side and other is

- a) Quality & Size
- b) size, complexity & quality
- c) Size & Complexity
- d) Complexity, speed & evolution

ANSWER: b) size, complexity & quality

9. Which maintenance process that works in fire fighting approach

- a) Boehm's Model
- b) Iterative Enhancement Model
- c) Quick Fix Model
- d) Reuse Model

ANSWER: c) Quick Fix Model

10.allows inclusion of maintenance requirements in the change specification

- a) Boehm's Model
- b) Iterative Enhancement Model
- c) Quick Fix Model
- d) Osborne Model

ANSWER: d) Osborne Model

1. Which of these are not the software maintenance financial reason.
 - a) Revenue Loss
 - b) Software Loss
 - c) Opportunity Loss
 - d) Productivity Loss

ANSWER: b) Software Loss

2. Losses due to problems with the software can be compared to probable cost of maintenance and
 - a) Return on Investment
 - b) Investment Opportunity
 - c) Software Productivity
 - d) Investment Returns

ANSWER: a) Return on Investment

3. The aspects of a maintenance team that lead to high maintenance costs are &
 - a) Quality, Staff turnover
 - b) Staff turnover, Software Quality
 - c) Domain expertise, Team
 - d) Staff turnover, Domain expertise

ANSWER: d) Staff turnover, Domain expertise

4. Which of these are not the functions performed by the Software Maintenance team
 - a) Locating information in system documentation
 - b) Creating new documents
 - c) Keeping system documentation up-to-date
 - d) Finding the source of system failures or problems

ANSWER: b) Creating new documents

5.method adapts by means of verifying that maintenance goals have been met; performance review to provide feedback to managers
 - a) Reuse
 - b) Hypothesizes
 - c) Osborne
 - d) Iterative

ANSWER: c) Osborne

6. The three step method adapted to do maintenance is

- a) Reuse
- b) Hypothesizes
- c) Osborne
- d) Iterative

ANSWER: d) Iterative

7. Maintenance is viewed as the activity involving the of existing program components

- a) Reuse
- b) Hypothesizes
- c) Osborne
- d) Iterative

ANSWER: a) Reuse

8 The Reuse-oriented Model has main steps

- a) 3
- b) 4
- c) 5
- d) 1

ANSWER: b) 4

9. Modifying the old system parts according to the new requirements is a step of

- a) Boehm's Model
- b) Iterative Enhancement Model
- c) Quick Fix Model
- d) Reuse Model

ANSWER: d) Reuse Model

10. A maintenance model requires complete documentation as starting point of each

- a) Step
- b) Hypothesizes
- c) Iteration
- d) Module

ANSWER: d) Osborne Model

- Software doesn't wear out, but it deteriorates (due to change).

Software points to the following interface programs: Engineering, User interface, window, pull-down menus in library etc.

computing environments is supported integrated with remote database and business applications.

- 7. AI software uses non-numerical algorithm to solve complex problem. Robotics, expert system, pattern recognition game playing

- Project needs adequate resources in terms of time, manpower, finance, material and knowledge-bank.

7. Project Communication Management

8. Configuration Management

work tasks:
work products:
Quality assurance points
Projects milestones

Testing – to answer over in the code.

- Deployment:
 - Delivery to the customer for evaluation
 - Customer provide feedback

- Software configuration management
 - Manages the effects of change.

- Risk management
 - Assesses risks that may effect that outcome of project or quality of product (i.e. software)

managers for a specific project.

- Deployment

```
graph TD; A[development] --> C[deployment]; B[testing] --> C; C --> D[monitoring]
```

deployment

delivery
support
feedback

- Operation and Maintenance: Normally this is the longest phase of the software life cycle. The system is installed and put into practical use. Maintenance involves correcting errors which were not discovered in earlier stages of the life-cycle.

- The model implies that once the product is finished, everything else is maintenance.

3. Assumes patience from customer - working version of program will not available until programs not getting change fully.

- Easy to use

Delivers software in small but usable pieces, each piece builds on pieces already delivered

- Incremental model focus more on delivery of operation product with each increment.

Makes heavy use of reusable software components with an extremely short development cycle

- Process modeling – Data object transforms to information flow necessary to implement business.

- Not appropriate when technical risks are high (heavy use of new technology)

- Model assist software engineer and customer to better understand what is to be built when requirement are uncertain(Fuzzy).

- Feedback from customer / end user will refine requirement and that is how iteration occurs during prototype to satisfy the needs of the customer.

- Developers get to build something immediately.

mechanism for defining requirement

construction

version of software.

- New product development projects
- Concept development projects

software to the customer as rapidly as feasible.

face-to-face conversation.

adjusts its behavior accordingly.

- Working together, stories are grouped for a deliverable increment next release.

functionality

acceptance testing

- each iteration, client-driven adaptive planning

- information on how to carry out some computation,
- a constraint on the development of the system.

Elicitation**Analysis****Specification****Validation**

- Example: “The system shall be available to all clinics during normal working hours (Mon-Fri, 0830-1730). Downtime during normal working hours shall not exceed 5 seconds in any one day”

MustHave

The system **shall validate** payment with the credit card processing company. *MustHave*

The system shall log in a customer within 5 seconds.

ShouldHave

Performance
requirements

Space
requirements

Privacy
requirements

Safety
requirements

- Use cases and scenarios
- Risk analysis

regulations, and standards

users required

from documented
procedures

[1] Preece, Rogers, and Sharp "Interaction Design: Beyond human-computer interaction", p214

All the above system types utilize different values of the constants used in Effort Calculations.

Embedded

3.6

1.20

Required turnabout time

Required development schedule

Required development schedule

environment

Required turnabout time	0.94	1.00	1.07	1.15
-------------------------	------	------	------	------

Required development schedule	1.23	1.08	1.00	1.04	1.10
-------------------------------	------	------	------	------	------

Embeddedc

2.8

1.20

- **Loss:** the risk becomes a reality and unwanted consequences or losses occur

Budget risk

- losing budgetary or personnel commitment

"What special characteristics of this product may threaten our project plan?"

- Staff size and experience

where they will have the most impact

Management Plan

- Draw a horizontal cutoff line in the table that indicates the risks that will be given further attention

- C = Total cost of developing 18 components is \$25,000
- RE = .80 x \$25,000 = \$20,000

- **Configuration** is generally understood to cover changes typically made by a system administrator.



- This also helps developers with debugging to check if configuration changes impacts the product's functionality.

current during the life-cycle of the project.

- Project Data
- Manuals

■ understand the likely paths for change

- A set of use-cases is developed by end-users.

full-experience
components

part.-experience
components

process

*The first guideline prescribes a default
allocation of costs among the first-level
WBS elements*

4. At this point, subproject managers are given the responsibility for decomposing each of the WBS elements into lower levels using their top-level allocation, staffing profile, and major milestone dates as constraints.

129

Software Design

Compiled by IT Department, SRMIST, KTR

Disclaimer:

The lecture notes have been prepared by referring to many books and notes prepared by the teachers. This document does not claim any originality and cannot be used as a substitute for prescribed textbooks.

CLR-2:Preparing the project plane based on the scope and calculating the project effort based on resources.

- User Interface Design
- Pattern Oriented Design
- Web Application Design
- Design Reuse
- Concurrent Engineering in Software Design
- Design Life-Cycle Management

- With each change request there will be a different version of the software design.
- Maintaining these design versions through the development process is very important.
- It is necessary to make sure that the right design is used for software construction.
- Software design should be robust and should also be able to take change requests without any problems.

- In such cases, the design is changed so that new factors of the features being added are incorporated.
- Software design development can be likened to designing a physical product.
- Suppose a new car model is to be developed.
- The car design is broken down into separate components and in the end assembling them will become a complete design for the car model.
- Various factors are considered during the design of the components.

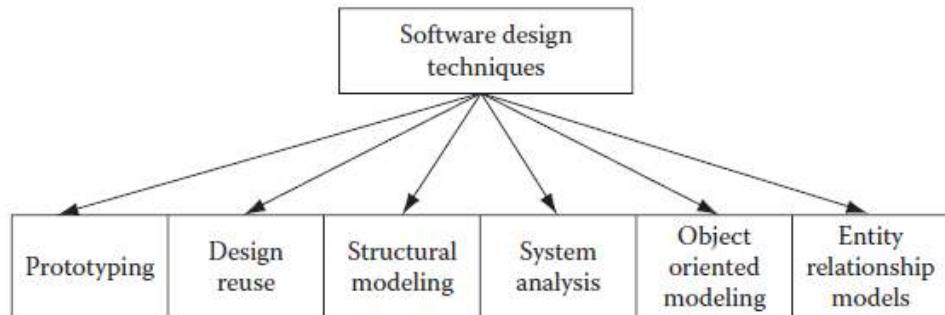
- During design, one consideration is also made that though each component is developed separately, after assembly the components should work with each other without any problems.
- That means assembling does not create any problems in the product itself.
- Similar considerations are also done when software products are designed.

- In the beginning, a software system may consist of only a few features.
- The feature set is expanded in future releases as and when it becomes necessary to include them in the system.
- If proper structure is not provided from the very beginning, the addition of these new features will make the system unstable.
- To deal with this problem, a technique called refactoring is used on these agile projects where incremental software development is done.

- Some of the design techniques that help make good software design include open architecture, modularity and scalability.
- The current trend of service-oriented architecture (SOA) has also helped tremendously in changing the design concepts.
- SOA is built on Web services and loose coupling of software components.
- The asynchronous messaging method of SOA is a vital aspect for developing Web-based applications.

be built during the software project.

- Some examples include prototypes, structural models, object-oriented design, systems analysis and entity relationship models.



techniques).

- Over the years, many software design techniques have evolved with the evolution of different programming paradigms.
- Starting with the early procedural programming paradigms, programming has evolved into present day "service-oriented architecture".
- Software design has kept the pace with these evolving paradigms, and thus it has also been evolving. So, we have early structural design paradigms to modern day SOA designs. Let us discuss some of these design techniques.

during the prototype demonstration sessions.

- This greatly helps in reducing the risks of not meeting customer expectations.
- In any case, customers do not care about internal workings of the application.
- They are always concerned about what the application screens look like and how the application behaves with different kind of inputs and events.

Starting with the early procedural programming paradigms, programming has evolved into present day "service-oriented architecture".

- This method of design reuse is known as internal design reuse.
- A more potent design reuse is becoming available after the advent of the open source paradigm and SOA.
- In the case of open source, the design reuse is in fact a case of copying existing design and then using it exactly as it is or modifying it to suit the needs.
- But in the case of SOA, there is no copying or modifying a software design.
- The existing design is utilized as it is.

service is available and the application uses this application/component.

- SOA is indeed leading to a reuse model that is going to transform the world of computing and the lives in years to come.

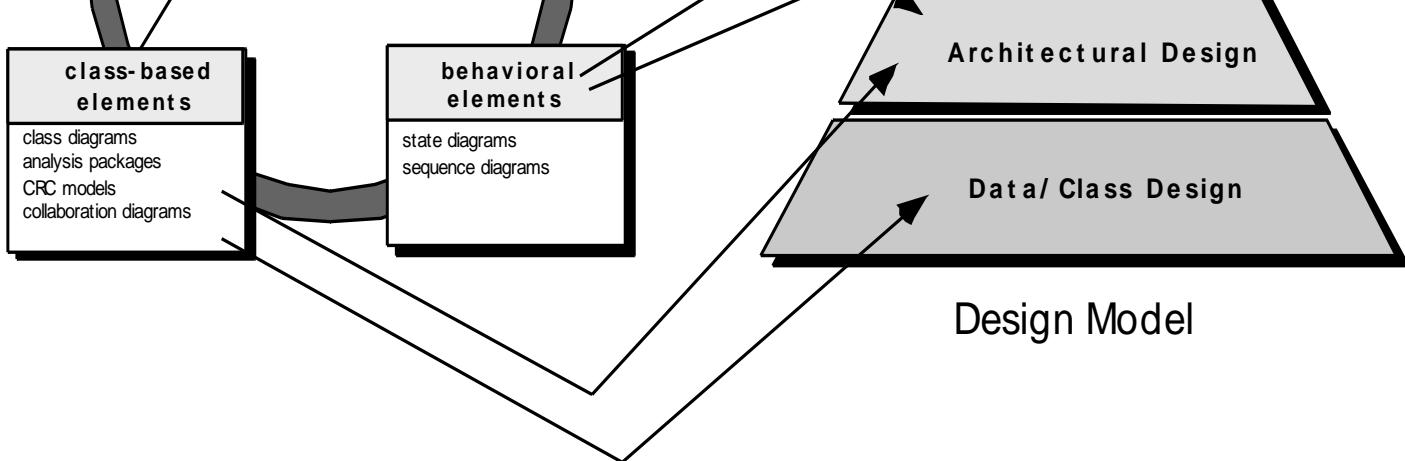
manageable parts.

- Breaking into parts for an application can best be done using a structural analysis.
- From requirement specifications, a feature set is made to decide what features will be in the application.
- This feature set is analyzed and broken down into smaller sets of features, which will go into different modules.
- This is represented in a structural model of the application.

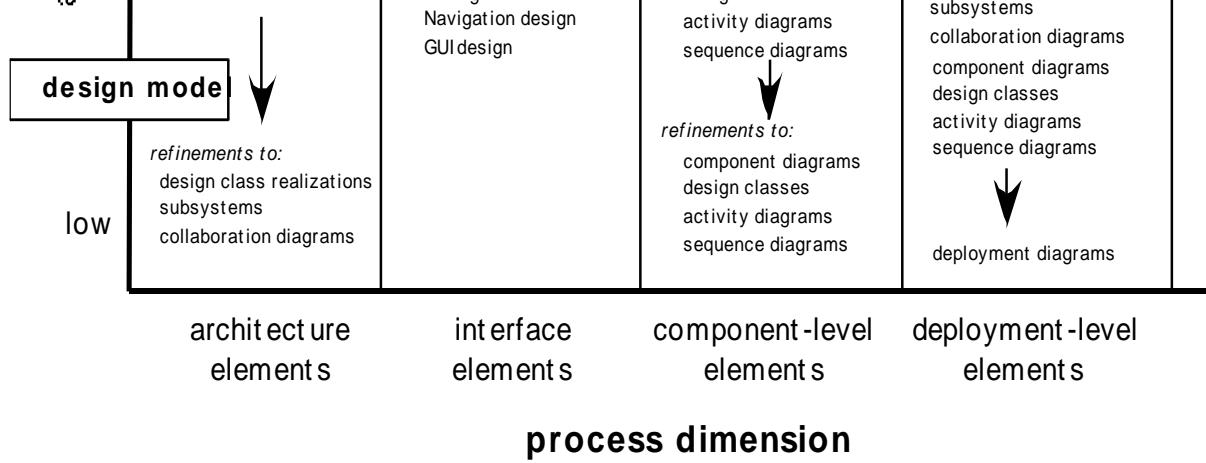
- So if we have a group of similar objects with somewhat different properties, then we can implement classes in such a way that a base parent class has child classes with different properties.
- This concept aligns very much to the real-world scenarios.
- Object-oriented design takes input from use cases, activity diagrams, user interfaces etc.

web site act with user activities, how these objects interact with the underlying software system of the bank, and how connections are made between the user and the website and between the website and the bank system.

- System analysis will analyze all these things.
- Based on the analysis, a system model can be made that will be used in developing the application.



- User Interface (UI)
 - External interfaces to other systems, devices, networks or other producers or consumers of information.
 - Internal interfaces between various design components
- Component Elements
 - Deployment Elements



- The architectural design description should address how the design architecture achieves requirements for performance, capacity, reliability, security, adaptability, and other system characteristics.

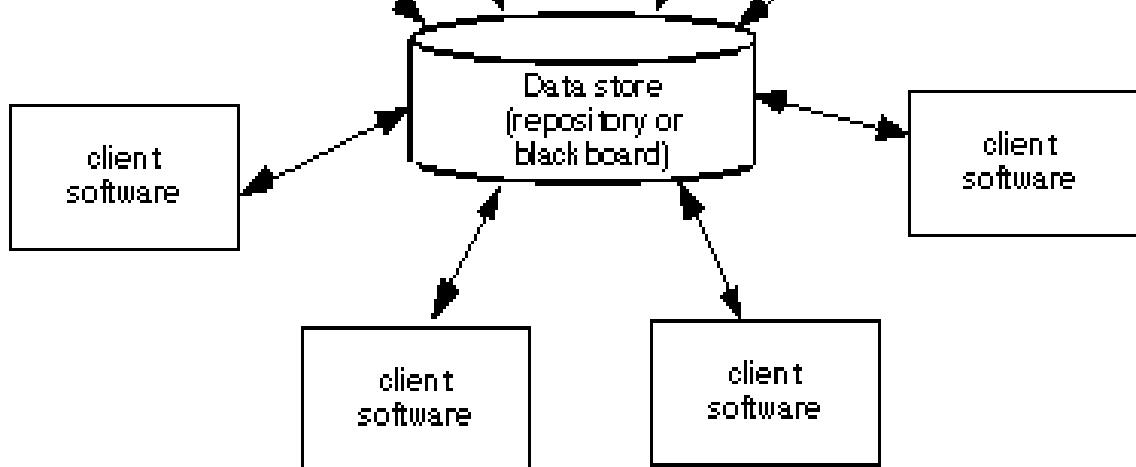
Families of related systems

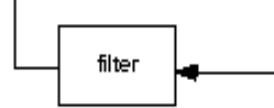
- The architectural design should draw upon repeatable patterns that are commonly encountered in the design of families of similar systems.
- In essence, the design should have the ability to reuse architectural building blocks.

- Representations of software architecture are an enabler for communication between all parties (stakeholders) interested in the development of a computer-based system.
- The architecture highlights early design decisions that will have a profound impact on all software engineering work that follows and, as important, on the ultimate success of the system as an operational entity.
- Architecture “constitutes a relatively small, intellectually graspable mode of how the system is structured and how its components work together”.

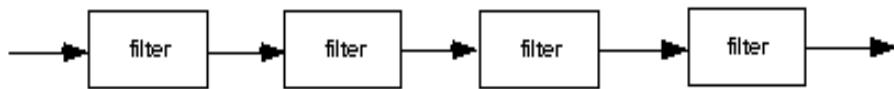
is "a representation of a whole system from the perspective of a related set of [stakeholder] concerns."

- (1) a set of components (e.g., a database, computational modules) that perform a function required by a system,
- (2) a set of connectors that enable “communication, coordination and cooperation” among components,
- (3) constraints that define how components can be integrated to form the system, and
- (4) semantic models that enable a designer to understand the overall properties of a system by analyzing the known properties of its constituent parts.

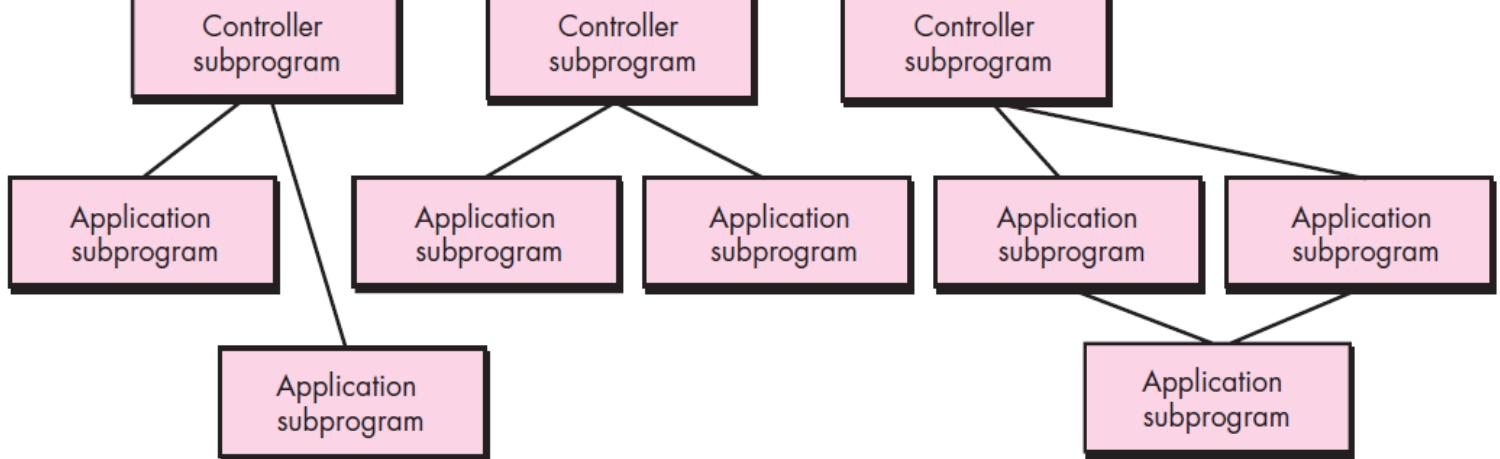




(a) pipes and filters

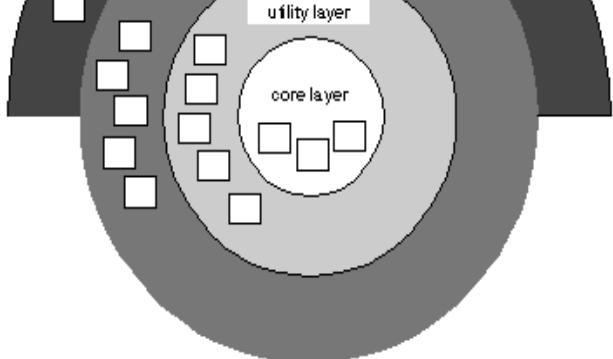


(b) batch sequential



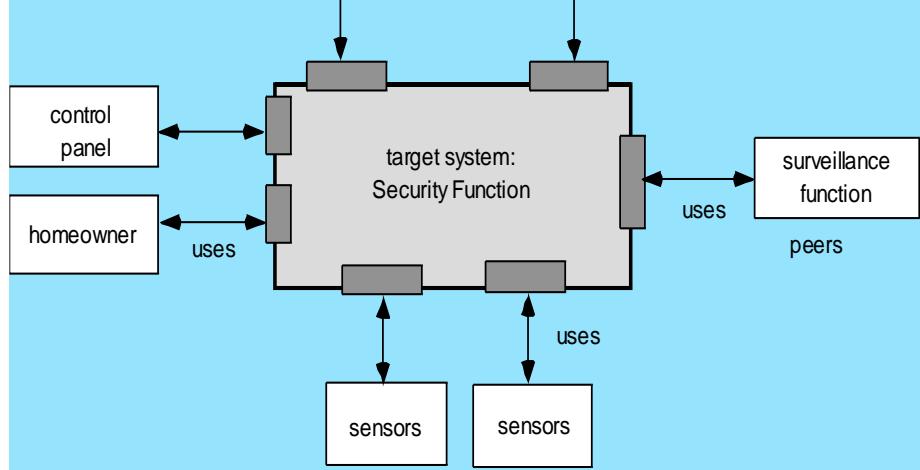
1998-1999

the characteristics and constraints of the system to be built, the architectural style and/or combination of patterns that best fits those characteristics and constraints can be chosen.



the application architecture.

- Distribution— the manner in which systems or components within systems communicate with one another in a distributed environment
 - A broker acts as a ‘middle-man’ between the client component and a server component.



Home security function.

- Detector – An abstraction that encompasses all sensing equipment that feeds information into the target system.
- The diagram illustrates the UML relationships for SafeHome security function archetypes.

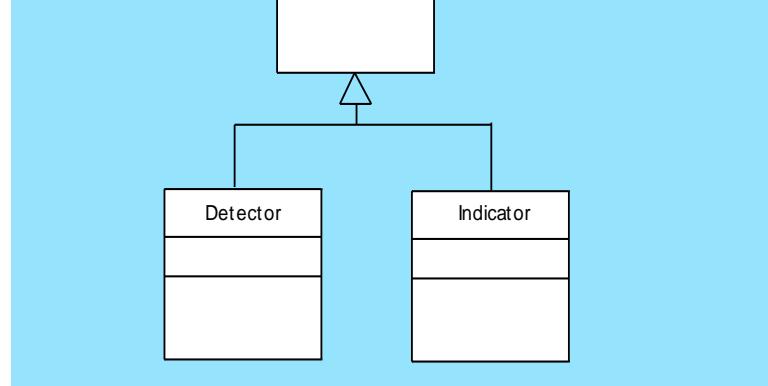
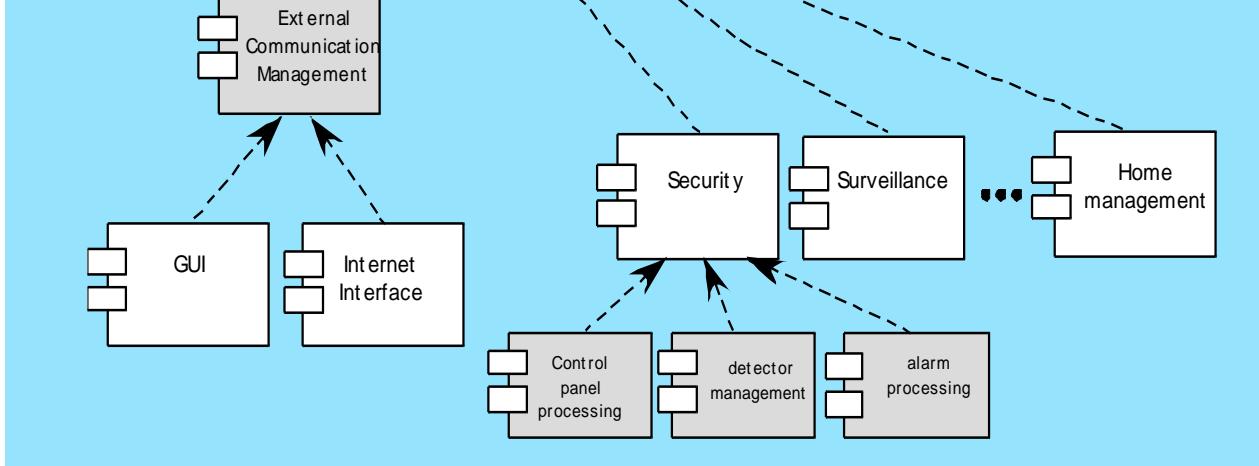


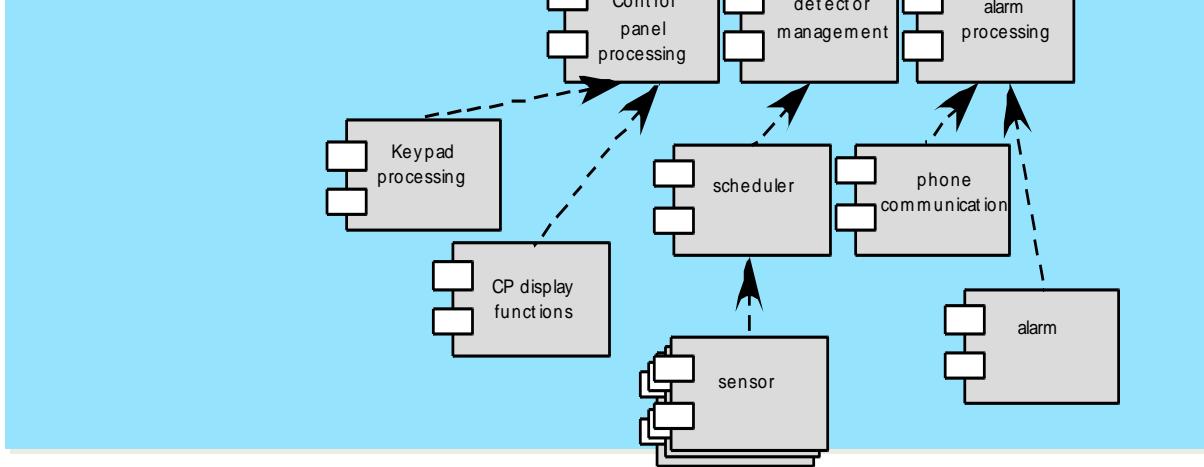
Figure 10.7 UML relationships for SafeHome security function archetypes
(adapted from [BOS00])



components might be defined to address the following functionality:

1. External communication management – coordinates communication of the security function with external entities such as other Internet-based systems and external alarm notification.
2. Control panel processing – manages all control panel functionality.
3. Detector management – coordinates access to all detectors attached to the system.
4. Alarm processing – verifies and acts on all alarm conditions.

- Components shown in figure above are elaborated to show additional details.
- For example, the detector management component interacts with a scheduler infrastructure component that implements polling of each sensor object used by the security system.
- Similar elaboration is performed for each of the components represented in figure above.



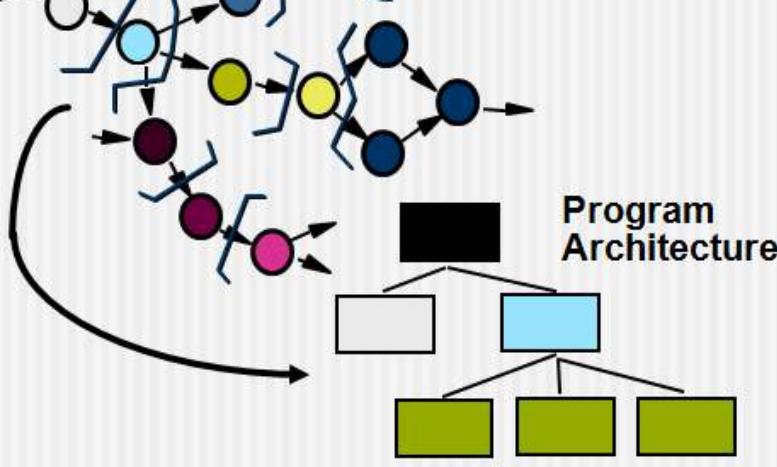
for a specific architectural style.

6. Critique candidate architectures (developed in step 3) using the sensitivity analysis conducted in step 5.

ADL

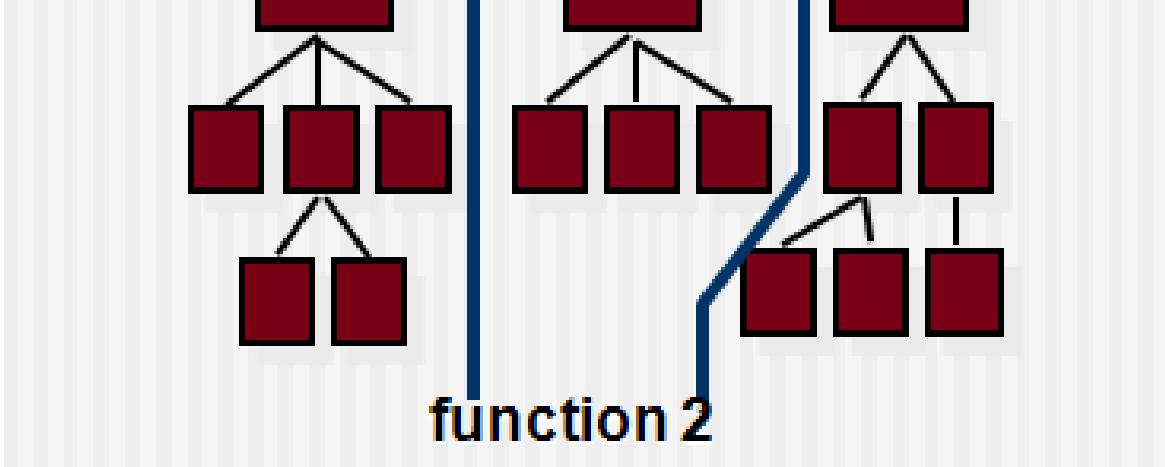
- Architectural description language (ADL) provides a semantics and syntax for describing a software architecture
- Provide the designer with the ability to:
 - Decompose architectural components
 - Compose individual components into larger architectural blocks and
 - Represent interfaces (connection mechanisms) between components.

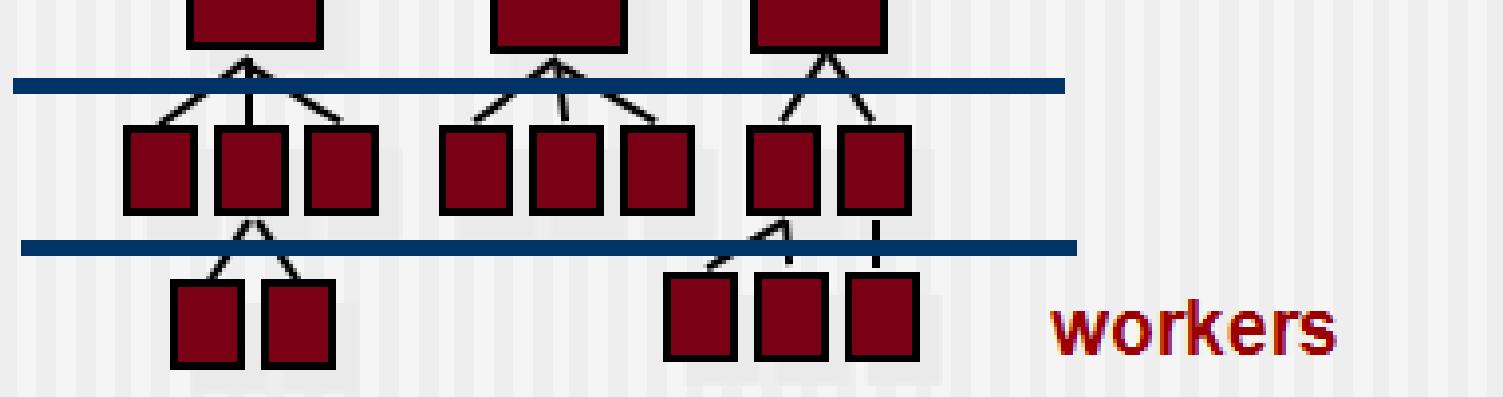
architectural design

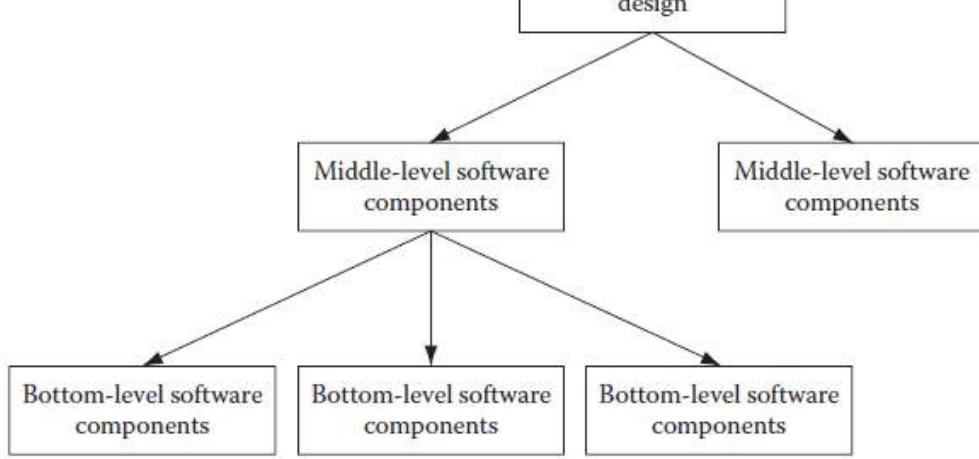


Why Partitioned Architecture?

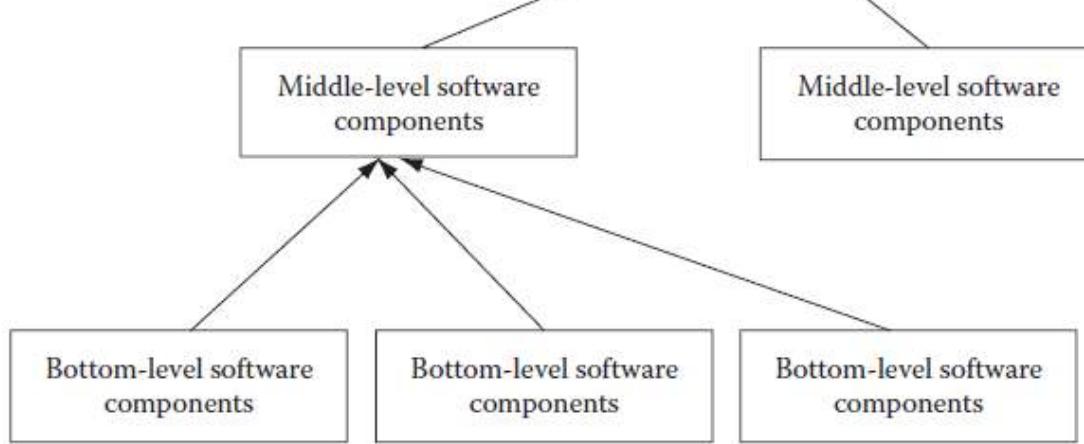
- Results in software that are easier to test
- Leads to software that are easier to maintain
- Results in propagation of fewer side effects
- Results in software that are easier to extend







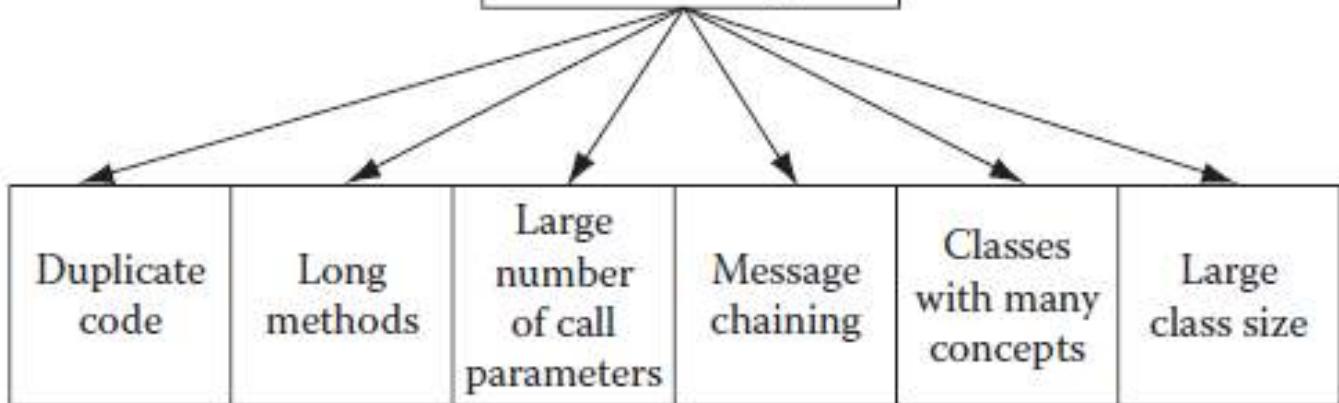
- The drawback of the top -down approach is that it is a risky model.
- The whole design has to be made in one go instead of making attempts to incrementally building the design, which is relatively a safer option.
- Generally, the top-down design approach is adopted on waterfall model-based projects.



during the iteration.

- To compensate for a sturdy and elaborate design upfront, the project team engages in refactoring (discussed next) the design to make sure that it does not become bulgy and unmanageable in later iterations.

refactoring



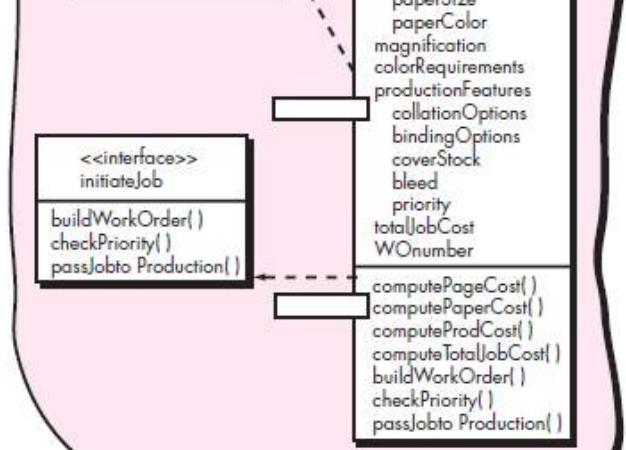
calls another method.

- When software code starts having these characteristics, then it is better to go for code cleaning or refactoring.
- Going for refactoring will be justified by savings in time due to better code reuse and make it easier to maintain code and scale up the product.
- Refactoring can be achieved by dividing cumbersome classes into smaller classes that can be managed and used in a better way.

less self-contained classes having less dependency on other classes.

- Increasing module coupling with increase in size of software product is always a concern.
- Frequent refactoring can help in reducing module coupling among classes.

- The overall intent of the software is to collect the customer's requirements at the front counter, cost a print job, and then pass the job on to an automated production facility.
- During requirements engineering, an analysis class called PrintJob was derived

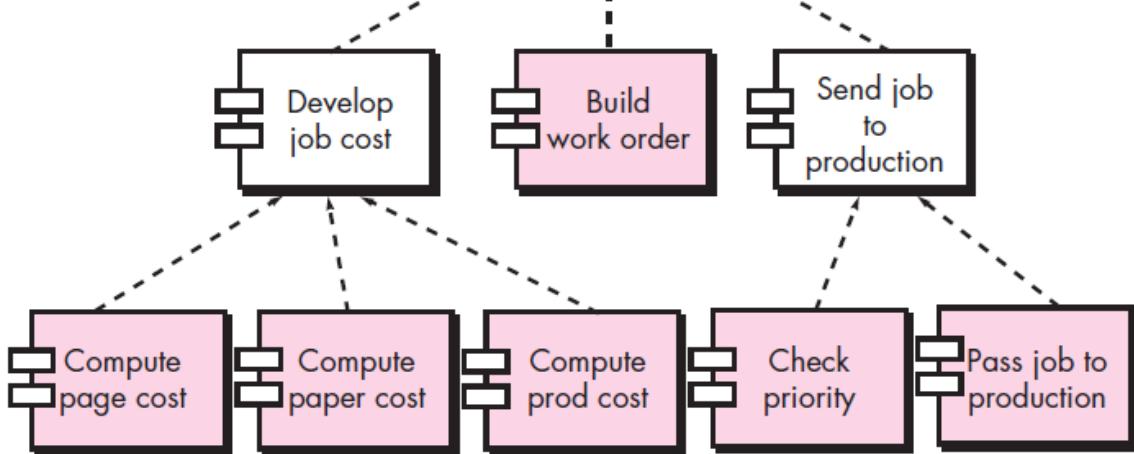


information to guide implementation.

- The original analysis class is elaborated to flush out all attributes and operations required to implement the class as the component PrintJob.
- Referring to the lower right portion of figure given, the elaborated design class PrintJob contains more detailed attribute information as well as an expanded description of operations required to implement the component.

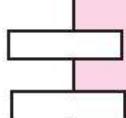
the processing required in the problem domain.

- Like object-oriented components, traditional software components are derived from the analysis model.
- In this case, however, the data flow-oriented element of the analysis model serves as the basis for the derivation.
- To illustrate this process of design elaboration for traditional components, again consider software to be built for a sophisticated print shop.
- A set of data flow diagrams would be derived during requirements modeling.



specifications provided by the customer.

- Data required to perform this function are: number of pages in the document, total number of documents to be produced, one- or two-side printing, color requirements, and size requirements.
- These data are passed to *ComputePageCost* via the module's interface.
- *ComputePageCost* uses these data to determine a page cost that is based on the size and complexity of the job—a function of all data passed to the module via the interface.



... page size = A, B, C, D
out: page cost
in: job size
in: color=1, 2, 3, 4
in: pageSize = A, B, C, D
out: BPC
out: SF

getJobData (numberPages, numberDocs,
sides, color, pageSize, pageCost)
accessCostsDB(jobSize, color, pageSize,
BPC, SF)
computePageCost()

job size (JS) =
numberPages * numberDocs;
lookup base page cost (BPC) ->
accessCostsDB (JS, color);
lookup size factor (SF) ->
accessCostDB (JS, color, size)
job complexity factor (JCF) =
 $1 + [(sides-1)*sideCost + SF]$
pagecost = BPC * JCF

together.”

- The Common Reuse Principle (CRP). “Classes that aren’t reused together should not be grouped together.”

Cohesion

- Conventional view:
 - The “single-mindedness” of a module
- OO view:
 - Cohesion implies that a component or class encapsulates only attributes and operations that are closely related to one another and to the class or component itself.

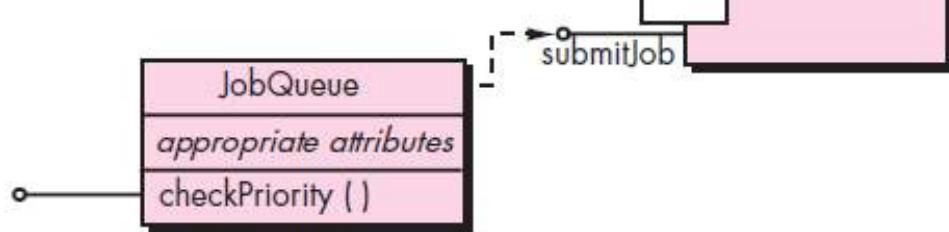
- Conventional view:
 - The degree to which a component is connected to other components and to the external world.
- OO view:
 - A qualitative measure of the degree to which classes are connected to one another.

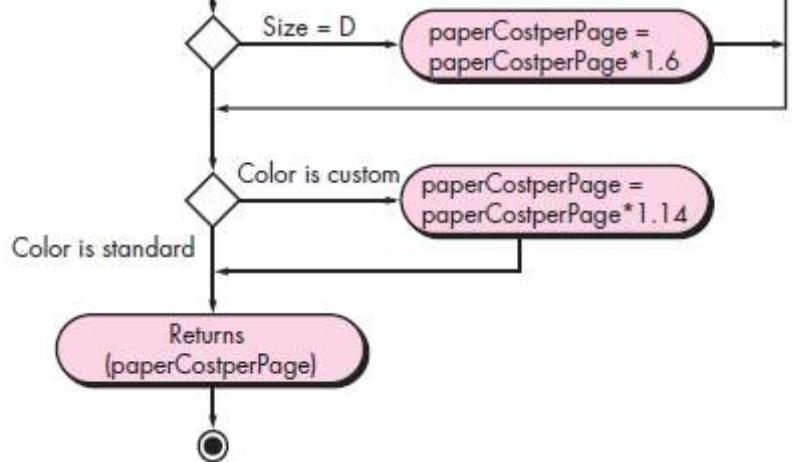
Step 5. Develop behavioral representations for classes required to manage them.

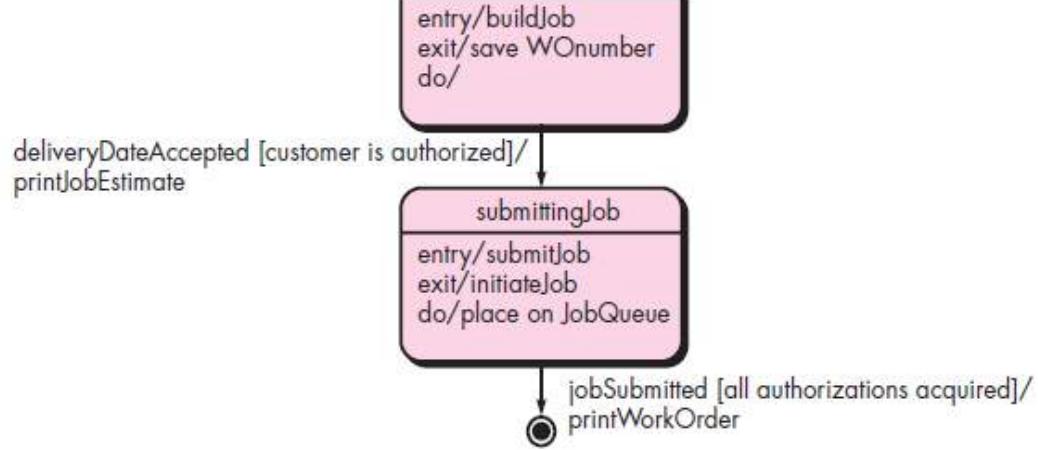
- Step 5. Develop and elaborate behavioral representations for a class or component.
- Step 6. Elaborate deployment diagrams to provide additional implementation detail.
- Step 7. Factor every component-level design representation and always consider alternatives.

:WorkOrder

:JobQueue







- Consider a Web-based video surveillance capability within SafeHomeAssured.com - Potential content components can be defined for the video surveillance capability:
 - (1) The content objects that represent the space layout (the floor plan) with additional icons representing the location of sensors and video cameras;
 - (2) The collection of thumbnail video captures (each an separate data object)
 - (3) The streaming video window for a specific camera.
- Each of these components can be separately named and manipulated as a package.

construct WebApp functional components that are identical in form to software components for conventional software.

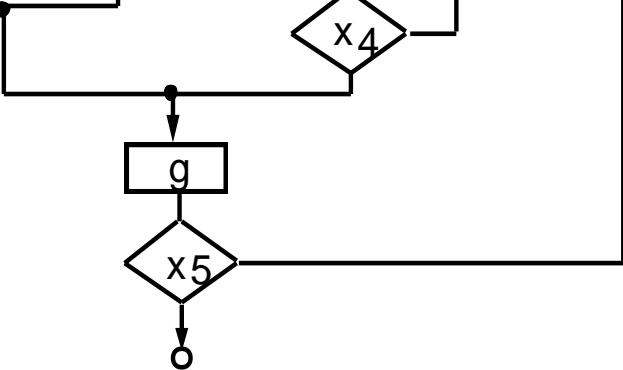
The approach:

1. Review the design description for the component
2. Use stepwise refinement to develop algorithm
3. Use structured programming to implement procedural logic
4. Use ‘formal methods’ to prove logic

close door.

**take key out;
find correct key;
insert in lock;
endif
pull/push door
move out of way;
end repeat**

- Sequence
- Conditional – If-then-else, Select-case
- Loops – Do-while, Repeat until
- Leads to more readable, testable code
- Can be used in conjunction with “proof of correctness”
- Important for achieving high quality, but not enough

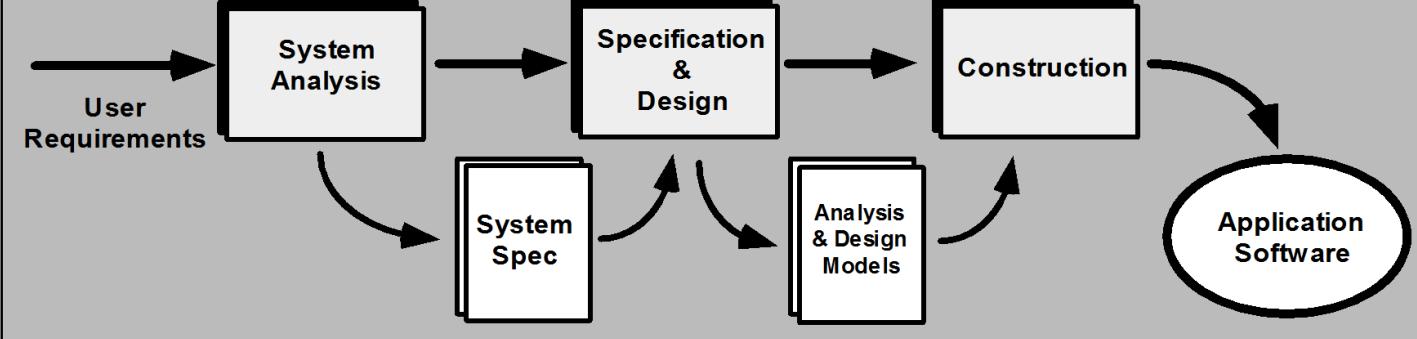


Rules					
no discount	✓				
apply 8 percent discount		✓	✓		
apply 15 percent discount				✓	✓
apply additional x percent discount	✓	✓			✓

- graphics can be generated from PDL
- enables declaration of data as well as procedure
- easier to maintain

- Are internally-developed reusable components available to implement the requirement?
- Are the interfaces for available components compatible within the architecture of the system to be built?
- At the same time, they are faced with the following impediments to reuse ...

- Few companies provide incentives to produce reusable program components.



- Is there duplication of the component's function within the domain?
- Is the component hardware-dependent?
- Does the hardware remain unchanged between implementations?
- Can the hardware specifics be removed to another component?
- Is the design optimized enough for the next implementation?
- Can we parameterize a non-reusable component so that it becomes reusable?
- Is the component reusable in many implementations with only minor changes?

- Sun JavaBeans

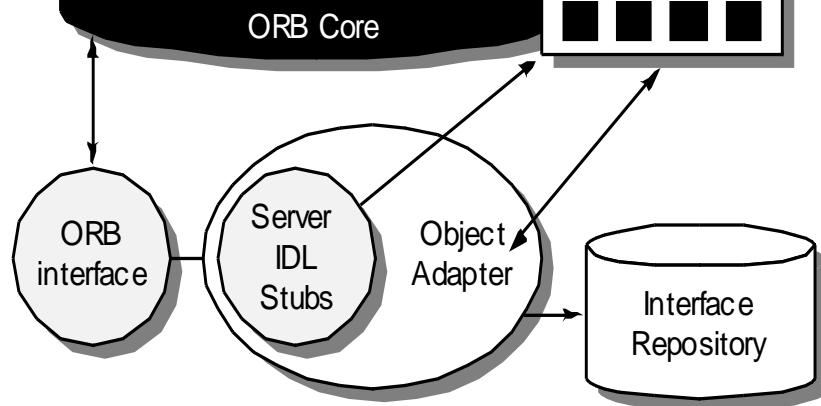
CBSE Activities

- Component qualification
- Component adaptation
- Component composition
- Component update

- Exception handling

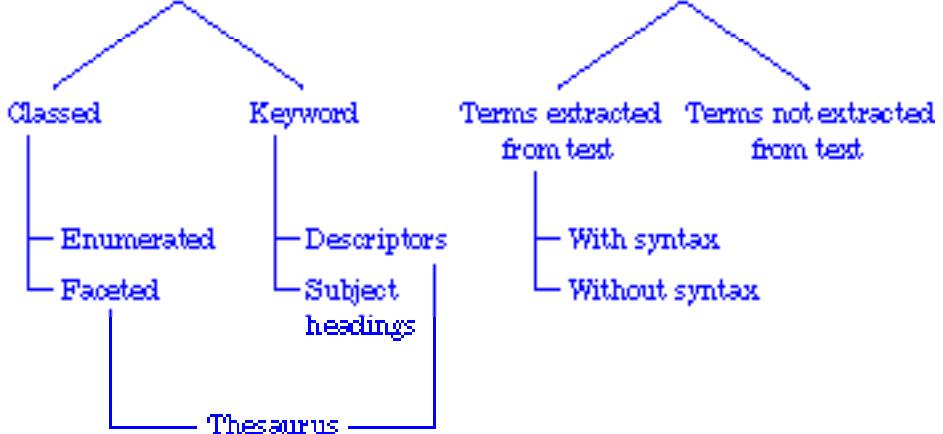
- An infrastructure must be established to bind components together.
- Architectural ingredients for composition include:
 - Data exchange model
 - Automation
 - Structured storage
 - Underlying object model

- An interface repository contains all necessary information about the service's request and response formats.



infrastructure developed using the Java programming language.

- The JavaBeans component system encompasses a set of tools, called the Bean Development Kit (BDK), that allows developers to
 - Analyze how existing Beans (components) work
 - Customize their behavior and appearance
 - Establish mechanisms for coordination and communication
 - Develop custom Beans for use in a specific application
 - Test and evaluate Bean behavior



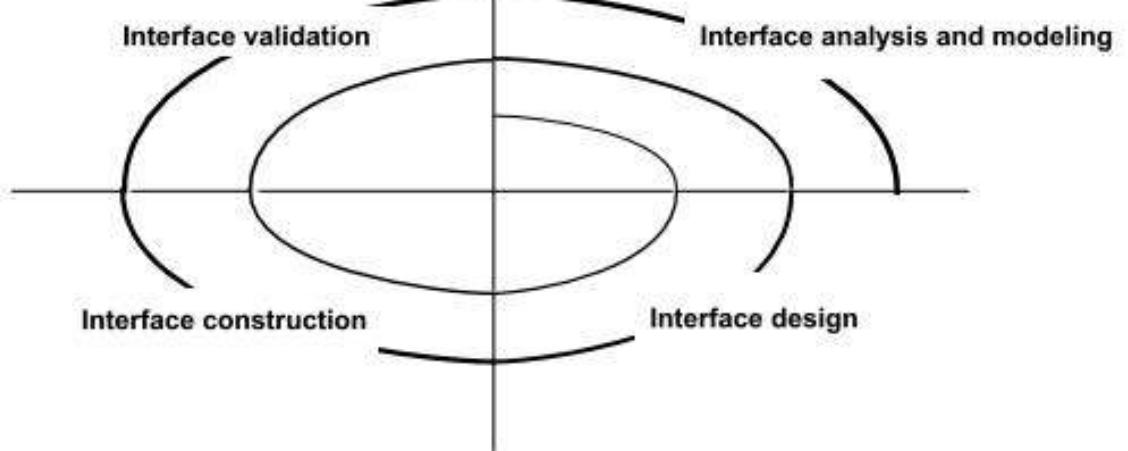
Typical Design Errors

- Lack of consistency
- Too much memorization
- No guidance / help
- No context sensitivity
- Poor response
- Arcane/unfriendly



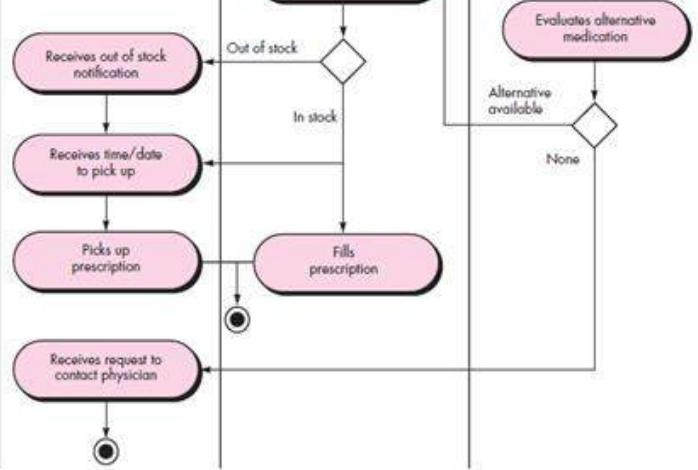
- Allow user interaction to be interruptible and undoable.
- Streamline interaction as skill levels advance and allow the interaction to be customized.
- Hide technical internals from the casual user.
- Design for direct interaction with objects that appear on the screen.

- Allow the user to put the current task into a meaningful context.
- Maintain consistency across a family of applications.
- If past interactive models have created user expectations, do not make changes unless there is a compelling reason to do so.



- What level of formal education does the average user have?
- Are the users capable of learning from written materials or have they expressed a desire for classroom training?
- Are users expert typists or keyboard phobic?
- What is the age range of the user community?
- Will the users be represented predominately by one gender?
- How are users compensated for the work they perform?
- Do users work normal office hours or do they work until the job is done?

- What work will the user perform in specific circumstances?
- What tasks and subtasks will be performed as the user does the work?
- What specific problem domain objects will the user manipulate as work is performed?
- What is the sequence of work tasks—the workflow?
- What is the hierarchy of tasks?
- Use-cases define basic interaction
- Task elaboration refines interactive tasks



Will mechanisms be available for moving directly to summary information for large collections of data?

- Will graphical output be scaled to fit within the bounds of the display device that is used?
- How will color be used to enhance understanding?
- How will error messages and warning be presented to the user?

- Response time
- Help facilities
- Error handling
- Menu and command labeling
- Application accessibility
- Internationalization

- Where have I been, where am I going?
The interface must facilitate navigation.
 - Provide a “map” (implemented in a way that is easy to understand) of where the user has been and what paths may be taken to move elsewhere within the WebApp.

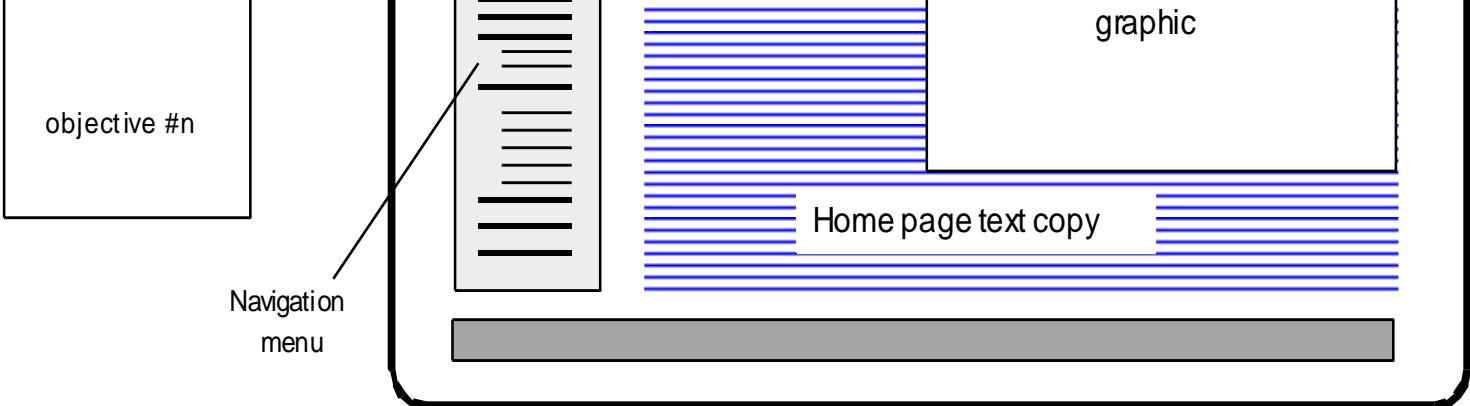
- Effective applications and services perform a maximum of work, while requiring a minimum of information from users.

builds it or the client-server environment that executes it.

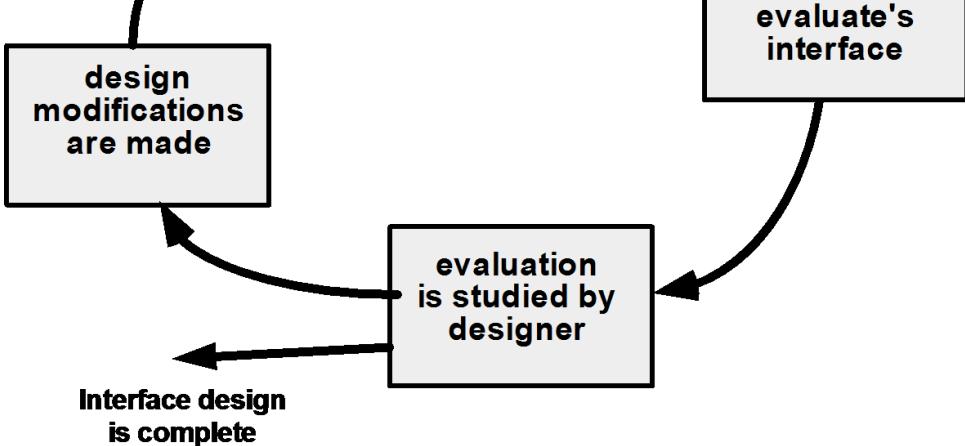
- Focus—The WebApp interface (and the content it presents) should stay focused on the user task(s) at hand.
- Fitt's Law—"The time to acquire a target is a function of the distance to and size of the target."
- Human interface objects—A vast library of reusable human interface objects has been developed for WebApps.

- Track state—When appropriate, the state of the user interaction should be tracked and stored so that a user can logoff and return later to pick up where she left off.
- Visible navigation—A well-designed WebApp interface provides “the illusion that users are in the same place, with the work brought to them.”

- Describe the interface layout for each state.
- Refine and review the interface design model.



- Consider resolution and browser window size when designing layout.



Discovering Analysis Patterns

- The most basic element in the description of a requirements model is the use case.
- A coherent set of use cases may serve as the basis for discovering one or more analysis patterns.
- A semantic analysis pattern (SAP) “is a pattern that describes a small set of coherent use cases that together describe a basic generic application.”

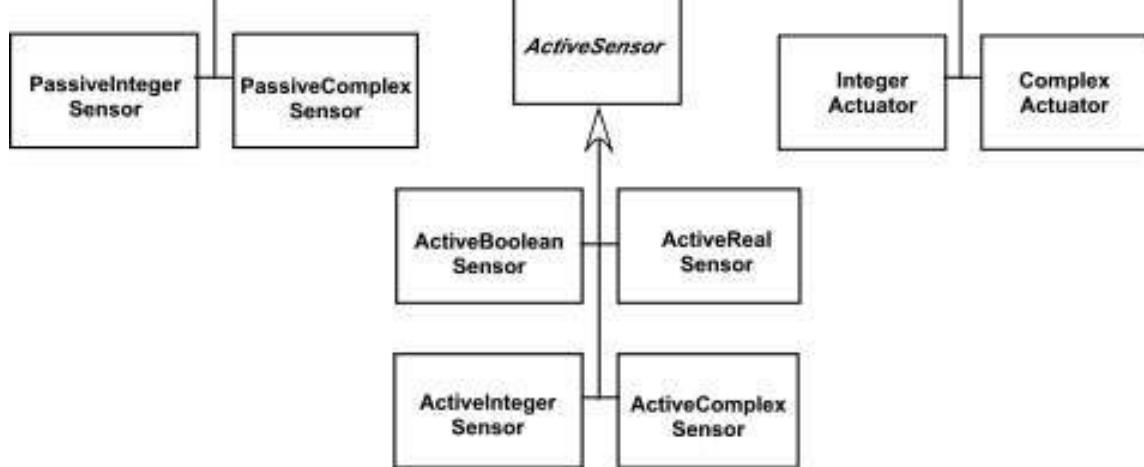
whether an object is inside 10 feet of the rear of the vehicle.

- It will automatically break the vehicle if the proximity sensor indicates an object within 3 feet of the rear of the vehicle.
- This use case implies a variety of functionality that would be refined and elaborated (into a coherent set of use cases) during requirements gathering and modeling.

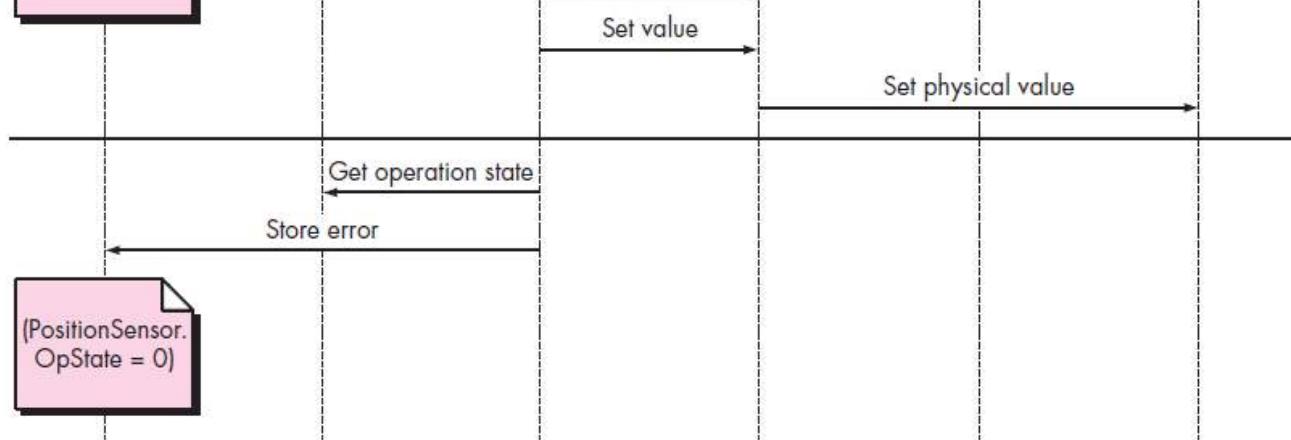
including attributes and operations.

- The Actuator-Sensor pattern uses a pull mechanism (explicit request for information) for Passive Sensors and a push mechanism (broadcast of information) for the Active Sensors.

- Each sensor and actuator should have a function implemented to check its own operation state.
- Each sensor and actuator should be able to test the validity of the values received or sent and set its operation state if the values are outside of the specifications.



classes since they should have basic functionalities such as querying the operation states.



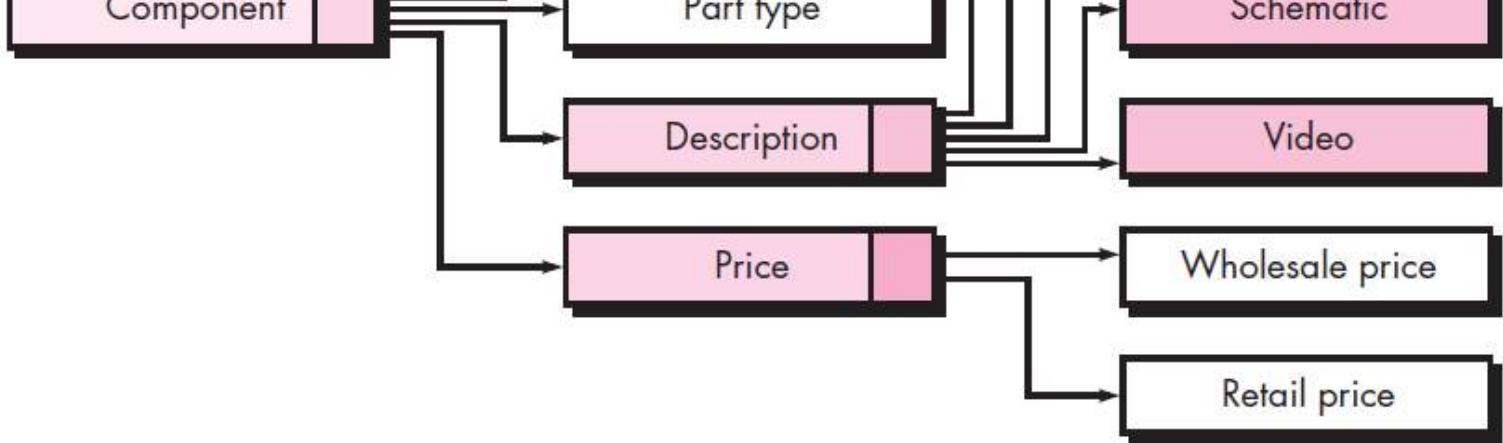
Reports that the operation state is `ERROR`.

- The ComputingComponent then sends the error code for a position sensor failure to the Fault Handler that will decide how this error affects the system and what actions are required.
- It gets the data from the sensors and computes the required response for the actuators.

content and imply other processing functions. All operations and functions are described in detail.

- **Configuration Analysis** - The environment and infrastructure in which the WebApp resides are described in detail.

- Content objects are extracted from use-cases
 - Examine the scenario description for direct and indirect references to content
- Attributes of each content object are identified
- The relationships among content objects and/or the hierarchy of content maintained by a WebApp
 - Relationships—Entity-relationship diagram or UML
 - Hierarchy—Data tree or UML



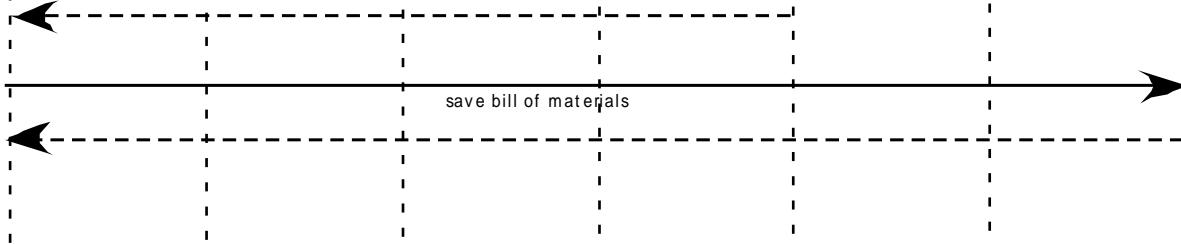


Figure 18.5 Sequence diagram for use-case:select SafeHome components

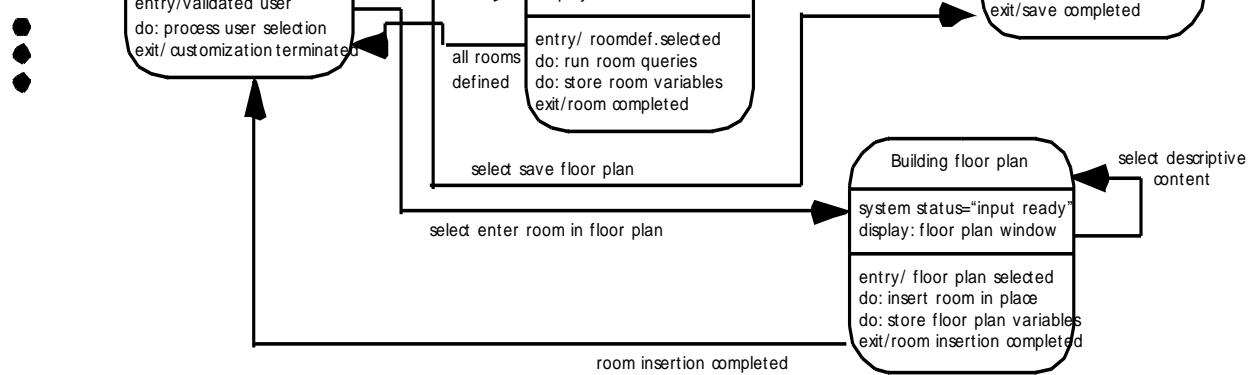


Figure 18.6 Partial state diagram for **new customer interaction**

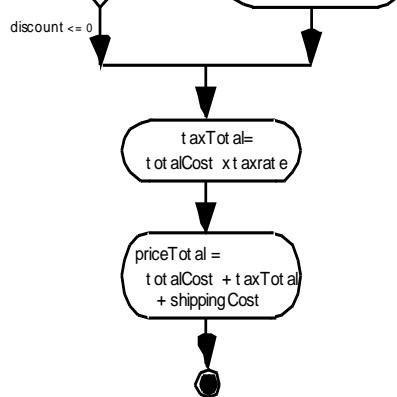


Figure 18.7 Activity diagram for `computePriceOperatic`

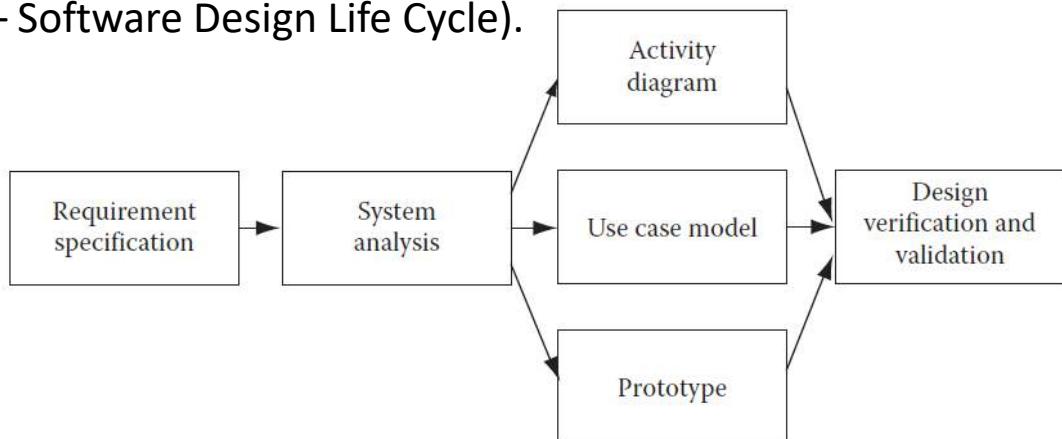
- Should navigation be accomplished via links, via search-based access, or by some other means?
- Should certain elements be presented to users based on the context of previous navigation actions?
- Should a navigation log be maintained for users?

- How should links external to the WebApp be handled, whether it is by overlaying the existing browser window or as a new browser window or as a separate frame?

design in their hands.

- Still some aspects about latter tasks can be done in advance.
- For instance, what development language will be used and how the application can be partitioned for development work can be decided at the design stage itself.
- Similarly, how maintenance and support functions will be done for the application can be determined at the design stage itself.
- Knowing in advance helps in taking care of issues that may arise in later stages.

below – Software Design Life Cycle).



Compiled by IT Department, SRMIST, KTR

Disclaimer:

The lecture notes have been prepared by referring to many books and notes prepared by the teachers. This document does not claim any originality and cannot be used as a substitute for prescribed textbooks.

CLR-3: Examine basic methodologies for software design, development, testing, closure and implementation

Automatic Code Generation

- Software Code Reuse
- Pair Programming
- Test-Driven Development
- Configuration Management
- Software Construction Artefacts

it not only includes software coding but also unit testing, integration testing, reviews and analysis.

- Construction is one of the most labor intensive phases in the software development life cycle.
- It comprises 30% or more of the total effort in software development.

- In fact, to shrink the construction time, many distributed teams, either internal or through contractors are deployed.
- The advantage to this is that these project teams do the software coding and other construction work in parallel with each other and thus the construction phase can be collapsed.

- The process itself should be efficient so that resource utilization can be optimized and thus cost of construction can be kept at a minimum.

- How well they do it depends on their experience, skills and the process they follow to do their job.
- Apart from these facilities, they also need some standards in their coding so that the work is fast as well as has other benefits like maintainability, readability and reusability (Figure in next slide).

Entity
relationship
diagram

Use cases

Using coding
standards,
techniques,
etc.

production
(software
product)



(Figure below - Software Construction Characteristics).

- Some of the coding standards include standards for code modularity, clarity, simplicity, reliability, safety and maintainability.

Modularity	Clarity	Reliability	Safety	Simplicity	Maintainability
------------	---------	-------------	--------	------------	-----------------

- Each time a particular functionality is needed in the software construction, it can be implemented using that particular module of software code.
- This increases software code reuse and thus enhances productivity of developers and code readability.

anybody reading the code could understand what a piece of code is supposed to do.

- There should also be ample white spaces in the code blocks, so that no piece of code should look crammed. White spaces enhance readability of written code.

on which the software product is to be built.

- When writing pieces of source code based on this structure, there will be little chance of defects entering into the source code.
- Generally during enhancements, the existing structure is not able to take load of additional source code and thus the structure becomes shaky.
- If the development team feels that this is the case, then it is far better to restructure the software design and then write a code based on the new structure than to add a spaghetti code on top of a crumbling structure.

- In these industries, the software product must be ensured to operate correctly and chances of error are less than 0.00001%.
- Industries like medicine and healthcare, road safety, hazardous material handling need foolproof software products to ensure that either human lives are saved (in case of medicine and healthcare) or human lives are not in danger.
- Here the software code must have inbuilt safety harnesses.

in the source code.

- Simplicity of written code can be enhanced by adopting best practices for many programming paradigms.
- For instance, in the case of object-oriented programming, abstraction and information hiding add a great degree of simplicity.
- Similarly, breaking the product to be developed into meaningful pieces that mimic real life parts makes the software product simple.

- To make sure that maintenance costs are under limit during software construction, it should be made sure that the source code is maintainable.
- It will be easy to change the source code for fixing defects during maintenance.

framework.

- In general, coding frameworks allow construction of the common infrastructure of basic functionality which can be extended later by the developers.
- This way of working increases productivity and allows for a robust and well structured software product.
- It is similar in approach to house building where a structure is built based on a solid foundation.

- Moreover, it is also a fact that it is cheaper to fix any defects found during construction at the phase level itself.
- If those defects are allowed to go in software testing (which is the next phase), then fixing those defects will become costlier.

Final
inspection



Code reviews



of deciding whether a piece of code is good enough or not.

- The developer reads those feedbacks and may decide to incorporate or to discard those suggestions.
- So this form of review is totally voluntary.
- Still, it is a powerful tool to eliminate defects or improve software code.

- The developer then can decide to incorporate those suggestions or discard them.

Reviews – Inspections

- Code inspections are final reviews of software code in which it is decided whether to pass a piece of code for inclusion into the main software build.

- With increasing hardware capacity, the size of software products has been increasing.
- Software product size affects the methods that can be used to construct specific sized software products.

automatic code generation, test-driven development, pair programming, etc.

- Structured programming enabled programmers to store large pieces of code inside procedures and functions.
- These pieces of code could be called by any other procedures or functions.
- This enabled programmers to structure their code in an efficient way.
- Code stored inside procedures could be reused anywhere in the application by calling it.

consists of a chassis, an engine, four wheels, body, and transmission.

- Each of these objects has some specific properties and specific functions.

from outside.

- This kind of representation of objects makes them robust and a system built on using them has relatively few problems.

- Then there are business analyst platforms developed by many ERP software vendors that generate code automatically when analysts configure the product.
- These analyst platforms are first built using any of the software product development methodologies.

- Generating all of the software code required to build a software application is still difficult.
- But some companies like Sun Microsystems are working to develop such a system.

utility is required is an example of code reuse.

- Code reuse in procedural programming techniques is achieved by creating special functions and utility libraries then using them in the source code.
- In object-oriented programming, code reuse is done at a more advanced level.

Libraries

Open source

Software as
a service

Inheritance



- It is known as “service oriented architecture” (SOA).

When it is the turn of the other developer to write the code, the first developer sits behind him and guides him on the requirements.

- So developers take turns for the coding and coaching work.
- This makes sure that each developer understands the big picture and helps them to write better code with lesser defects.

So here, tests drive software development, and hence it is appropriately named test-driven development.

- If the version control management is not handled properly, then many developers may start working on a wrong version of source code, and thus a lot of rework may be needed in the end.
- There is one more dimension to configuration management for the construction phase.

- Thus, development is halted until the build is rebuilt with the correct code.
- Imagine what may happen in the case of distributed teams located at far-flung locations with different time zones and a central build is being maintained.
- It will be difficult to communicate and manage the build process in such a scenario.

message and immediately tries to fix the build.

- Once the build is fixed, then other developers can check-in their code.
- Thus, configuration management plays an important role in construction phase.

Review reports are also generated after reviews are conducted.

Disclaimer:

The lecture notes have been prepared by referring to a book. This document does not claim any originality and cannot be used as a substitute for prescribed textbooks.

■ Test Artifacts

performance requirements.

component or a software system.

analysis, development testing, usability testing, qualification testing, acceptance testing, and installation testing

- Both must be written but don't constitute part of the installed software product

- Boundary value violations

- Once a set of errors are corrected, more errors occur, and testing appears to enter an endless loop

- A systematic test approach is applied

integration/testing process.



some subset of tests is re-run

These slides are designed to accompany *Software Engineering: A Practitioner's Approach*, 7/e
(McGraw-Hill 2009). Slides copyright 2009 by Roger Pressman.



builds and integrates

cluster

Integration testing

- After a smoke test is completed, detailed test scripts are executed

This means that testing operations in a stand alone fashion (the conventional unit testing approach) is usually ineffective in the object-oriented context.

5. Each functional component is unit tested.

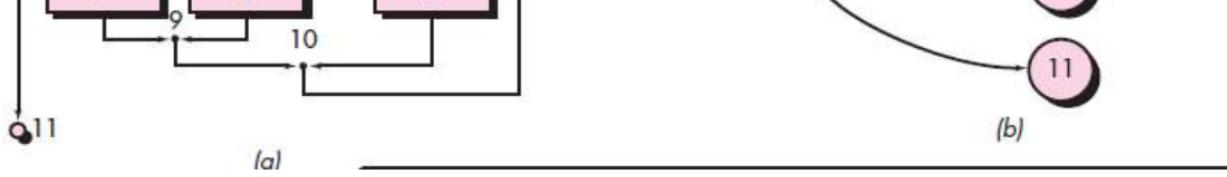
- All functional requirements are satisfied
- All behavioral characteristics are achieved

release of the software product to the entire customer base

tester attempts to break the program.

- The architectural design is well understood; documentation is available and organized

- An edge does not intersect or cross over another edge



Therefore cyclomatic complexity = 4

```
20 printf("End of list\n");
21 return 0;
22 } // End functionZ
```




```
while (currenttemp >= MINIMUM_TEMPERATURE)
```

Unstructured loops

- (4) behaviour or performance errors, and
- (5) initialization and termination errors.

7. What effect will specific combinations of data have on system operation?



or preferences
Background color: white
Text color: default color
or preferences

4. **Timing modeling.** The nodes are program objects, and the links are the sequential connections between those objects. Link weights are used to specify the required execution times as the program executes.

- From each equivalence class, test cases are selected so that the largest number of attributes of an equivalence class are exercised at once

only with the specific task at hand

- Input: {true condition} Eq classes: {true condition}, {false condition}

X →

One input item at a time

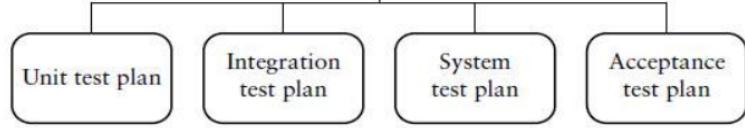
X →

L9 orthogonal array

5. When to test. What are the schedules for tests. What items need to be available.

6. When to stop testing. It is not economically feasible or practical to plan to test until all defects have been revealed. This is a goal that testers can never be sure they have reached.

- Usually staff from one or more groups cooperates in test plan development.



- Once testing is thoroughly done for these parts, then you should start testing low priority areas.

features, functionalities and quality aspects. It may consist of both explicit and implicit type of testing.

...), dynamic test point is added to static test point to get total test points (..., for system...

FP QC

TTH

- A precondition for monitoring a project is the existence of a project plan.

4. Measure and analyze results. - Measurements for monitoring and controlling must be collected, organized, and analyzed. They are then used to compare the actual achievements with standards, goals, and plans.

manage versions, releases, and revisions of documents, code, plans, and reports.

interrupt with the proper functioning of the program.

- A use case is a description of a particular use of the software by a user. In this technique, the test cases are designed to execute different business scenarios and end-user functionalities. Use case testing helps to identify test cases that cover the entire system.

- In this technique, the source code of a program is leveraged to find every executable path. This helps to determine all the faults within a particular code.

- In exploratory testing, the test design and test execution are performed concurrently.

- **History and Audit Trail:** You should be able to see its pass/fail history as well as general changes and who has run/modified the test case.

- Generally, when a certain level of quality of the application is reached, then test execution stops.

Defect
closure

etc

- Like test summary, you can include some simple metrics like Detect density, % of fixed defects.

(e.g., by increasing the number of hours worked per day or by increasing output per hour worked).

- It enables one to decide if a piece of software is ready to be released. Defect density is counted per thousand lines of code also known as KLOC



MANAGEMENT

Disclaimer:

The lecture notes have been prepared by referring to many books and notes prepared by the teachers. This document does not claim any originality and cannot be used as a substitute for prescribed textbooks.

CLR-2: Acquire the latest industry knowledge, tools and comply to the latest global standards for project management



- Maintenance Life Cycle
- Maintenance Techniques
- Software Release
- Software Maintenance

all this is your magnum opus and you need to be careful.

- You need to make sure that all your tasks are completed

- Releases may also referred to as launches or increments.

Final release timing will depend on a few factors, including where your product

business is in its market lifecycle and your available resources.

You're introducing new features to current customers

- You're introducing new features to new or different customer segments
- You're re-releasing your current product using different technology (often called "re-platforming")

- What do you hope will happen for the business once the product has been released?
- By when?

- The release planning session is an opportunity to get buy-in from your team. This is especially important given that most of the attendees will lead the efforts of actually building the product.

- A release management process incorporates all of the following:
- Planning
- Phased Communication and Supporting Team Engagement
- Reasonable stages to Readiness
- Forward adjustment on Plan

- This general plan will also provide expectations of major product changes (or dependencies) for products that may depend on your roadmap or platform.
- Plans will be revisited after each iteration. For this reason, a tracking of an external release target (with quarterly, monthly, or other precision) can be helpful.
- It will still set the expectation and trust with your customers, but can be refined by you as the plan progresses.

- Use this template to engage your greater team, who may be supporting multiple products in the portfolio only when needed.
- A standard for launch also sets expectation for when these teams will be needed internally.

- A release status will enable communication to your internal stakeholders, while feature status workflows enable granular visibility into the readiness of the feature and its current status with respect to development, staging, or QA environments.

Estimate cost of providing support	Selection of software version to be shipped	Decision for alpha, beta or regular release	Create walk around for known defects	Provide training to support staff	Make customer support strategy
------------------------------------	---	---	--------------------------------------	-----------------------------------	--------------------------------

- It is a constant struggle that calls for good product release strategies.
- Depending on the situation, the project manager may need to convince the management to cut short some of the product features to meet the deadline as well as meet quality standards.

or beta release of the product.

- In that case, the product will be released only among a few selected groups and not in the market as a whole. The controlled product release is the best option in these conditions.

- The cost of support, depending on the number of estimated users, walk around, and remaining bugs should be figured out.
- These measures will ensure that the product is transitioned into market without facing major difficulties.

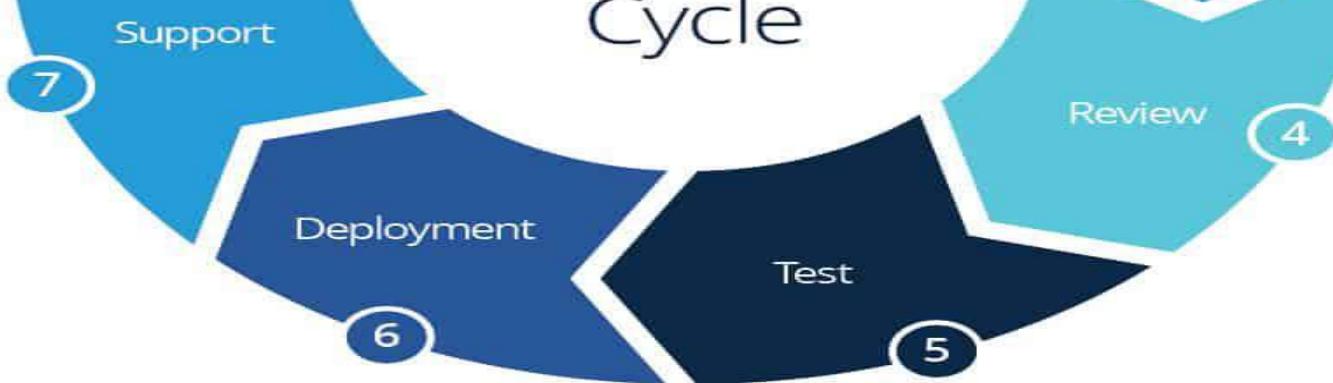
software interfaces	hardware interfaces	Create master data	Create test data	Create user accounts	infrastructure for installation
---------------------	---------------------	--------------------	------------------	----------------------	---------------------------------

your software product.

- You need to make sure that you have developed and tested all the hardware and software interfaces for integrating your product, with existing legacy systems and infrastructure.

Therefore, prepare a list of your own requirements and hand it over to your customer's support team so that they are prepared when you arrive for implementation.

Cycle



the training.

- Apart from the user manual, you also need to prepare a tutorial to include probable scenarios that may arise during operation of the product.

This will lead to a waste of your support team's time.

- It is lot better to train them now, during user training, rather than face user requests later

Software
defects

New user
requirement

Changed user
requirement

Technology
obsolescence

Better
technology

- To list various factors which adversely affect software maintenance.
- To know types of software maintenance, viz. corrective, adaptive, perfective, and preventive maintenance.
- To understand the Software maintenance life cycle.

- To appreciate need for technology change management, which is a process of identifying, selecting and evaluating new technologies.
- To understand software maintenance documentation.

- In software engineering a software needs to be 'serviced' so that it is able to meet the changing environment (such as business and user needs) where it functions this servicing of software.

Due to changes in government regulations or students to maintain competition with other software that exist in the same category.

- **Improving the Software to Support User Requirements**

To enhance functionality in a software, to improve performance .

- **Facilitating Future Maintenance Work**

Include restructuring of the software code and database used in the software.

familiarity	<p>understanding of how the system functions, and why it has been developed to function in that manner, must be preserved at all costs.</p> <p>The incremental change in each release over the life time of the system is approximately constant.</p>
Law of conservation of organizational stability	<p>Over a program's lifetime, its rate of development is approximately constant and independent of the resources devoted to system development.</p>

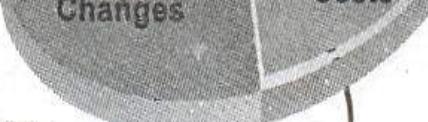
Law of declining quality	Poorly modified systems lead to introduction of defects; & The quality of E-type systems will appear to be declining as newer products emerge.
Law of feedback system	To sustain continuous change or evolution, & to minimize threats of software decay & loss of familiarity, feedback to monitor the performance is must. Feedback helps to collect metrics on the systems and maintenance efforts performance.

- Legacy system are generally associated with high maintenance costs.
- The root cause of this expense is the degraded structure that results from prolonged maintenance.

lifetime of change.

Obsolete support software	Support software may not be available for a particular platform, or no longer be supported by its original vendor or any other organization.
Obsolete hardware	Legacy system's hardware may have been discontinued.

	them.
Poorly documented	Documentation is often missing or inconsistent.
Poorly understood	As a consequence of system complexity and poor documentation, software maintainers often understand the legacy system poorly.



How much will be the cost
of maintaining a system
over its lifetime?

What will be the number
of change requests?

What will be the costs
involved in maintaining a
system over the next year?

- Costs involved in implementing change depend on the maintainability of system components.

Organizational Environment	<ul style="list-style-type: none">• Change in business policies.• Competition in market.
Operational Environment	<ul style="list-style-type: none">• Hardware platform.• Software specifications.

- Complexity of programs.
- Program Structure.

Software Maintenance
team

- Staff turnover.
- Domain expertise.

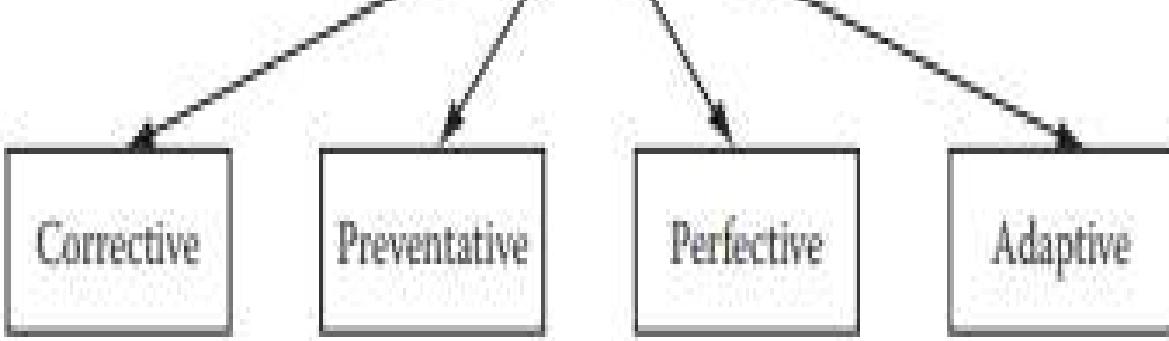
The aspects of a maintenance team that lead to high maintenance costs are:

- Staff turnover
- Domain expertise

these defects are removed.

- Change in user requirements: The business organization that was using the software product has seen a change in business transactions or business workflows that are not supported by the software product

- This situation has led to including maintainability characteristics during the entire product development cycle.
- A lot of work remains to be done during the maintenance phase of any software product. How to manage these activities so that costs can be minimized is an area of concern yet to be resolved.



rectify them.

- The maintenance team then makes a plan and fixes those defects.
- After application of the patch containing the fixes, the software starts running without these defects

- This kind of maintenance may involve changing the interface or porting the application to another hardware/ software platform

- For all these kinds of requirements, a perfective maintenance may be needed.

- In such cases, preventive maintenance on the software product can make sure that the product will be useful even after these environmental changes occur

When the organization faces such a case, it is left with no alternative but to either get an entirely different software product that will replace the existing one or do maintenance of an existing product to make it usable.

- The losses due to problems with the software can be compared to probable cost of maintenance and an ROI (return on investment) can be done.
- If we get a desirable ROI then it is better to go for maintenance.



- Osborne's model
- Iterative enhancement model
- Reuse oriented model



- What are the advantages of such a model and why is it still used?
- In the appropriate environment it can work perfectly well.

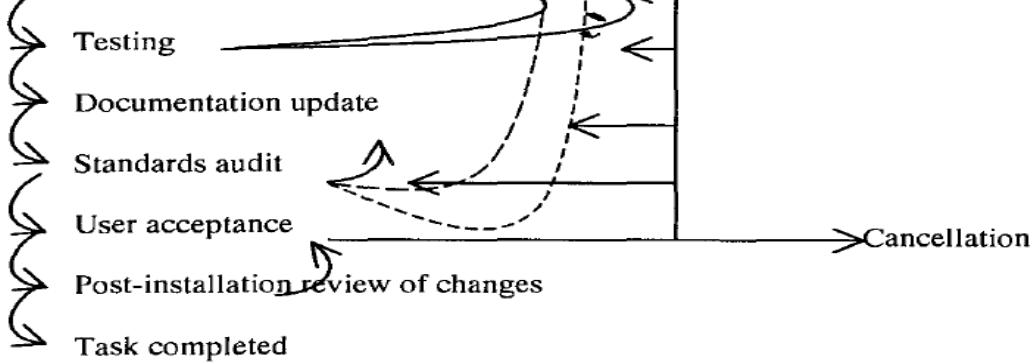
Results

New version of software

Software in use

and cost-benefit evaluations to a set of proposed changes.

- The approved changes are accompanied by their own budgets which will largely determine the extent and type of resource expended.



- A means of verifying that maintenance goals have been met;
- Performance review to provide feedback to managers.

Redesign
Current Version
& Implement

Characterize
Proposed
Modifications



Existing documentation for each stage is:

- Requirement documentation, Design Documentation, Source code documentation, & Test documentation

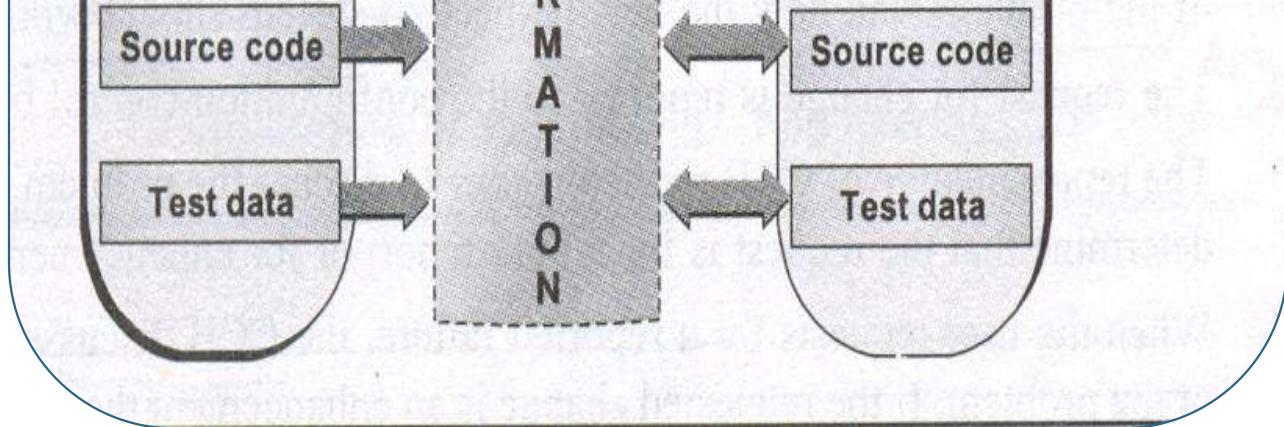
Test

Analysis

Test

Analysis

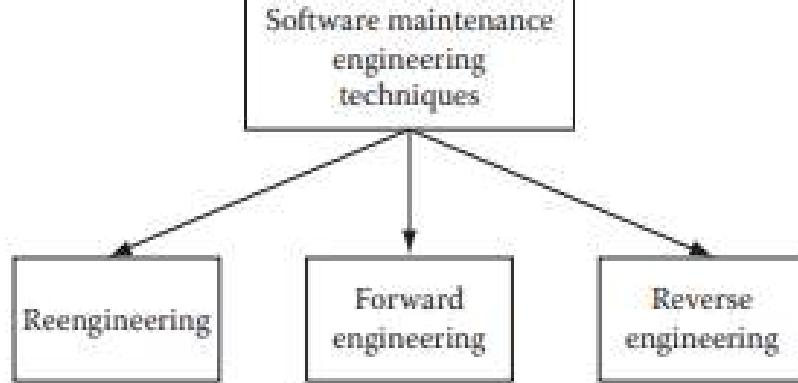
- A detailed framework is required for classification of components for reuse.



- Integrating the modified parts into the new system.

- It makes a lot of sense to go for an iterative approach.
- This approach is similar to the concept of iterative software development.

the long run is a much cheaper alternative compared to no testing/cursory testing and later spending money in providing support.



- Even if documentation is there, it is not up to date.

on the situation.

- Some of the common maintenance techniques include reengineering, reverse engineering, and forward engineering.

specifically analyzed to find out the root cause of the defect.

- Once this analysis is successful, then fixing that defect becomes easy.

- In such cases, the reverse engineering technique is adopted.
- Using this technique, similar components or product parts are constructed as compared to existing product components/parts.
- This way the software product functionality is changed as the new constructed parts will have the desired functionality.

- All new extended development is based on the existing design and construction methods and will be made for the same hardware/software platform.

- The high priority features will be definitely added and the low priority features for that iteration will be added if any time remains in the iteration.
- Our SaaS vendor does not release alpha or beta releases of its product as they do not serve mass markets.

production instances.

- Since there are no immediate customers who will be available for doing user acceptance testing, the internal testing team does the user acceptance testing as well.

maintenance programmers have moved to a different company. In order to take advantage of state-of-the-art parallel machines, the contractor wants the software to be reimplemented on a parallel platform.

- Briefly describe the techniques that will be needed to accomplish the task.
- How would you go about performing the job, bearing in mind the merits of software reuse?

to use Solaris 2.x. This also meant the users had to retrain and learn the use of new commands. Some administrative practices became out of date as a result of the upgrade.

The end result was a more efficient and cost-effective system but the cost of accommodating the upgrade went well beyond the retail price of the new software.

Software engineering point: encapsulation of parts of the code means that execution of a particular part of a program cannot accidentally modify variables that have no relevance to that section of code.

A major step forward is to take the data out of the programs altogether, to store it in external tables or files. VAT upgrades can now be carried out by providing new data files and the programs need not be modified.

Software engineering point: proper separation of data from code avoids unnecessary code modification.

VAT rates and the eligibility of different goods in different contexts are in fact nothing to do with system developers and maintainers. They are set and amended by Government bodies. Even

essentially a black box that takes in information from the program and returns a current rate.

Software engineering point: true interoperability between software systems using properly researched and formulated interfaces is the route to real flexibility and is a quantum leap forward in building maintainable systems.

and construction of the airbag finally commissioned.

Because of the tendency to treat software change in a less formal way, the software "airbag" will be bolted onto the car with no regard to safety considerations or appropriate design. Issues of safety, correct placing, and how other components are affected are unlikely to be considered until problems with the bolted-on version arise.

Mcq

1. Which Test Document is used to define the Exit Criteria of Testing?
 - a. Defect Report
 - b. Test Summary Report
 - c. Test Case
 - d. Test Plan**
2. Which of the following is not a part of a test design document?
 - a) Test Plan
 - b) Test Design Specification
 - c) Test Case Specification
 - d) Test Log**
3. During _____ it is ensured that defects are logged and get fixed.
 - a. Test bed
 - b. Defect tracking**
 - c. Test reporting
 - d. Defect scenario
4. The document helps in determining whether the product is ready for release
 - a. Test reporting**
 - b. Master test plan
 - c. Test log
 - d. Defect report
5. The exact structure of the master test plan and its content does not depend on this factor.
 - a. Type of organization
 - b. Documentation standards followed by the organization
 - c. Level of project formality
 - d. Anatomy of a test case**
6. _____ involves managing different versions of test cases, keeping track of changes in them, keeping a separate repository of test case.
 - a. Defect tracking
 - b. Test reporting
 - c. Test Plan
 - d. Test case management**
7. _____ preparation involves installing the application on a machine that is accessible to all test teams.
 - a. Test bed**
 - b. Test case
 - c. Test log

- d. Test script
8. _____ is a common way of measuring performance of a test team.
- a. Defect count per hour per day
 - b. Cyclomatic complexity
 - c. No of defects per team
 - d. Debug line of code
9. When an expected result is not specified in test case template then _____.
- a. We cannot run the test.
 - b. It may be difficult to repeat the test.
 - c. It may be difficult to determine if the test has passed or failed.**
 - d. We cannot automate the user inputs.
10. ITG stands for
- a) instantaneous test group
 - b) integration testing group
 - c) individual testing group
 - d) independent test group
- Ans D
11. By collecting _____ during software testing, it is possible to develop meaningful guidelines to halt the testing process.
- a) Failure intensity
 - b) Testing time
 - c) Metrics
 - d) All of the mentioned
- Ans C
12. In which testing level the focus is on customer usage?
- a) Alpha Testing
 - b) Beta Testing
 - c) Validation Testing
 - d) Both Alpha and Beta
- Ans A
13. Which of the following term describes testing?
- a) Finding broken code
 - b) Evaluating deliverable to find errors
 - c) A stage of all projects
 - d) None of the mentioned
- Ans B
14. White Box techniques are also classified as
- a) Design based testing
 - b) Structural testing

- c) Error guessing technique
- d) None of the mentioned

Ans B

15. Which of the following is/are White box technique?

- a) Statement Testing
- b) Decision Testing
- c) Condition Coverage
- d) All of the mentioned

Ans D

16. Boundary value analysis belong to?

- a) White Box Testing
- b) Black Box Testing
- c) White Box & Black Box Testing
- d) None of the mentioned

Ans B

17. Which of the following is not part of the Test document?

- a. Test Case
- b. Requirements Traceability Matrix [RTM]
- c. Test strategy
- d. Project Initiation Note [PIN]

Ans D

18. Alpha testing is done at

- a) Developer's end
- b) User's end
- c) Developer's & User's end
- d) None of the mentioned

ANS A

19. Which of the following is not a valid phase of SDLC (Software Development Life Cycle)?

- a. Testing Phase
- b. Requirement Phase
- c. Deployment phase
- d. Testing closure

Ans D

1. Which Testing is performed first?

- a. Black box testing
- b. White box testing
- c. Dynamic testing
- d. Static testing**

2. Verification and Validation uses _____.
- a. Internal and External resources respectively.**
 - b. Internal resources only.
 - c. External resources only.
 - d. External and Internal resources respectively.

3. _____ refers to a different set of tasks ensures that the software that has been built is traceable to Customer Requirements.
- a. Verification
 - b. Requirement engineering
 - c. Validation**
 - d. System Design

4. Which of the following is not a software testing generic characteristics?
- a) Different testing techniques are appropriate at different points in time**
 - b) Testing is conducted by the developer of the software or an independent test group
 - c) Testing and debugging are different activities, but debugging must be accommodated in any testing strategy
 - d) Equivalent class analysis

5. Minimum of four test data are available in _____
- a. Boundary Value Analysis**
 - b. Equivalence Class Partitioning
 - c. Dynamic testing
 - d. Requirement engineering

6. Focus Testing comes under _____
- a. Performance Testing
 - b. Acceptance Testing
 - c. Usability Testing**
 - d. Component Testing

7. Verification is also known as:
- a) Dynamic testing
 - b) Static testing**
 - c) Unit testing
 - d) Integration testing

8. Which of the following is not part of the Test document?

- a) Test Case
- b) Requirements Traceability Matrix
- c) Test Strategy
- d) Project Initiation Note

9. Functional testing is a -----?

- a) Test design technique
- b) Test level
- c) SDLC Model
- d) Test type

10. What is error guessing in software testing?

- a) Test control management techniques
- b) Test verification techniques
- c) Test execution techniques
- d) Test case design/ data management techniques**

1. The order in which test levels are performed is:

- a) Unit, Integration, Acceptance, System
- b) Unit, System, Integration, Acceptance
- c) Unit, Integration, System, Acceptance
- d) It depends on the nature of a project**

2. System testing is a

- a) Black box testing**
- b) Grey box testing
- c) White box testing
- d) Both a and b

3. Test cases are designed during which of the following stages?

- a) Test recording
- b) Test configuration
- c) Test planning
- d) Test specification**

4. In order to control cost, defects should ideally be detected in which phase:

- a) Coding
- b) Design
- c) Implementation
- d) Requirements Gathering**

5. Error guessing is a:

- a) Test verification techniques
- b) Test data management techniques**
- c) Test control management techniques
- d) Test execution techniques

6. What are the various Testing Levels?

- a) Unit Testing
- b) System Testing
- c) Integration Testing
- d) All of the mentioned**

7. Exhaustive testing is

- a) always possible
- b) practically possible
- c) impractical but possible**
- d) impractical and impossible

8. Which of the below is not a part of the Test Plan?

- a)Schedule
- b)Risk
- c)Incident reports**
- d)Entry and exit criteria

9. What is component testing?

- a)White-box testing**
- b)Grey box testing
- c)Black box testing
- d)Both a & c

10. ----- are the problems that threaten the success of a project but which has not yet happened.

- a)Risk**
- b)Bug
- c)Failure
- d>Error

18CSC206J Software Engineering and Project Management

Unit V MCQs

1. Which of the following is true about Corrective Maintenance?
 - A. It includes modifications and updatations done in order to correct or fix problems, which are either discovered by user or concluded by user error reports.
 - B. It includes modifications and updatations applied to keep the software product up-to date and tuned to the ever-changing world of technology and business environment.
 - C. It includes modifications and updates done in order to keep the software usable over long period of time.
 - D. It includes modifications and updatations to prevent future problems of the software.
2. What type of software testing is generally used in Software Maintenance?
 - A. Regression Testing
 - B. System Testing
 - C. Integration Testing
 - D. Unit Testing
3. What are the activities of Program modularization and Source code translation?
 - A. Forward engineering
 - B. Reverse Engineering
 - C. Reengineering
 - D. Reverse Engineering and Reengineering
4. In how many categories a maintenance is being classified?
 - A. two
 - B. three
 - C. four
 - D. five
5. Which of the following is not a cause for software maintenance for a typical product?
 - A. It is not possible to guarantee that a software is defect-free even after thorough testing.
 - B. The deployment platform may change over time.
 - C. The user's needs may change over time.
 - D. Software undergoes wear and tear after long usage.
6. Which of the following types of maintenance consumes the maximum effort for a typical software?
 - A. Adaptive

- B. Corrective
- C. Preventive
- D. **Perfective**

7. State whether the following statement is TRUE or FALSE.

The cost benefits derived from reengineering are realized largely due to decreased maintenance and support costs for the new software product.

- A. **True**
 - B. False
8. How much effort is typically expended by a software organization on software maintenance?
- A. 20 percent
 - B. 40 percent
 - C. **60 percent**
 - D. 80 percent
9. Preventive maintenance is implementing changes in existing or new requirements of user.
- A. True
 - B. **False**
10. What is a software patch?
- A. Required or Critical Fix
 - B. **Emergency Fix**
 - C. Daily or routine Fix
 - D. None of the mentioned
11. What does ACT stand for in Boehm model for software maintenance?
- A. Actual change track
 - B. Annual change track
 - C. **Annual change traffic**
 - D. Actual change traffic
12. If a software product costed Rs. 10,000,000 for development, compute the annual maintenance cost given that every year approximately 5 per cent of the code needs modification.
- A. 1.25 Million/Annum
 - B. **1 Million/Annum**
 - C. 1.05 Million/Annum
 - D. 2 Million/Annum

13. Which Software-end factors affecting maintenance Cost
- A. Structure of Software Program
 - B. Programming Language
 - C. Dependence on external environment
 - D. All mentioned above
14. Corrective maintenance is the type of maintenance that is most frequently carried out on a typical software product.
- A. True
 - B. False
15. Software reengineering process model includes restructuring activities for which of the following work items?
- A. code
 - B. documentation
 - C. data
 - D. all of the above
16. The only deliverable work product for a successful project is the working program.
- a) True
 - b) False
- Answer:B
17. What involves preparing software for external release and keeping track of the system versions that have been released for customer use?
- a) System building
 - b) Release management
 - c) Change management
 - d) Version management
- Answer:B
18. What is related to the overall functionality of the delivered software?
- a) Function-related metrics
 - b) Product-related metrics
 - c) Size-related metrics
 - d) None of the mentioned
- Answer:A

19. The foundation of the Releases module is the releases This enables the modeling of upcoming releases within a hierarchicalstructure. (choose only one)

- a) Plan
- b) Tree
- c) Graph
- d) Cycle

Answer:B

20. A release represents a group of changes in one or more applications that will be available for distribution at the same time. After defining the release, you define

- a) Plan
- b) Tree
- c) Graph
- d) Cycles

Answer:D

21. After defining the releases tree, you define and review the requirements in the Requirements module and assign them to

- a) Trees and cycles
- b) Graphs and cycles
- c) Plans and cycles
- d) Releases and cycles

Answer:D

22. After assigning requirements to releases and cycles, you assign each test set folder in the Test Lab module to a cycle. A is a group of test instances in a Quality Center project designed to achieve specific test goals.

- a) test set
- b) test tree
- c) test Graph
- d) test Cycle

Answer:A

23. If an application flaw is detected while running a test set, you can submit a defect. Quality Center automatically creates a link between the

- a) test suite, associated release and cycle, and the new defect
- b) test case, associated release and cycle, and the new defect
- c) test run, associated release and cycle, and the new defect
- d) associated release and cycle, and the new defect

Answer:C

24. A defect can be submitted to a Quality Center project from any module at any stage of the application management process.

- a) True
- b) False

Answer:A

25. You define releases and cycles in Quality Center using the module.

- a) Releases module
- b) Requirements module
- c) Test Plan module
- d) Release module

Answer:A

26. You can add user-defined fields and change the label of any of the fields in the Releases module.

- a) True
- b) False

Answer:A

27. You can use the to restrict and dynamically change the fields and values in the Releases module.

- a) QC Script Editor
- b) VBScript Editor
- c) Script Editor
- d) QuickTest Script Editor

Answer:C

28. A release represents a group of changes in one or more applications that will be available for distribution at the same time. Each release can contain a cycles.

- a) two
- b) three
- c) five
- d) number of

Answer:D

29. A cycle represents a development and QA cycle based on the project timeline. Both releases and cycles have defined

- a) start and end dates
- b) start and end times
- c) start and end modules
- d) start and end timelines

Answer:A

30. While reviewing the Coverage Progress graph, the indicates the distribution of the assigned requirements within the release's cycles.

- a) Coverage Progress curve
- b) Assigned requirements curve
- c) Requirements curve
- d) Assigned curve

Answer:B

31. The Planned coverage curve below the Assigned requirements curve indicates the percentage of requirements that are covered throughout the

- a) Project
- b) Year
- c) Plan
- d) release

Answer:D

32. The Progress tab displays statistics and graphs to provide visual indications of the current progress of your release or cycle.

- a) True
- b) False

Answer:A

33. The Passed coverage curve is below the Executed coverage curve. It indicates the test runs in which the most recent status is

- a) Passed
- b) Failed

Answer:A

34. The Quality tab helps you draw conclusions quickly and see the number of defects submitted over the course of a

- a) release
- b) cycle
- c) Both A & B
- d) None

Answer:C

35. You can move a release folder or a release to a different location in the releases tree.

Moving a release folder also moves its releases and cycles.

- a) True
- b) False

Answer:A

36. You can delete a release folder, release, or cycle. When you delete a folder, the releases and cycles under the folder are also deleted. When you delete a release, the cycles under it are not deleted.

- a) True
- b) False

Answer:B

37. You can create a duplicate of a cycle within the same release.

- a) True
- b) False

Answer:A

38. You cannot move a release folder or a release to a new location in the releases tree by dragging it.

- a) True
- b) False

Answer:B

18CSC206J – SOFTWARE ENGINEERING AND PROJECT MANAGEMENT

SAMPLE QUESTIONS - MODEL EXAM – PATTERN – May 2021

Part A - MCSA (Multiple Choice Single Answer)

Part B – MCSA / MCMA (Multiple Choice Multiple Answers)

Unit 4 & Unit 5

PART A

1. Match the following:

List I	List II
(P) Condition coverage	(i) Black-box testing
(Q) Equivalence class partitioning	(ii) System testing
(R) Volume testing	(iii) White-box testing
(S) Alpha testing	(iv) Performance testing

a) P-ii, Q- iii, R-i, S-iv

b) **P-iii, Q- i, R-iv, S-ii**

c) P-iii, Q- iv, R-ii, S-i

d) P-iii, Q- i, R-ii, S-iv

Ans: b) P-iii, Q- i, R-iv, S-ii

2. The cyclomatic complexity of the flow graph of a program provides

- a) **an upper bound for the number of tests that must be conducted to ensure that all statements have been executed at least once**
- b) a lower bound for the number of tests that must be conducted to ensure that all statements have been executed at most once

- c) an upper bound for the number of tests that must be conducted to ensure that all statements have been executed at most once
- d) a lower bound for the number of tests that must be conducted to ensure that all statements have been executed at least once

Ans: a) an upper bound for the number of tests that must be conducted to ensure that all statements have been executed at least once

3. Testing begins at _____ level and works “outward” toward the integration of the entire computer based system.

- a) **Component**
- b) Micro
- c) Unit
- d) System

Ans: a) Component

4. _____ refers to the set of tasks that ensure that software correctly implements a specific function.

- a) Software debugging
- b) Error correction
- c) Verification
- d) Validation

Ans: c) Verification

5. Which testing is focuses on the design and construction of the software architecture.

- a) Validation testing
- b) Unit testing
- c) System testing
- d) **Integration testing**

Ans: d) Integration testing

6. _____ serve to replace modules that are subordinate the component to be tested.

- a) Driver
- b) **Stub**
- c) Module

- d) Interface

Ans: Stub

7. Cyclomatic Complexity cannot be applied in _____.

- a) Re-engineering
- b) Risk Management
- c) Test Planning
- d) Reverse engineering**

Ans: d) Reverse engineering

8. Which kind of testing helps to ensure that changes do not introduce unintended behaviour or additional errors.

- a) Regression Testing**
- b) Integration Testing
- c) System Testing
- d) Validation Testing

Ans: a) Regression Testing

9. Test cases are designed during

- a) Test recording
- b) Test configuration
- c) Test planning
- d) Test specification**

Ans: d) Test specification

10. _____ is conducted at one or more end-user sites.

- a) Alpha test
- b) Beta test**
- c) Validation test
- d) Unit test

Ans: b) Beta test

11. Executing a system in a manner that demands resources in abnormal quantity, frequency or volume is

- a) Security testing
- b) Stress testing**
- c) Recovery testing
- d) Performance testing

Ans: b) Stress testing

12. Which debugging approach can be used successfully in small programs?

- a) Brute force
- b) Cause elimination
- c) Backtracking**
- d) Automated debugging

Ans: c) Backtracking

13. Which of the following is not the attributes of a good test?

- a) high probability of finding an error
- b) redundant**
- c) best of breed
- d) neither too simple nor too complex

Ans: b) redundant

14. White box testing sometimes called

- a) Basis path testing
- b) Flow graph testing
- c) Glass-box testing**
- d) Behavioral resting

Ans: c) Glass-box testing

15. Given a flow graph with 10 nodes, 13 edges and one connected components, the number of regions and the number of predicate(decision) nodes in the flow graph will be

- a) 5,4
- b) 4,5

c) 3,1

d) 13,8

Ans: a) 5,4

16. _____ is a test case design technique that complements equivalence partitioning.

- a) **Boundary value analysis**
- b) Orthogonal array testing
- c) Graph based testing
- d) Model based testing

Ans: a) Boundary value analysis

17. Which testing is an integration testing approach that is commonly used when “shrink-wrapped” software products are being developed?

- a) Regression Testing
- b) Integration testing
- c) **Smoke testing**
- d) Validation testing

Ans: c) Smoke testing

18. _____ serves as a blue print to conduct software testing activities

- a) **Test Plan**
- b) Test Scope
- c) Test Strategy
- d) Test Estimation

Ans: a) Test Plan

19. Error guessing is a

- a) *Test verification techniques*
- b) *Test data management techniques*
- c) *Test control management techniques*
- d) *Test execution techniques*

Ans: b) Test data management techniques

20. _____ is an environment that contains all the hardware and software needed to test a software component or a software system.

- a) Test case
- b) Test item
- c) **Test bed**
- d) Test criteria

Ans: c) Test bed

21. In which phase of the Software Development Life Cycle (SDLC) users or developers may discover errors or problems that cause the entire process to start over again ?

- A. System Analysis
- B. System Design
- C. System Implementation
- D. System Maintenance

Answer D. System Maintenance

22. Changes made to the system to reduce the future system failure chances is called

- A. Preventive Maintenance
- B. Adaptive Maintenance
- C. Corrective Maintenance
- D. Perfective Maintenance

Answer A. Preventive Maintenance

23. Which of the following is not part of software reengineering process model ?

- E. Forward Engineering
- F. Inventory Analysis
- G. Prototyping
- H. Reverse Engineering

Answer C. Prototyping

24. Re-structuring of software results in

- A. Higher Quality Programs
- B. Increased Maintenance Cost

- C. Difficulty in software testing
- D. Increase in ROI

Answer A. Higher Quality Programs

25. For minor changes done in source code ----- is released but for major changes a new ----- will be released

- A. Patch, Entity
- B. Entity, Version
- C. Version, Patch
- D. Patch, Version

Answer : D : Patch, Version

Part B - MCMA (Multiple Choice Multiple Answers Questions)

1. Which of the following is/are white box technique(s) ?

- A) Statement Coverage
- B) Path testing
- C) State transition testing
- D) Data flow testing

Answer : A, B, D

2. Which of the following are components of a test plan?

- A) Features to be tested
- B) Incident reports
- C) Risks
- D) Schedule

Answer : A, C, D

3. Which of the following is true about test coverage criteria?

- A) A measure of test coverage criteria is the percentage of user requirements that are not covered

- B) Test coverage criteria can be measured in terms of items exercised by a test suite
- C) Test coverage criteria are often used when specifying test completion criteria
- D) A measure of test coverage criteria is the percentage of faults found

Answer : B, C

- 4. Increasing the quality of the software, by better development methods, will affect the time needed for testing by
 - A) Reducing the testing time
 - B) Reducing the number of defects
 - C) Reducing the test coverage
 - D) Eliminating Unit Testing

Answer : A, B

- 5. Which one of the following statement(s) about system testing is/are true?
 - A) System tests are often performed by independent teams
 - B) Functional testing is used more than structural testing
 - C) Faults found during system tests can be costly to fix
 - D) End-users should be involved in system tests

Answer : A, B, C

- 6. Unit testing standard includes :
 - A. Syntax testing
 - B. Equivalence partitioning
 - C. Stress testing
 - D. Decision coverage

Answer : A, B, D

- 7. Which of the following statement(s) is/are TRUE ?
 - A. In a system, two different failures may have different severities.
 - B. A system is necessarily more reliable after debugging for the removal of a fault.
 - C. Every fault will affect the reliability of a system
 - D. Undetected errors may lead to faults and eventually to incorrect behaviour

Answer : A, D

8. Which of the following is/are not form(s) of functional testing ?

- A. Boundary value analysis
- B. Smoke testing
- C. Performance testing
- D. Security testing

Answer : C, D

9. A configuration management system normally provides:

- A. Linkage of customer requirements to version numbers
- B. Facility to compare test results with expected results
- C. The precise differences in versions of software component source code
- D. Restricted access to the source code library

Answer : A, C, D

10. Incremental testing approaches include

- A. Bottom up
- B. Top down
- C. Big-bang
- D. Functional incrimination

Answer : A, B, D

11. Which of the following is/are maintenance process model(s)?

- A. Quick Fix Model
- B. Function Point Analysis
- C. Osborne's Model
- D. COCOMO

Answer : A, C

12. The reverse engineering maintenance technique is followed when

- A. There is no information as to how the product is constructed
- B. it is almost impossible to do any modification in the source code
- C. We have ample documentation about the existing product
- D. The existing product needs to be extended to fulfill new customer needs

Answer : A, B

13. Which of the following is NOT true about Corrective Maintenance?

- A. It includes modifications and updates done in order to correct or fix problems, which are either discovered by user or concluded by user error reports.
- B. It includes modifications and updates applied to keep the software product up-to date and tuned to the ever-changing world of technology and business environment.
- C. It includes modifications and updates done in order to keep the software usable over long period of time.
- D. It includes modifications and updates to prevent future problems of the software.

Answer : B, C, D

14. Which Software-end factors affect maintenance Cost?

- A. Structure of Software Program
- B. Programming Language
- C. Dependence on external environment
- D. Number of test engineers deployed

Answer : A,B,C

15. Software Maintenance must be performed in order to:

- A. Catch defects early in the software development cycle
- B. Correct faults encountered by end users
- C. Implement enhancements requested by customers
- D. Migrate legacy software to suit latest technology trends

Answer : B,C,D

PART A

- 1) The process of generating analysis and design documents is known as
- a) Software engineering
 - b) Software re-engineering
 - c) Reverse engineering
 - d) Re-engineering

Answer- C

- 2) Which of the following manuals is not a user documentation?
- a) Beginner's Guide
 - b) Installation guide
 - c) Reference Guide
 - d) SRS

Answer- D

- 3) When Testing should be stopped?

- A)When manager asks to stop
- B)When leader asks to stop
- C)When enough money are spend on testing
- D)It depends on risk associated with that project.

Answer- D

- 4) Bug status is set to postpone due to _____.

- A)Priority of that bug may low.
- B)Lack of time for the release.
- C)The bug may not be the major effect in the software.
- D) Data may be unavailable.

Answer- D

- 5) What is the main task of test planning?

- a)Measuring and analyzing results
- b)Evaluating exit criteria and reporting
- c)Determining the test approach
- d)Preparing the test specification

Answer- C

PART B

1) Which of the following is/are the uses of software testing tools?

- i. Test tools are used in reconnaissance.
- ii. Test tools help in managing the testing process.
- a) i only
- b) ii only
- c) Both i and ii
- d) None of the above

Answer- C

2) Which of the following is/are the purposes of using software testing tools?

- i. To improve the efficiency of test activities by automating repetitive tasks.
- ii. To automate the activities that require significant resources when done manually.
- iii. To automate the activities that cannot be executed manually.
- a) i and ii only
- b) ii and iii only
- c) i and iii only
- d) All i, ii and iii

Answer- D

3)

Followings are major tasks of Test planning activity. Arrange them in correct order.

1. Determine Test approach
2. Determine Required test resources
3. Determine scope and risks and identify objectives of testing
4. Implement the test policy and/or the test strategy
5. Determine the exit criteria
6. Schedule test analysis and design tasks, test implementation, execution and evaluation

A) 3,1,2,5,4,6

B) 3,1,2,5,6,4

C) 3,2,1,5,4,6

D) 3,1,4,2,6,5

Answer- D

4) The modification of the software to match changes in the ever changing environment, falls under which category of software maintenance?

- a) Corrective
- b) Adaptive
- c) Perfective
- d) Preventive

Answer- B

5) Which regression test selection technique exposes faults caused by modifications?

- a) Efficiency

- b) Precision
- c) Generality
- d) Inclusiveness

Answer- D

UNIT-4

1)An "Online TimeSheet Management System" web based software was developed by Suraj and Animesh of CSE Department. Suraj developed 3 main master high level modules and Animesh developed 3 detail low level modules. The main master entry screen functionalities was handled by Suraj and detailed timesheet entry screens and report generation of all data was handled by Animesh. High and low-level modules are bundled based on the control and data processing they provide for a specific program feature. They need to test the functionalities after bundling the modules , which strategy would best suit for this kind of application

- A. Top-Down Integration
- B. Module Level Integration
- C. Bottom-Up Integration
- D. Sandwich Integration

ANSWER: D

2)Consider the below given function Add_element procedure , which was used to insert elements into a list

```
void Add_element (int list[], int elem[], int cnt)
{
    int i,j,key;
    for (i=0; i<=cnt; i++) elem[i] = i;
    for (i=2; i<=cnt; i++)
    {
        key = elem[i];
        j = 1;
        while (list[elem[j-1]] > list[key])
        {
            elem[j] = elem[j-1];
            j--;
        }
    }
}
```

```

    elem[j] = key;
}

for (int l=0;l<cnt;l++)
printf("%d\n",elem[l]);

}

```

After creating a suitable control graph for the function , find how many edges and nodes would be present in it respectively?

- A. 14,11
- B. 14 , 12
- C. 12,12
- D. 13,11

ANSWER: B

3)Create a Calculator program in C language . The Pseudo code for the program follows the guidelines given below

1. Add individual functions for each arithmetic calculation.
2. In case of Subtraction , a check is made to ensure that the result would be positive always
3. In case of division, a check is made to ensure that the result is not less than 1
4. Add a main function which would invoke the arithmetic functions after getting user inputs.

Sketch the control flow graph and find the cyclomatic complexity of the entire program

- A. 5
- B. 6
- C. 7
- D. 8

ANSWER: C

4)"eLetter" is a web based application used to send the communication of wards internal marks to their respective parents. The software uses SQL as backend . A database by name - "LetterDB" is created with

tables to student details like Regno, Name, Parents Name , Parent Contact no, Email and another detail table which contains the internal assessment marks obtained by the students. The student master table and assessment detail table are linked using Regno. The "LetterDB" contains sql user defined functions, stored procedures, triggers to aid in calculating the total internal marks based on individual assessment marks. The Front end application generates the Letter to Parent report and send that through parent email id. The Testing team is now assigned with the job of doing White box testing, Identify on which of the following is covered under white box testing?

- A. Checks whether the letter to parent report is generated
- B. Checks and validates the database functions, triggers and procedures
- C. checks whether the student details are printed
- D. checks the web application for behavioural or performance errors

ANSWER: B

5)The employees of a software company are given access to their employee portal with a valid username and password field. The portal login accepts a minimum of 8 characters and maximum of 12 characters . Which of the following combination of test cases would be a valid as per the Boundary Value Analysis test case design methodology

- A. Test Cases : Password of length 7 characters, 8 characters,9-11 characters,12 characters and more than 12 characters
- B. Test Cases : Password of length 6 characters,7 characters,8-11 characters,12 characters and more than 12 characters
- C. Test Cases : Password of length 7 characters,8 characters,9 characters,11 characters and 12 characters
- D. Test Cases : Password of length 7 characters,8 characters,9 -12 and more than 12 characters

ANSWER: A

6)A simple online bulletin system was developed to send out centralized communication to the students of computer science department. The application was estimated to have 20 function points and 10 functional characteristics which are based on user importance,usage-intensity,complexity , uniformity and interfacing systems. The application has 8 dynamic quality characteristics and 4 static quality characteristics. The factors influencing the environment of a software application are test tools with rating - 4 , development tests(rating - 4) and Testware determined by usable dataset (rating - 2). The factors influencing the productivity is based on skillset of human resources with a rating of 0.8 . The control factors which can be applied externally is about 10% of the primary test hours . With the help of the above data calculate the Total Test hours for online Bulletin System

A. 13400 hours

B. 14784 hours

C. 19958 hours

D. 15584 hours

ANSWER: B

7)Black box test case design also known as.....design technique.

A. Structure based

B. Specification based

C. Experience based

D. System based

ANSWER: B

8)The main focus of boundary value analysis design methodology is to

A. Explore the output errors

B. Explore input errors

C. Rectify the errors

D. Report the errors

ANSWER: B

9)Which metric is used to the number of defects confirmed in software/module during a specific period of operation or development against the size of the project counted in KLOC?

A. Total Errors Count

B. Status of Defect

C. Count of path traversed

D. Density of Defect

ANSWER: D

10) Analyze the following conversation in a toll booth

Toll officer: "What type of ticket do you require, single or return? " IF the customer wants 'return'

Toll Officer: "return within 4 hrs or more ?"

IF the customer replies 'within 4 hrs'

Toll officer: "Amount to be paid Rs 15"

ELSE

Toll Officer: "Amount to be paid Rs 20"

ENDIF

ELSE

Toll Officer: "Amount to be paid Rs 10"

ENDIF

what would be the minimum no of tests required to achieve 100% statement coverage?

A. 1

B. 2

C. 3

D. 4

ANSWER: C

11) What are testing methodologies that may be part of an organization's testing strategies?

A. Analysis, function based, contact based and non-reactive strategy

B. Analysis, non-functional based, control based and non- Regression strategy

C. Analytical, Model based, Methodical and Reactive strategy

D. Characteristic, unsystematic, contract based and active strategy

ANSWER: C

12)A numeric field accepts age of a candidate registering for a Government job. The age for applying is greater than 22 and less than or equal to 28 only. Based on the above statement specify which of the following covers the MOST boundary values?

- A. 18,22,28,30
- B. 12,22,25,30
- C. 22,23,25,28
- D. 18,24,28,29

ANSWER: D

13)Certain People apply for personal load to a Privatized bank in order start up a new business. The criteria for approving loan are as follows

- a) Less than 20 – Refuse loan
- b) Age between 21 – 35 – Approve Personal loan with a limit of 2 lac
- c) Age older than 35 – 50 – Approve personal loan with limit of up to 5 lac

Considering the above scenario,Which of the following sets of values lie in the equivalence classes?

- A. Valid Test Data - 20,36,51
- B. Valid Test Data - 19 ,30,43
- C. Valid Test Data - 21,35,44
- D. Valid Test Data – 20,35,50

ANSWER: B

14)Given below are some of the various testing activities performed by a software team

- I. Person checks if the button shadow works properly in his system
- II. Checks performed to test the feature in multiple browsers especially the browsers suggested by the client
- III. Checks performed if the box-shadow works when integrated with the entire page or the site
- IV. Checks performed to analyse and see whether the features suggested by the client is feasible

Which of the above steps falls under verification testing?

A. I & II

B. I & III

C. II & IV

D. II & III

ANSWER: B

15)The Testing team performs various testing and logs in certain defects, which are fixed by the Development team on an incremental basis and builds are given for further testing. What kind of testing would be performed before delivering it to the testing team?

A. System Testing

B. Integration Testing

C. Smoke Testing

D. Functional Testing

ANSWER: C

16)The end users and stake holders are performing the testing to ensure which its working as per their requirements and needs, what type of testing strategy do they perform?

A. Performance Testing

B. Alpha Testing

C. Beta Testing

D. User Acceptance Testing

ANSWER: D

17)The QA testing team re-executes a small subset of tests that have already been conducted and ensure that the changes are not creating negative effects by using the following testing strategy

A. Alpha Testing

B. Beta Testing

C. Regression Testing

D. Smoke Testing

ANSWER: C

18)The QA team of a Library Management System project defines the list of tests to be performed, the level of tests to be covered and their relationship with the requirements. The team also frames the test implementation strategy; testing efforts to perform the test, test policy in align with the organization, their exceptions and impact. What is the name of the document framed by the testing team, which covers the above activities?

A. Master Test Plan

B. Project Plan Document

C. Test Plan Document

D. Test Framework

ANSWER: A

18CSC206J SOFTWARE ENGINEERING AND PROJECT MANAGEMENT
UNIT-IV
SOFTWARE TESTING

1 MARKS

S.NO	QUESTIONS	LEVEL	CLO	PG. NO
1.	End result of Software Requirement Analysis is _____. a. Functional and Behavioral b. Architectural and Structural c. Usability and Reliability d. Algorithmic and Data Structure	1	4	R5 153
2.	Which Testing is performed first? a. Black box testing b. White box testing c. Dynamic testing d. Static testing	1	4	R5 194
3.	Verification and Validation uses _____. a. Internal and External resources respectively. b. Internal resources only. c. External resources only. d. External and Internal resources respectively.	1	4	R51 89
4.	Testing beyond normal operational capacity is _____. a. Load testing b. Performance testing c. Stress testing d. Dynamic testing	1	4	R1 479
5.	The expected results of the software is _____. a. Only important in system testing b. Only used in component testing c. Most useful when specified in advance d. Derived from the code	1	4	R1 712
6.	When an expected result is not specified in test case template then _____. a. We cannot run the test. b. It may be difficult to repeat the test. c. It may be difficult to determine if the test has passed or failed. d. We cannot automate the user inputs.	2	4	R5 173
7.	Test cases are created in which phase? a. Test Specification b. Test Planning c. Test Requirement d. Test Configuration	1	4	R1 449
8.	_____ refers to a different set of tasks ensures that the software that has been built is traceable to Customer Requirements.	1	4	R1 450

a. Verification			
b. Requirement engineering			
c. Validation			
d. None of the above			
9. Which granularity level of testing checks the behavior of module cooperation?	1	4	R1 459
a) Unit Testing			
 b) Integration Testing			
c) Acceptance Testing			
d) Regression Testing			
10. Which testing is an integration testing approach that is commonly used when “shrink-wrapped” software products are being developed?	1	4	R1 463
a) Regression Testing			
b) Integration testing			
 c) Smoke testing			
d) Validation testing			
11. Which of the following is / are not a Iterative Model?	1	4	R1 61
a. RAD			
b. Incremental			
 c. V model			
d. Spiral Model			
12. Which is not true in case of Unit Testing? [REF 1, PAGE:466]	1	4	R1 466
a. It decreases the software development speed.			
b. It can't be expected to catch every error in a program.			
c. In this tester evaluates if individual units of source code are fit for use.			
d. It is usually conducted by the development team.			
13. Focus Testing comes under _____.	1	4	R1 540
a. Performance Testing			
b. Acceptance Testing			
 c. Usability Testing			
d. Component Testing			
14. Difference between Retesting and Regression Testing is _____.	1	4	R5 197
a. Retesting ensures the original fault has been removed where as regression testing looks for unexpected side-effects.			
b. Retesting looks for unexpected side-effects where as regression testing ensures the original fault has been removed.			
c. Retesting is done after faults are fixed where as regression testing is done earlier			
d. Retesting is done by developers whereas regression testing is done by independent testers			
15. First component of the DFD is _____ .	1	4	R1 187
a. Process			
b. Flow			
c. Entity			
d. Level			

Minimum of four test data are available in _____ .	1	4	R1
a. Boundary Value Analysis			498
b. Equivalence Class Partitioning			
c. Both A and B			
d. None of these.			
17. Which coupling should be avoided in software? [REF 1, PAGE:289]	1	4	R1
a. Data coupling			289
b. Content Coupling			
c. Control coupling			
d. Stamp coupling.			
18. Cyclomatic Complexity cannot be applied in _____ .	1	4	R1
a. Re-engineering			488
b. Risk Management			
c. Test Planning			
d. Reverse engineering			
19. Data classification is done by which type of Decision Tree?	1	4	R1
a. Regression Tree			715
b. Boosted Tree			
c. Classification Tree			
d. Bagging Tree			
20. Which of the following is not a software testing generic characteristics?	1	4	R5
a) Different testing techniques are appropriate at different points in time			335
b) Testing is conducted by the developer of the software or an independent test group			
c) Testing and debugging are different activities, but debugging must be accommodated in any testing strategy			
d) None of the mentioned			

4 MARKS:

1. **What is software testing?** L2 CLO4
In most quality standards documents, software testing is divided into two parts: “validation” and “verification.” While verification implies that the developed software is working as intended by checking the requirement specifications, design, source code, etc., in static mode, validation implies that the software has been validated to be working after running it and checking whether all functionality meets the requirements. Verification techniques are also known as static testing, since the source code is not run to do testing. Each work product including requirement specifications, design, and source code during software development is tested using static methods.
2. **How does software testing help in increasing quality of a software product?** L2 CLO4
It is a fact that the exact number of defects in a software product is difficult to find. At best it can be predicted using some defect estimation tools. It is also impossible to detect all defects in a software product. Nevertheless, finding and fixing critical bugs up to an acceptable limit as per expectations is important. If there are more defects in the product after the product enters production, then the project team will be in big trouble. The support costs for a bug ridden product will be too high. So, less than required testing is a certain call for rebuke from stakeholders. So the software testing helps in providing a quality software product which is bug free.

3. What techniques are used for testing software?

L2 CLO4

Verification, Validation, Test Prioritization, Effort Estimation, Test Point Analysis, Defect Tracking

4. List out the Problems with Traditional Development Model with neat diagram.

L2 CLO4

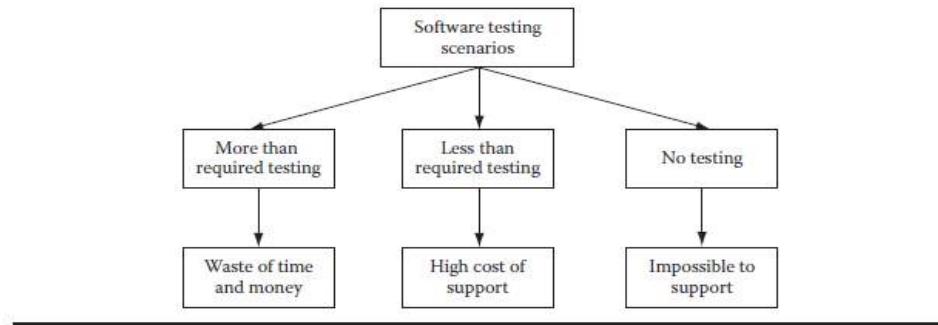


Figure 13.1 Software testing scenarios.

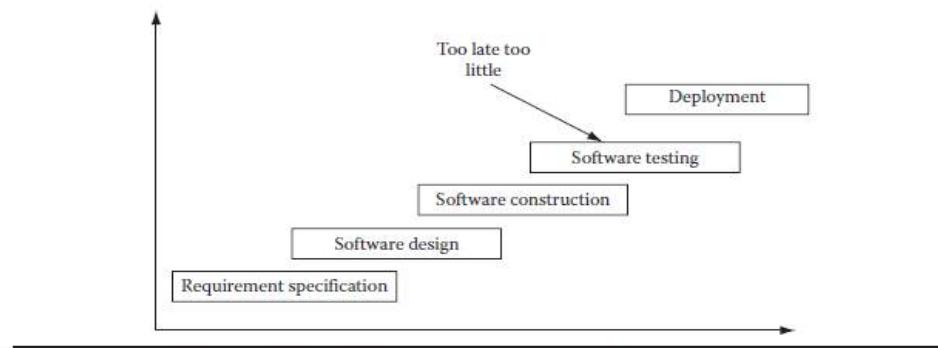


Figure 13.2 Traditional software development model (too little, too late testing).

5. Describe about Verification and Validation.

L2 CLO4

In most quality standards documents, software testing is divided into two parts: “validation” and “verification.” While verification implies that the developed software is working as intended by checking the requirement specifications, design, source code, etc., in static mode, validation implies that the software has been validated to be working after running it and checking whether all functionality meets the requirements.

Verification techniques are also known as static testing, since the source code is not run to do testing. Figure 13.3 shows that each work product including requirement specifications, design, and source code during software development is tested using static methods. The requirement specifications are reviewed for completeness, clarity, design ability, testability, etc. The software design is reviewed for robustness, security, implementability, scalability, complexity, etc. The source code is reviewed for dead code, unused variables, faulty logic, constructs, etc.

Once the source code is ready to be run as a system, validation testing can be started. Validation testing is also known as dynamic testing as, in this case, the source code is actually run to determine that it is running per specifications. During validation, unit, integration, system, and finally user acceptance testing are performed. Unit testing is done to ensure each unit piece of source code is free from defects. Once unit testing is done, then this piece of code is integrated with the main source code build. But before integrating to the main build, it is strongly advisable to do local integration testing on the developer’s own computer. Only when the source code runs smoothly and all integration tests pass, the source code should be integrated with the main build.

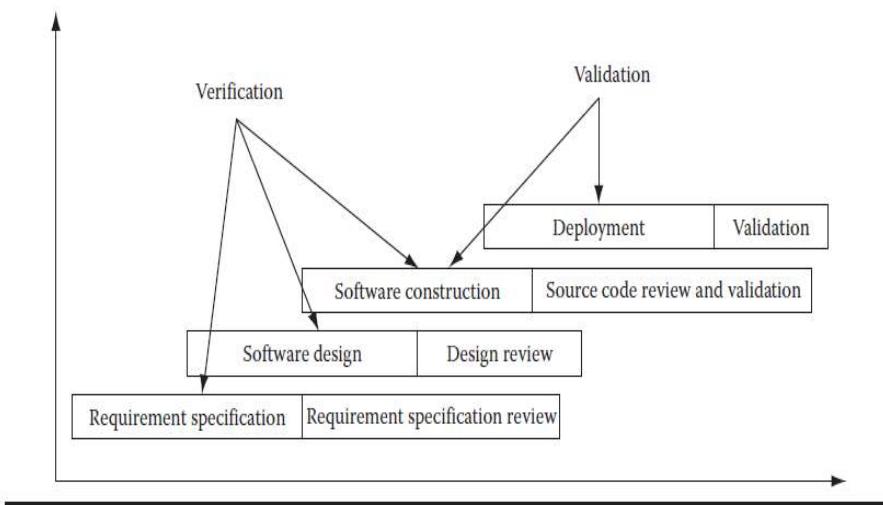


Figure 13.3 Software verification and validation.

6. Write short notes on Risk management in Software testing.

L2 CLO4

The test manager should also do plan for all known risks that could impact the test project. If proper risk mitigation planning is not done, and a mishap occurs, then the test project schedule could be jeopardized, costs could escalate, and/or quality could go down.

Some of the risks that can have severe, adverse impact on a test project include an unrealistic schedule, resource unavailability, skill unavailability, frequent requirement changes, etc. Requirement changes pose a serious threat to testing effort, because for each requirement change, the whole test plan gets changed. The test team has to revise its schedule for additional work as well as to assess impact of the change on the test cases they have to recreate.

For test professional resources, a good alternative resource planning is required. The test manager should, in consultation with human resource manager, keep a line of test professionals who may join in case one is needed on his project.

For scheduling problems, the test manager has to ensure in advance that schedules do not get affected. He has to keep a buffer in the schedule for any eventuality.

To keep a tab on the project budget, the test manager has to ensure that the schedule is not unrealistic and also has to load his test engineers appropriately. If some test engineers are not loaded adequately, then project costs may go higher. For this reason, if any test professionals do not have enough assignments on one project, they should be assigned work from other projects.

7. State the importance of effort estimation?

L2 CLO4

For making scheduling, resource planning, and budget for a test project, the test manager should make a good effort estimate. Effort estimate should include information such as project size, productivity, and test strategy. While project size and test strategy information comes after consultation with the customer, the productivity figure comes from experience and knowledge of the team members of the project team.

The wideband Delphi technique uses brainstorming sessions to arrive at effort estimate figures after discussing the project details with the project team. This is a good technique because the people who will be assigned the project work will know their own productivity levels and can figure out the size of their assigned project tasks from their own experience. Initial estimates from each team member are then discussed with other team members in an open environment. Each person has his own estimate. These estimates are then unanimously condensed into final estimate figures for each project task.

Effort estimation is one area where no test manager can have a good grasp, at the initial stages of the project. This is because not many details are clear about the project. As the project unfolds, after executing some of its related tasks, things become clearer. At that stage, any test manager can comfortably give an effort estimate for the remaining project tasks.

8. Explain about Test Automation.

L2 CLO4

Most testing tasks are done manually, as they are still difficult to automate. Wherever automation is possible, it can be evaluated. Care should also be taken not to do automation blindly. This is because the initial effort for automation is more than manual testing.

Testing tasks include requirements and design document review, test case scenario creation, test case creation, test case execution, test case management, and defect tracking.

Out of these tasks, test case execution and test case management are the only tasks for which good automation tools are available.

9. Describe about Defect Tracking.

L2 CLO4

Defect tracking is one of the most important activities in a test project [13]. During defect tracking it is ensured that defects are logged and get fixed. All defects and their fixing are tracked carefully (Figure 13.6).

Defect count per hour per day is a common way of measuring performance of a test team. If the testing is done for an in-house software product, traditionally, it used to not be a performance evaluation measurement. What really counted was the number of defects found in production when the software product was deployed and used by end users. But it is too late a performance measurement. What if many of the test team members left before the product was deployed? In fact this is a reality, given the high attrition rate (as much as 20% at many corporations) of software professionals. Once they are gone, there is no point in measuring the performance. Thus, a better measurement would allow for more immediate results. This is achieved by measuring the defect count per hour per day. Then there is the case of outsourced test projects. If the contract is only for testing up to deployment and not afterward, then measurement does not make sense after the contract has ended.

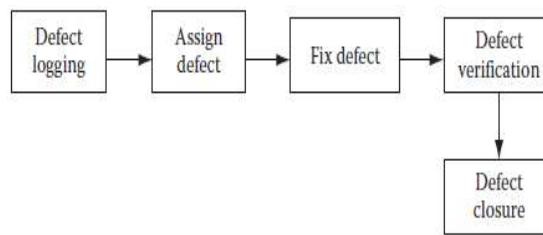


Figure 13.6 Defect life cycle.

A good defect tracking application should be deployed on a central server that is accessible to all test and development teams. Each defect should be logged in such a way that it could be understood by both development and testing teams.

10. Write short notes on Software Testing in Iterative Model. [REF-5. PAGE: 197]

L2 CLO4

In an iterative model, each iteration is a short cycle. So the amount of testing in each iteration is also small. Thus, unlike in waterfall model, software testing has a lesser role in the iterative development life cycle.

Generally, software defects tend to increase with the size of software products. Since in iteration mode the software product is small, there will be fewer defects in the product. Although in reality, as the software product grows in size over many

iterations, the number of defects per line of software code is bound to increase. In iterative development, regression testing is also a big issue. In each iteration, there will be a large number of regression test cases to run. As the product size increases with iterations, the set of regression test cases also increases. It becomes a liability after a while. Manually running all those regression tests takes a lot of time, which becomes a hindrance for the release schedule. In such cases, the best option is to go for automation of these regression test cases. Automated test cases take much less time (sometimes if the manual running of test cases was taking 5 days, after automation it took only 5 h) to run.

12 Marks:

- | | | |
|---|----------|------|
| 1. Discuss in detail Test Strategy and Planning. | L2 | CLO4 |
| 2. Explain a case study on Software testing. | L3 | CLO4 |
| 3. (i) List out the Problems with Traditional Development Model with neat diagram
(ii) Describe about Verification and Validation. | L2
L2 | CLO4 |
| 4. (i) Write short notes on Risk management in Software testing.
(ii) State the importance of effort estimation. | L2
L1 | CLO4 |
| 5. (i) Explain about Test point analysis and its components.
(ii) Elaborate in detail the Test Automation. | L2
L2 | CLO4 |
| 6. (i) Describe about Defect Tracking.
(ii) Write short notes on Software Testing in Iterative Model. | L2
L2 | CLO4 |
| 7. Elaborate in detail Test Project Monitoring and Control with neat sketch. | L2 | CLO4 |
| 8. Describe in detail the techniques used for testing software. | L2 | CLO4 |

UNIT-5

1) Which of these are incorrect for the product plan?

- A. No need to revise product plan frequently
- B. Product plan guides the launch of various product development activities
- C. Product plan can be revised for every sprint
- D. Product plan can be revised based in team suggestion

ANSWER: A

2) A program that is used in a real-world environment necessarily must change or become progressively less useful in that environment. State the law

- A. Law of increasing complexity
- B. Law of conservation of familiarity
- C. Law of continuing change
- D. Law of conservation of organizational stability

ANSWER: C

3) The aspects of a maintenance team that lead to high maintenance costs are &

- A. Quality, Staff turnover
- B. Staff turnover, Software Quality
- C. Domain expertise, Team
- D. Staff turnover, Domain expertise

ANSWER: D

4). method adapts by means of verifying that maintenance goals have been met; performance review to provide feedback to managers.

- A. Reuse
- B. Boehm's Model

C. Osborne

D. Iterative

ANSWER: C

5)Analyze and find out from these loss which is not the financial reason for software maintenance.

A. Revenue Loss

B. Software Loss

C. Opportunity Loss

D. Productivity Loss

ANSWER: B

6).....prepared by your team is up to date and in synch with the version of your software product, which you will implement at the customer site.

A. Product Manual

B. Project Manual

C. Hardware and Software Manual

D. User Manual

ANSWER: D

7)Judge which characteristic Results due to combination of other system factors, such as complexity, poor documentation and lack of inexperienced personnel symbolize?

A. Complex software

B. High maintenance cost

C. Obsolete hardware

D. Poorly documented

ANSWER: B

8)Identify among these which session is a descendent of the product roadmap.

- A. Release Maintenance
- B. Phased communication
- C. Release Planning
- D. Forward adjustment

ANSWER: C

9)Identify the key Components of Software Maintenance Framework.

- A. User requirements
- B. Opportunity
- C. Productivity
- D. Revenue

ANSWER: A

10)Identify the key Factor that does not affect Software Maintenance.

- A. Relationship of Software product and Environment
- B. Relationship of Software product and User
- C. Relationship of Software product and Software Maintenance team
- D. Relationship of Software product and product implementation

ANSWER: D

11)The operating environment in which a software product runs in operation includes the hardware and software platform as well as the interfaces for human and other machine interactions. If any of these change over time, it becomes difficult to run the software product. In such cases, what type of Maintenance would you suggest?

- A. Corrective Maintenance
- B. Adaptive Maintenance
- C. Preventive Maintenance

D. Perfective maintenance

ANSWER: B

12) If the entire team of the project management has agreed on all the functionalities being implemented in the product as well as the required quality, which of the following phase will be preferred by team.

- A. Decision of alpha, beta version
- B. Regular release
- C. Training to support staff
- D. Customer support strategy

ANSWER: B

13) Even after thorough reviews and testing, the software product contains many defects when it goes into production. The software vendor instructs his maintenance team to create a patch to rectify them. A patch can be created to rectify those defects. Analyze the case and provide the type of maintenance needed.

- A. Corrective Maintenance
- B. Adaptive Maintenance
- C. Perfective Maintenance
- D. Preventive Maintenance

ANSWER: A

14) Generally after a lapse of time, there are likely changes in business or operative environment, or there may be changes in hardware/software environment. Many of these changes can be perceived in advance and can be adopted. Which maintenance types is suitable for this case?

- A. Corrective Maintenance
- B. Adaptive Maintenance
- C. Perfective Maintenance
- D. Preventive Maintenance

ANSWER: D

15)The operating environment in which a software product runs in operation includes the hardware and software platform as well as the interfaces for human and other machine interactions. If any of these change over time, then which of the following type can be preferred to make the product to be a consistent one?

- A. Corrective Maintenance
- B. Adaptive Maintenance
- C. Perfective Maintenance
- D. Preventive Maintenance

ANSWER: B

16)A business workflow may have changed, a business transaction may have changed, or an altogether new business transaction was represented in the software product. For all these kinds of requirements, Identify the type of maintenance needed?

- A. Corrective Maintenance
- B. Adaptive Maintenance
- C. Perfective Maintenance
- D. Preventive Maintenance

ANSWER: C

17)You are the software project manager. Software maintenance activities reduce ROI currently. This in turn has impact in the overall project. At this situation, what maintenance model will you use?

- A. Quick fix model
- B. Boehm's model
- C. Osborne's model
- D. Iterative enhancement model

ANSWER: B

18) An organization has purchased ERP from its vendor. During the course of usage, some of the defects were identified like, not able to generate monthly reports, some quality attributes missing, some security issues, etc., Which maintenance model will you suggest for this case by analyzing the given scenario?

- A. Quick fix model
- B. Boehm's model
- C. Osborne's model
- D. Iterative enhancement model

ANSWER: A

18CSC206J SOFTWARE ENGINEERING AND PROJECT MANAGEMENT
UNIT-V
SOFTWARE REUSE

1 MARKS

S.NO	QUESTIONS	LEVEL	CLO	PG. NO
1.	_____ will provide a step-by-step guide for using the product under scenarios. a. User Training b. User Requirements c. User Maintenance d. User Validation	1	5	203
2.	If the software has some defects, then it will take a _____ to rectify it. a. Corrective Maintenance b. Adaptive Maintenance c. Preventive Maintenance d. Perfective maintenance	1	5	205
3.	_____ is necessary to do so that the software product becomes reusable. a. Corrective Maintenance b. Adaptive Maintenance c. Preventive Maintenance d. Perfective maintenance	1	5	205
4.	_____ is needed when there is a change in the business environment, and thereby users need additional/modified functionality in the software product to do their tasks. a. Corrective Maintenance b. Adaptive Maintenance c. Preventive Maintenance d. Perfective maintenance	1	5	205
5.	_____ on the software product can make sure that the product will be useful even after environmental changes occur. a. Corrective Maintenance b. Adaptive Maintenance c. Preventive Maintenance d. Perfective maintenance	1	5	205
6.	A _____ analysis can be done, to see if it is more profitable to conduct a Maintenance program on the software or keep using it as it is. a. Profit/Loss b. Test c. Maintenance d. Corrective	1	5	206
7.	In which model there is no planning involved in the whole process and is mostly an ad hoc Approach?	1	5	207

a.	Quick Fix Model				
b.	Boehm's Model				
c.	Osborne's Model				
d.	Iterative Enhancement Model				
8.	_____ Model is based on economic models and often involves calculating ROI, for any planned maintenance.	1	5	207	
	a. Quick Fix Model				
	b. Boehm's Model				
	c. Osborne's Model				
	d. Iterative Enhancement Model				
9.	A quality assurance plan should accompany the maintenance plan in which model?	1	5	206	
	a. Quick Fix Model				
	b. Boehm's Model				
	c. Osborne's Model				
	d. Iterative Enhancement Model				
10.	_____ model is based on the similar concept of iterative software development.	1	5	207	
	a. Quick Fix Model				
	b. Boehm's Model				
	c. Osborne's Model				
	d. Iterative Enhancement Model				
11.	_____ type of process is adopted for component-based software products.	1	5	207	
	a. Quick Fix Model				
	b. Boehm's Model				
	c. Osborne's Model				
	d. Reuse Oriented Model				
12.	_____ is also known as reuse engineering.	1	5	208	
	a. Reverse Engineering				
	b. Reengineering				
	c. Forward Engineering				
	d. Both a & b				
13.	_____ technique is most useful when nonexistent or sketchy documentation is available for the software product	1	5	208	
	a. Reverse Engineering				
	b. Reengineering				
	c. Forward Engineering				
	d. Both a & b				
14.	_____ the opposite of reverse engineering.	1	5	209	
	a. Reverse Engineering				
	b. Reengineering				
	c. Forward Engineering				
	d. Both a & b				
15.	_____ is also known as Renovation and Reclamation	1	5	209	

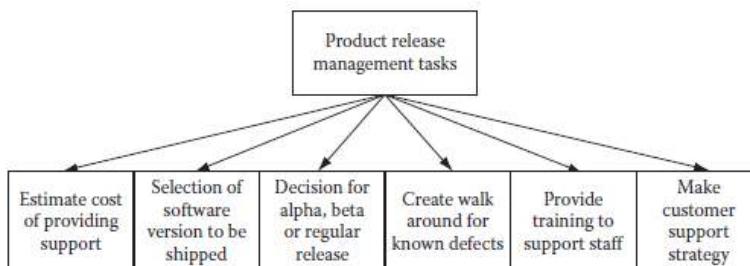
- a. Reverse Engineering
- b. Reengineering
- c. **Forward Engineering**
- d. Both a & b

4 MARKS:

1. Explain about Product Release Management.

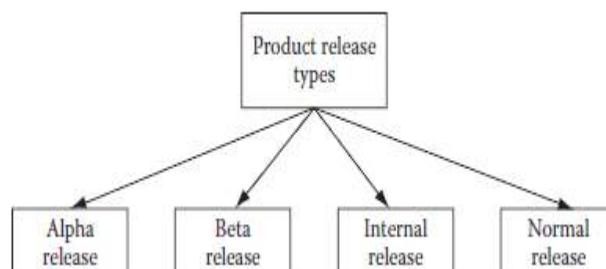
L2 CLO5

Project teams working for software product vendors struggle to keep pace with release of the software product. There is pressure from the market to launch new versions by certain dates. New features are to be added, porting the product to new platforms, old features are to be enhanced, existing bugs are to be removed, and yet it has to meet the deadline. It is a constant struggle that calls for good product release strategies. Depending on the situation, the project manager may need to convince the management to cut short some of the product features to meet the deadline as well as meet quality standards.



2. Explain the idea list for software product release.

L2 CLO5



A product release is the process of launching a new product for a specific market or user base. In software development, a product release is sometimes done with a beta version so that core developers/users can assist with debugging and feedback prior to the release of the actual software.

3. Analyze the reasons for maintenance in software products.

L2 CLO5

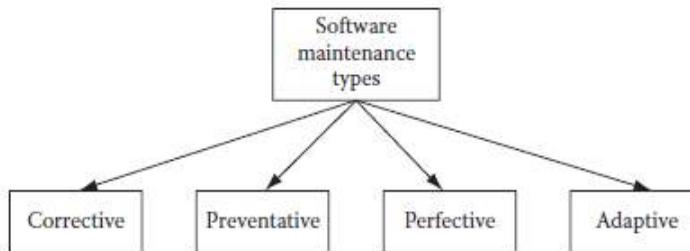
1. Technology obsolescence : The software platform (operating system, medium of user, interface) or the hardware platform on which the software product runs gets obsolete.

2. Software defects: There are major software defects in the product and it is difficult to operate. For this reason, a software patch may be needed to be applied so that these defects are removed.

3. Change in user requirements: The business organization that was using the software product has seen a change in business transactions or business workflows that are not supported by the software product.

4. List out the software maintenance types.

L2 CLO5



Corrective: after thorough reviews and testing, the software product contains many defects when it goes into production. These defects are uncovered as users start using the application. They are logged with the support staff and after a sizable number of errors are detected, the software vendor instructs his maintenance team to create a patch to rectify them.

Adaptive: The operating environment in which a software product runs in operation includes the hardware and software platform as well as the interfaces for human and other machine interactions. If any of these change over time, it becomes difficult to run the software product. In such cases, it becomes necessary to do adaptive maintenance so that the software product becomes reusable.

Perfective: This kind of maintenance is needed when there is a change in the business environment, and thereby users need additional/modified functionality in the software product to do their tasks. A business workflow may have changed, a business transaction may have changed, or an altogether new business transaction was represented in the software product.

Preventive: Generally after a lapse of time, there are likely changes in business or operative environment, or there may be changes in hardware/software environment. These changes are bound to occur and they affect the way the software product operates. Many of these changes can be perceived in advance.

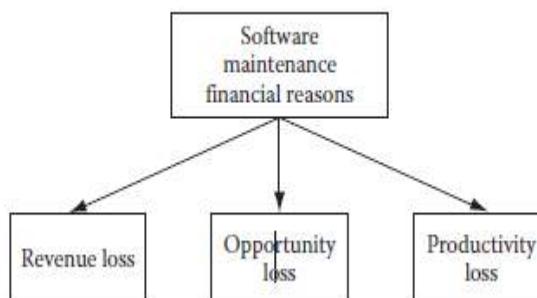
5. Classify the financial reasons for software maintenance.

L2 CLO5

1. Loss in business revenue: It may happen that business transactions are faulty and thus the business may lose revenue.

2. Opportunity loss: Sometimes there could be some business opportunity in the marketplace, but due to some software problems it could not be availed.

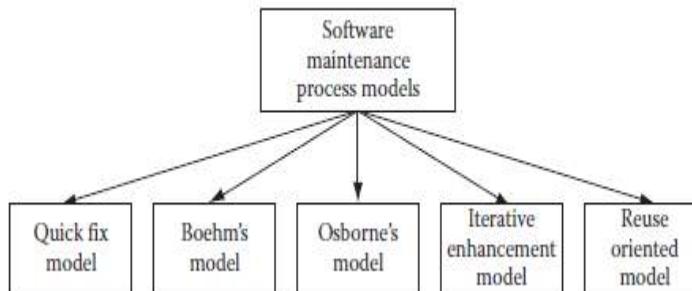
3. Productivity loss: If the software product becomes difficult to operate due to many walk arounds or lengthy processing then productivity will become lower for business personnel.



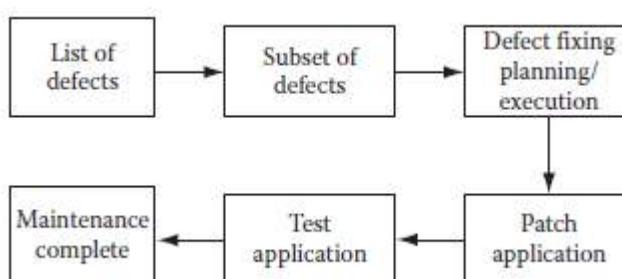
6. Sketch the software maintenance model.

L1 CLO5

Software maintenance process models have been defined and Some of the popular ones include the quick fix model, Boehm's model, Osborne's model, iterative enhancement model, and reuse oriented model



7. Demonstrate the Quick Fix Model. L3 CLO5
- This is the simplest of maintenance models; whenever any defects with the software products are found they are immediately fixed. There is no planning involved in the whole process and it is mostly an ad hoc approach.
8. Examine Reuse Oriented model. L2 CLO5
- This type of process is adopted for component-based software products. For fixing many defects, existing components are analyzed and then the appropriate changes are made.
9. Define Iterative Enhancement Model. L1 CLO5
- This model is based on the similar concept of iterative software development. All software defects and change requests are logged and then a small set from this list is taken for making fixes. This set is prepared based on the priority of changes required. High priority fixes are done before low priority fixes.
10. Compare and contrast Boehm's Model Osborne's Model L2 CLO5
- Boehm's model:** Boehm's model is based on economic models and often involves calculating ROI, for any planned maintenance. If ROI turns out to be good, then it is carried out or else it is dropped.
- Osborne's model:** Osborne realized that difficulties in carrying out maintenance work are due to gaps in communication. He proposed four steps to prevent this situation. He stated that a maintenance plan should include all change requests in the form of maintenance requirements. A quality assurance plan should accompany the maintenance plan. Metrics should be developed to measure and assess quality of work carried out during maintenance.
11. Design Software Maintenance Life Cycle. L2 CLO5
- Software maintenance life cycle, testing is a crucial phase. This phase also consumes a lot of time and effort. But the value addition in all this effort and time spent helps in reducing defects, which in the long run is a much cheaper alternative compared to no testing/cursory testing and later spending money in providing support.



12. Write about software maintenance engineering techniques. L2 CLO5
- Maintenance of software products sometimes becomes a tough proposition. There is no proper documentation that can be used for understanding how the product is designed and constructed. Sometimes there is no documentation at all. Even if documentation is there, it is not up to date. This out-of-date documentation is not of much use for any maintenance work. Sometimes even if the documentation is up to

date, the maintenance work is difficult due to dirty design or construction work. All these situations call for some specific techniques for maintenance work depending on the situation. Some of the common maintenance techniques include reengineering, reverse engineering, and forward engineering.

13. Differentiate Reverse Engineering and Forward Engineering.

L2 CLO5

Reverse engineering technique is most useful when nonexistent or sketchy documentation is available for the software product. Due to unavailability of documentation, there is no information as to what the design is and how the product is constructed. In such a situation, it is almost impossible to do any modification in the source code for any maintenance work.

In such cases, the reverse engineering technique is adopted. Using this technique, similar components or product parts are constructed as compared to existing product components/parts. This way the software product functionality is changed as the new constructed parts will have the desired functionality.

Forward engineering is just the opposite of reverse engineering. In this situation, we have ample documentation about the existing product. Due to new customer needs, the existing product needs to be extended so that the new needs can be fulfilled. All new extended development is based on the existing design and construction methods and will be made for the same hardware/software platform.

12 Marks:

1. Explain in detail about Project Release Management. L2 CLO5
2. List out the software maintenance types and describe in detail. L2 CLO5
3. Distinguish Briefly about Software maintenance process model. L2 CLO5
4. Illustrate about Software maintenance techniques in detail L2 CLO5
5. Discuss in detail about Software maintenance Life cycle. L2 CLO5
6. Write down the steps for Sofware product implementation. L2 CLO5

18CSC206J – SOFTWARE ENGINEERING AND PROJECT MANAGEMENT

SAMPLE QUESTIONS - MODEL EXAM – PATTERN – May 2021

Part A - MCSA (Multiple Choice Single Answer)

Part B – MCSA / MCMA (Multiple Choice Multiple Answers)

Unit 4 & Unit 5

PART A

1. Match the following:

List I	List II
(P) Condition coverage	(i) Black-box testing
(Q) Equivalence class partitioning	(ii) System testing
(R) Volume testing	(iii) White-box testing
(S) Alpha testing	(iv) Performance testing

a) P-ii, Q- iii, R-i, S-iv

b) P-iii, Q- i, R-iv, S-ii

c) P-iii, Q- iv, R-ii, S-i

d) P-iii, Q- i, R-ii, S-iv

Ans: b) P-iii, Q- i, R-iv, S-ii

2. The cyclomatic complexity of the flow graph of a program provides

- a) **an upper bound for the number of tests that must be conducted to ensure that all statements have been executed at least once**
- b) a lower bound for the number of tests that must be conducted to ensure that all statements have been executed at most once

- c) an upper bound for the number of tests that must be conducted to ensure that all statements have been executed at most once
- d) a lower bound for the number of tests that must be conducted to ensure that all statements have been executed at least once

Ans: a) an upper bound for the number of tests that must be conducted to ensure that all statements have been executed at least once

3. Testing begins at _____ level and works “outward” toward the integration of the entire computer based system.

- a) **Component**
- b) Micro
- c) Unit
- d) System

Ans: a) Component

4. _____ refers to the set of tasks that ensure that software correctly implements a specific function.

- a) Software debugging
- b) Error correction
- c) Verification
- d) Validation

Ans: c) Verification

5. Which testing is focuses on the design and construction of the software architecture.

- a) Validation testing
- b) Unit testing
- c) System testing
- d) **Integration testing**

Ans: d) Integration testing

6. _____ serve to replace modules that are subordinate the component to be tested.

- a) Driver
- b) **Stub**
- c) Module

- d) Interface

Ans: Stub

7. Cyclomatic Complexity cannot be applied in _____.

- a) Re-engineering
- b) Risk Management
- c) Test Planning
- d) Reverse engineering**

Ans: d) Reverse engineering

8. Which kind of testing helps to ensure that changes do not introduce unintended behaviour or additional errors.

- a) Regression Testing**
- b) Integration Testing
- c) System Testing
- d) Validation Testing

Ans: a) Regression Testing

9. Test cases are designed during

- a) Test recording
- b) Test configuration
- c) Test planning
- d) Test specification**

Ans: d) Test specification

10. _____ is conducted at one or more end-user sites.

- a) Alpha test
- b) Beta test**
- c) Validation test
- d) Unit test

Ans: b) Beta test

11. Executing a system in a manner that demands resources in abnormal quantity, frequency or volume is

- a) Security testing
- b) Stress testing**
- c) Recovery testing
- d) Performance testing

Ans: b) Stress testing

12. Which debugging approach can be used successfully in small programs?

- a) Brute force
- b) Cause elimination
- c) Backtracking**
- d) Automated debugging

Ans: c) Backtracking

13. Which of the following is not the attributes of a good test?

- a) high probability of finding an error
- b) redundant**
- c) best of breed
- d) neither too simple nor too complex

Ans: b) redundant

14. White box testing sometimes called

- a) Basis path testing
- b) Flow graph testing
- c) Glass-box testing**
- d) Behavioral resting

Ans: c) Glass-box testing

15. Given a flow graph with 10 nodes, 13 edges and one connected components, the number of regions and the number of predicate(decision) nodes in the flow graph will be

- a) 5,4
- b) 4,5

c) 3,1

d) 13,8

Ans: a) 5,4

16. _____ is a test case design technique that complements equivalence partitioning.

a) **Boundary value analysis**

b) Orthogonal array testing

c) Graph based testing

d) Model based testing

Ans: a) Boundary value analysis

17. Which testing is an integration testing approach that is commonly used when “shrink-wrapped” software products are being developed?

a) Regression Testing

b) Integration testing

c) **Smoke testing**

d) Validation testing

Ans: c) Smoke testing

18. _____ serves as a blue print to conduct software testing activities

a) **Test Plan**

b) Test Scope

c) Test Strategy

d) Test Estimation

Ans: a) Test Plan

19. Error guessing is a

a) *Test verification techniques*

b) *Test data management techniques*

c) *Test control management techniques*

d) *Test execution techniques*

Ans: b) Test data management techniques

20. _____ is an environment that contains all the hardware and software needed to test a software component or a software system.

- a) Test case
- b) Test item
- c) **Test bed**
- d) Test criteria

Ans: c) Test bed

21. In which phase of the Software Development Life Cycle (SDLC) users or developers may discover errors or problems that cause the entire process to start over again ?

- A. System Analysis
- B. System Design
- C. System Implementation
- D. System Maintenance

Answer D. System Maintenance

22. Changes made to the system to reduce the future system failure chances is called

- A. Preventive Maintenance
- B. Adaptive Maintenance
- C. Corrective Maintenance
- D. Perfective Maintenance

Answer A. Preventive Maintenance

23. Which of the following is not part of software reengineering process model ?

- E. Forward Engineering
- F. Inventory Analysis
- G. Prototyping
- H. Reverse Engineering

Answer C. Prototyping

24. Re-structuring of software results in

- A. Higher Quality Programs
- B. Increased Maintenance Cost

- C. Difficulty in software testing
- D. Increase in ROI

Answer A. Higher Quality Programs

25. For minor changes done in source code ----- is released but for major changes a new ----- will be released

- A. Patch, Entity
- B. Entity, Version
- C. Version, Patch
- D. Patch, Version

Answer : D : Patch, Version

Part B - MCMA (Multiple Choice Multiple Answers Questions)

1. Which of the following is/are white box technique(s) ?

- A) Statement Coverage
- B) Path testing
- C) State transition testing
- D) Data flow testing

Answer : A, B, D

2. Which of the following are components of a test plan?

- A) Features to be tested
- B) Incident reports
- C) Risks
- D) Schedule

Answer : A, C, D

3. Which of the following is true about test coverage criteria?

- A) A measure of test coverage criteria is the percentage of user requirements that are not covered
- B) Test coverage criteria can be measured in terms of items exercised

- by a test suite
- C) Test coverage criteria are often used when specifying test completion criteria
 - D) A measure of test coverage criteria is the percentage of faults found

Answer : B, C

- 4. Increasing the quality of the software, by better development methods, will affect the time needed for testing by
 - A) Reducing the testing time
 - B) Reducing the number of defects
 - C) Reducing the test coverage
 - D) Eliminating Unit Testing

Answer : A, B

- 5. Which one of the following statement(s) about system testing is/are true?
- A) System tests are often performed by independent teams
 - B) Functional testing is used more than structural testing
 - C) Faults found during system tests can be costly to fix
 - D) End-users should be involved in system tests

Answer : A, B, C

- 6. Unit testing standard includes :
 - A. Syntax testing
 - B. Equivalence partitioning
 - C. Stress testing
 - D. Decision coverage

Answer : A, B, D

- 7. Which of the following statement(s) is/are TRUE ?
 - A. In a system, two different failures may have different severities.
 - B. A system is necessarily more reliable after debugging for the removal of a fault.
 - C. Every fault will affect the reliability of a system
 - D. Undetected errors may lead to faults and eventually to incorrect behaviour

Answer : A, D

8. Which of the following is/are not form(s) of functional testing ?

- A. Boundary value analysis
- B. Smoke testing
- C. Performance testing
- D. Security testing

Answer : C, D

9. A configuration management system normally provides:

- A. Linkage of customer requirements to version numbers
- B. Facility to compare test results with expected results
- C. The precise differences in versions of software component source code
- D. Restricted access to the source code library

Answer : A, C, D

10. Incremental testing approaches include

- A. Bottom up
- B. Top down
- C. Big-bang
- D. Functional incrimination

Answer : A, B, D

11. Which of the following is/are maintenance process model(s)?

- A. Quick Fix Model
- B. Function Point Analysis
- C. Osborne's Model
- D. COCOMO

Answer : A, C

12. The reverse engineering maintenance technique is followed when

- A. There is no information as to how the product is constructed
- B. it is almost impossible to do any modification in the source code
- C. We have ample documentation about the existing product
- D. The existing product needs to be extended to fulfill new customer needs

Answer : A, B

13. Which of the following is NOT true about Corrective Maintenance?

- A. It includes modifications and updates done in order to correct or fix problems, which are either discovered by user or concluded by user error reports.
- B. It includes modifications and updates applied to keep the software product up-to date and tuned to the ever-changing world of technology and business environment.
- C. It includes modifications and updates done in order to keep the software usable over long period of time.
- D. It includes modifications and updates to prevent future problems of the software.

Answer : B, C, D

14. Which Software-end factors affect maintenance Cost?

- A. Structure of Software Program
- B. Programming Language
- C. Dependence on external environment
- D. Number of test engineers deployed

Answer : A,B,C

15. Software Maintenance must be performed in order to:

- A. Catch defects early in the software development cycle
- B. Correct faults encountered by end users
- C. Implement enhancements requested by customers
- D. Migrate legacy software to suit latest technology trends

Answer : B,C,D



Interface, window, pull-down menus in library etc.

... AI software uses non-numerical algorithm to solve complex problem. Robotics, expert system, pattern recognition games playing

- Delivery to the customer for evaluation

earlier stages of the life-cycle.

The model implies that once the product is inferior, everything else is maintenance.

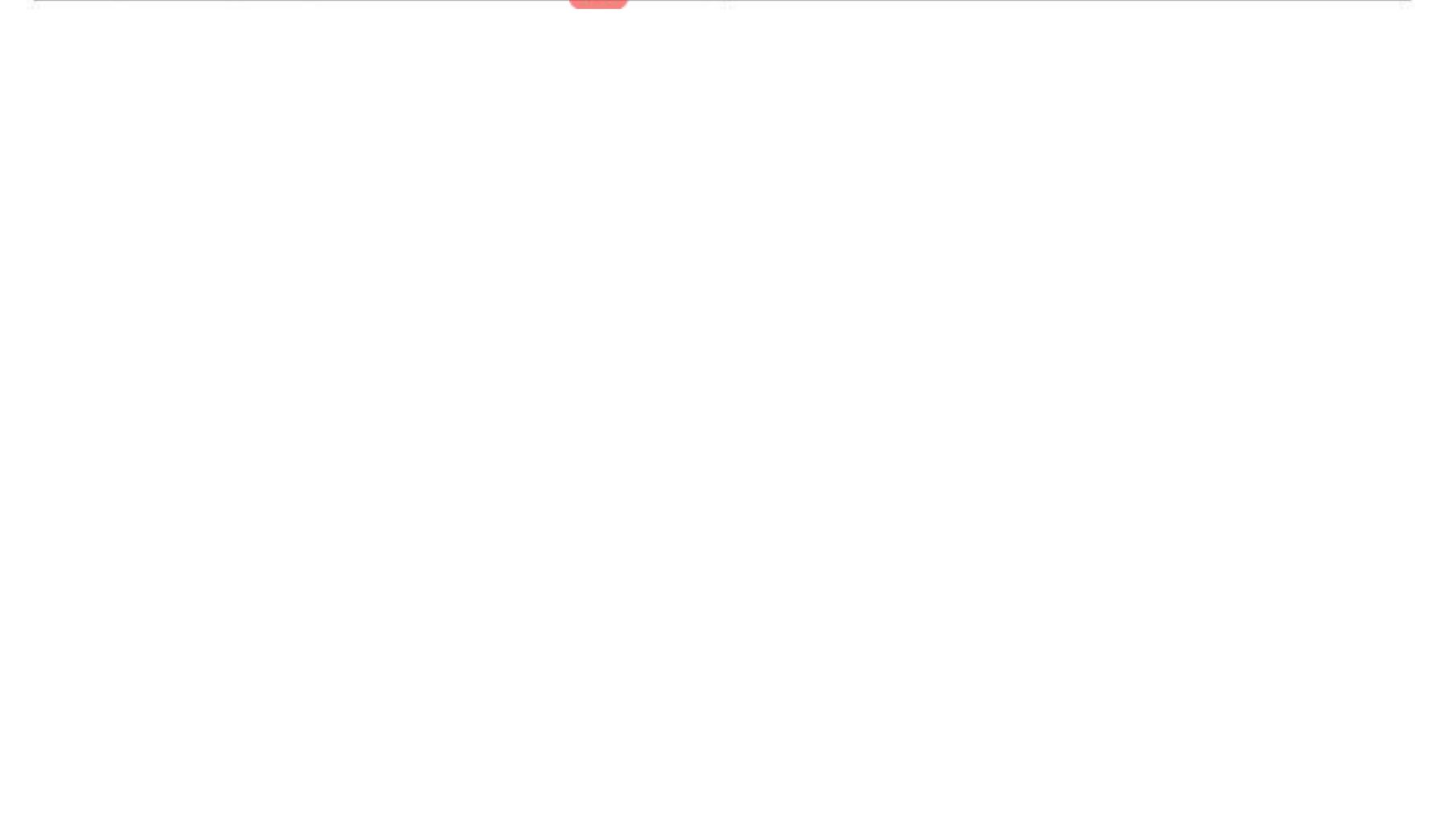
getting change fully.

Makes heavy use of reusable software components with an extremely short development cycle

built when requirement are uncertain(Fuzzy).

prototype to satisfy the needs of the customer.























Required development schedule

Required development schedule









- **Loss:** the risk becomes a reality and unwanted consequences or losses occur

- losing budgetary or personnel commitment

plan?"

- Staff size and experience

Management Plan

- Draw a horizontal cutoff line in the table that indicates the risks that

- C = Total cost of developing 18 components is \$25,000

-- -- -- -- -- -- -- -- -- --

NOTE: Stakeholders - A person or group of people who own a share in a business.

the product's functionality.

current during the life-cycle of the project.

4. At this point, subproject managers are given ~~the~~ responsibility for decomposing each of the WBS elements 129



LEARN · LEAP · LEAD

& PROJECT MANAGEMENT



Edit with WPS Office

Software Design

Compiled by IT Department,SRMIST, KTR



Disclaimer:

The lecture notes have been prepared by referring to many books and notes prepared by the teachers. This document does not claim any originality and cannot be used as a substitute for prescribed textbooks.

technology project.

CLR-2:Preparing the project plane based on the scope and calculating the project effort based on resources.



Edit with WPS Office

Outcomes (CLO):

*Functional Oriented
for Software Design.*

*Object Oriented
Approach for
Software Design.*



Edit with WPS Office

- Component Level Design
- User Interface Design
- Pattern Oriented Design
- Web Application Design
- Design Reuse
- Concurrent Engineering in Software Design
- Design Life-Cycle Management

first and later its parts are designed.

- In bottom up approach, the software product parts are designed first and later they are joined to create design of the entire product.
- With each change request there will be a different version of the software design.
- Maintaining these design versions through the development process is very important.
- It is necessary to make sure that the right design is used for software



Edit with WPS Office

- Refactoring is a design technique which is used in iterative models.
 - When the design become bulky after many iterations of development, it starts getting crumbling against new features being added to the product.
 - In such cases, the design is changed so that new factors of the features being added are incorporated.
 - Software design development can be likened to designing a physical product.
 - Suppose a new car model is to be developed.
-



Edit with WPS Office

the body is one of the prime considerations.

- Similarly, an aerodynamic body helps in keeping the car from rolling over during accidents and thus it is a prime safety factor that the car body should be aerodynamic.
- During design, one consideration is also made that though each component is developed separately, after assembly the components should work with each other without any problems.
- That means assembling does not create any problems in the product itself.
- Similar considerations are also done when software products are designed



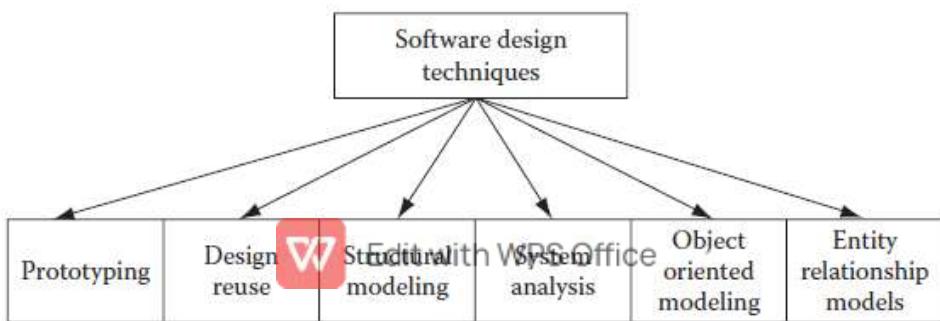
Edit with WPS Office

support the software system (Figure below-Characteristics of a good software design).

- In these days, most software systems are built incrementally.
 - In the beginning, a software system may consist of only a few features.
 - The feature set is expanded in future releases as and when it becomes necessary to include them in the system.
 - If proper structure is not provided from the very beginning, the addition of these new features will make the system unstable.
-

- Some of the design techniques that help make good software design include open architecture, modularity and scalability.
- The current trend of service-oriented architecture (SOA) has also helped tremendously in changing the design concepts.
- SOA is built on Web services and loose coupling of software components.
- The asynchronous messaging method of SOA is a vital aspect for

- Software design on any project may consist of many work products, which together can be termed the software design for the software product that will be built during the software project.
- Some examples of software design techniques include:



specific design often termed as low level design.

- The platform-specific design or low level design will be in terms of a good database model and a good application model (Figure above - Software design techniques).
- Over the years, many software design techniques have evolved with the evolution of different programming paradigms.
- Starting with the early procedural programming paradigms, programming has evolved into present day "service-oriented architecture".
- Software design has kept the pace with these evolving paradigms, and thus

- A miscommunication or misunderstanding between the customer and the project team gets cleared once the difference of opinion are sorted out early on during the prototype demonstration sessions.
- This greatly helps in reducing the risks of not meeting customer expectations.
- In any case, customers do not care about internal workings of the application.
- They are always concerned about what the application screens look like



Edit with WPS Office

be depicted in prototypes, as program logic is mostly not visible and cannot be developed in prototypes.

- Starting with the early procedural programming paradigms, programming has evolved into present day "service-oriented architecture".



Edit with WPS Office

these components, then it is possible to use these pieces of information in many components by reusing them.

- It will reduce effort in designing the product.
- This method of design reuse is known as internal design reuse.
- A more potent design reuse is becoming available after the advent of the open source paradigm and SOA.
- In the case of open source, the design reuse is in fact a case of copying existing design and then using it exactly as it is or modifying it to suit the needs

- Using this information, the design is done for the application.
- There is an assumption that as if the application/component provided as a service is available and the application uses this application/component.
- SOA is indeed leading to a reuse model that is going to transform the world of computing and the lives in years to come.



Edit with WPS Office

- For ease of working, maintenance, and breaking development tasks to allocate to group of developers, it is essential that an application is broken down into manageable parts.
- Breaking into parts for an application can best be done using a structural analysis.
- From requirement specifications, a feature set is made to decide what features will be in the application.
- This feature set is analyzed and broken down into smaller sets of features,

- These child classes inherit all the properties of their parent class, and they can have some more properties of their own in addition.
- So if we have a group of similar objects with somewhat different properties, then we can implement classes in such a way that a base parent class has child classes with different properties.
- This concept aligns very much to the real-world scenarios.
- Object-oriented design takes input from use cases, activity diagrams, user interfaces etc.

objects and events.

- The system analysis will be concerned with user activities, what objects on the web site act with user activities, how these objects interact with the underlying software system of the bank, and how connections are made between the user and the website and between the website and the bank system.
- System analysis will analyze all these things.
- Based on the analysis, a system model can be made that will be used in

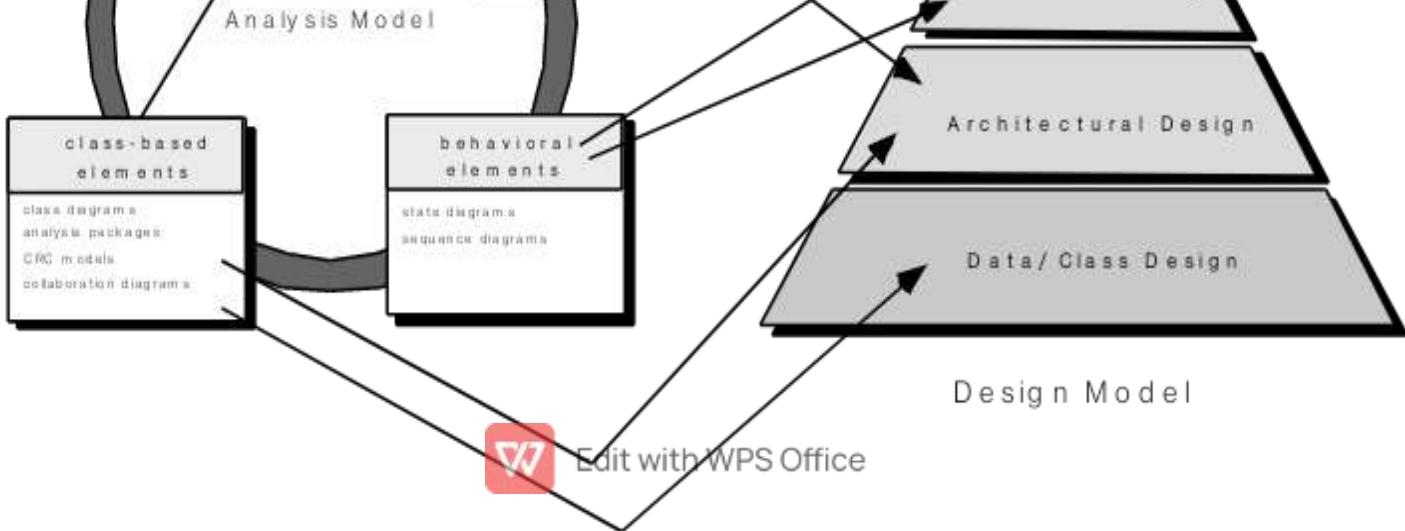


Edit with WPS Office

- This kind of representation helps to make a clean database design.



Edit with WPS Office



- Patterns and “styles”.
- Interface Elements
 - User Interface (UI)
 - External interfaces to other systems, devices, networks or other producers or consumers of information.
 - Internal interfaces between various design components
- Component Elements
- Deployment Elements



Edit with WPS Office



processing that manipulates the data and interact via the invocation of methods.

Extra-functional properties

- The architectural design description should address how the design architecture achieves requirements for performance, capacity, reliability, security, adaptability, and other system characteristics.

Families of related systems

- The architectural design  Edit with WPS Office should draw upon repeatable patterns that are commonly encountered in the design of families of similar systems

Architecture - Significance

- Representations of software architecture are an enabler for communication between all parties (stakeholders) interested in the development of a computer-based system.
- The architecture highlights early design decisions that will have a profound impact on all software engineering work that follows and, as important, on the ultimate success of the system as an operational entity.

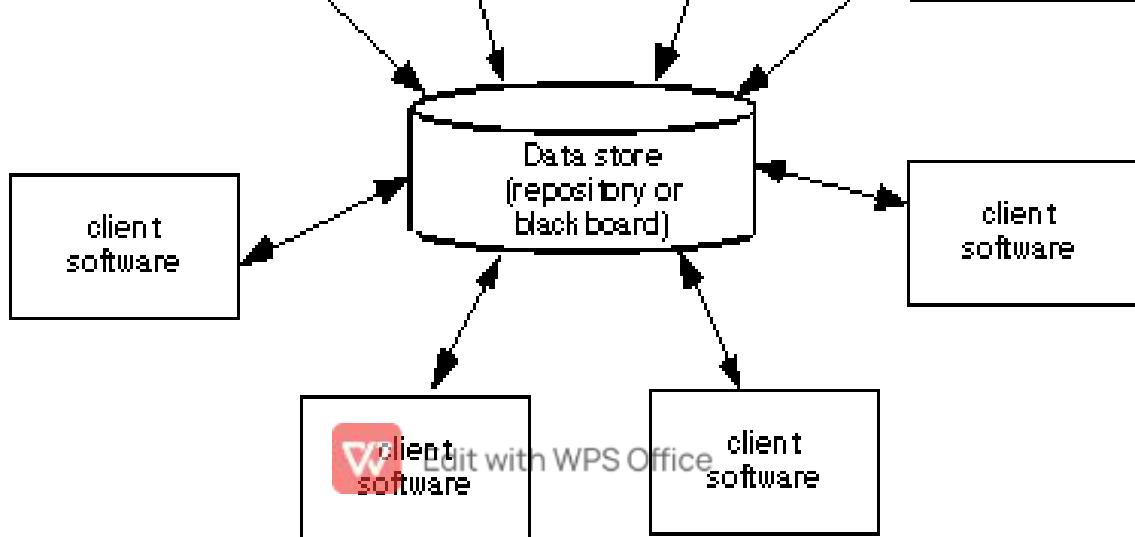
- The IEEE Standard defines an architectural description (AD) as a “a collection of products to document an architecture.”
 - The description itself is represented using multiple views, where each view is “a representation of a whole system from the perspective of a related set of [stakeholder] concerns.”



Edit with WPS Office

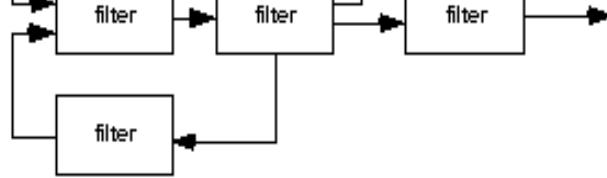
Architectural Styles

- Each style describes a system category that encompasses:
 - (1) a set of components (e.g., a database, computational modules) that perform a function required by a system,
 - (2) a set of connectors that enable “communication, coordination and cooperation” among components,
 - (3) constraints that define how components can be integrated to form the system, and





Edit with WPS Office



(a) pipes and filters

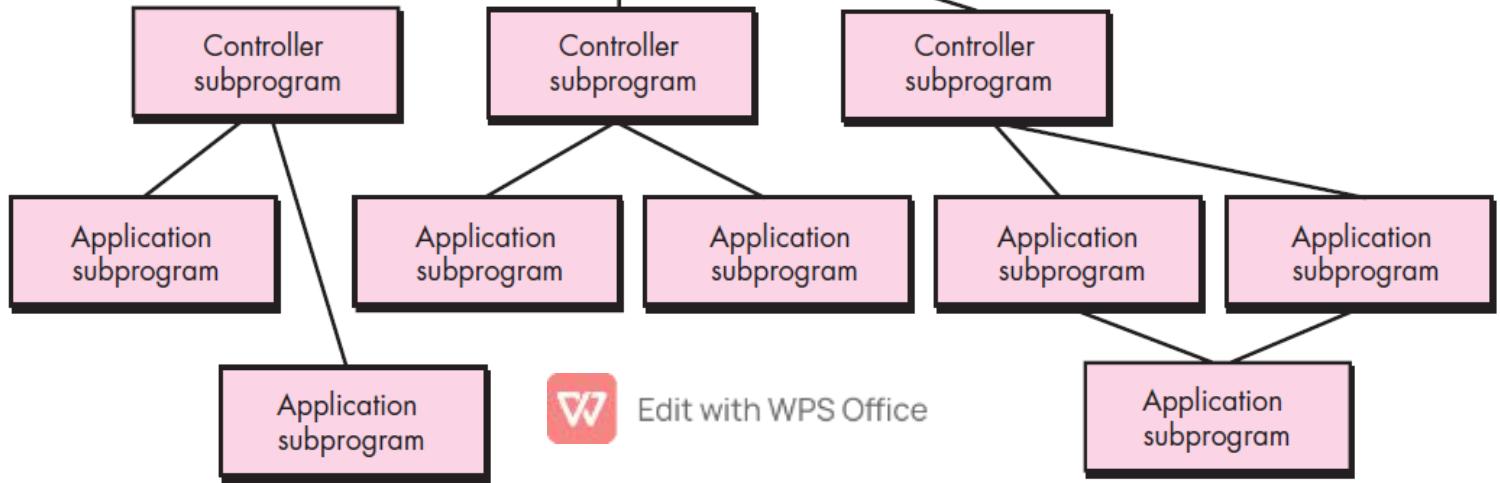


EditWith WPS Office

sequential components (filters) to transform it.



Edit with WPS Office



Edit with WPS Office

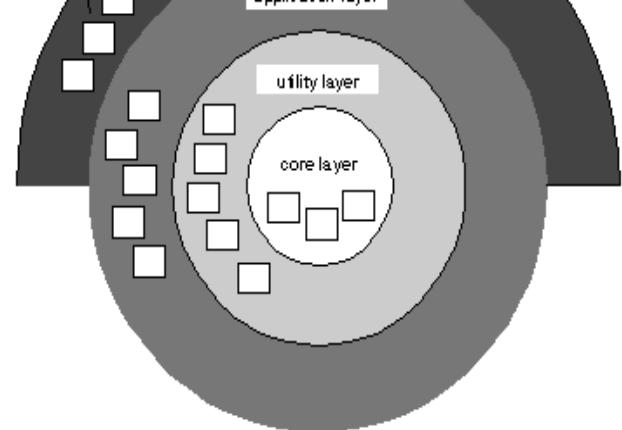
must be applied to manipulate the data.

- Communication and coordination between components are accomplished via message passing.



Edit with WPS Office

- These architectural styles are only a small subset of those available.
- Once requirements engineering uncover the characteristics and constraints of the system to be built, the architectural style and/or combination of patterns that best fits those characteristics and constraints can be chosen.



Edit with WPS Office

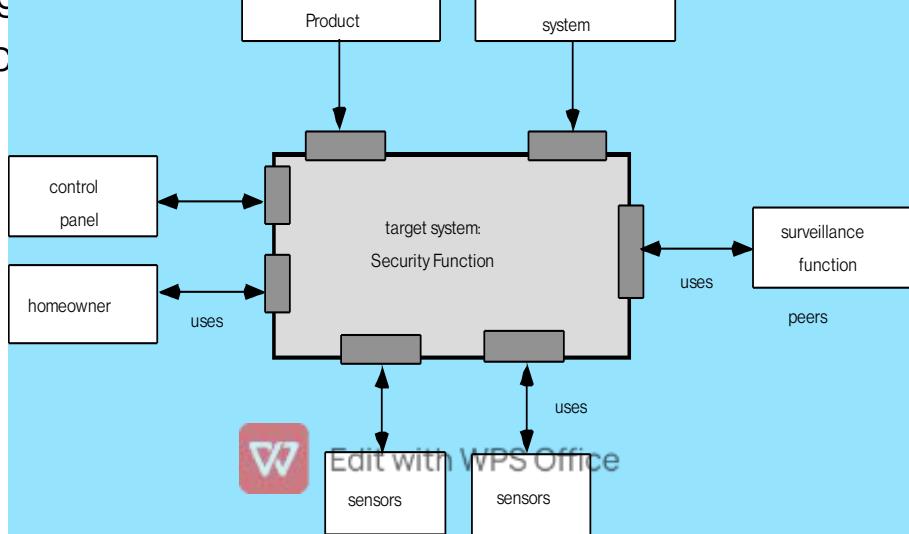
retrieval capability of a DBMS to the application architecture.

- An application level persistence pattern that builds persistence features into the application architecture.
- Distribution – the manner in which systems or components within systems communicate with one another in a distributed environment
 - A broker acts as a ‘middle-man’ between the client component and a server component.



Edit with WPS Office

The below diagram illustrates the architecture of the SafeHome security system.



Node – represents a cohesive collection of input and output elements of the home security function.

- Detector – An abstraction that encompasses all sensing equipment that feeds information into the target system.
- The diagram illustrates the UML



Edit with [WPS Office](#)

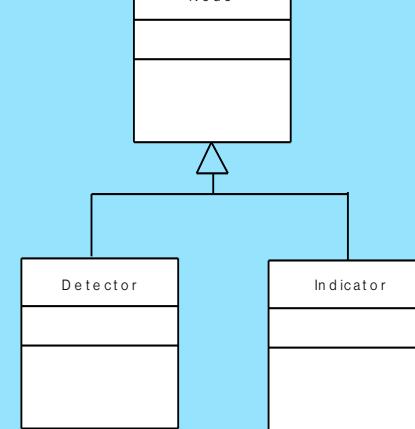


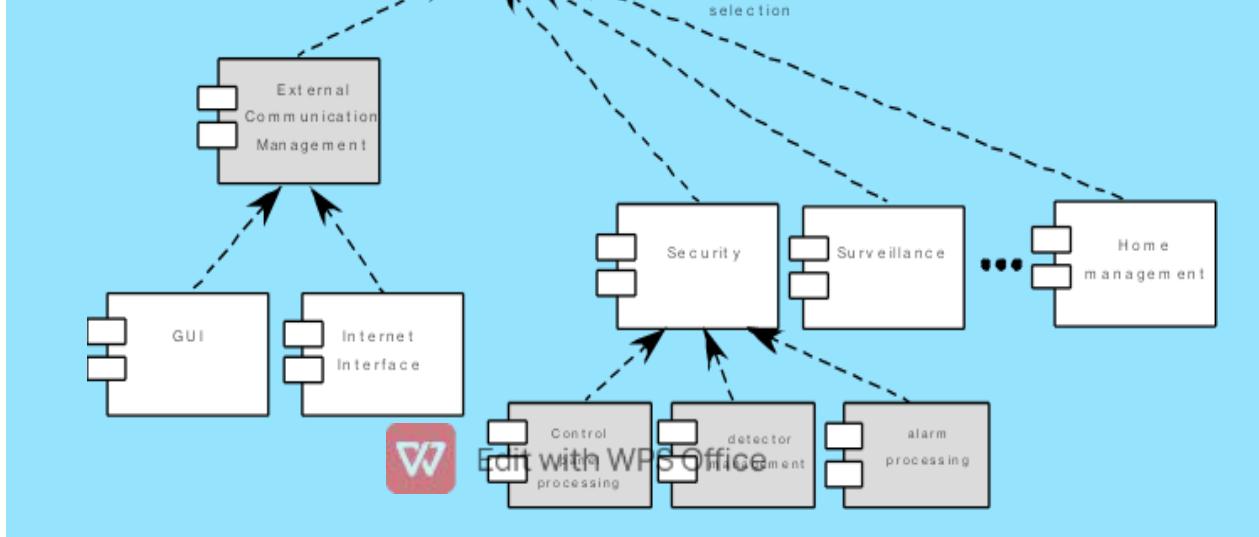
Figure 10.7 UML relationships for SafeHome security function archetypes
(adapted from [BOS00])

abstractions that must be further refined as architectural design proceeds.

- For example, Detector might be refined into a class hierarchy of sensors.



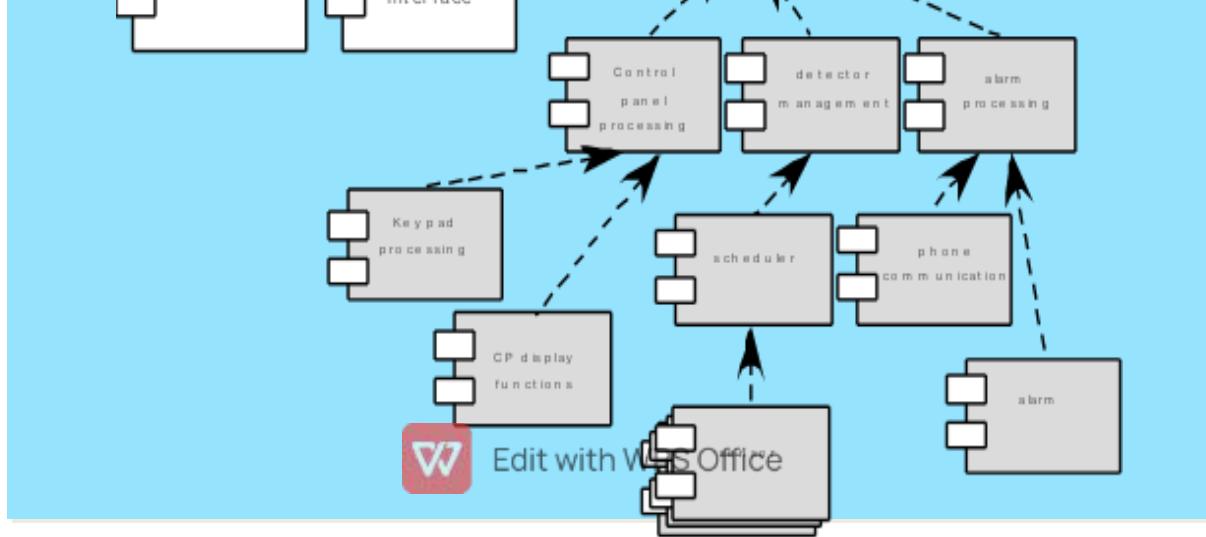
Edit with WPS Office



components but have no business connection to the application domain.

- In regards to the SafeHome home security function example, the set of top-level components might be defined to address the following functionality:
 1. External communication management – coordinates communication of the security function with external entities such as other Internet-based systems and external alarm notification
 2. Control panel processing – manages all control panel functionality.

- To accomplish this, an actual instantiation of the architecture is developed.
- The figure below illustrates an instantiation of the SafeHome architecture for the security system.
- Components shown in figure above are elaborated to show additional details.
- For example, the detector management component interacts with a scheduler infrastructure component that implements polling of each sensor object used by the security system.



4. Evaluate quality attributes by considering each attribute in isolation.
5. Identify the sensitivity of quality attributes to various architectural attributes for a specific architectural style.
6. Critique candidate architectures (developed in step 3) using the sensitivity analysis conducted in step 5.

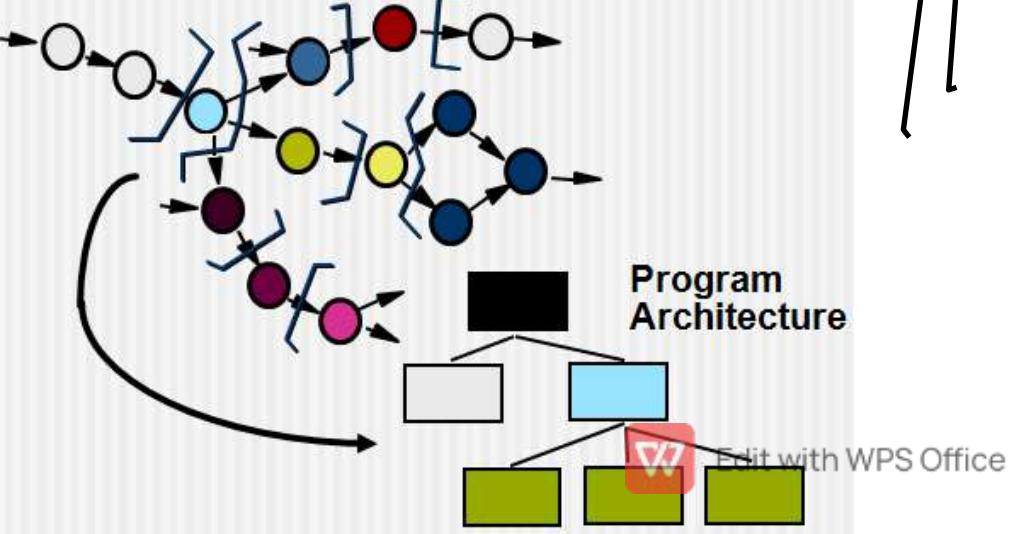


Edit with WPS Office

- Constrained dependencies represent constraints on the relative flow of control among a set of activities.

ADL

- Architectural description language (ADL) provides a semantics and syntax for describing a software architecture
- Provide the designer with the ability to:
 - Decompose architectural components

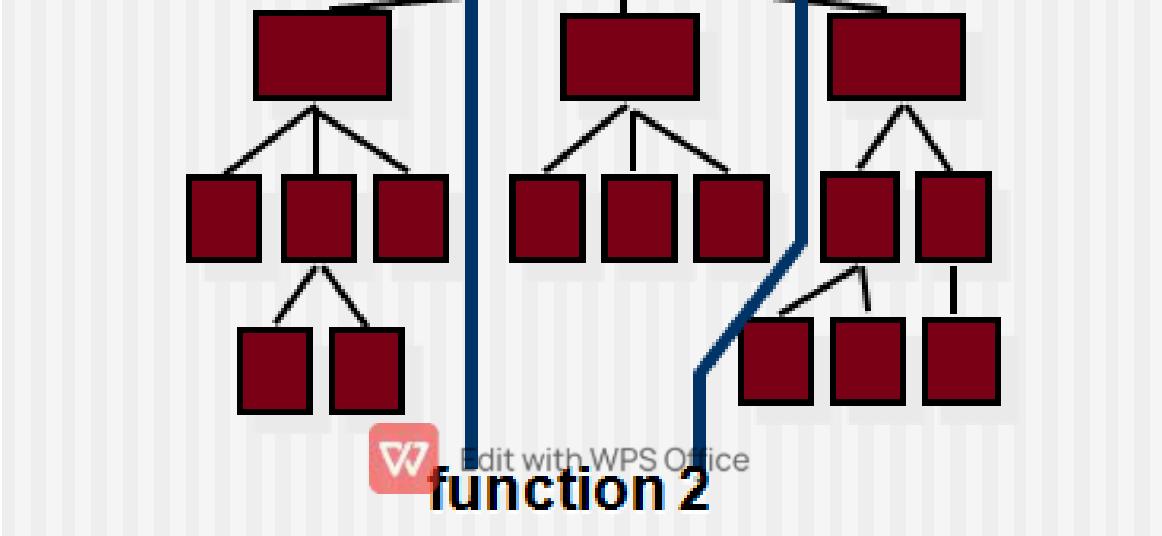


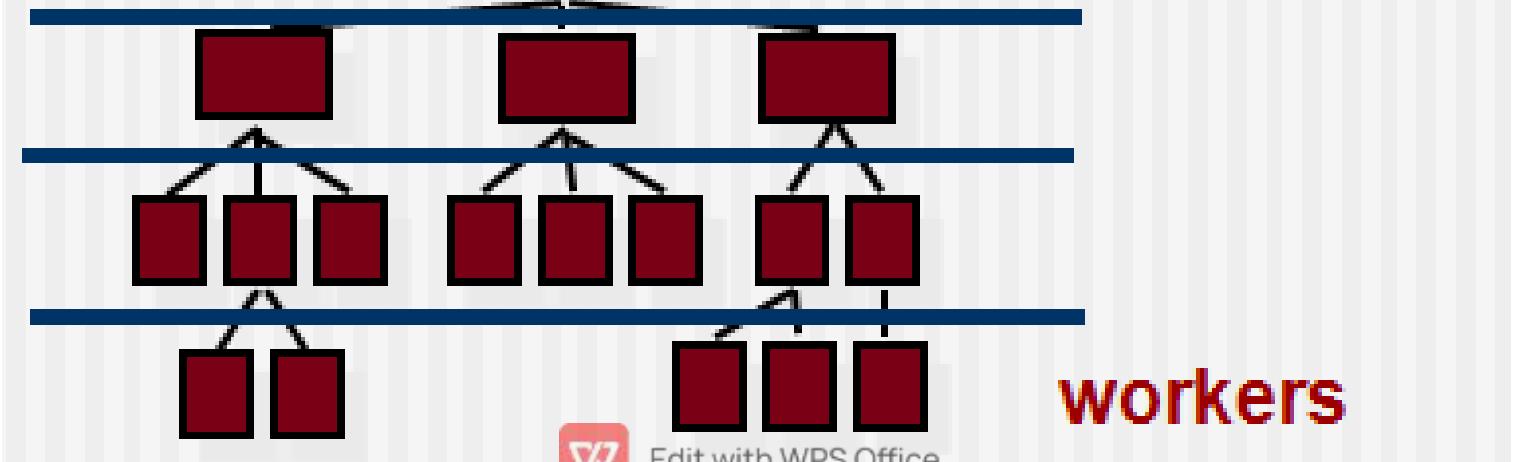
architectural design



Why Partitioned Architecture?

- Results in software that are easier to test
- Leads to software that are ~~easy~~ controllable to maintain
- Results in propagation of fewer side effects

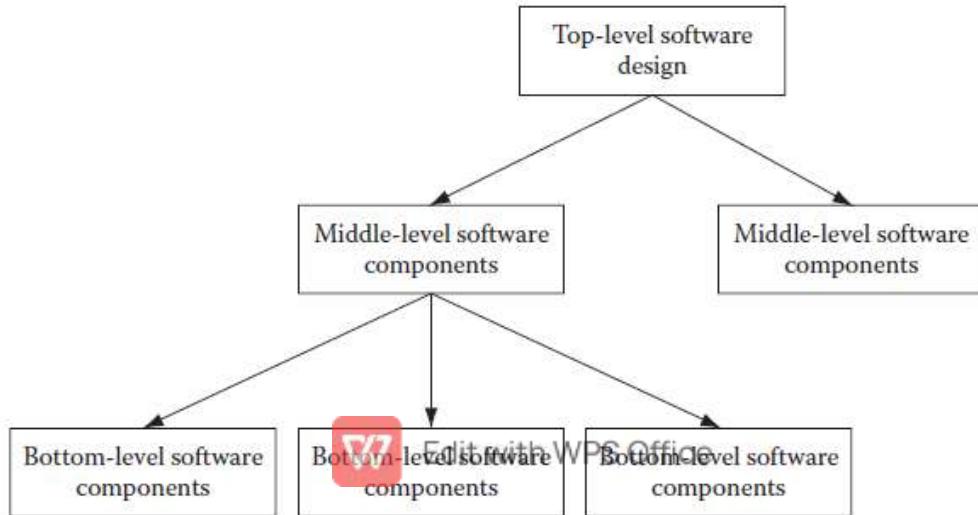




Edit with WPS Office

example
in the p

provided

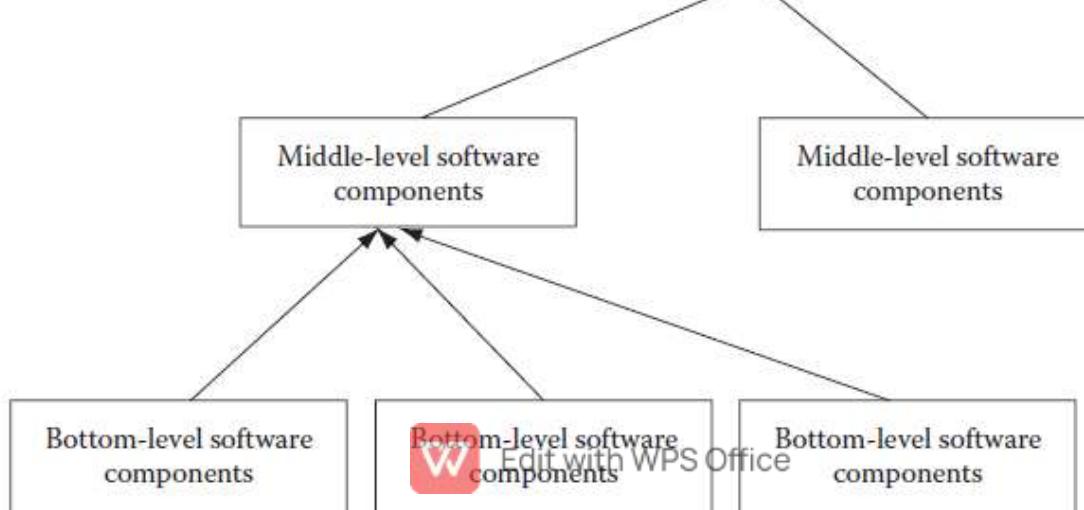


inside a single framework.

- So a fragmented and dissimilar approach for designing different parts of the product is avoided.
- The drawback of the top -down approach is that it is a risky model.
- The whole design has to be made in one go instead of making attempts to incrementally building the design, which is relatively a safer option.
- Generally, the top-down design approach is adopted on waterfall model-based projects.



Edit with WPS Office



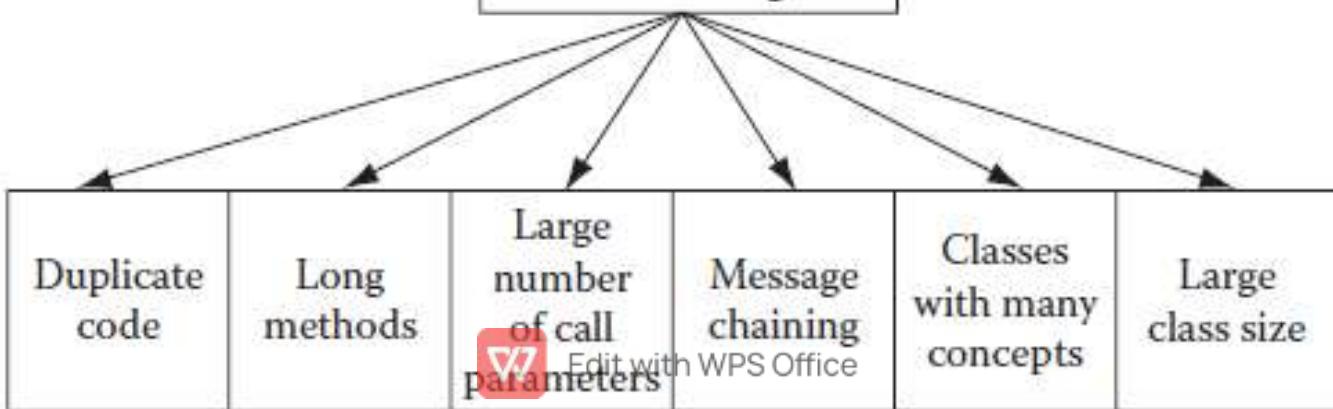
software design from the beginning of the project.

- In each iteration, a design is thought of for the requirements that are taken during the iteration.
- To compensate for a sturdy and elaborate design upfront, the project team engages in refactoring (discussed next) the design to make sure that it does not become bulgy and unmanageable in later iterations.



Edit with WPS Office

Cases requiring refactoring



number of parameters, too much communication between classes resulting from a large number of calls for methods in code, and message chaining by calling one method which in turn calls another method.

- When software code starts having these characteristics, then it is better to go for code cleaning or refactoring.
- Going for refactoring will be justified by savings in time due to better code reuse and make it easier to maintain code and scale up the product.
- Refactoring can be achieved by dividing cumbersome classes into smaller



Edit with WPS Office

refactoring.



Edit with WPS Office

- Service Oriented Architecture (SOA) architecture provides great help here.
- SOA architecture essentially promotes loose coupling, and this implies more or less self-contained classes having less dependency on other classes.
- Increasing module coupling with increase in size of software product is always a concern.
- Frequent refactoring can help in reducing module coupling among classes.

data structures that are required to implement the processing logic, and an interface that enables the component to be invoked and data to be passed to it.



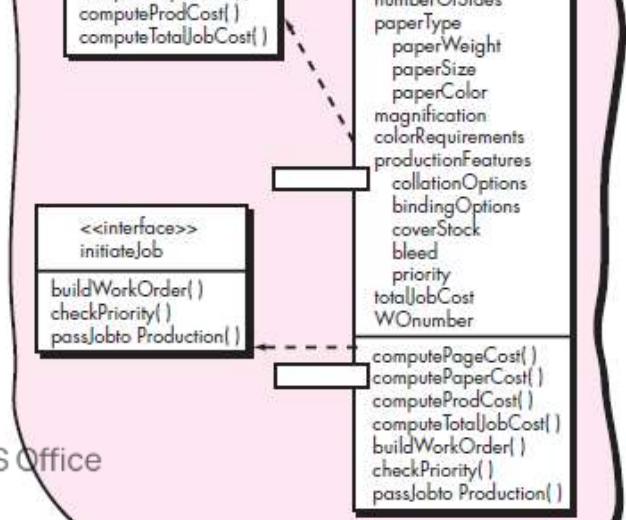
Edit with WPS Office

elaboration, consider software to be built for a sophisticated print shop.

- The overall intent of the software is to collect the customer's requirements at the front counter, cost a print job, and then pass the job on to an automated production facility.
- During requirements engineering, an



Edit with WPS Office



component box.

- Component level design begins at this point.
- The details of the component PrintJob must be elaborated to provide sufficient information to guide implementation.
- The original analysis class is elaborated to flush out all attributes and operations required to implement the class as the component PrintJob.
- Referring to the lower right portion of figure given, the elaborated design class PrintJob contains more detailed attribute information as well as an expanded description of operations required to implement the component.

- (2) A Problem domain component that implements a complete or partial function that is required by the customer, or
 - (3) An Infrastructure component that is responsible for functions that support the processing required in the problem domain.
- Like object-oriented components, traditional software components are derived from the analysis model.
 - In this case, however, the data flow-oriented element of the analysis model serves as the basis for the derivation.
 - To illustrate this process of design elaboration for traditional components,

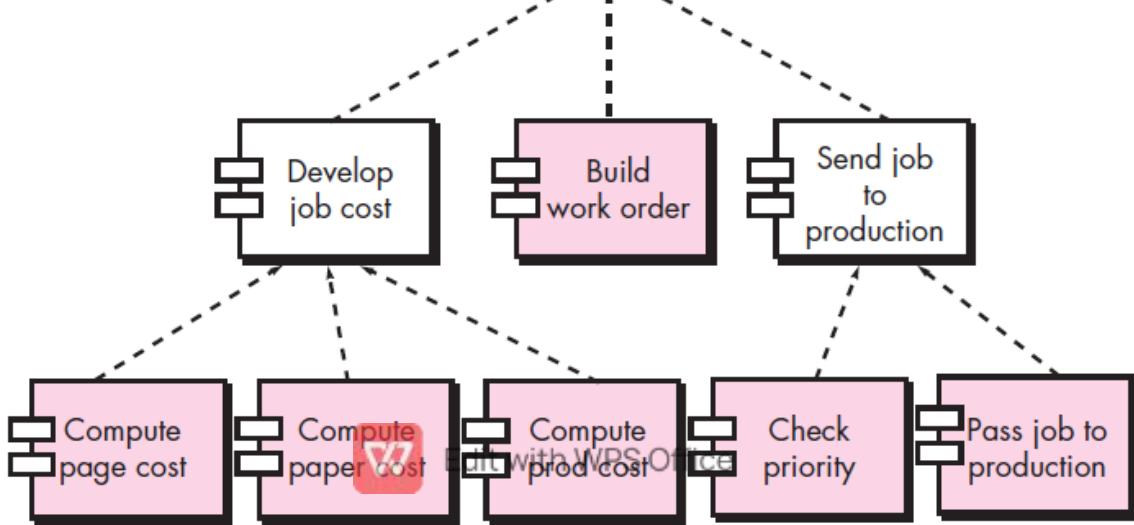


Diagram.

- To illustrate this process, consider the module *ComputePageCost*.
- The intent of this module is to compute the printing cost per page based on specifications provided by the customer.
- Data required to perform this function are: number of pages in the document, total number of documents to be produced, one- or two-side printing, color requirements, and size requirements.
- These data are passed to *ComputePageCost* via the module's interface.
- *ComputePageCost* uses these data to determine a page cost that is based



Edit with WPS Office

in: sides= 1, 2
in: color=1, 2, 3, 4
in: page size = A, B, C, D
out: page cost
in: job size
in: color=1, 2, 3, 4
in: pageSize = A, B, C, D
out: BPC
out: SF

getJobData (numberPages, numberDocs,
sides, color, pageSize, pageCost)
accessCostsDB(jobSize, color, pageSize,
BPC, SF)
computePageCost()

job size (JS) =
numberPages * numberDocs;
lookup base page cost (BPC) ->
accessCostsDB (JS, color);
lookup size factor (SF) ->
accessCostDB (JS, color, size)
job complexity factor (JCF) =
 $1 + [(sides-1)*sideCost + SF]$
pagecost = BPC * JCF



Edit with WPS Office



Edit with WPS Office

The Release Reuse Equivalence Principle (RER). “The granule of reuse is the granule of release.”

- The Common Closure Principle (CCP). “Classes that change together belong together.”
- The Common Reuse Principle (CRP). “Classes that aren’t reused together should not be grouped together.”



Edit with WPS Office

Dependencies and Inheritance

- It is a good idea to model dependencies from left to right and inheritance from bottom (derived classes) to top (base classes).

Cohesion

- Conventional view:
 - The “single-mindedness” of a module
- OO view:
 - Cohesion implies that a component or class encapsulates only



Edit with WPS Office

Coupling

- Conventional view:
 - The degree to which a component is connected to other components and to the external world.
- OO view:
 - A qualitative measure of the degree to which classes are connected to one another.



Edit with WPS Office

- Inclusion or import
- External



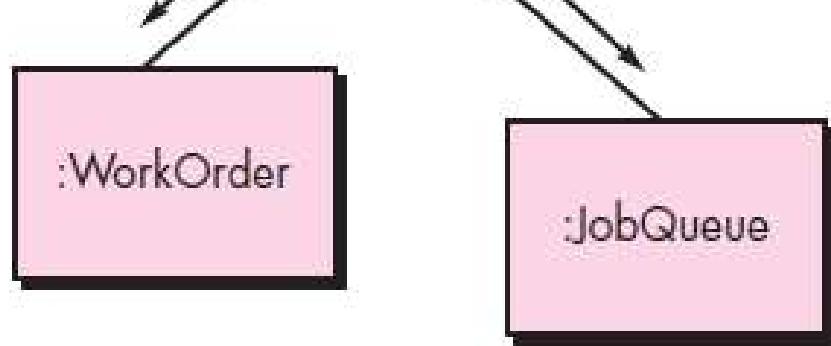
Edit with WPS Office

Step 3c. Elaborate attributes and define data types and data structures required to implement them.

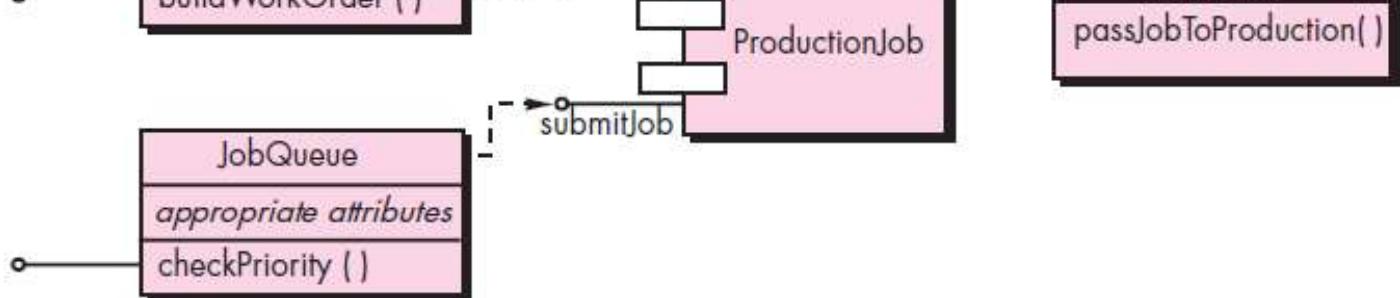
- Step 3d. Describe processing flow within each operation in detail.
- Step 4. Describe persistent data sources (databases and files) and identify the classes required to manage them.
- Step 5. Develop and elaborate behavioral representations for a class or component.
- Step 6. Elaborate deployment diagrams to provide additional implementation detail.
- Step 7. Elaborate deployment diagrams to provide additional implementation detail.



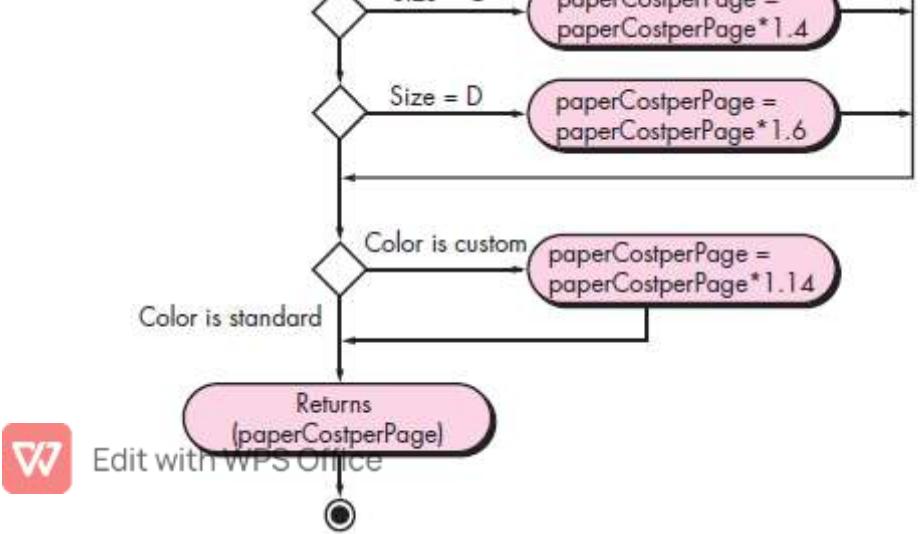
Edit with WPS Office



Edit with WPS Office



Edit with WPS Office



Edit with WPS Office

getElectronicSignature

formingJob

entry/buildJob
exit/save WOnumber
do/

deliveryDateAccepted [customer is authorized]/
printJobEstimate

submittingJob

entry/submitJob
exit/initiateJob
do/place on JobQueue



Edit with WPS Office

jobSubmitted [all authorizations acquired]/
printWorkOrder

- Focuses on content objects and the manner in which they may be packaged for presentation to a WebApp end-user.
- Consider a Web-based video surveillance capability within SafeHomeAssured.com - Potential content components can be defined for the video surveillance capability:
 - (1) The content objects that represent the space layout (the floor plan) with additional icons representing the location of sensors and video cameras;
 - (2) The collection of thumbnail video captures (each an separate data object)

- To achieve these (and many other) capabilities, it is suggested to design and construct WebApp functional components that are identical in form to software components for conventional software.



Edit with WPS Office

Algorithm Design

- The closest design activity to coding
- The approach:
 1. Review the design description for the component
 2. Use stepwise refinement to develop algorithm
 3. Use structured programming to implement procedural logic
 4. Use ‘formal methods’ to prove logic



Edit with WPS Office

walk through;
close door.

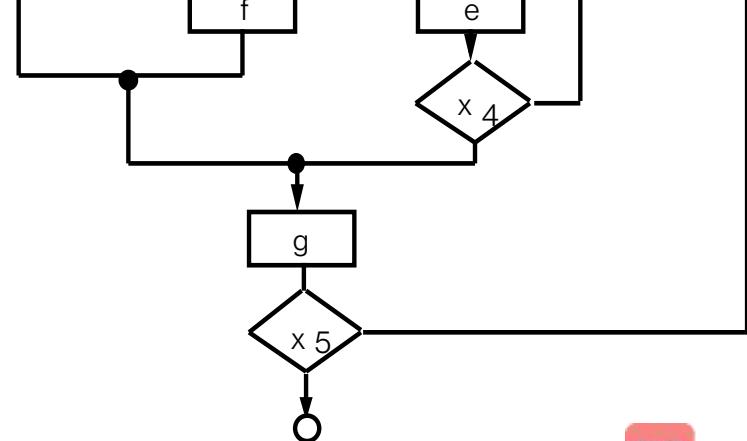
turn knob clockwise;
if knob doesn't turn, then
 take key out;
 find correct key;
 insert in lock;
endif
pull/push door
move out of way;
end repeat



Edit with WPS Office

Structured Programming

- Uses a limited set of logical constructs:
 - Sequence
 - Conditional – If-then-else, Select-case
 - Loops – Do-while, Repeat until
- Leads to more readable, testable code
- Can be used in conjunction with “proof of correctness”
W Edit with WPS Office
- Important for achieving high quality, but not enough



special discount	F	T	F	T	F	T
Rules						
no discount	✓					
apply 8 percent discount		✓	✓			
apply 15 percent discount				✓	✓	
apply additional x percent discount	✓		✓			✓



Edit with WPS Office

- machine readable, no need for graphics input
- graphics can be generated from PDL
- enables declaration of data as well as procedure
- easier to maintain



Edit with WPS Office

- Are commercial off-the-shelf (COTS) components available to implement the requirement?
 - Are internally-developed reusable components available to implement the requirement?
 - Are the interfaces for available components compatible within the architecture of the system to be built?
- At the same time, they are faced with the following impediments to reuse ...

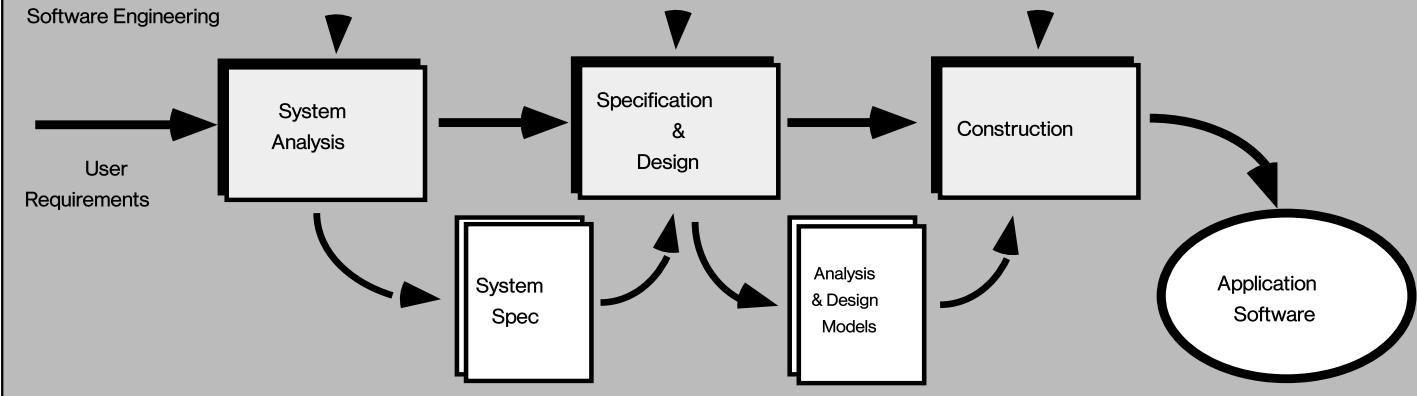


Edit with WPS Office

- Many companies continue to encourage of software development methodologies which do not facilitate reuse.
- Few companies provide incentives to produce reusable program components.



Edit with WPS Office



Edit with WPS Office

- Is component functionality required on future implementations?
- How common is the component's function within the domain?
- Is there duplication of the component's function within the domain?
- Is the component hardware-dependent?
- Does the hardware remain unchanged between implementations?
- Can the hardware specifics be removed to another component?
- Is the design optimized enough for the next implementation?
- Can we parameterize a non-reusable component so that it becomes

- A standard should exist, e.g.,
 - OMG/CORBA
 - Microsoft COM
 - Sun JavaBeans

CBSE Activities

- Component qualification



Edit with WPS Office

- Embedded design assumptions including the use of specific numerical or non-numerical algorithms.
- Exception handling



Edit with WPS Office

CBSE Activities – Composition

- An infrastructure must be established to bind components together.
- Architectural ingredients for composition include:
 - Data exchange model
 - Automation
 - Structured storage
 - Underlying object model



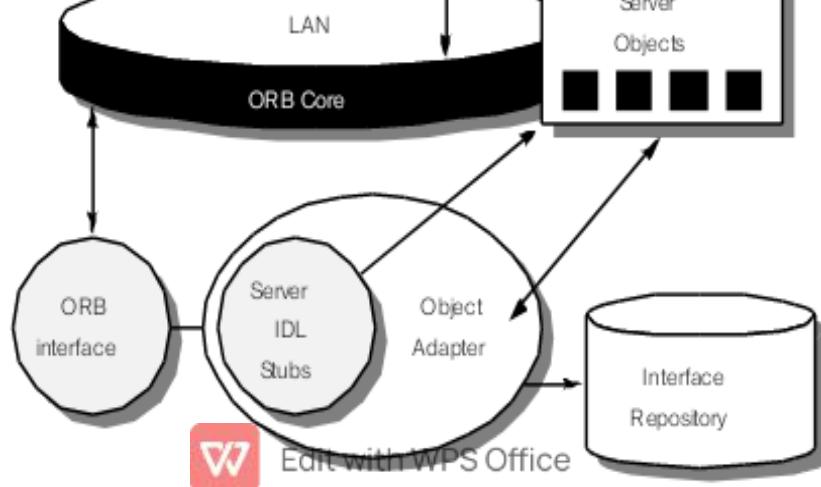
Edit with WPS Office

ORB server.

- Requests are made via an IDL or dynamically at run time.
- An interface repository contains all necessary information about the service's request and response formats.

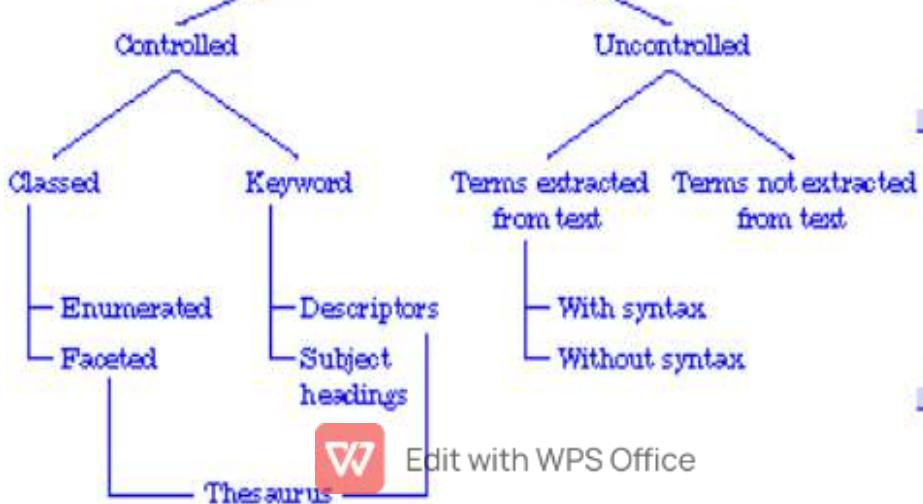


Edit with WPS Office



SUN JavaBeans

- The JavaBeans component system is a portable, platform independent CBSE infrastructure developed using the Java programming language.
- The JavaBeans component system encompasses a set of tools, called the Bean Development Kit (BDK), that allows developers to
 - Analyze how existing Beans (components) work
 - Customize their behavior and appearance
 - Establish mechanisms for coordination and communication
 - Develop custom Beans for use in a specific application



Edit with WPS Office



Edit with WPS Office

Typical Design Errors

- Lack of consistency
- Too much memorization
- No guidance / help
- No context sensitivity
- Poor response



Edit with WPS Office



unnecessary or undesired actions.

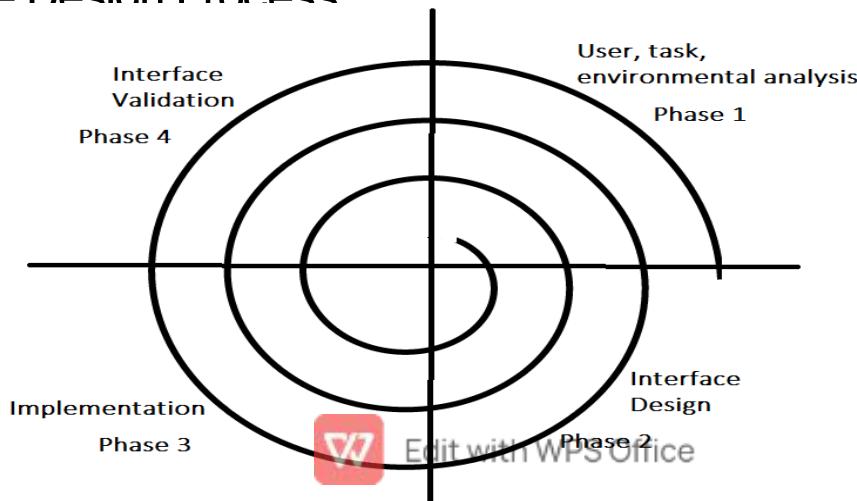
- Provide for flexible interaction.
- Allow user interaction to be interruptible and undoable.
- Streamline interaction as skill levels advance and allow the interaction to be customized.
- Hide technical internals from the casual user
- Design for direct interaction with objects that appear on the screen

Make the Interface Consistent

- Allow the user to put the current task into a meaningful context.
- Maintain consistency across a family of applications.
- If past interactive models have created user expectations, do not make changes unless there is a compelling reason to do so.



User Interface Design Process



User Analysis

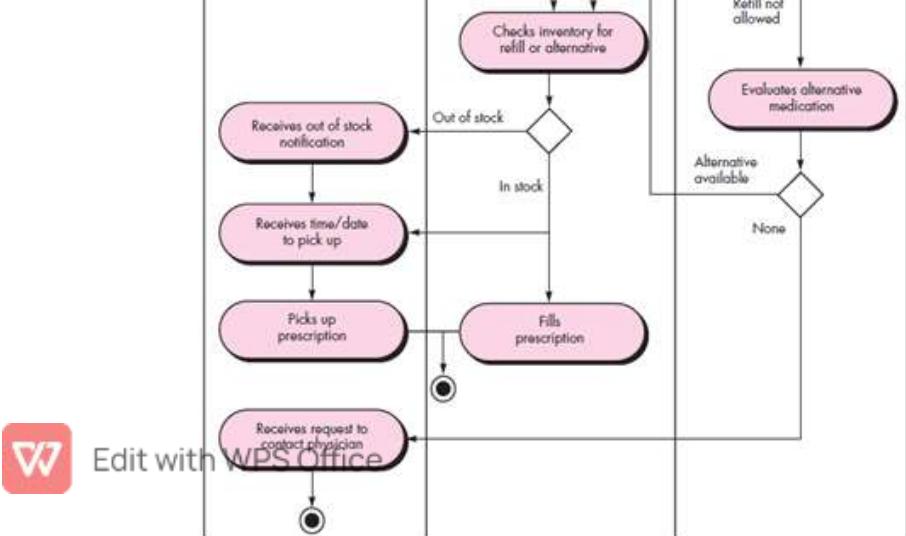
- Are users trained professionals, technician, clerical, or manufacturing workers?
- What level of formal education does the average user have?
- Are the users capable of learning from written materials or have they expressed a desire for classroom training?
- Are users expert typists or keyboard phobic?
- What is the age range of the user community?
- Will the users be represented predominately by one gender?

Task Analysis and Modeling

- Answers the following questions ...
 - What work will the user perform in specific circumstances?
 - What tasks and subtasks will be performed as the user does the work?
 - What specific problem domain objects will the user manipulate as work is performed?
 - What is the sequence of work tasks – the workflow?
 - What is the hierarchy of tasks?



Edit with WPS Office



Edit with WPS Office

understanding?

- Will mechanisms be available for moving directly to summary information for large collections of data?
- Will graphical output be scaled to fit within the bounds of the display device that is used?



Edit with WPS Office

- How will color be used to enhance understanding?

Design Issues

- Response time
- Help facilities
- Error handling
- Menu and command labeling
- Application accessibility
- Internationalization



Edit with WPS Office

- What links are live?
 - What content is relevant?
-
- Where have I been, where am I going?
The interface must facilitate navigation.
 - Provide a “map” (implemented in a way that is easy to understand) of where the user has been and what paths may be taken to move elsewhere within the WebApp



Edit with WPS Office

- Work is carefully and continuously saved, with full option for the user to undo any activity at any time.
- Effective applications and services perform a maximum of work, while requiring a minimum of information from users.



Edit with WPS Office

- Efficiency – The design of the WebApp and its interface should optimize the user's work efficiency, not the efficiency of the Web engineer who designs and builds it or the client-server environment that executes it.
- Focus – The WebApp interface (and the content it presents) should stay focused on the user task(s) at hand.
- Fitt's Law – “The time to acquire a target is a function of the distance to and size of the target.”
- Human interface objects – A vast library of reusable human interface



Edit with WPS Office

- Readability – All information presented through the interface should be readable by young and old.
- Track state – When appropriate, the state of the user interaction should be tracked and stored so that a user can logoff and return later to pick up where she left off.
- Visible navigation – A well-designed WebApp interface provides “the illusion that users are in the same place, with the work brought to them.”



Edit with WPS Office

Develop a procedural representation of the user's interaction with the interface.

- Develop a behavioral representation of the interface.
- Describe the interface layout for each state.
- Refine and review the interface design model.

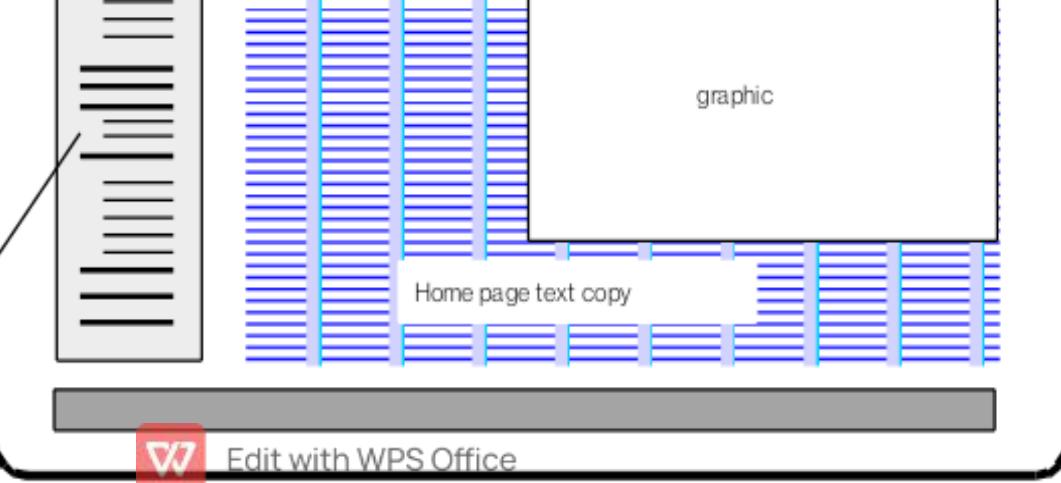


Edit with WPS Office

objective #5

objective #n

Navigation
menu



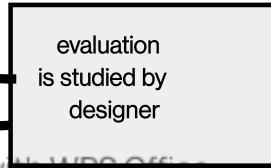
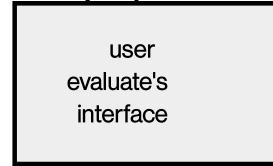
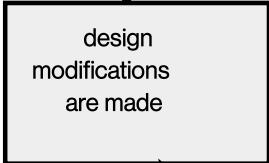
- Don't extend your real estate with the scrolling bar.
- Consider resolution and browser window size when designing layout.



Edit with WPS Office



Edit with WPS Office
Interface design
is complete



Edit with WPS Office
Interface design
is complete



Edit with WPS Office
Interface design
is complete

- Once the pattern has been discovered, it is documented.

Discovering Analysis Patterns

- The most basic element in the description of a requirements model is the use case.
- A coherent set of use cases may serve as the basis for discovering one or more analysis patterns.
- A semantic analysis pattern (SAP) “is a pattern that describes a small set of



Edit with WPS Office

operator can maintain orientation as the vehicle moves in reverse.

- The control software also monitors a proximity sensor to determine whether an object is inside 10 feet of the rear of the vehicle.
- It will automatically break the vehicle if the proximity sensor indicates an object within 3 feet of the rear of the vehicle.
- This use case implies a variety of functionality that would be refined and elaborated (into a coherent set of use cases) during requirements gathering and modeling.



Edit with WPS Office

But in a more general case, a widely applicable pattern is discovered -->

Actuator-Sensor



Edit with WPS Office

- Although many of the sensors and actuators look quite different, their behavior is similar enough to structure them into a pattern.
- The pattern shows how to specify the sensors and actuators for a system, including attributes and operations.
- The Actuator-Sensor pattern uses a pull mechanism (explicit request for information) for Passive Sensors and a push mechanism (broadcast of information) for the Active Sensors.

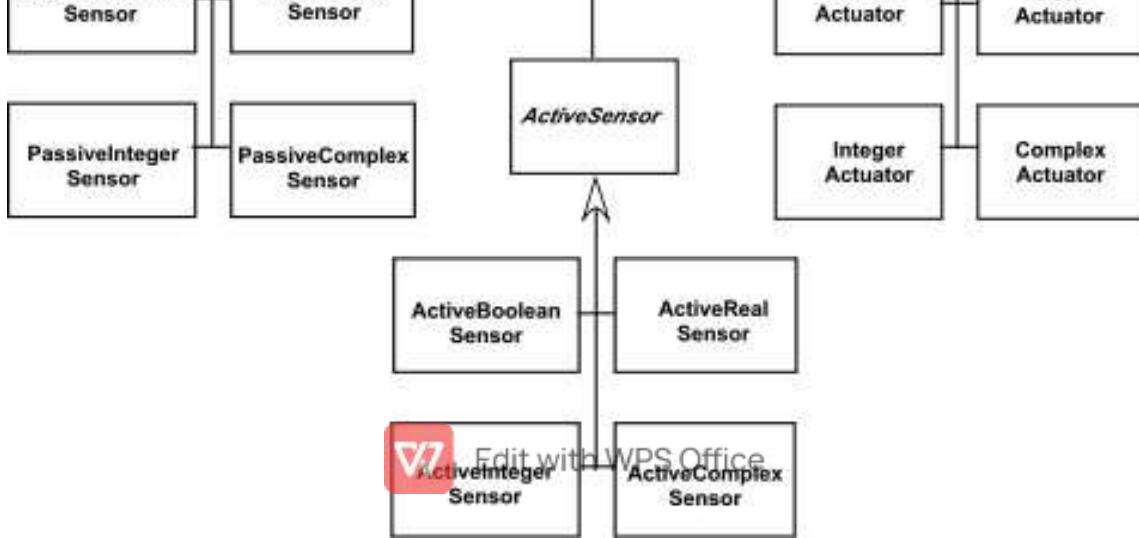


Edit with WPS Office

- Each actuator must have some method to invoke the appropriate response determined by the ComputingComponent.
- Each sensor and actuator should have a function implemented to check its own operation state.
- Each sensor and actuator should be able to test the validity of the values received or sent and set its operation state if the values are outside of the specifications.



Edit with WPS Office



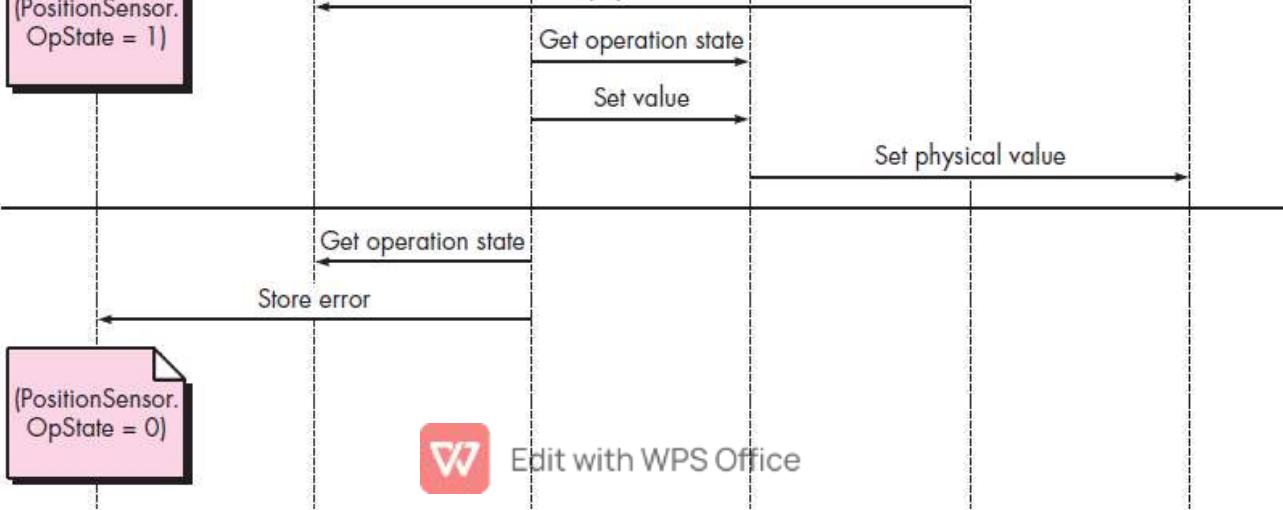
Edit with WPS Office

be easily represented in terms of primitive data types, such as a radar device.

- Nonetheless, these devices should still inherit the interface from the abstract classes since they should have basic functionalities such as querying the operation states.



Edit with WPS Office



Edit with WPS Office

System and their Software counterparts.

- In the lower part of the diagram, below the horizontal line, the Position Sensor reports that the operation state is zero.
- The ComputingComponent then sends the error code for a position sensor failure to the Fault Handler that will decide how this error affects the system and what actions are required.
- It gets the data from the sensors and computes the required response for the actuators.



Edit with WPS Office

- **Functional Analysis** - The usage scenarios (use-cases) created as part of interaction analysis define the operations that will be applied to WebApp content and imply other processing functions. All operations and functions are described in detail.
- **Configuration Analysis** - The environment and infrastructure in which the WebApp resides are described in detail.

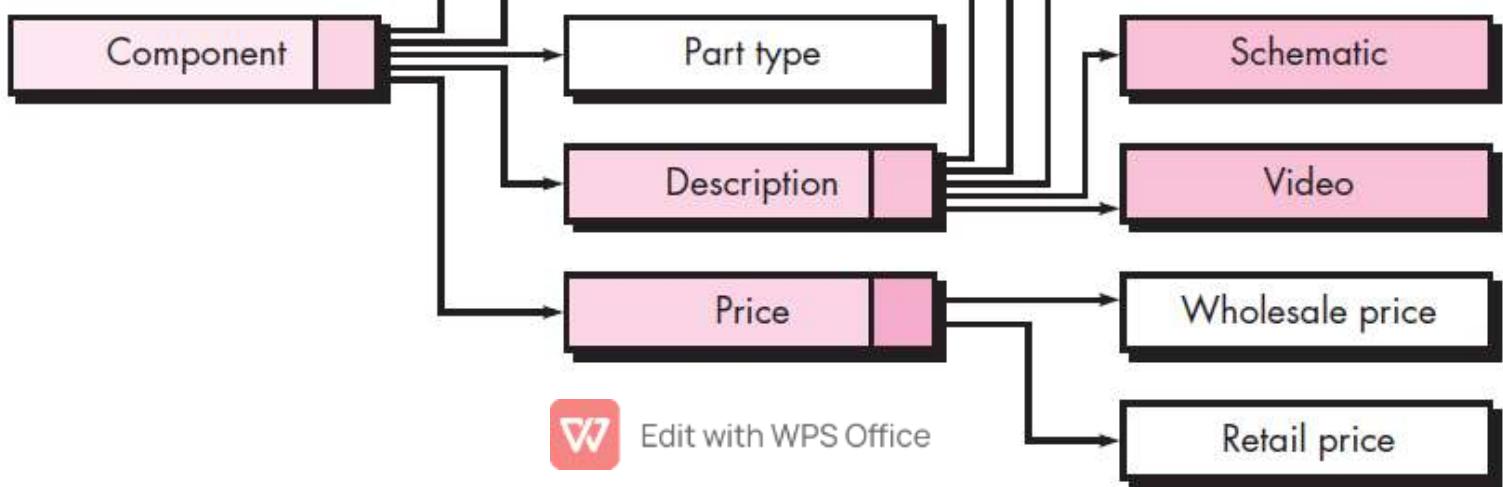
of the business

The Content Model

- Content objects are extracted from use-cases
 - Examine the scenario description for direct and indirect references to content
- Attributes of each content object are identified
- The relationships among content objects and/or the hierarchy of content maintained by a WebApp



Edit with WPS Office



Edit with WPS Office



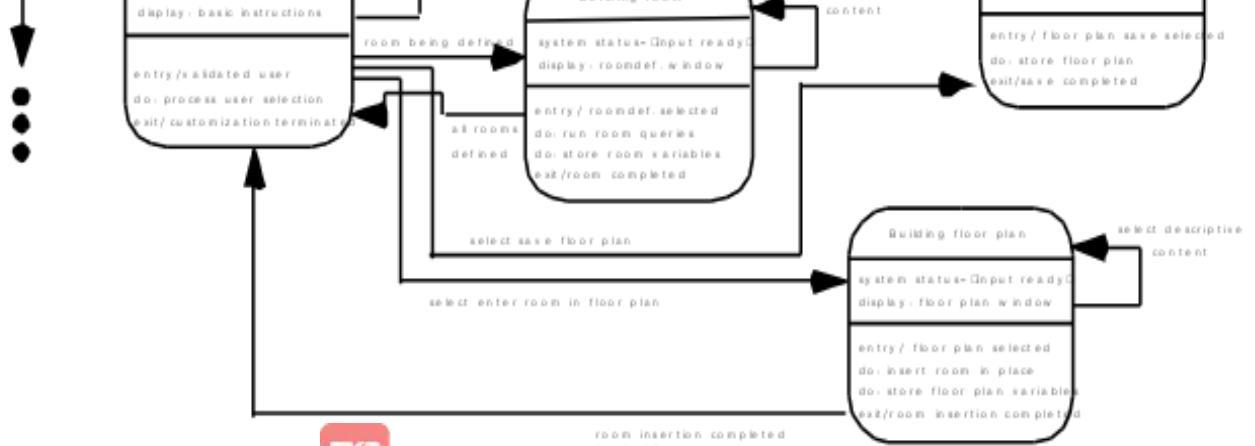
Edit with WPS Office



Figure 18.5 Sequence diagram for use-case: *select SafeHome components*



Edit with WPS Office

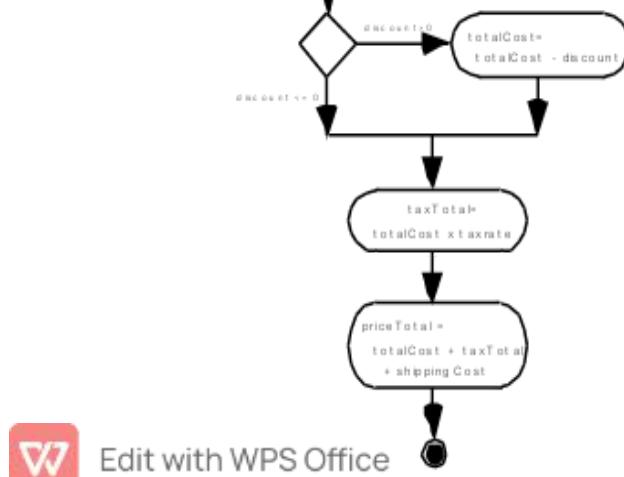


Edit with WPS Office

Figure 18.6 Partial state diagram for new customization interaction



Edit with WPS Office



Edit with WPS Office

Figure 18.7 Activity diagram for `computePrice` operation

- Testing requirements should be defined



Edit with WPS Office

- Should navigation to related groups of elements be given priority over navigation to a specific element.
- Should navigation be accomplished via links, via search-based access, or by some other means?
- Should certain elements be presented to users based on the context of previous navigation actions?



Edit with WPS Office

- For which user category should optimal navigation be designed?
- How should links external to the WebApp be handled, whether it is by overlaying the existing browser window or as a new browser window or as a separate frame?



Edit with WPS Office

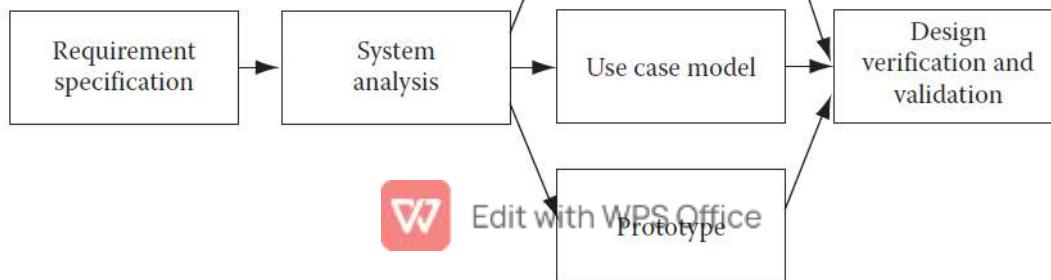
- Until all details about design are made, development cannot be started.
- So, the development team cannot start their job until they have a software design in their hands.
- Still some aspects about latter tasks can be done in advance.
- For instance, what development language will be used and how the application can be partitioned for development work can be decided at the design stage itself.
- Similarly, how maintenance and support functions will be done for the



Edit with WPS Office

and validated through design reviews.

- Once the design is reviewed and approved, then the design phase is over (Figure below – Software Design Life Cycle).





Edit with WPS Office



Edit with WPS Office



Edit with WPS Office

Compiled by IT Department, SRMIST, KTR



Disclaimer:

The lecture notes have been prepared by referring to many books and notes prepared by the teachers. This document does not claim any originality and cannot be used as a substitute for prescribed textbooks.



Edit with WPS Office



Edit with WPS Office

- Object-Oriented Programming
- Automatic Code Generation
- Software Code Reuse
- Pair Programming
- Test-Driven Development
- Configuration Management
- Software Construction Artefacts

software deployment are all equally crucial tasks.

- Furthermore, the process of software construction itself consists of many tasks; it not only includes software coding but also unit testing, integration testing, reviews and analysis.
- Construction is one of the most labor intensive phases in the software development life cycle.
- It comprises 30% or more of the total effort in software development.



Edit with WPS Office

work is divided not only among developers, but also small teams are formed to work on parts of the software build.

- In fact, to shrink the construction time, many distributed teams, either internal or through contractors are deployed.
- The advantage to this is that these project teams do the software coding and other construction work in parallel with each other and thus the construction phase can be collapsed.



Edit with WPS Office

The whole process of construction should follow a proven process so that the produced code is maintainable, testable and reliable.

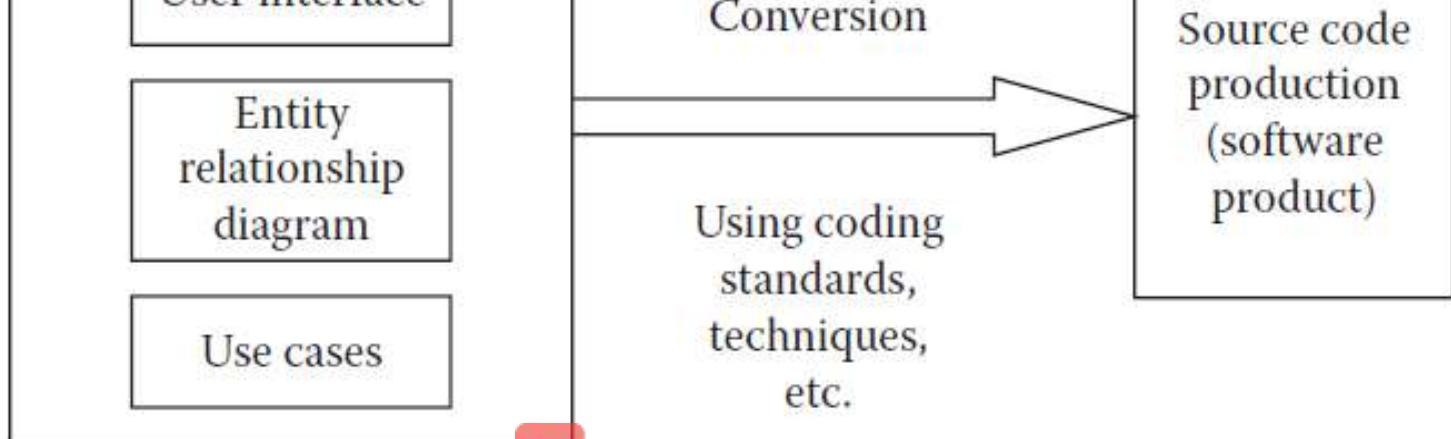
- The process itself should be efficient so that resource utilization can be optimized and thus cost of construction can be kept at a minimum.



Edit with WPS Office

CONVERTING THE SPECIFICATIONS INTO SOFTWARE CODE IS TOTALLY DEPENDENT ON THE CONSTRUCTION TEAM.

- HOW WELL THEY DO IT DEPENDS ON THEIR EXPERIENCE, SKILLS AND THE PROCESS THEY FOLLOW TO DO THEIR JOB.
- APART FROM THESE FACILITIES, THEY ALSO NEED SOME STANDARDS IN THEIR CODING SO THAT THE WORK IS FAST AS WELL AS HAS OTHER BENEFITS LIKE MAINTAINABILITY, READABILITY AND REUSABILITY (FIGURE IN NEXT SLIDE).



Edit with WPS Office

- A uniform coding standard across all construction teams working on the same project will make sure that these issues can be minimized if not eliminated (Figure below - Software Construction Characteristics).
- Some of the coding standards include standards for code modularity, clarity, simplicity, reliability, safety and maintainability.



Edit with WPS Office



Edit with WPS Office

- Each major function should be contained inside a software code module.
- The module should contain not only structure, but it should also process data.
- Each time a particular functionality is needed in the software construction, it can be implemented using that particular module of software code.
- This increases software code reuse and thus enhances productivity of developers and code readability.



Edit with WPS Office

- Standard naming conventions should be used so that the code has ample clarity.
- There should be sufficient documentation inside the code block, so that anybody reading the code could understand what a piece of code is supposed to do.
- There should also be ample white spaces in the code blocks, so that no piece of code should look crammed. White spaces enhance readability of written code.



Edit with WPS Office

- Reliable source code can be achieved by first designing the software product with future enhancement in consideration as well as by having a solid structure on which the software product is to be built.
- When writing pieces of source code based on this structure, there will be little chance of defects entering into the source code.
- Generally during enhancements, the existing structure is not able to take load of additional source code and thus the structure becomes shaky.
- If the development team feels that this is the case, then it is far better to

Industries where human lives are concerned and that human lives could be in danger because of faulty machine operation or exposure to a harmful environment.

- In these industries, the software product must be ensured to operate correctly and chances of error are less than 0.00001%.
- Industries like medicine and healthcare, road safety, hazardous material handling need foolproof software products to ensure that either human lives are saved (in case of medicine and healthcare) or human lives are not in danger.

- Simplicity makes the code readable and will help in removing any defects found in the source code.
- Simplicity of written code can be enhanced by adopting best practices for many programming paradigms.
- For instance, in the case of object-oriented programming, abstraction and information hiding add a great degree of simplicity.
- Similarly, breaking the product to be developed into meaningful pieces that mimic real life parts makes the software product simple.

more than 70% of all costs including software development,

implementation, and maintenance.

- To make sure that maintenance costs are under limit during software construction, it should be made sure that the source code is maintainable.
- It will be easy to change the source code for fixing defects during maintenance.



Edit with WPS Office

- In object oriented programming, what base classes are to be made, which will be used throughout construction, is a subject that is part of the coding framework.
- In general, coding frameworks allow construction of the common infrastructure of basic functionality which can be extended later by the developers.
- This way of working increases productivity and allows for a robust and well structured software product.

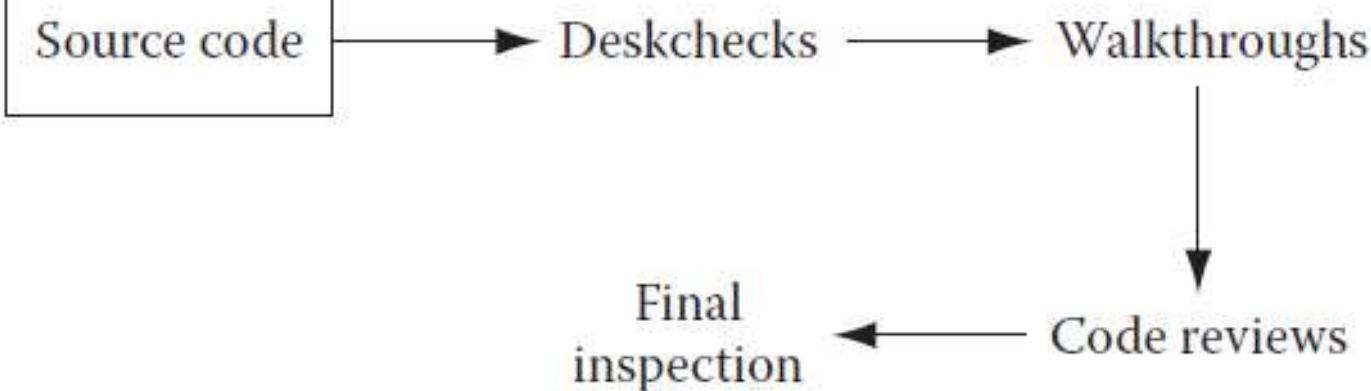


Edit with WPS Office

- Any construction rework means wasting a lot of effort already put in.
- Moreover, it is also a fact that it is cheaper to fix any defects found during construction at the phase level itself.
- If those defects are allowed to go in software testing (which is the next phase), then fixing those defects will become costlier.



Edit with WPS Office



Edit with WPS Office

- While inspections provide the final go/no go decision for approval of a piece of code, other methods are less formal and are meant for removing defects instead of deciding whether a piece of code is good enough or not.



Edit with WPS Office

members.

- These team members review the code and send feedback and comments to the developer as suggestions for improvement in the code.
- The developer reads those feedbacks and may decide to incorporate or to discard those suggestions.
- So this form of review is totally voluntary.
- Still, it is a powerful tool to eliminate defects or improve software code.



Edit with WPS Office

through his piece of code.

- The team members then make suggestions for improvement, if any.
- The developer then can decide to incorporate those suggestions or discard them.



Edit with WPS Office

generated and is reviewed by the entire team.

Reviews – Inspections

- Code inspections are final reviews of software code in which it is decided whether to pass a piece of code for inclusion into the main software build.



Edit with WPS Office

- As it is well known in the industry, the early software products were of small size due to limited hardware capacity.
- With increasing hardware capacity, the size of software products has been increasing.
- Software product size affects the methods that can be used to construct specific sized software products.



Edit with WPS Office

advancement in computer science, different programming techniques evolved.

- These include structured programming, object-oriented programming, automatic code generation, test-driven development, pair programming, etc.



Edit with WPS Office

- Using structured programming, large programs could be constructed that could be used for making large commercial and business applications.
- Structured programming enabled programmers to store large pieces of code inside procedures and functions.
- These pieces of code could be called by any other procedures or functions.
- This enabled programmers to structure their code in an efficient way.
- Code stored inside procedures could be reused anywhere in the application by calling it.

because objects contain both data and structure.

- Widely used as an example in object-oriented programming books, a car consists of a chassis, an engine, four wheels, body, and transmission.
- Each of these objects has some specific properties and specific functions.



Edit with WPS Office

behavior of the object is visible or perceived.

- Unnecessary details about the object are hidden and in fact are not available from outside.
- This kind of representation of objects makes them robust and a system built on using them has relatively few problems.



Edit with WPS Office

- Unfortunately, this is still a dream. Some CASE and modeling tools are available that generate software code. But they are not sophisticated. They are also not complete.
- Then there are business analyst platforms developed by many ERP software vendors that generate code automatically when analysts configure the product.
- These analyst platforms are first built using any of the software product development methodologies



Edit with WPS Office

- Some of these code types include control statements such as loop statements, if statements, etc., and database access, etc.
- Generating all of the software code required to build a software application is still difficult.
- But some companies like Sun Microsystems are working to develop such a system.



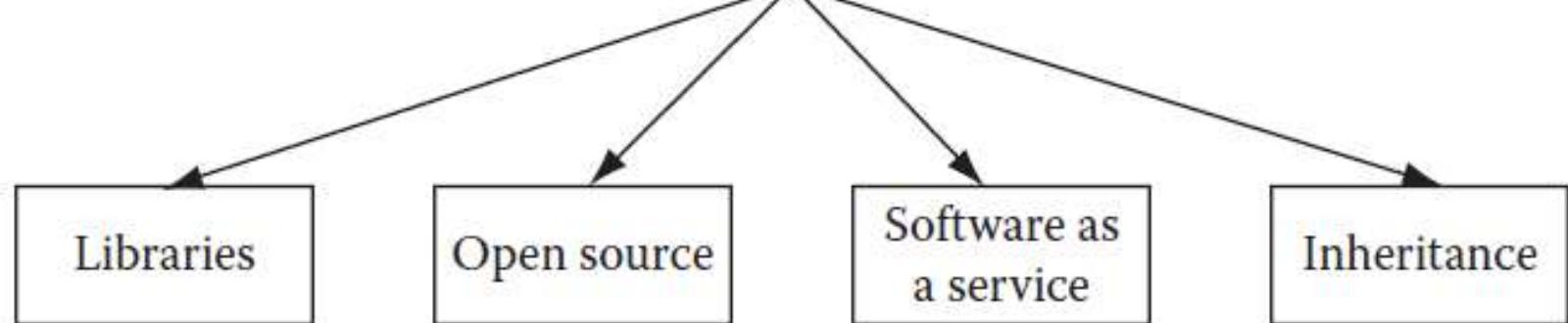
Edit with WPS Office

Making a block of source code to create a functionality or general utility library and using it at all places in the source code wherever this kind of functionality or utility is required is an example of code reuse.

- Code reuse in procedural programming techniques is achieved by creating special functions and utility libraries then using them in the source code.
- In object-oriented programming, code reuse is done at a more advanced level.



Edit with WPS Office



Edit with WPS Office

- Apart from creating and using libraries and general purpose classes for code reuse, a more potent code reuse source has evolved recently.
- It is known as “service oriented architecture” (SOA).



Edit with WPS Office

and guides him through the requirements (functional, nonfunctional).

- When it is the turn of the other developer to write the code, the first developer sits behind him and guides him on the requirements.
- So developers take turns for the coding and coaching work.
- This makes sure that each developer understands the big picture and helps them to write better code with lesser defects.



Edit with WPS Office

- Once it is proved that their logic is perfect, only then they write the source code.
- So here, tests drive software development, and hence it is appropriately named test-driven development.



Edit with WPS Office

- So it happens that the development team ends up with many versions of a source code during the project.
- If the version control management is not handled properly, then many developers may start working on a wrong version of source code, and thus a lot of rework may be needed in the end.
- There is one more dimension to configuration management for the construction phase.



Edit with WPS Office

- When the build is broken, then no other developer can check in his code.
- Thus, development is halted until the build is rebuilt with the correct code.
- Imagine what may happen in the case of distributed teams located at far-flung locations with different time zones and a central build is being maintained.
- It will be difficult to communicate and manage the build process in such a scenario.



Edit with WPS Office

- If the build fails, then the developer who had checked-in in the code gets the message and immediately tries to fix the build.
- Once the build is fixed, then other developers can check-in their code.
- Thus, configuration management plays an important role in construction phase.



Edit with WPS Office

well understood, and changing any source code will be easy.

- Review reports are also generated after reviews are conducted.



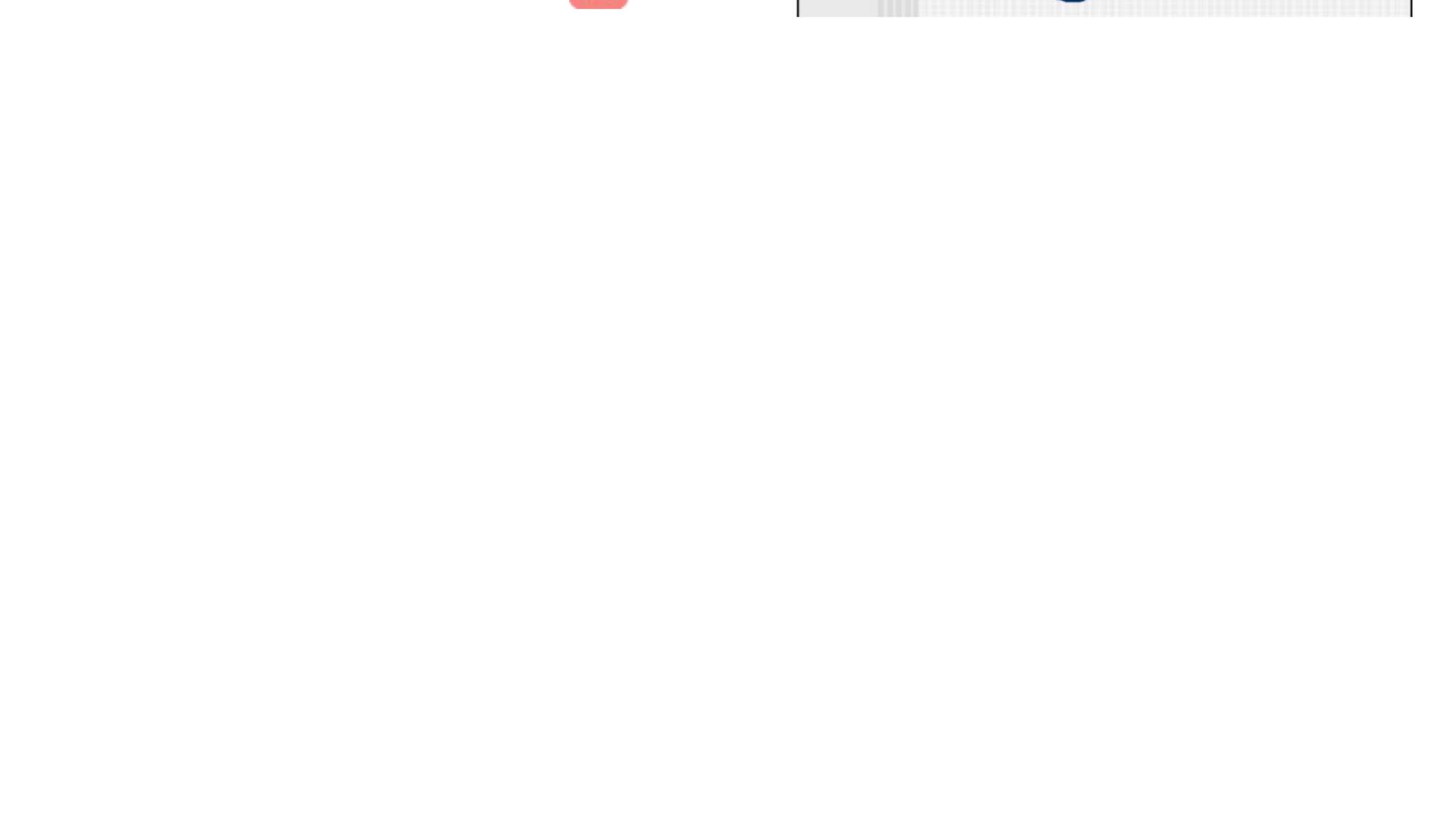
Edit with WPS Office



Edit with WPS Office



Edit with WPS Office



need for drivers and stubs

- Daily testing gives managers and practitioners a realistic assessment of the progress

Creating tests

- Understandable

- An edge must start and terminate at a node

11

$t=1$

(b)


```
20 printf("End of list\n");
```


- (5) Initialization and termination errors.



- Input: {true condition} Eq classes: {true condition}, {false condition}





they have reached.















testing.







- 5.1 Use Case Testing

identify test cases that cover the entire system.

- Mootor test plan should align with test policy and test strategy. It

reached, then test execution stops.

Test, Performance Test, System Test.. etc

This information should be displayed visually by using color indicator,
graph and highlighted table

compliance, project schedule compliance, and quality (number of



ENGINEERING & PROJECT

MANAGEMENT



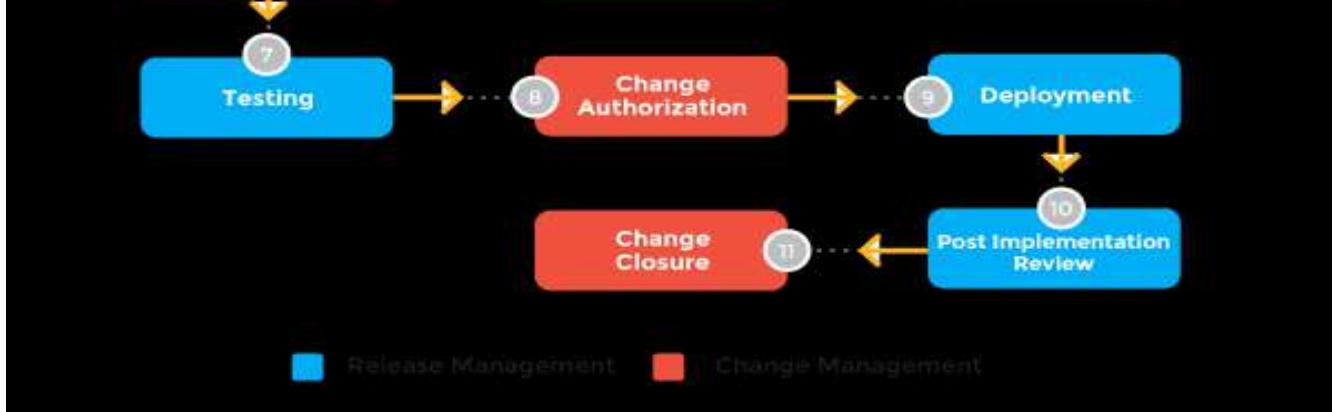
Edit with WPS Office

—itative factors (methodologies for software design, development, testing, closure and implementation

CLR-2: Acquire the latest industry knowledge, tools and comply to the latest global standards for project management



Edit with WPS Office



Edit with WPS Office

- Maintenance Cost
- Maintenance Process
- Maintenance Life Cycle
- Maintenance Techniques
- Software Release



Edit with WPS Office

- However, do not run fast in anticipation of wrapping things as early as possible. After all this is your magnum opus and you need to be careful.
- You need to make sure that all your tasks are completed



Edit with WPS Office

- Alpha and beta versions of the software typically precede its release.
- Releases may also referred to as launches or increments.



Edit with WPS Office

- Your release rhythm will depend on a few factors, including where your product business is in its market lifecycle and your available resources.



Edit with WPS Office

- You're introducing new features to current customers
- You're introducing new features to new or different customer segments
- You're re-releasing your current product using different technology
(often called “re-platforming”)

why the release matters.

- What do you hope will happen for the business once the product has been released?
- By when?



Edit with WPS Office

buy-in from your team. This is especially important given that most of the attendees will lead the efforts of actually building the product.



Edit with WPS Office

dependable release management process.

- A release management process incorporates all of the following:



Edit with WPS Office

been through during estimation.

- This general plan will also provide expectations of major product changes (or dependencies) for products that may depend on your roadmap or platform.
- Plans will be revisited after each iteration. For this reason, a tracking of an external release target (with quarterly, monthly, or other precision) can be helpful.



Edit with WPS Office

- It will still set the expectation and trust with your customers, but can be refined

- As a product manager, establishing a launch or release template will enable you to create a "gold standard" for major delivery.
- Use this template to engage your greater team, who may be supporting multiple products in the portfolio only when needed.
- A standard for launch also sets expectation for when these teams will be needed

QUESTION

- A release status will enable communication to your internal stakeholders, while feature status workflows enable granular visibility into the readiness of the feature and its current status with respect to development, staging, or QA environments.



Edit with WPS Office

regular reviews to ensure plans are on track.



Edit with WPS Office

Estimate cost of providing support	Selection of software version to be shipped	Decision for alpha, beta or regular release	Create walk around for known defects	Provide training to support staff	Make customer support strategy
------------------------------------	---	---	--------------------------------------	-----------------------------------	--------------------------------



Edit with WPS Office

has to meet the deadline.

- It is a constant struggle that calls for good product release strategies.
- Depending on the situation, the project manager may need to convince the management to cut short some of the product features to meet the deadline as well as meet quality standards

the best solution for meeting quality standards.

- If the software vendor is not too sure about product quality, then he may opt for an alpha or beta release of the product.
- In that case, the product will be released only among a few selected groups and not in the market as a whole. The controlled product release is the best option in these conditions



Edit with WPS Office

- Walk around for known issues, estimated number of critical bugs still remaining in the product, training for the support staff, etc., should be done.
- The cost of support, depending on the number of estimated users, walk around, and remaining bugs should be figured out.
- These measures will ensure that the product is transitioned into market without facing major difficulties.



Edit with WPS Office

Check software interfaces	Check hardware interfaces	Create master data	Create test data	Create user accounts	Check infrastructure for installation
---------------------------	---------------------------	--------------------	------------------	----------------------	---------------------------------------



Edit with WPS Office

to be there for installing your software product.

- You need to make sure that you have developed and tested all the hardware and software interfaces for integrating your product, with existing legacy systems and infrastructure.



Edit with WPS Office

production team or customer's team.

- Therefore, prepare a list of your own requirements and hand it over to your customer's support team so that they are prepared when you arrive for implementation.



Edit with WPS Office

common issues that occur when implementing a software product: business alignment from the organizational view and acceptance from human view.



Edit with WPS Office

software purchase (more than hardware and software requirements together).



Edit with WPS Office

Management Cycle



Edit with WPS Office

- Give this list to the end users and ask them to select one user per role who will receive the training.
- Apart from the user manual, you also need to prepare a tutorial to include probable scenarios that may arise during

how to use the product in those circumstances.

- This will lead to a waste of your support team's time.
- It is lot better to train them now, during user training, rather than face user requests later

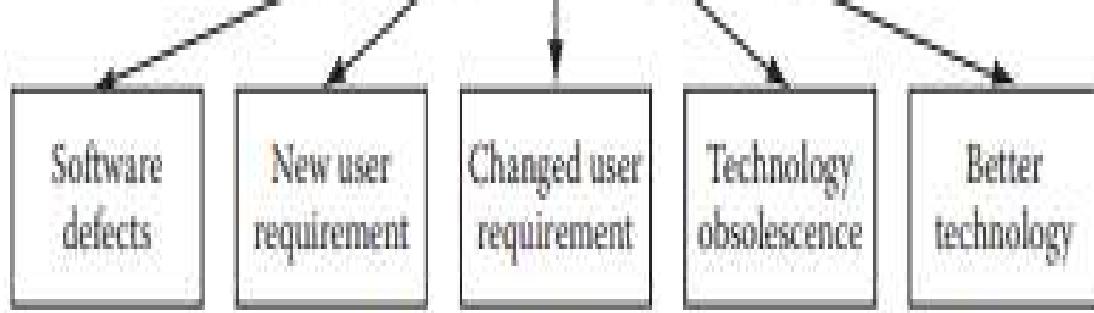


Edit with WPS Office

MAINTENANCE



Edit with WPS Office



Edit with WPS Office

- To list various factors which adversely affect software maintenance.
- To know types of software maintenance, viz. corrective, adaptive, perfective, and preventive maintenance.
- To understand the Software maintenance life cycle

software.

- To appreciate need for technology change management, which is a process of identifying, selecting and evaluating new technologies.
- To understand software maintenance documentation.



Edit with WPS Office

- Techniques for maintenance
- Tools for Software maintenance
- Technology change management(TCM)
- Software maintenance documentation.

- In software engineering a software needs to be ‘serviced’ so that it is able to meet the changing environment (such as business and user needs) where it functions this servicing of software.



Edit with WPS Office

Due to changes in government regulations or students to maintain competition with other software that exist in the same category.

- Improving the Software to Support User Requirements



Edit with WPS Office

To enhance functionality in a software, to improve performance .

to enhance functionality in a software, to improve performance .

- **Facilitating Future Maintenance Work**

Include restructuring of the software code and database



Edit with WPS Office

used in the software.

complex unless some actions are taken to reduce the complexity.

- Lehman stated five laws for software maintenance and evolution of large systems.



Edit with WPS Office

Law of conservation of familiarity	<p>For E-Systems to efficiently continue to evolve a deep understanding of how the system functions, and why it has been developed to function in that manner, must be preserved at all costs.</p> <p>The incremental change in each release over the life time of the system is approximately constant.</p>
Law of conservation of organizational stability	<p>Over a program's lifetime, its rate of development is approximately constant and independent of the resources devoted to system development.</p>

	can have negative affects to the ability to be comprehended along with its ability to evolve.
Law of declining quality	Poorly modified systems lead to introduction of defects; & The quality of E-type systems will appear to be declining as newer products emerge.
Law of feedback system	To sustain continuous change or evolution, & to minimize threats of software decay & loss of familiarity, feedback to monitor the performance is must. <small>Edit with WPS Office</small>

- Legacy system are generally associated with high maintenance costs.
- The root cause of this expense is the degraded structure that results from prolonged maintenance.

must have occurred over a legacy system's lifetime of change.

Obsolete support software	Support software may not be available for a particular platform, or no longer be supported by its original vendor or any other organization.
Obsolete hardware	Legacy system's hardware may have been discontinued. <small>Edit with WPS Office</small>

	proper working of the organizations which operate them.
Poorly documented	Documentation is often missing or inconsistent.
Poorly understood	As a consequence of system complexity and poor documentation, software maintainers often understand the legacy system poorly.

Predicting
System
Changes

Maintenance
Costs

What will be the number
of change requests?

How much will be the cost
of maintaining a system
over its lifetime?

What will be the costs
involved in maintaining a
system over the next year?



Edit with WPS Office

- Implementing change degrades the system structure and hence reduces its maintainability.
- Costs involved in implementing change depend on the maintainability of system components.



Edit with WPS Office

Organizational Environment	<ul style="list-style-type: none">• Change in business policies.• Competition in market.
Operational Environment	<ul style="list-style-type: none">• Hardware platform.• Software specifications.



Edit with WPS Office

Software Product

- Quality of documentation.
- Complexity of programs.
- Program Structure.

Software Maintenance team

- Staff turnover.
- Domain expertise.



Edit with WPS Office

crucial. When the organization faces such a case, it is left with no alternative but to either get an entirely different software product that will replace the existing one or do maintenance of an existing product to make it usable.



Edit with WPS Office



Edit with WPS Office

Maintenance team



Edit with WPS Office

- Adding new functions to the system
- Finding the source of system failures or problems
- Managing change to the system as they are made.

The aspects of a maintenance team that lead to high maintenance costs are:

- Staff turnover



Edit with WPS Office

- Software defects: There are major software defects in the product and it is difficult to operate. For this reason, a software patch may be needed to be applied so that these defects are removed.
- Change in user requirements: The business organization that was using the software product has seen a change in business transactions or business workflows that are not supported by the software product

entire product development cycle.

- A lot of work remains to be done during the maintenance phase of any software product. How to manage these activities so that costs can be minimized is an area of concern yet to be resolved.

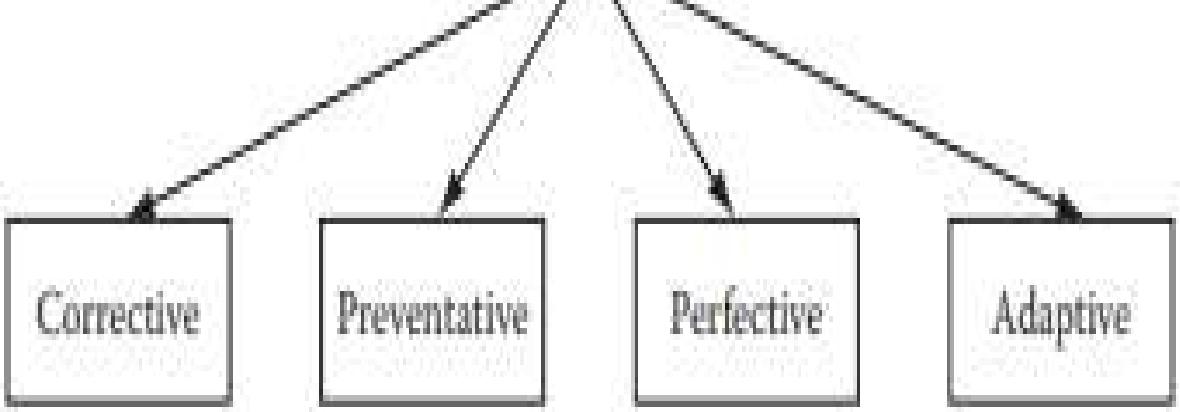


Edit with WPS Office

- preventive
- perfective maintenance



Edit with WPS Office



Edit with WPS Office

number of errors are detected, the software vendor instructs

his maintenance team to create a patch to rectify them.

- The maintenance team then makes a plan and fixes those defects.
- After application of the patch containing the fixes, the software starts running without these defects



Edit with WPS Office

If any of these change over time, it becomes difficult to run the software product. In such cases, it becomes necessary to do adaptive maintenance so that the software product becomes reusable.

- This kind of maintenance may involve changing the interface or porting the application to another hardware/ software platform



Edit with WPS Office

may have changed, or an altogether new business transaction was represented in the software product.

- For all these kinds of requirements, a perfective maintenance may be needed



Edit with WPS Office

software product operates.

- Many of these changes can be perceived in advance.
- In such cases, preventive maintenance on the software product can make sure that the product will be useful even after these



Edit with WPS Office

- Moreover, large enterprise software products are that much crucial.
- When the organization faces such a case, it is left with no alternative but to either get an entirely different software product that will replace the existing one or do maintenance



Edit with WPS Office

2. Opportunity loss: Sometimes there could be some business opportunity in the marketplace, but due to some software problems it could not be availed.



Edit with WPS Office

- The losses due to problems with the software can be compared to probable cost of maintenance and an ROI (return on investment) can be done.
- If we get a desirable ROI then it is better to go for maintenance.



Edit with WPS Office

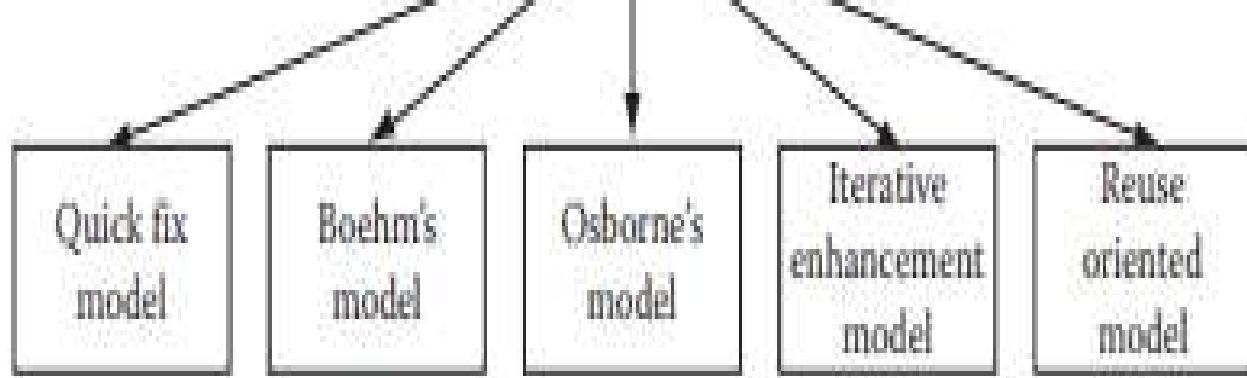


Edit with WPS Office

- Boehm's model
- Osborne's model
- Iterative enhancement model
- Reuse oriented model



Edit with WPS Office



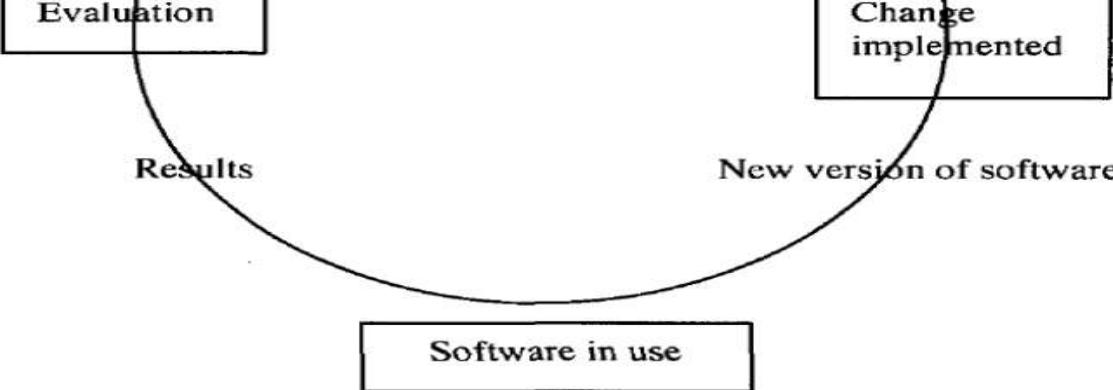
Edit with WPS Office

- It is easy to see how the model emerged historically, but it cannot be dismissed as a purely historical curiosity because, like the code-and-fix model, it is still used.
- What are the advantages of such a model and why is it still used?



Edit with WPS Office

- In the appropriate environment it can work perfectly well.



Edit with WPS Office

applying particular strategies and cost-benefit evaluations to a set of proposed changes.

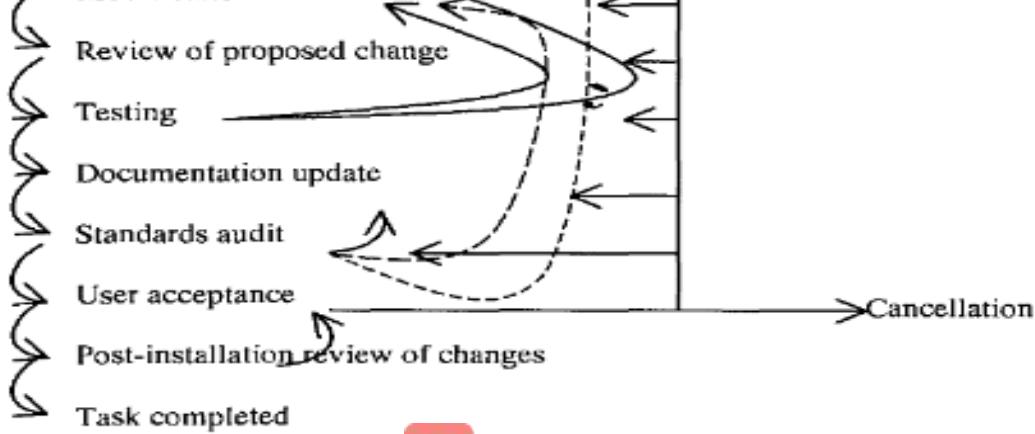
- The approved changes are accompanied by their own budgets which will largely determine the extent and type of resource expended



Edit with WPS Office

- Thus, the maintenance process is driven by the maintenance manager's decisions which are based on the balancing of objectives against constraints.





Edit with WPS Office

specification;

- a software quality assurance program which establishes quality assurance requirements;
- a means of verifying that maintenance goals have been met;
- performance review to provide feedback to managers.



Edit with WPS Office

Redesign
Current Version
& Implement

Characterize
Proposed
Modifications



Existing documentation for each stage is:

- Requirement documentation, Design Documentation, Source code documentation, & Test documentation

Test

Analysis

Test

Analysis

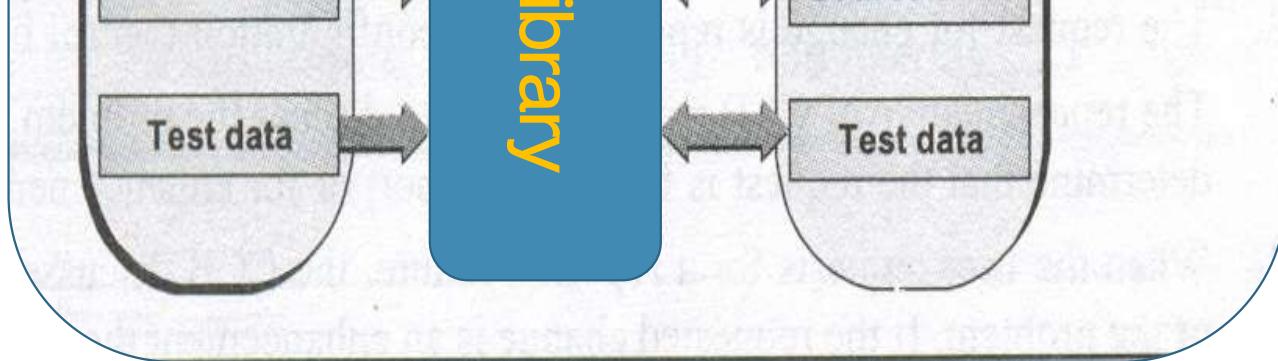


Edit with WPS Office

- A detailed framework is required for classification of components for reuse.



Edit with WPS Office



Edit with WPS Office

- Modifying the old system parts according to the new requirements, and
- Integrating the modified parts into the new system.



Edit with WPS Office

- Either the list of defects can be taken as a whole or a subset of defects from this list can be taken for a fixing plan.
- It makes a lot of sense to go for an iterative approach.
- This approach is similar to the concept of iterative

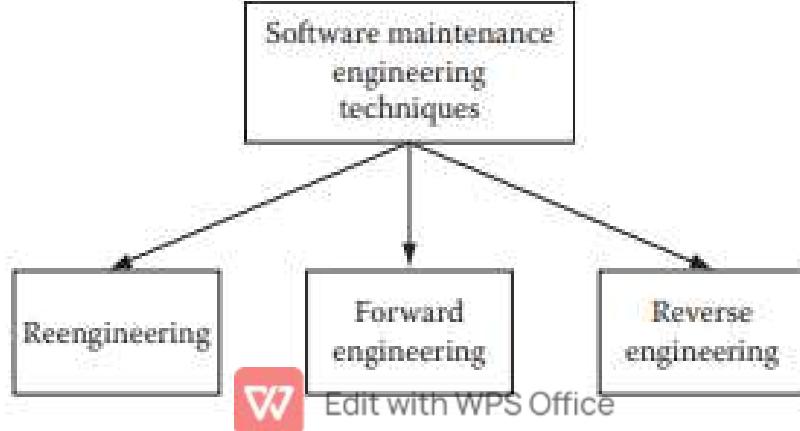


Edit with WPS Office

- This phase also consumes a lot of time and effort.
- But the value addition in all this effort and time spent helps in reducing defects, which in the long run is a much cheaper alternative compared to no testing/cursory testing and later spending money in providing support.



Edit with WPS Office



constructed.

- Sometimes there is no documentation at all.
- Even if documentation is there, it is not up to date.



Edit with WPS Office

- All these situations call for some specific techniques for maintenance work depending on the situation.
- Some of the common maintenance techniques include reengineering, reverse engineering, and forward engineering.

- When any maintenance work is needed, from the list of defects, each defect is specifically analyzed to find out the root cause of the defect.
- Once this analysis is successful, then fixing that defect becomes easy.



Edit with WPS Office

- In such cases, the reverse engineering technique is adopted.
- Using this technique, similar components or product parts are constructed as compared to existing product components/parts.
- This way the software product functionality is changed as the new constructed parts will have the desired functionality.



Edited with WPS Office

- Due to new customer needs, the existing product needs to be extended so that the new needs can be fulfilled.
- All new extended development is based on the existing design and construction methods and will be made for the same



Edit with WPS Office

Navigation icons: back, forward, search, etc.

The high priority features will be definitely added and the low priority features for that iteration will be added if any time remains in the iteration.

- Our SaaS vendor does not release alpha or beta releases of its product as they do not serve mass markets.



Edit with WPS Office

New version is tested thoroughly by their testing team, and no major

defects are passed in the production instances.

- Since there are no immediate customers who will be available for doing user acceptance testing, the internal testing team does the user acceptance testing as well.



Edit with WPS Office

maintenance programmers have moved to a different company. In order to take advantage of state-of-the-art parallel machines, the contractor wants the software to be reimplemented on a parallel platform.

- Briefly describe the techniques that will be needed to accomplish the task.
- How would you go about performing the job, bearing in mind the merits of software reuse?



Edit with WPS Office

applications that previously ran on Solaris 1.x had to be modified in order to use Solaris 2.x. This also meant the users had to retrain and learn the use of new commands. Some administrative practices became out of date as a result of the upgrade.

The end result was a more efficient and cost-effective system but the cost of accommodating the upgrade went well beyond the retail price of the new software.



Edit with WPS Office

Later modifications to produce modular code would address the latter problem.

Software engineering point: encapsulation of parts of the code means that execution of a particular part of a program cannot accidentally modify variables that have no relevance to that section of code.

A major step forward is to take the data out of the programs altogether, to store it in external tables or files. VAT upgrades can now be carried out by providing new data files and the programs need not be modified.

Software engineering point: proper separation of data from code avoids unnecessary code modification.

VAT rates and the ~~eligibility~~ of different goods in different contexts are in fact nothing to do with system developers and maintainers. They are set and amended by Government bodies. Even

should access central data sources e.g. a central VAT server, which is essentially a black box that takes in information from the program and returns a current rate.

Software engineering point: true interoperability between software systems using properly researched and formulated interfaces is the route to real flexibility and is a quantum leap forward in building maintainable systems.



Edit with WPS Office

current quantity and safety regulations, the design could then be approved and construction of the airbag finally commissioned.

Because of the tendency to treat software change in a less formal way, the software "airbag" will be bolted onto the car with no regard to safety considerations or appropriate design. Issues of safety, correct placing, and how other components are affected are unlikely to be considered until problems  with the bolted-on version arise.



Edit with WPS Office