K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109
FIRST INTERNAL TEST QUESTION PAPER 2023-24 EVEN SEMESTER

SET: A

USN			
Degree :	B.E	Semester :	VI
Branch :	Computer Science & Design	Course Code :	21CS61
Course Title :	Software Engineering and Project Management	Date :	27/05/24
Duration :	1 Hr (60 Minutes)	Max Marks :	20

Note: Answer **ONE full** question from each Module.

K-Levels: K1-Remebering, K2-Understanding, K3-Applying, K4-Analyzing, K5-Evaluating, K6-Creating

Q No.	Question	Marks	CO	K-Level
	Module 1			
1(a)	**Identify** the nature of Software.With a neat graphical representation compare failure rate of software with failure rate of Hardware.	8	CO1	K3
(b)	**Obtain** the definition of Software. Also explain software engineering.	4	CO1	K3
	OR			
2(a)	**Make use of** neat diagram and explain the evolutionary protype model of software development process.	8	CO1	K3
(b)	**Make use of** neat diagram and explain the spiral model of software development process.	4	CO1	K3
	Module 2			
3(a)	**Identify** requirement engineering tasks and explain the same.	8	CO2	K3
	OR			
4(a)	**Identify** the meaning of Requirement elicitation in requirement engineering.Obtain UML activity diagram of Safe home application for eliciting requirements.	8	CO2	K3

Sd (SUSHMA A
Name & Signature of Course In charge:

[illegible] Dr. S Welcha
Name & Signature of Module Coordinator:

Deepa
HOD

Principal [illegible]

Selected.

K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109
I INTERNAL ASSESSMENT 2023 – 24 EVEN SEMESTER

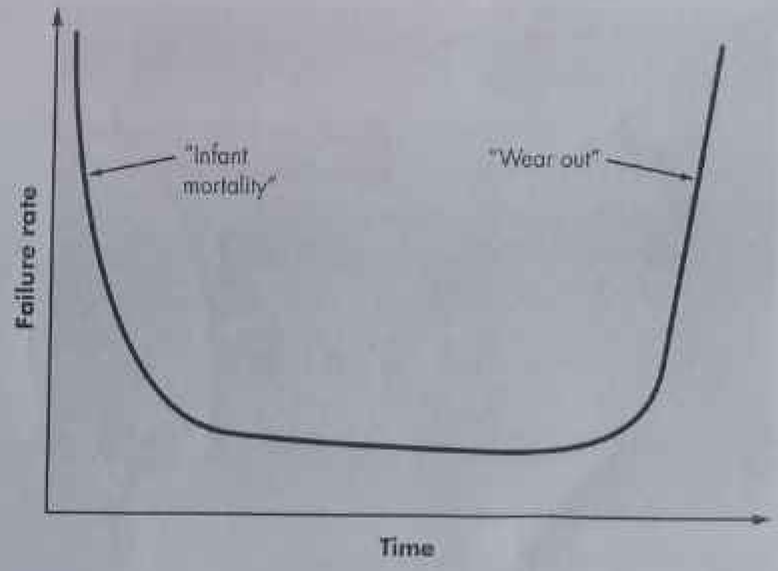
SCHEME AND SOLUTION

SET-A

Degree	:	B.E	Semester :	VI
Branch	:	COMPUTER SCIENCE AND DESIGN	Course Code :	21CS61
Course Title	:	SOFTWARE ENGINEERING AND PROJECT MANAGEMENT	Max Marks :	20

Q No.	POINTS	Marks
1(a)	Software is developed or engineered, it is not manufactured in the classical sense. Although some similarities exist between software development and hardware manufacturing, the two activities are fundamentally different. In both activities, high quality is achieved through good design, but the manufacturing phase for hardware can introduce quality problems that are nonexistent for software. Both activities require the construction of a "product," but the approaches are different Software doesn't "wear out." ■ Indicates that hardware exhibits relatively high failure rates early in its life (these failures are often attributable to design or manufacturing defects). ■ Defects are corrected and the failure rate drops to a steady-state level (hopefully, quite low) for some period of time. ■ As time passes, however, the failure rate rises again as hardware components suffer from the cumulative effects of dust, vibration, abuse, temperature extremes, and many other environmental maladies. ■ Following fig depicts failure rate as a function of time for hardware. The relationship, often called the "bathtub curve	2.5M+ 2.5M

FIGURE 1.1
Failure curve for hardware

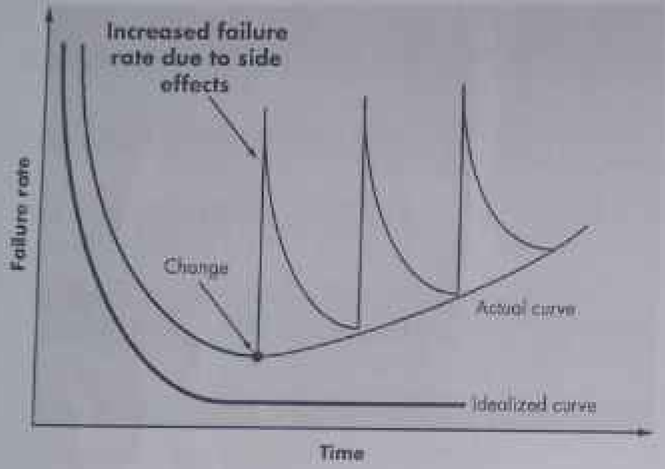


■

- ✓ Software is not susceptible to the environmental maladies that cause hardware to wear out.
- ✓ In theory, therefore, the failure rate curve for software
- ✓ should take the form of the "idealized curve" hown in the below fig.
- ✓ Undiscovered defects will cause high failure rates early in the life of a program.
- ✓ However, these are corrected and the curve flattens as shown in the below fig.

During its life,software will undergo change. As changes are made, it is likely that errors will be introduced, causing the failure rate curve to spike as shown in the "actual curve" as shown in the below fig .

+3M



1(5) Definition of Software and Software engineering — 2M + 2M

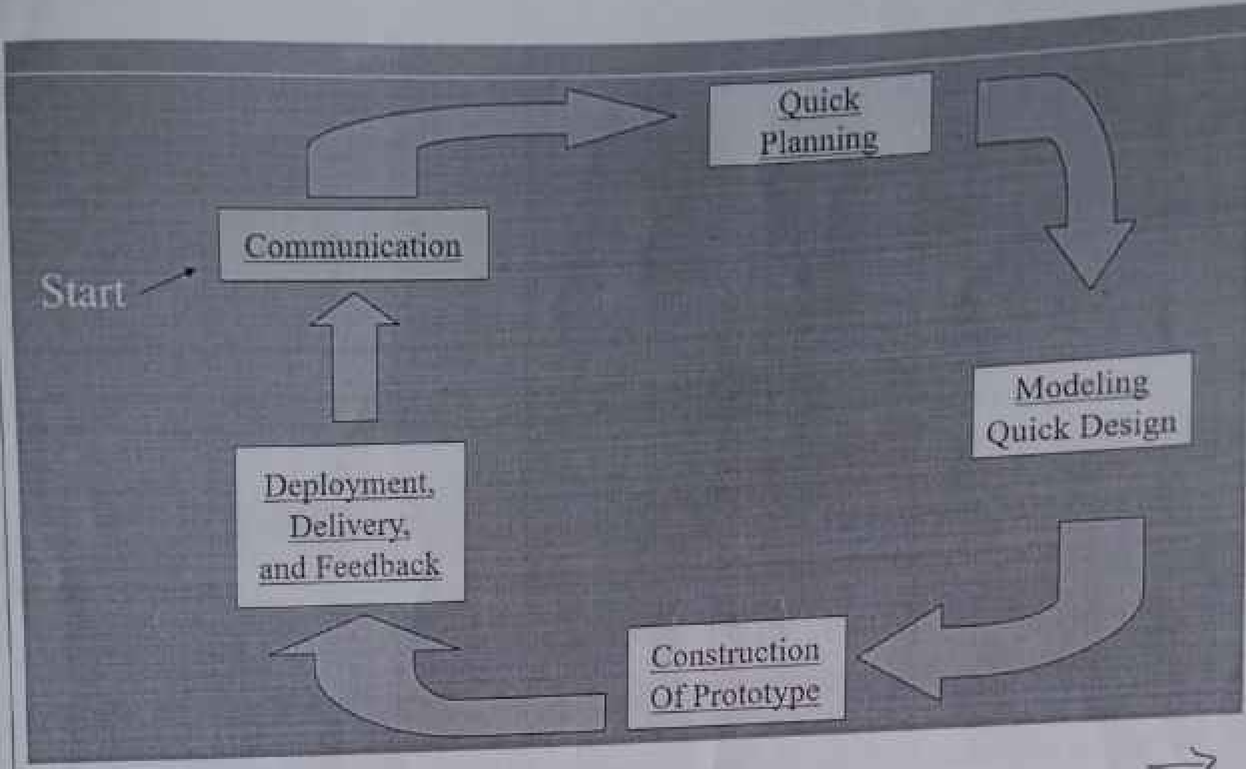
- ✧ Software engineering is an engineering discipline that is concerned with all aspects of software production from the early stages of system specification through to maintaining the system after it has gone into use.
- ✧ Engineering discipline
  - Using appropriate theories and methods to solve problems bearing in mind organizational and financial constraints.
- ✧ All aspects of software production
  - Not just technical process of development. Also project management and the development of tools, methods etc. to support software production.

(OR)

2(a) **Prototyping**

Model

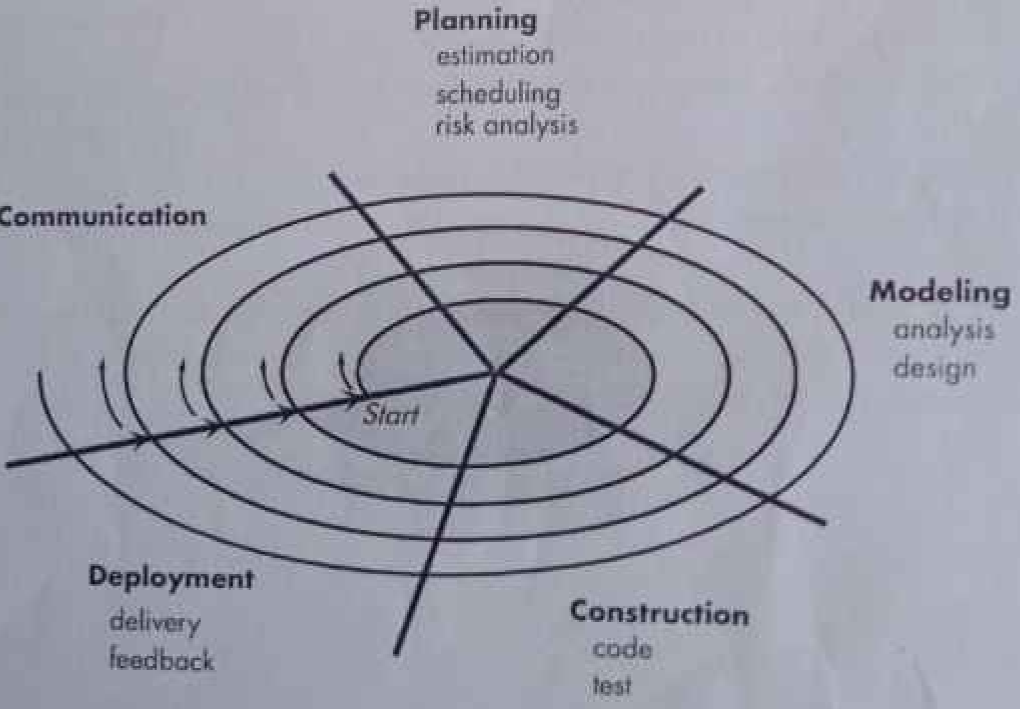
(Diagram) Prototyping is defined as the process of developing a working replication of a product or system that has to be engineered



→

Explanation of PROTYPE MODEL .

9(5) Spiral model and its explanation.



Module 2

3(a) Seven distinct tasks of Requirement ngineeing

Inception
Elicitation
Elaboration
Negotiation
Specification
Validation
Requirements Management

2m

Some of these tasks may occur in parallel and all are adapted to the needs of the project
All strive to define what the customer wants
All serve to establish a solid foundation for the design and construction of the software

Explanation of Each task. ⟶ -6 =8

(OR)

4(a) **Eliciting requirements is difficult because of**

- **Problems of scope → identify the boundaries of the system.**
- **Problems of understanding → domain , computing environment.**
- **Problems of Volatility → requirements may change over time.**

Elicitation may be accomplished through two activities:

- **Collaborative Requirements Gathering**

Quality Function Deployment.

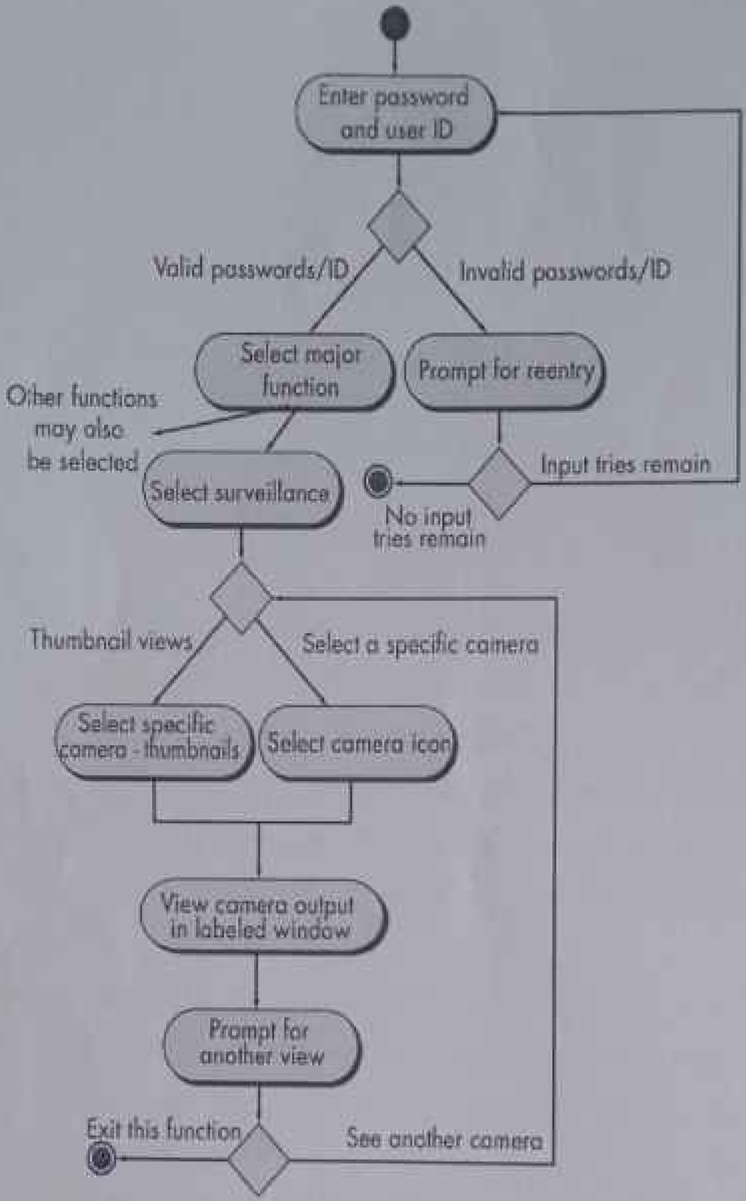
Elicitation Work Products

- A statement of **need and feasibility**
- A bounded statement of **scope** for the system or product
- A list of customers, users, and other **stakeholders** who participated in requirements elicitation
- A description of the **system's technical environment**
- **A list of requirements** (organized by function) and the domain constraints that apply to each
- A set of preliminary **usage scenarios** (in the form of use cases) that provide insight into the use of the system or product under different operating conditions
- Any **prototypes** developed to better define requirements

5
4

FIGURE 6.5

Activity diagram for Access camera surveillance via the Internet—display camera views function.



K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109
FIRST INTERNAL TEST QUESTION PAPER 2023-24 EVEN SEMESTER

SET: B

USN										

Degree	: B.E	Semester	:	VI
Branch	: Computer Science & Design	Course Code	:	21CS61
Course Title	: Software Engineering and Project Management	Date	:	27/05/24
Duration	: 1 Hr (60 Minutes)	Max Marks	:	20

Note: Answer **ONE full** question from each Module.
K-Levels: K1-Remebering, K2-Understanding, K3-Applying, K4-Analyzing, K5-Evaluating, K6-Creating

Q No.	Questions	Marks	CO	K-Level
	Module 1			
1(a)	**Make use of** neat diagram and explain the Incremental model of software development process.	8	CO1	K3
(b)	**Identify** broad categories of software application domains.	4	CO1	K3
	OR			
2(a)	**Make use of** neat diagram and explain the waterfall model of software development process.	8	CO1	K3
(b)	**Model** the general software process(process Framework) with a neat diagram.	4	CO1	K3
	Module 2			
3(a)	**Identify** requirement engineering tasks and explain the same.	8	CO2	K3
	OR			
4(a)	**Obtain** the use case UML diagram for the development of SafeHome control panel and also elaborate detailed description of usecase.	8	CO2	K3

[illegible] (SUSHMA A)
Name & Signature of Course In charge:

Dr. [illegible]
Name & Signature of Module Coordinator:

[illegible]
HOD

[illegible]
Principal

K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109
I INTERNAL ASSESSMENT 2023 – 24 EVEN SEMESTER

SCHEME AND SOLUTION

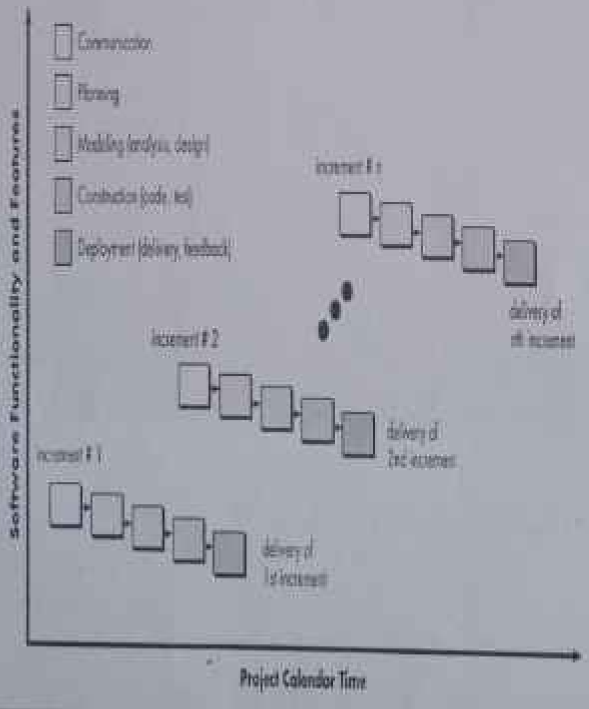
SET-B

Degree	: B.E	Semester :	VI
Branch	: COMPUTER SCIENCE AND DESIGN	Course Code :	21CS61
Course Title	: SOFTWARE ENGINEERING AND PROJECT MANAGEMENT	Max Marks :	20

Q No.	POINTS	Marks
	Module 1	
1(a)	▪ Used when requirements are well understood ▪ Multiple independent deliveries are identified ▪ Each linear sequence produces deliverable "increments" of the software ▪ Work flow is in a linear (i.e., sequential) fashion <u>within</u> an increment and is staggered <u>between</u> increments ▪ Iterative in nature; focuses on an operational product with each increment ▪ The incremental model combines elements of linear and parallel process flows	2.5M+ 2.5M + 3M =8M

FIGURE 2.5

The incremental model


Communication
Planning
Modeling (analysis, design)
Construction (code, test)
Deployment (delivery, feedback)
increment # n
delivery of nth increment
increment # 2
delivery of 2nd increment
increment # 1
delivery of 1st increment
Software Functionality and Features
Project Calendar Time

▪

1. As a result of use and/or evaluation, a plan is developed for the next increment.
2. The plan addresses the modification of the core product to better meet the needs of the customer and the delivery of additional features and functionality.
3. This process is repeated following the delivery of each increment, until the complete

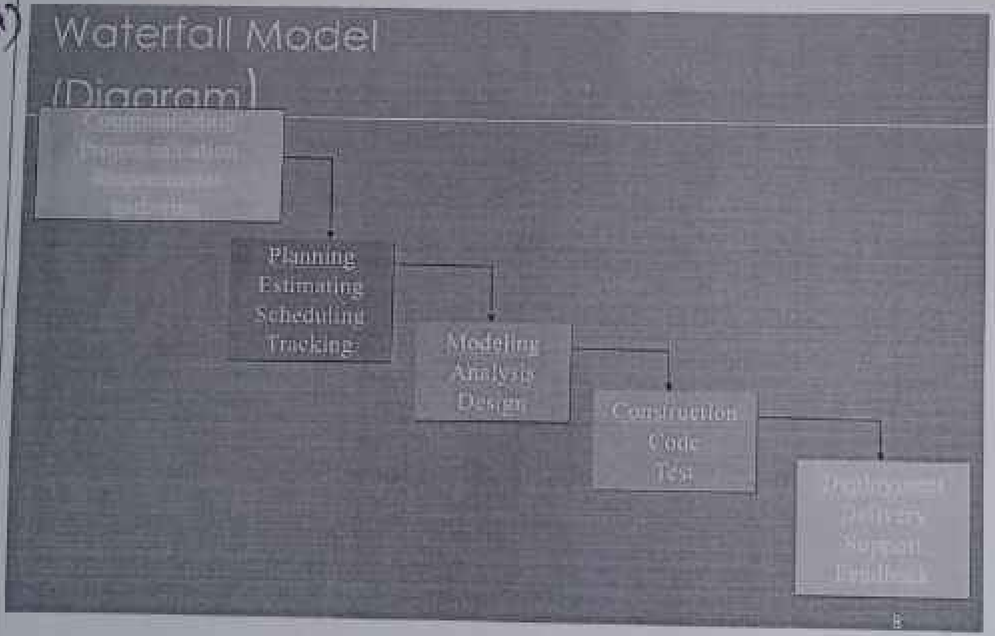
4. The incremental process model focuses on the delivery of an operational product with each increment

1(b)
- system software
- application software
- engineering/scientific software
- embedded software
- product-line software
- WebApps (Web applications)
- AI software

1M +

Explanation of each application domains → 3M

2(a)



3M +

WATER FALL MODEL AND EXPLAINATION → 5M

2(b) Communication

Involves communication among the customer and other stake holders; encompasses requirements gathering

Planning

Establishes a plan for software engineering work; addresses technical tasks, resources, work products, and work schedule

Modelling (Analysis, Design)

Construction (Code, Test)
Combines code generation and testing to uncover errors
Deployment
Involves delivery of software to the customer for evaluation and feedback

- Milestone can be defined as recognizable endpoint of software project activity. At each milestone, report must be generate,
- Software Quality Assurance is a kind of Umbrella activity that is applied throughout the software process.

4M

A process is a collection of activities, actions, and tasks that are performed when some work product is to be created to reach goal.

A process framework establishes the foundation for a complete software engineering process by identifying a small number of framework activities that are applicable to all software projects, regardless of their size or complexity.

In addition, the process framework encompasses a set of umbrella activities

Module 2

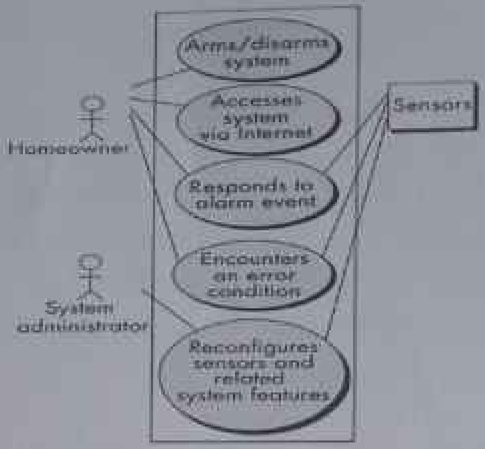
3(a) Seven distinct tasks
Inception
Elicitation
Elaboration
Negotiation
Specification
Validation
Requirements Management

Some of these tasks may occur in parallel and all are adapted to the needs of the project
All strive to define what the customer wants
All serve to establish a solid foundation for the design and construction of the software

Explanation of Each task.

(OK) $\longrightarrow$ 6

4(a) Use case: InitiateMonitoring
Primary actor: Homeowner.
Goal in context: To set the system to monitor sensors when the homeowner leaves the house or remains inside.
Preconditions: System has been programmed for a password and to recognize various sensors.
Trigger: The homeowner decides to "set" the system, i.e., to turn on the alarm functions.

	FIGURE 5.2 **UML use case diagram for *SafeHome* home security function**  Arms/disarms system Accesses system via Internet Responds to alarm event Encounters an error condition Reconfigures sensors and related system features Homeowner System administrator Sensors Detailed description of Usecase diagram	4M

[illegible] (SUSHMA A)
COURSE INCHARGE

[illegible]
HOD

K.S. INSTITUTE OF TECHNOLOGY, BENGALURU - 560109
SECOND INTERNAL TEST QUESTION PAPER 2023-24 EVEN SEMESTER

KSIT

SET: A

USN | | | | | | | | | |

Degree	:	B.E	Semester	:	VI
Branch - Stream	:	Computer Science & Design	Course Type / Code	:	21CS61
Course Title	:	Software Engineering & Project Management	Date	:	27/06/24
Duration	:	1 Hr (60 minutes)	Max Marks	:	20

Note: Answer **ONE full** question from each Module.
K-Levels: K1-Remebering, K2-Understanding, K3-Applying, K4-Analyzing, K5-Evaluating, K6-Creating

Q No.	Questions	Marks	CO	K-Level
	Module2			
1(a)	**Obtain** the definition of Requirement Analysis. Also discuss the types of Requirement analysis models.	4	CO2	K3
	OR			
2(a)	**Identify** Requirement analysis rules of thumb.	4	CO2	K3
	Module3			
3(a)	**Obtain** the meaning of an Agile process. Outline an analysis of agility and compare the cost of change with traditional software process models.	7	CO3	K3
(b)	**Identify** Agility principles proposed by the Agile agency.	5		K3
	OR			
4(a)	**Design** Scrum process flow and using a neat pictorial representation explain Scrum Agile process model.	7	CO3	K3
(b)	**Model** the Dynamic agile system development with a neat diagram.	5		K3
	Module 4			
5(a)	**Obtain** activities covered by Project management team.	4	CO4	K3
	OR			
6(a)	**Identify** the categories of software projects and brief them.	4	CO4	K3

[illegible] (SUSHMA A)
Name & Signature of Course In charge:

Deepa . S . R
Deepa
Name & Signature of Module Coordinator:

Deepa
HOD

[illegible]
Principal
Selected

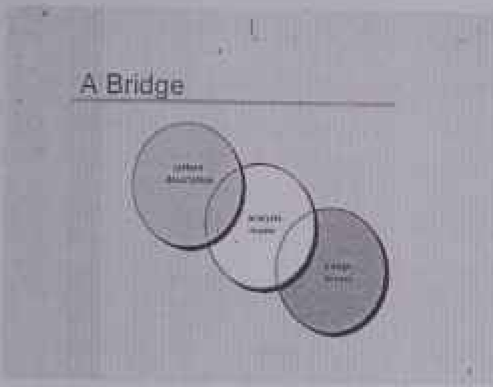
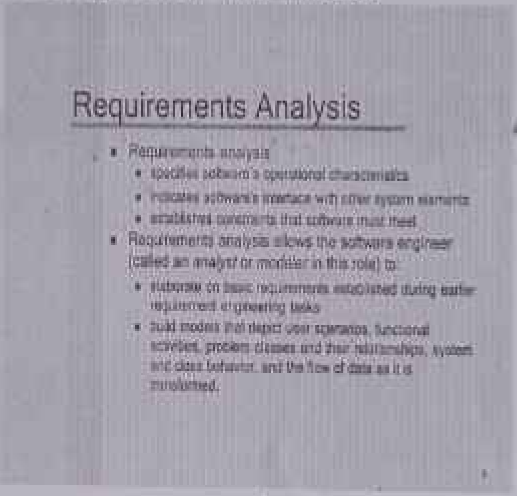
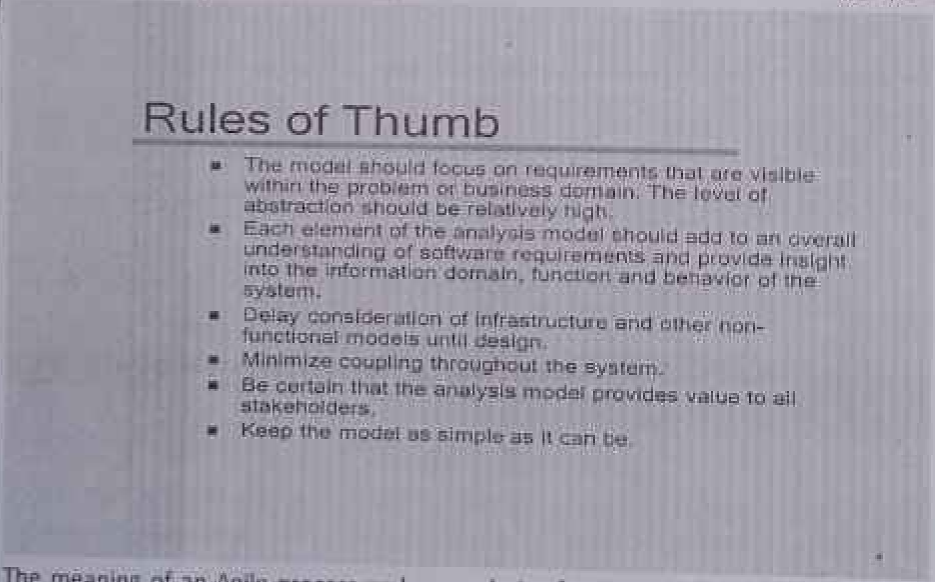
70

K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109
II INTERNAL ASSESSMENT 2023 – 24 EVEN SEMESTER

SCHEME AND SOLUTION

SET-A

Degree	: B.E	Semester :	VI
Branch	: COMPUTER SCIENCE AND DESIGN	Course Code :	21CS61
Course Title	: SOFTWARE ENGINEERING AND PROJECT MANAGEMENT	Max Marks :	20

Q No.	POINTS	Marks
	Module 3	
1(a)	Definition of Requirement Analysis. Also the types of Requirement analysis models  A Bridge  Requirements Analysis • Requirements analysis ◦ specifies software's operational characteristics ◦ indicates software's interface with other system elements ◦ establishes constraints that software must meet • Requirements analysis allows the software engineer (called an analyst or modeler in this role) to: ◦ elaborate on basic requirements established during earlier requirement engineering tasks ◦ build models that depict user scenarios, functional activities, problem classes and their relationships, system and class behavior, and the flow of data as it is transformed.	4M
2(a)	Requirement analysis rules of thumb. with Explanation  Rules of Thumb • The model should focus on requirements that are visible within the problem or business domain. The level of abstraction should be relatively high. • Each element of the analysis model should add to an overall understanding of software requirements and provide insight into the information domain, function and behavior of the system. • Delay consideration of infrastructure and other non-functional models until design. • Minimize coupling throughout the system. • Be certain that the analysis model provides value to all stakeholders. • Keep the model as simple as it can be.	4M
3(a)	The meaning of an Agile process and an analysis of agility and the cost of change with traditional software process models.	

Scrum process flow AND EXPLAINATION

4(a) Pictorial Representation → Explaination

5M + 3M

Scrum

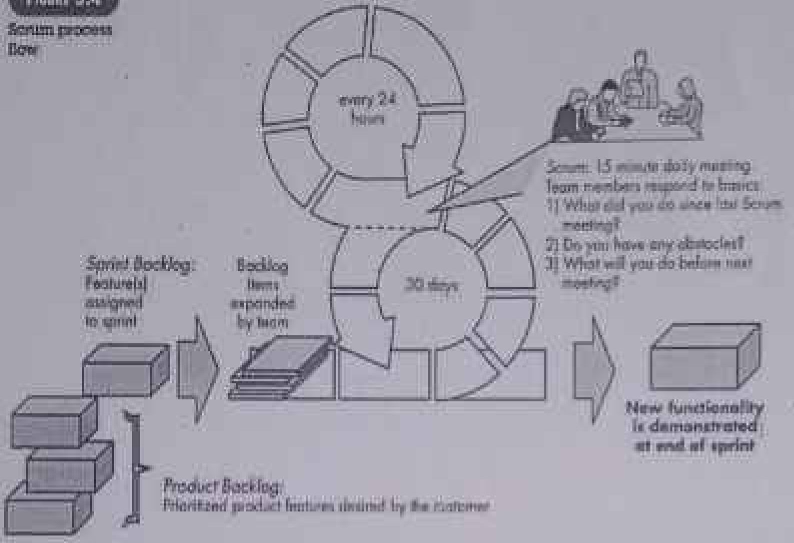
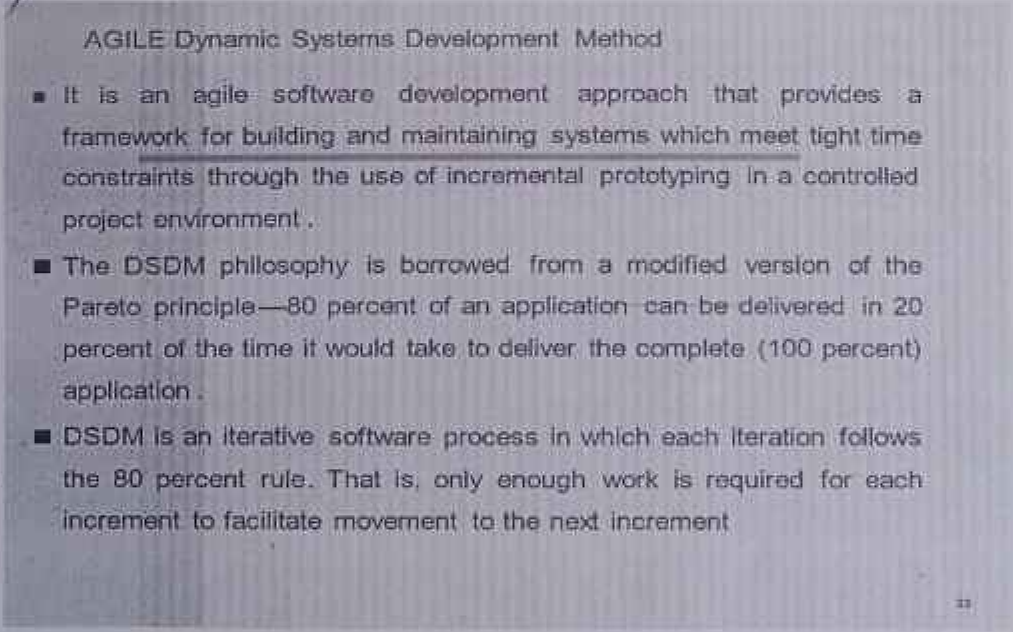

every 24 hours
Scrum: 15 minute daily meeting
Team members respond to basics:
1) What did you do since last Scrum meeting?
2) Do you have any obstacles?
3) What will you do before next meeting?
Sprint Backlog: Feature(s) assigned to sprint
Backlog items expanded by team
30 days
New functionality is demonstrated at end of sprint
Product Backlog: Prioritized product features desired by the customer

Figure 3.4 Scrum process flow

The Dynamic agile system development with a neat diagram.

4(b)

7M



AGILE Dynamic Systems Development Method

- It is an agile software development approach that provides a framework for building and maintaining systems which meet tight time constraints through the use of incremental prototyping in a controlled project environment .
- The DSDM philosophy is borrowed from a modified version of the Pareto principle—80 percent of an application can be delivered in 20 percent of the time it would take to deliver the complete (100 percent) application .
- DSDM is an iterative software process in which each iteration follows the 80 percent rule. That is, only enough work is required for each increment to facilitate movement to the next increment

- The remaining detail can be completed later when more business requirements are known or changes have been requested and accommodated.

the DSDM life cycle that defines three different iterative cycles, preceded by two additional life cycle activities:

Feasibility study—establishes the basic business requirements and constraints associated with the application to be built and then assesses whether the application is a viable candidate for the DSDM process.

Business study—establishes the functional and information requirements that will allow the application to provide business value

Functional model iteration—produces a set of incremental prototypes that demonstrate functionality for the customer.

Activities covered by Project management team

5(a)

Activities covered by project management

Feasibilitystudy

Isprojecttechnicallyfeasible and worthwhilefrom a businespoint of view?

Planning

Only done if projectis feasible

Execution

Implemenplan, but plan may be changedas we go along

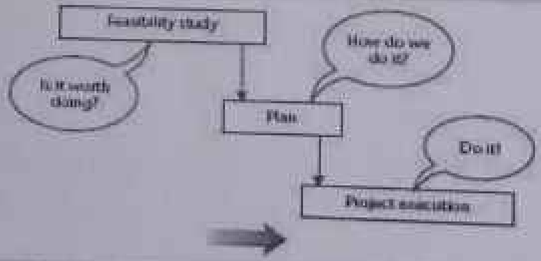


Figure 1.2 The feasibility study/plan/execution cycle

Pictorial representation

Explaination

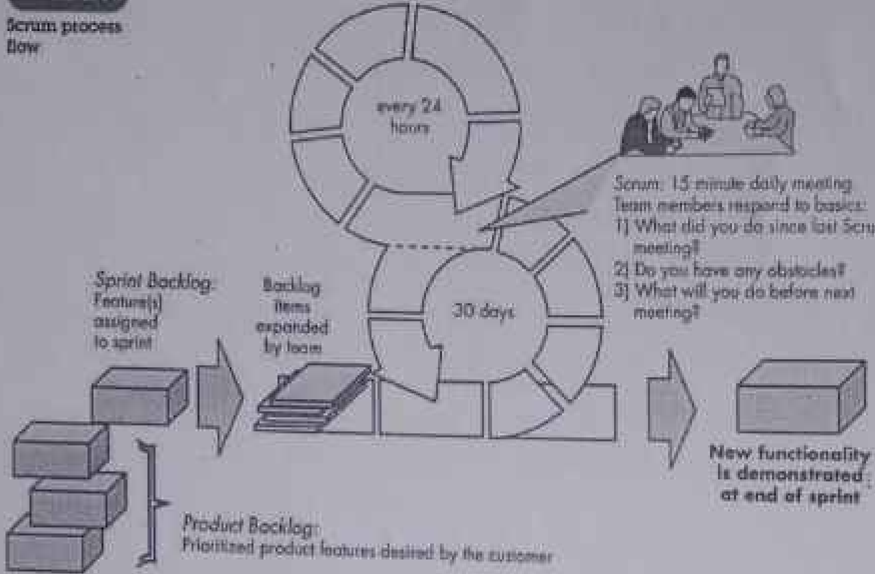
Scrum process flow AND EXPLAINATION

4(a) Pictorial Representation → Explaination

Scrum

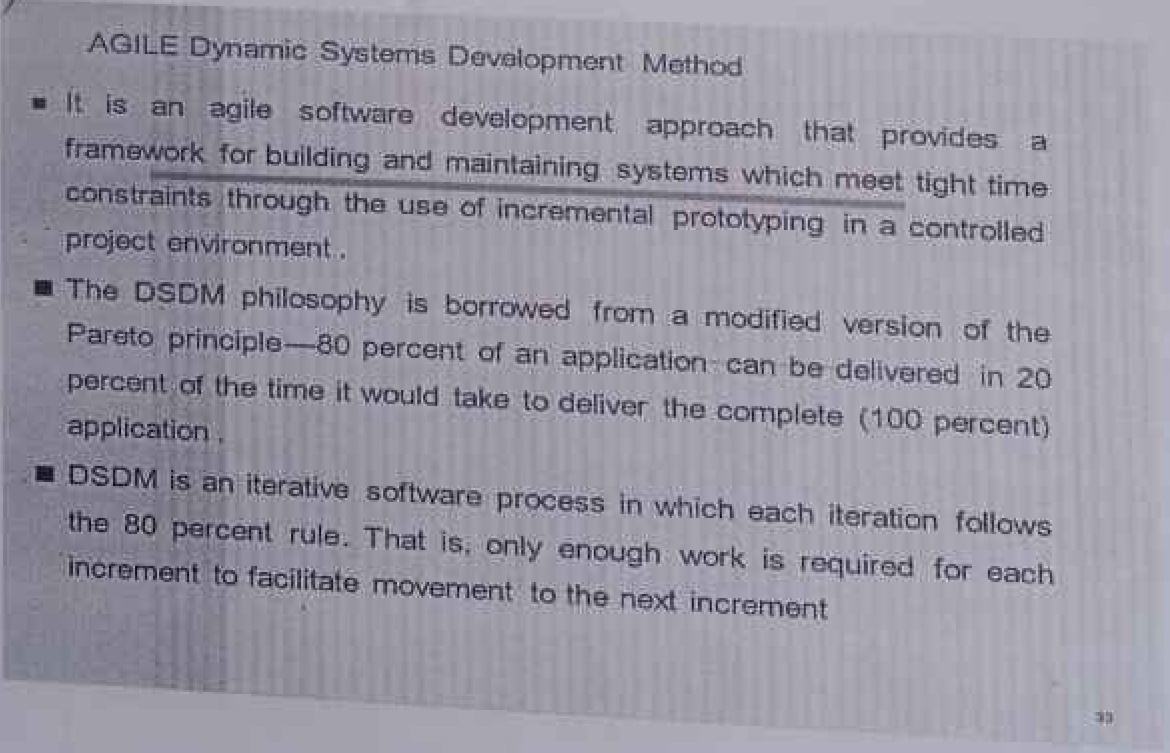
FIGURE 3.4

Scrum process flow


every 24 hours
30 days
Scrum: 15 minute daily meeting
Team members respond to basics:
1) What did you do since last Scrum meeting?
2) Do you have any obstacles?
3) What will you do before next meeting?
Sprint Backlog: Feature(s) assigned to sprint
Backlog items expanded by team
New functionality is demonstrated at end of sprint
Product Backlog: Prioritized product features desired by the customer

The Dynamic agile system development with a neat diagram.

4(b)



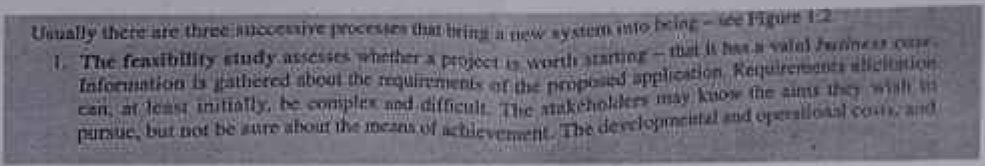
AGILE Dynamic Systems Development Method

- It is an agile software development approach that provides a framework for building and maintaining systems which meet tight time constraints through the use of incremental prototyping in a controlled project environment.
- The DSDM philosophy is borrowed from a modified version of the Pareto principle—80 percent of an application can be delivered in 20 percent of the time it would take to deliver the complete (100 percent) application.
- DSDM is an iterative software process in which each iteration follows the 80 percent rule. That is, only enough work is required for each increment to facilitate movement to the next increment

33

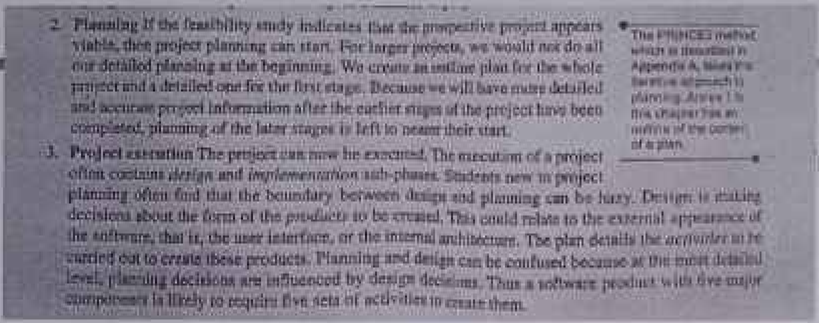
A software is not only concerned with the actual writing of software in fact where a software application is BOUGHT off the shelf there may be no software writing as such but this is still fundamental software project because so many of the other activitIES associates with software will still be present

For larger project detail planning will not be done at the beginning and an outline plan for the whole project is done and a detailed plan for the first stage



Usually there are three successive processes that bring a new system into being – see Figure 1.2.

1. **The feasibility study** assesses whether a project is worth starting – that it has a valid *business case*. Information is gathered about the requirements of the proposed application. Requirements elicitation can, at least initially, be complex and difficult. The stakeholders may know the aims they wish to pursue, but not be sure about the means of achievement. The developmental and operational costs, and

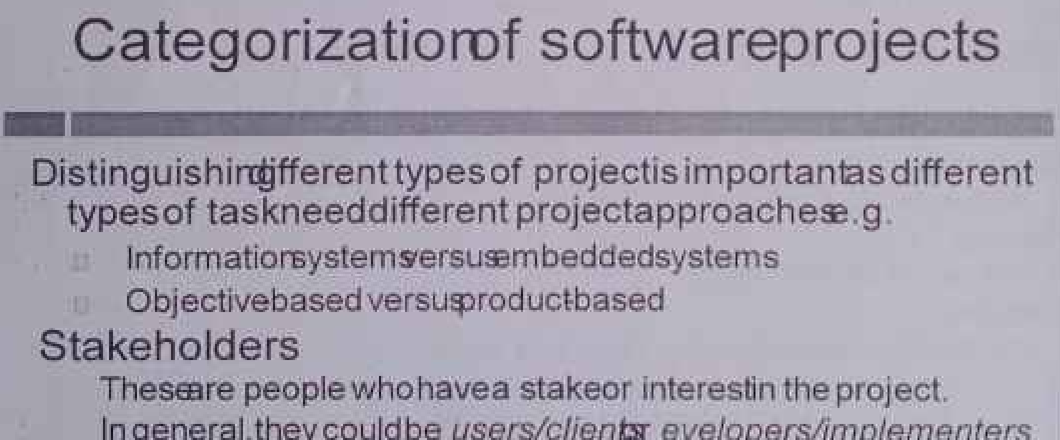


2. **Planning** If the feasibility study indicates that the prospective project appears viable, then project planning can start. For larger projects, we would not do all our detailed planning at the beginning. We create an outline plan for the whole project and a detailed one for the first stage. Because we will have more detailed and accurate project information after the earlier stages of the project have been completed, planning of the later stages is left to nearer their start.

The PRINCE2 method, which is described in Appendix A, takes an iterative approach to planning. Annex 1 to this chapter has an outline of the content of a plan.

3. **Project execution** The project can now be executed. The execution of a project often contains *design* and *implementation* sub-phases. Students new to project planning often find that the boundary between design and planning can be hazy. Design is making decisions about the form of the *products* to be created. This could relate to the external appearance of the software, that is, the user interface, or the internal architecture. The plan details the *activities* to be carried out to create these products. Planning and design can be confused because at the most detailed level, planning decisions are influenced by design decisions. Thus a software product with five major components is likely to require five sets of activities to create them.

The categories of software projects and brief them



Categorizationof softwareprojects

Distinguishingdifferent typesof projectis importantasdifferent typesof taskneeddifferent projectapproachese.g.

- Informationsystemsversusembeddedsystems
- Objectivebased versusproductbased

Stakeholders

Theseare people whohavea stakeor interestin the project.
In general,they couldbe *users/clients*or *evelopers/implementers*
Theycouldbe:

- Withinthe projectteam
- Outsidethe projectteam,but withinthe sameorganization
- Outsideboth the projectteamand the organization

[illegible] (SUSHMA A).
COURSE INCHARGE

[illegible]
HOD

K.S. INSTITUTE OF TECHNOLOGY, BENGALURU - 560109

SECOND INTERNAL TEST QUESTION PAPER 2023-24 EVEN SEMESTER

SET: B

USN										

Degree	: B.E		Semester	: VI
Branch - Stream	: Computer Science & Design		Course Type / Code	: 21CS61
Course Title	: Software Engineering & Project Management		Date	: 27/06/24
Duration	: 1 Hr (60 minutes)		Max Marks	: 20

Note: Answer **ONE full** question from each Module.

K-Levels: K1-Remebering, K2-Understanding, K3-Applying, K4-Analyzing, K5-Evaluating, K6-Creating

Q No.	Questions	Marks	CO	K-Level
	Module2			
1(a)	**Obtain** the concepts of data modeling with an example each.	4	CO2	K3
	OR			
2(a)	**Identify** requirement modelling approaches with neat diagram and explain each of them.	4	CO2	K3
	Module3			
3(a)	**Model** the development method of extreme programming (XP) in Agile. With a need to diagram, summarize XP values.	7	CO3	K3
(b)	**Obtain** key Human traits required to exist among the Agile software development team members.	5		K3
	OR			
4(a)	**Identify** philosophy of Agile Adaptive software development process model and discuss with a neat diagram.	7	CO3	K3
(b)	**Model** Feature driven agile system development with a neat diagram.	5		K3
	Module 4			
5(a)	**Obtain** the definition of a Project in software engineering and explain The ISO 12207 standard software development life cycle with a neat diagram.	4	CO4	K3
	OR			
6(a)	**Obtain** activities covered by project management team.	4	CO4	K3

[illegible] (SUSHMA [illegible])
Name & Signature of Course In charge:

Deepa. S.R
Deepa
Name & Signature of Module Coordinator:

Deepa
HOD

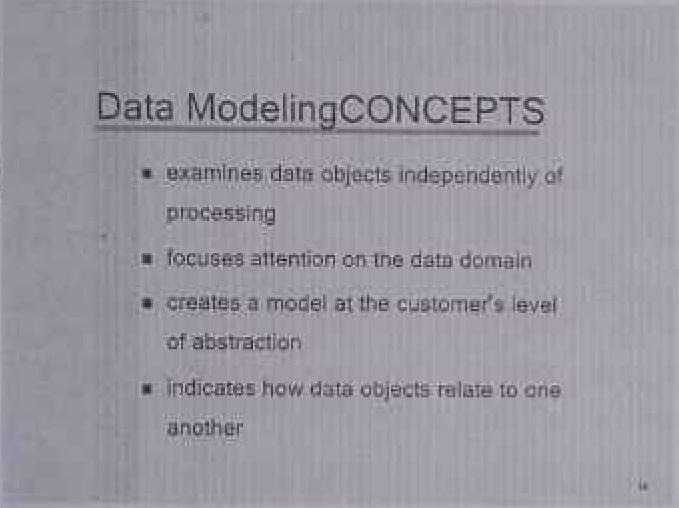
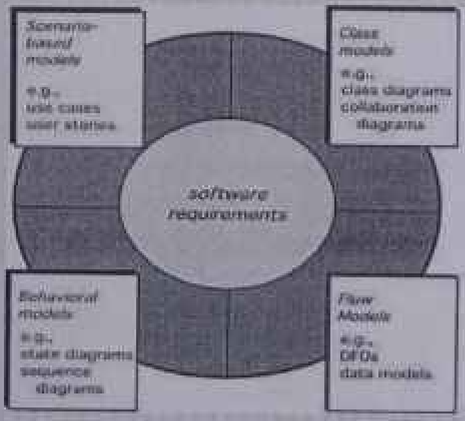
[illegible]
Principal

K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109
II INTERNAL ASSESSMENT 2023 – 24 EVEN SEMESTER

SCHEME AND SOLUTION

SET-B

Degree	: B.E	**Semester** :	VI
Branch	: COMPUTER SCIENCE AND DESIGN	**Course Code** :	21CS61
Course Title	: SOFTWARE ENGINEERING AND PROJECT MANAGEMENT	**Max Marks** :	20

Q No.	POINTS	Marks
	Module 3	
1(a)	The concepts of data modeling with an example each. Explanation of following data modeling concepts  Data ModelingCONCEPTS ▪ examines data objects independently of processing ▪ focuses attention on the data domain ▪ creates a model at the customer's level of abstraction ▪ indicates how data objects relate to one another Explaination of Concepts	4m
2(a)	Requirement modelling approaches with neat diagram Explaination of all modelling approaches Elements of Requirements Analysis  These slides are designed to accompany Software Engineering: A Practitioner's Approach, 7/e (McGraw-Hill, 2009). Slides copyright 2009 by Roger Pressman.	4M

Scenario-based models
e.g., use cases user stories
Class models
e.g., class diagrams collaboration diagrams
software requirements
Behavioral models
e.g., state diagrams sequence diagrams
Flow Models
e.g., DFDs data models

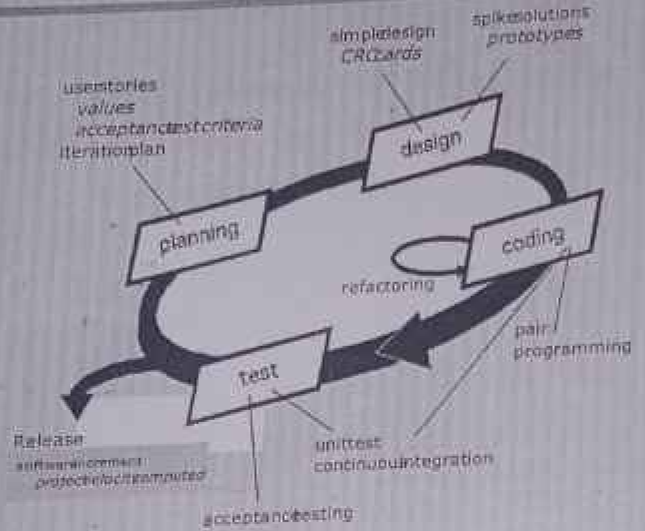
The development method of extreme programming (XP) in Agile. With a need to diagram, XP values.
Explanation of following model

3(a)

Extreme Programming (XP)

pictorial Model – 4m

Explanation_4m



23

XP Values
In order to achieve effective communication between software engineers and other stakeholders (e.g., to establish required features and functions for the software), XP emphasizes close, yet informal (verbal) collaboration between customers and developers, the establishment of effective metaphors3 for communicating important concepts, continuous feedback, and the avoidance of voluminous documentation as a communication medium.

XP restricts developers to design only for immediate needs, rather than consider future needs. The intent is to create a simple design that can be easily implemented in code). If the design must be improved, it can be refactored4 at a later time. Feedback is derived from three sources: the implemented software itself, the customer, and other software team members

An agile XP team must have the discipline (courage) to design for today, recognizing that future requirements may change dramatically, thereby demanding substantial rework of the design and implemented code. By following each of these values, the agile team inculcates respect among it members, between other stakeholders and team members, and indirectly, for the software itself key Human traits required to exist among the Agile software development team members.

3(5)

Human Factors

- Agile development focuses on the talents and skills of individuals, molding the process to specific people and teams.
- key traits ~~must exist among the people on an agile team and the team itself~~
  - **Competence.** In an agile development context, "competence" encompasses innate talent, specific software-related skills, and overall knowledge of the process that the team has chosen to apply. ex Conceptual thinking, Determination, Communication skills (written), Communication skills (verbal), Creative thinking, Discipline, Organising, Responding flexibly, Listening etc:
  - **Common focus.** Although members of the agile team may perform different tasks and bring different skills to the project, all should be focused on one goal—to deliver a working software increment to the customer within the time promised (**deliver a working software increment**)

—7M—

19

Collaboration: Software engineering (regardless of process) is about assessing, analyzing, and using information that is communicated to the software team; creating information that will help all stakeholders understand the work of the team;

Decision-making ability. Any good software team must be allowed the freedom to control its own destiny. This implies that the team is given autonomy—decision-making authority for both technical and project issues.

Ex Daily Stand-ups, **Self-Organizing Teams:** Agile teams have the autonomy to self-organize around tasks and roles. For instance, rather than having a project manager assign tasks, team members discuss and decide who will take on which tasks based on their skills, interests, and availability.

Fuzzy problem-solving ability. Software managers must recognize that the agile team will continually have to deal with ambiguity.

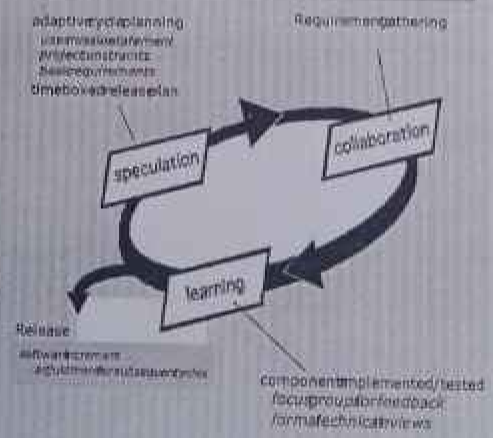
Mutual trust and respect:

Self-organization. In the context of agile development, self-organization implies three hings: (1) the agile team organizes itself for the work to be done, (2) the team organizes he process to best accommodate its local environment, (3) the team organizes the work schedule to best achieve delivery of the software increment.

Microsoft: Microsoft has adopted Agile practices across various product teams, Office 365, and Windows. spotify .
Netflix: teams are encouraged to experiment and iterate quickly, with a strong emphasis on automation and continuous delivery. This agility has been crucial in llowing Netflix to rapidly adapt to changing market demands and maintain its leadership in the streaming industry

4(a) philosophy of Agile Adaptive software development process model and discuss with a neat diagram.

Adaptive Software Development



29

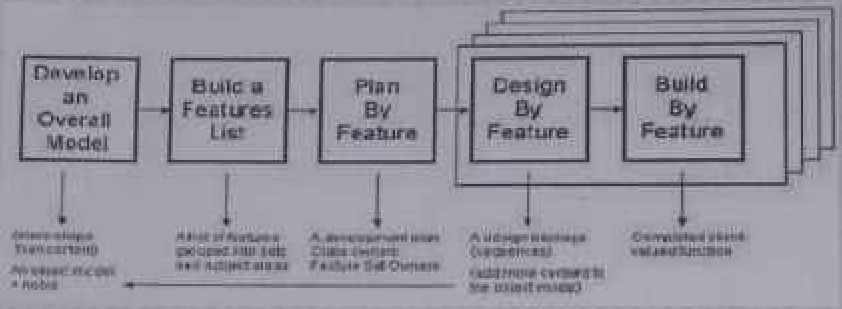
Pictorial model = 3M

Explaination = 4M

Model pictorial representation 4 M
Explanation 4 M

4(b) Feature driven agile system development with a neat diagram.

Feature Driven Development



Reprinted with permission of Peter Coad

These slides are designed to accompany Software Engineering: A Practitioner's Approach, 7/e (McGraw-Hill, 2009). Slides copyright 2009 by Roger Pressman. 42

5(a) The definition of a Project in software engineering and explain The ISO 12207 standard software development life cycle with a neat diagram.

define a project = 1M

ISO 12207 life-cycle

Requirements analysis

- Requirements elicitation: what does the client need?
- Analysis: converting 'customer facing' requirements into equivalents that developers can understand
- Requirements will cover
  - Functions, Quality, Resource constraints i.e. costs
- Architecture design
  - Based on *system requirements*
  - Defines components of system: hardware, software, organizational
  - *Software requirements* will come out of this
- Code and test
  - Of individual components
- Integration
  - Putting the components together

Lifecycle pictorial = 2M
Explanation = 2.
Total = 4M

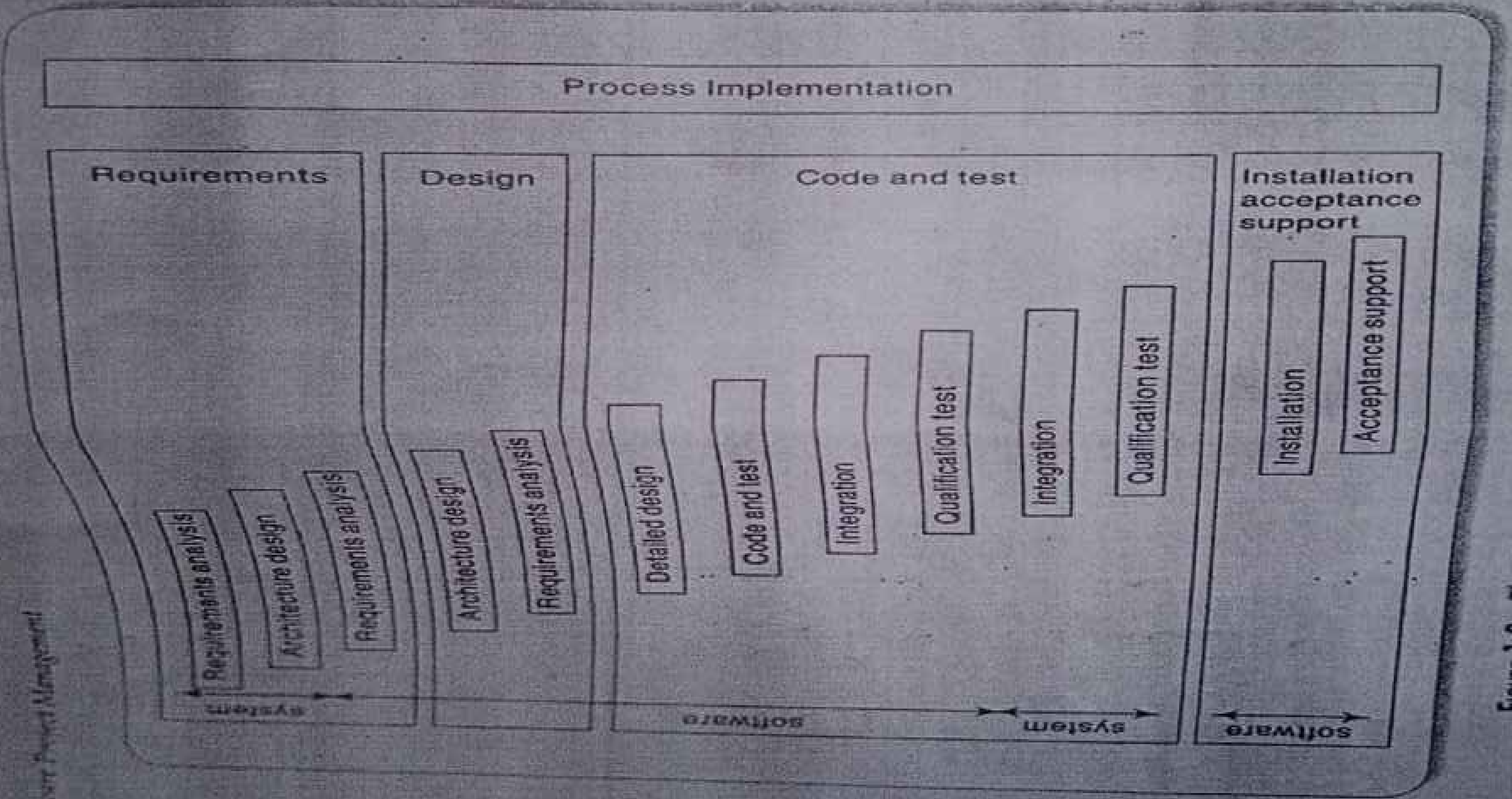


Figure 1.2 Th- ISO

ISO12207...

- Qualificationtesting
  - Testingthe *system*(notjustthe *software*)
- Installation
  - Theprocessof makingthesystemoperational
  - Includes setting up standing data, settingsystem parametersinstallingon operationalhardwareplatformsusertrainingetc
- Acceptancesupport
  - Includingmaintenanceand enhancement

10

6(a) ACtivities covered by project management team.

4M

Activities covered by project management

Feasibilitystudy

Isprojecttechnicallyfeasible and worthwhilefrom a businesspoint of view?

Planning

Only doneif projectis feasible

Execution

Implementplan, but plan may be changedas we go along

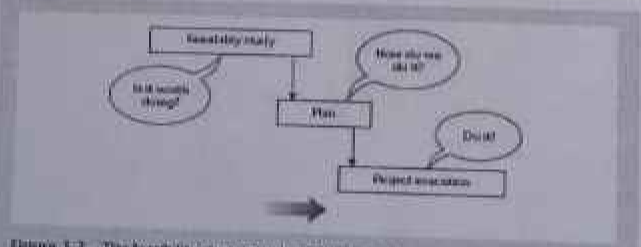
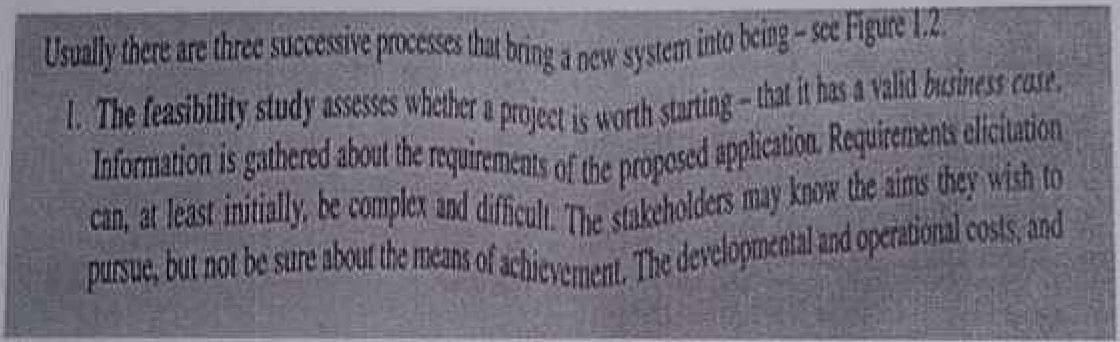


Figure 1.2 The feasibility study/plan/execution cycle

9

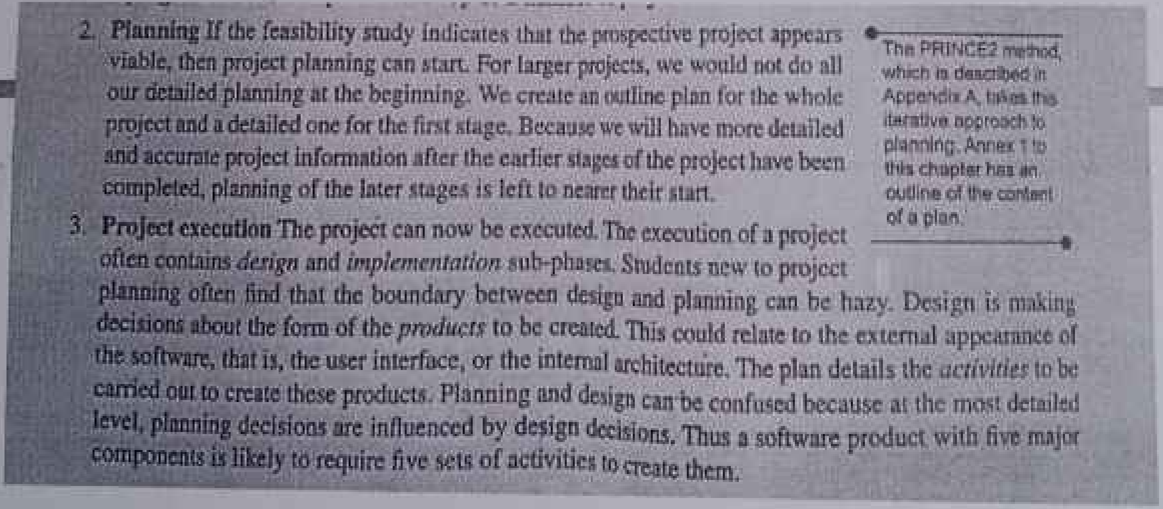
A software is not only concerned with the actual writing of software in fact where a software application is BOUGHT off the shelf there may be no software writing as such but this is still fundamental software project because so many of the other activitIES associates with software will still be present

For larger project detail planning will not be done at the beginning and an outline plan for the whole project is done and a detailed plan for the first stage



Usually there are three successive processes that bring a new system into being – see Figure 1.2.

1. **The feasibility study** assesses whether a project is worth starting – that it has a valid *business case*. Information is gathered about the requirements of the proposed application. Requirements elicitation can, at least initially, be complex and difficult. The stakeholders may know the aims they wish to pursue, but not be sure about the means of achievement. The developmental and operational costs, and



2. **Planning** If the feasibility study indicates that the prospective project appears viable, then project planning can start. For larger projects, we would not do all our detailed planning at the beginning. We create an outline plan for the whole project and a detailed one for the first stage. Because we will have more detailed and accurate project information after the earlier stages of the project have been completed, planning of the later stages is left to nearer their start.
3. **Project execution** The project can now be executed. The execution of a project often contains *design* and *implementation* sub-phases. Students new to project planning often find that the boundary between design and planning can be hazy. Design is making decisions about the form of the *products* to be created. This could relate to the external appearance of the software, that is, the user interface, or the internal architecture. The plan details the *activities* to be carried out to create these products. Planning and design can be confused because at the most detailed level, planning decisions are influenced by design decisions. Thus a software product with five major components is likely to require five sets of activities to create them.

The PRINCE2 method, which is described in Appendix A, takes this iterative approach to planning. Annex 1 to this chapter has an outline of the content of a plan.

[illegible] (SUSHMA A)
COURSE INCHARGE

[illegible]
HOD

K.S. INSTITUTE OF TECHNOLOGY, BENGALURU - 560109
THIRD INTERNAL TEST QUESTION PAPER 2023-24 EVEN SEMESTER

SET: A

USN | | | | | | | | | | |

Degree :	B.E	Semester :	VI
Branch - Stream :	COMPUTER SCIENCE & DESIGN	Course Type / Code :	21CS61
Course Title :	SOFTWARE ENGINEERING & PROJECT MANAGEMENT	Date :	29/07/24
Duration :	60 Minutes	Max Marks :	20

Note: Answer **ONE full** question from each module.
K-Levels: K1-Remebering, K2-Understanding, K3-Applying, K4-Analyzing, K5-Evaluating, K6-Creating

Q No.	Questions	Marks	CO	K-Level
	Module-5			
1(a)	**Obtain** BOEHM'S quality model and explain neatly with a diagram	7	CO5	K3
(b)	**Identify** the differences between Verification versus Validation in Software Engineering .Discuss Software testing process with levels of testing.	5	CO5	K3
	OR			
2(a)	**Make use of** V-process model of Testing and elaborate Testing phase and show how it is an extension of water fall model development.	7	CO5	K3
(b)	**Obtain the** Product and process Metrics and explain Product versus process quality management with a neat diagram showing sequence of processes and deliverables.	5	CO5	K3
	Module-4			
3(a)	**Identify** the differences between Software development life cycle and Project Management life cycles with suitable diagrams.	8	CO4	K3
	OR			
4(a)	**Obtain** the types of stakeholders in Software Engineering . Discuss Project management control cycle.	8	CO4	K3

[illegible] (SUSHMA A)
Name & Signature of Course In charge:

Deepa S.R
Deepa
Name & Signature of Module Coordinator:

Deepa
HOD

[illegible]
Principal
[illegible]

20 copies

K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109
III INTERNAL ASSESSMENT 2023– 24 EVEN SEMESTER

SCHEME AND SOLUTION

SET-A

Degree	: B.E	Semester :	VI
Branch	: COMPUTER SCIENCE AND DESIGN	Course Code :	21CS61
Course Title	: SOFTWARE ENGINEERING AND PROJECT MANAGEMENT	Max Marks :	20

Q No.	POINTS
	Module 3

(a) What is Boehm's Software Quality Model

- It's a software quality through a hierarchical structure of attributes
- This model is similar to the McCall Quality Model but encompasses a wider range of characteristics, including hardware performance related ones.
- Boehm's model categorizes quality attributes into three levels:

1. primary uses (high-level characteristics)
2. intermediate constructs (mid-level characteristics)
3. primitive constructs (basic characteristics)

·quality model diagram —
·Explaination—

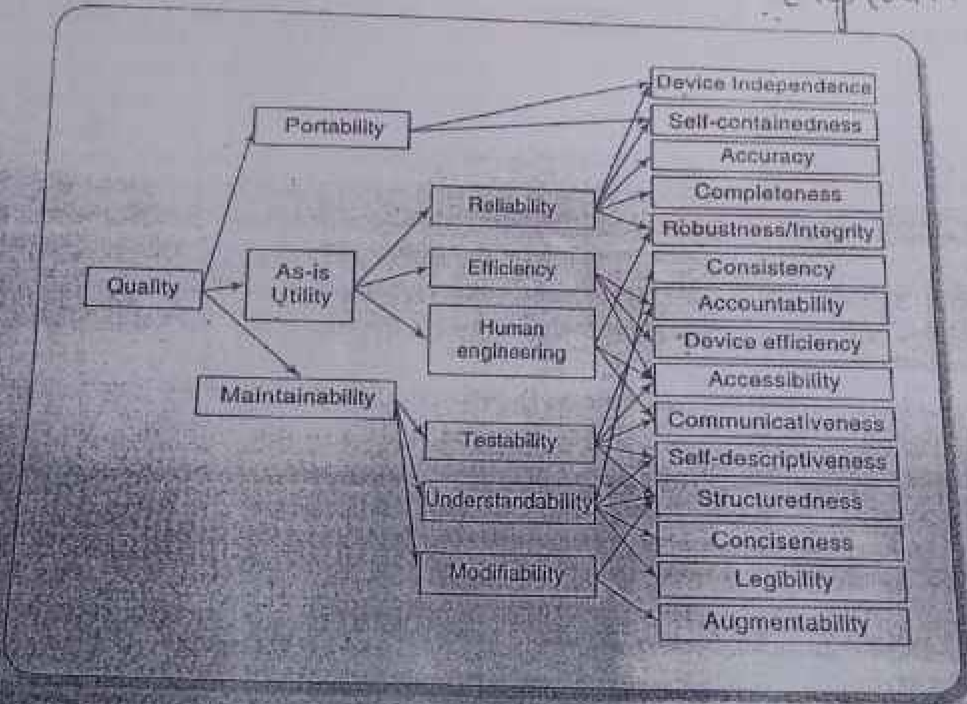


FIGURE 13.3 Boehm's quality model

so program testing would

be produced so user acceptance tests would still be valid

1 (b)

Verification versus validation

The objectives of both verification and validation techniques are very similar. Both these techniques have been designed to help remove errors in software. In spite of the apparent similarity between their objectives, the underlying principles of these two bug detection techniques and their applicability are very different. The main differences between these two techniques are the following:

- Verification is the process of determining whether the output of one phase of software development conforms to that of its previous phase. Validation is the process of determining whether fully developed software conforms to its requirements specification. We can therefore say that the objective of verification is to check if the artifacts produced after a phase conforms to that of the previous phase. For example, a verification step can be to check if the design documents produced after the design step conform to the requirements specification. On the other hand, validation is applied to the fully developed and integrated software to check if it satisfies the customer's requirements. The primary techniques used for verification include review, simulation, and formal verification. On the other hand, validation techniques are primarily based on product testing.
- Verification is carried out during the development process to check if the development activities are being carried out correctly, whereas validation is carried out towards the end of the development process to check if the right product as required by the customer has been developed. Verification techniques can be viewed as an attempt to achieve phase containment of errors. Phase containment of errors has been acknowledged to be a cost-effective way to eliminate program bugs, and accepted as an important software engineering principle.

All the boxes shown in the right hand side of the V-process model of Figure 13.9 correspond to verification activities except the system testing block which corresponds to validation activity.

Comparison of verification & validation

- Levels of testing

Levels of testing

A software product is normally tested at three different stages or levels. These three testing stages are

- Unit testing
- Integration testing
- System testing

During unit testing, the individual components (or units) of a program are tested. For every module, unit testing is carried out as soon as the coding for it is complete. Since every module is tested separately, there is a good scope for parallel activities during unit testing. The objective of integration testing is to check whether the modules have any errors pertaining to interfacing with each other.

Unit testing is referred to as testing in the small, whereas integration and system testing are referred to as testing in the large. After testing all the units individually, the units are integrated over a number of steps and tested after each step of integration (integration testing). Finally, the fully integrated system is tested (system testing).

3.?

OR

The V-process model can be seen as expanding the activity box 'testing' in the waterfall model. Each step has a matching validation process which can, where defects are found, cause a loop back to the corresponding development stage and a reworking of the following steps. Ideally this feeding back should occur only where a discrepancy has been found between what was specified by a particular activity and what was actually implemented in the next lower activity on the descent of the V loop. For example, the system designer might have written that a calculation be carried out in a certain way. A developer building code to meet this design might have misunderstood what was required. At system testing stage, the original designer would be responsible for checking that the software is doing what was specified and this would discover the coder's misreading of that document.

Explaination of V-process Model — 3M

- By using product- based approach to planning and control ,it focuses on products which is convenient.
- It is easier to measure products qualities in a completed computer application rather than during its development.
- Trying to use the attributes of intermediate products created at earlier stages to predict the quality of the final application is difficult.
- An alternative approach is to scrutinize the quality of the processes used to develop software product.
- The system development process comprises a number of activities linked so that the output from one activity is the input to the next.
- Errors can be entered at any stage.
- When errors are not removed at early stages it becomes more expensive to correct at later stages.
- Each development step that passes before the error us found increases the amount of rework needed.
- An error in the specification found in testing will mean rework at all the stages between specification and testing. Each successive step of development is also more detailed and less able to absorb change.

Explaination of Testing phase = 2m

Total = 5M

2(b) Product and Process metrics

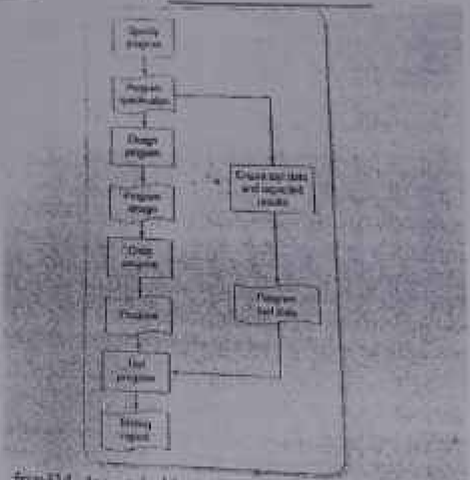


Figure 13.4 An example of the sequence of processes and deliverables

Diagram =

& Explanation = 4

Errors should therefore be eradicated by careful examination of the deliverables of each step before they are passed on. One way of doing this is by having the following process requirements for each step.

- *Entry requirements*, which have to be in place before an activity can start. An example would be that a comprehensive set of test data and expected results be prepared and approved before program testing can commence.
- *Implementation requirements*, which define how the process is to be conducted. In the testing phase, for example, it could be laid down that whenever an error is found and corrected, all test runs must be repeated, even those that have previously been found to run correctly.
- *Exit requirements*, which have to be fulfilled before an activity is deemed to have been completed. For example, for the testing phase to be recognized as being completed all tests will have to have been run successfully with no outstanding errors.

3(a)

MODULE 4

Software Development & Project Management Life Cycles

- Software development life cycle (SDLC) denotes the stages through which a software is developed.
- During SDLC, starting from its conception, developers carry out several processes / methodologies till the software is fully developed and deployed at client side.
- Stages includes - Requirements gathering, analysis, detailed design, coding, testing, and deployment.

- Meanwhile, contrast to SDLC, Project Management Life Cycle (PMLC) typically starts before software development activities start & continues for entire duration of SDLC.
- Software project manager, in this cycle, carries out several processes / methodologies to perform required software project management activities
- Stages include - Initiation, planning, execution, monitoring, controlling, and closing.

Comparison of SDLC & PMLC

- Activities carried out by developers during SDLC & PMLC are grouped into phases.
- The different phases of both cycles are shown in below fig:

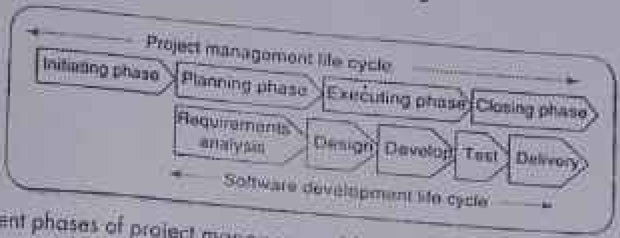


FIGURE 1.8 Different phases of project management life cycle and software development life cycle

- The initiating and planning phases of the project management life cycle often start before the software development life cycle begins.
- The executing phase of project management overlaps with the entire software development life cycle.
- By the time the software development process starts, the initiating phase of project management is almost complete.
- Both cycles interact closely to ensure the successful delivery of the software project.

Explaination of Different phase of project Mangement life cycle, with a neat diagroum = 4M

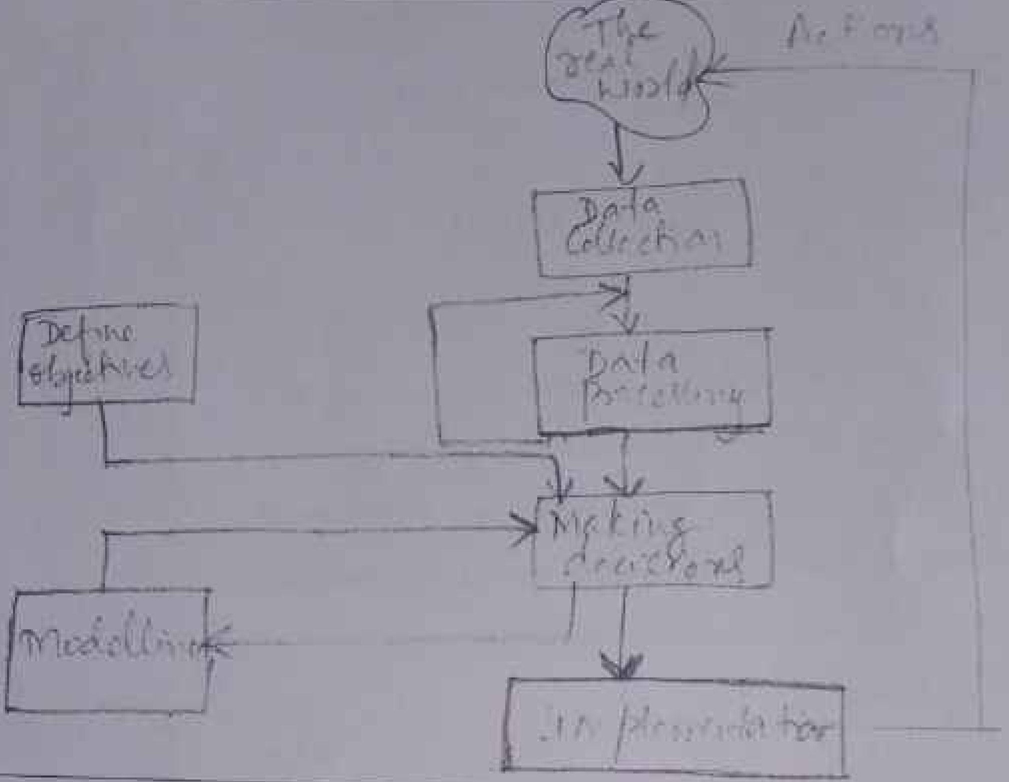
Total = 8M

1(a)

Stakeholders

Definition = 1M

Types of Stakeholders Definitions and Explaination = 3M

 The real world Actions Data Collection Data Processing Define objectives Making decisions Modelling Implementation	Total = 8M
The Project Control cycle Diagram & Explaination	= 4M

[illegible] (SUSHMA A)
COURSE INCHARGE

[illegible]
HOD

K.S. INSTITUTE OF TECHNOLOGY, BENGALURU - 560109
THIRD INTERNAL TEST QUESTION PAPER 2023-24 EVEN SEMESTER

KSIT

SET: B

USN										

Degree	:	B.E	Semester	:	VI
Branch - Stream	:	COMPUTER SCIENCE & DESIGN	Course Type / Code	:	21CS61
Course Title	:	SOFTWARE ENGINEERING & PROJECT MANAGEMENT	Date	:	29/07/24
Duration	:	60 Minutes	Max Marks	:	20

Note: Answer **ONE full** question from each module.

K-Levels: K1-Remebering, K2-Understanding, K3-Applying, K4-Analyzing, K5-Evaluating, K6-Creating

Q No.	Questions	Marks	CO	K-Level
	Module-5			
1(a)	**Make use of** table of good software attributes and discuss the Importance of Software quality. Define Software quality.	7	CO5	K3
(b)	**Obtain** different types of Software testing approaches and elaborate levels of testing.	5	CO5	K3
	OR			
2(a)	**Obtain** BOEHM'S quality model and explain neatly with a diagram.	7	CO5	K3
(b)	**Identify** the place of Software quality in step wise with a neat diagram.	5	CO5	K3
	Module-4			
3(a)	**Identify** the objectives and sub objectives of the project. What measure of effectiveness could be used to check the success in achieving the objectives of the project?	8	CO4	K3
	OR			
4(a)	**Obtain** the differences between traditional and modern project management Practices. Discuss Principal Project management processes with a neat diagram.	8	CO4	K3

[illegible] (SUSHMA A)
Name & Signature of Course In charge:

Deepa .S.R
[illegible]
Name & Signature of Module Coordinator:

[illegible]
HOD -CSD

[illegible]
Principal

K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109
III INTERNAL ASSESSMENT 2023 – 24 EVEN SEMESTER

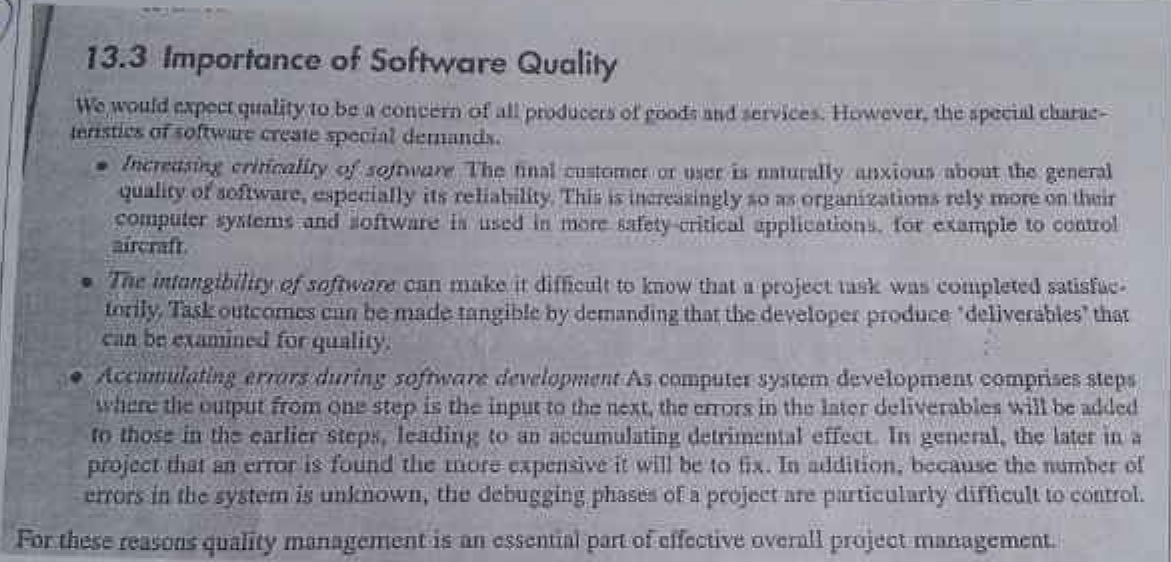
SCHEME AND SOLUTION

SET-B

Degree	:	B.E	Semester :	VI
Branch	:	COMPUTER SCIENCE AND DESIGN	Course Code :	21CS61
Course Title	:	SOFTWARE ENGINEERING AND PROJECT MANAGEMENT	Max Marks :	20

Q No.	POINTS	Marks
	Module 3	

1(a)



13.3 Importance of Software Quality

We would expect quality to be a concern of all producers of goods and services. However, the special characteristics of software create special demands.

- *Increasing criticality of software* The final customer or user is naturally anxious about the general quality of software, especially its reliability. This is increasingly so as organizations rely more on their computer systems and software is used in more safety-critical applications, for example to control aircraft.
- *The intangibility of software* can make it difficult to know that a project task was completed satisfactorily. Task outcomes can be made tangible by demanding that the developer produce 'deliverables' that can be examined for quality.
- *Accumulating errors during software development* As computer system development comprises steps where the output from one step is the input to the next, the errors in the later deliverables will be added to those in the earlier steps, leading to an accumulating detrimental effect. In general, the later in a project that an error is found the more expensive it will be to fix. In addition, because the number of errors in the system is unknown, the debugging phases of a project are particularly difficult to control.

For these reasons quality management is an essential part of effective overall project management.

7M
= 3M
+

Table 12.1 *Software quality criteria*

Quality factor	*Software quality criteria*
Correctness	traceability, consistency, completeness
Reliability	error tolerance, consistency, accuracy, simplicity
Efficiency	execution efficiency, storage efficiency
Integrity	access control, access audit
Usability	operability, training, communicativeness, input/output volume, input/output rate
Maintainability	consistency, simplicity, conciseness, modularity, self-descriptiveness
Testability	simplicity, modularity, instrumentation, self-descriptiveness
Flexibility	modularity, generality, expandability, self-descriptiveness
Portability	modularity, self-descriptiveness, machine independence, software system independence
Reusability	generality, modularity, software system independence, machine independence, self-descriptiveness
Interoperability	modularity, communications commonality, data commonality

4M

4(b)

13.12 Testing

The final judgement of the quality of a software application is whether it actually works correctly when executed. This section looks at aspects of the planning and management of testing. A major headache with testing is estimating how much testing remains at any point. This estimate of the work still to be done depends on an unknown, the number of bugs left in the code. We will briefly discuss how we can deal with this problem.

In Chapter 4, the *V-process model* was introduced as an extension to the waterfall process model. Figure 13.9 gives a diagrammatic representation of this model. This stresses the necessity for validation activities that match the activities creating the products of the project.

5M

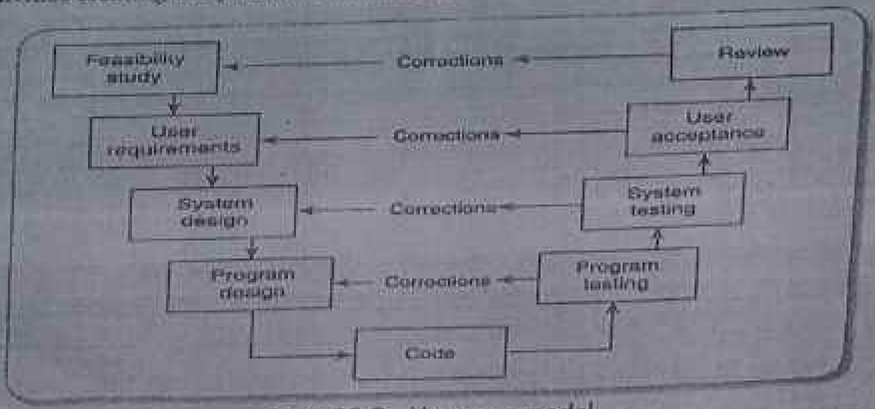


FIGURE 13.9 V-process model

OR

V-process model with diagram – 3M

Explaination – 2M

2(a) What is Boehm's Software Quality Model

- It's a software quality through a hierarchical structure of attributes.
- This model is similar to the McCall Quality Model but encompasses a wider range of characteristics, including hardware performance–related ones.
- Boehm's model categorizes quality attributes into three levels:

1. primary uses (high–level characteristics)
2. intermediate constructs (mid–level characteristics)
3. primitive constructs (basic characteristics)

Explaination – 2M

Diagram – 5M

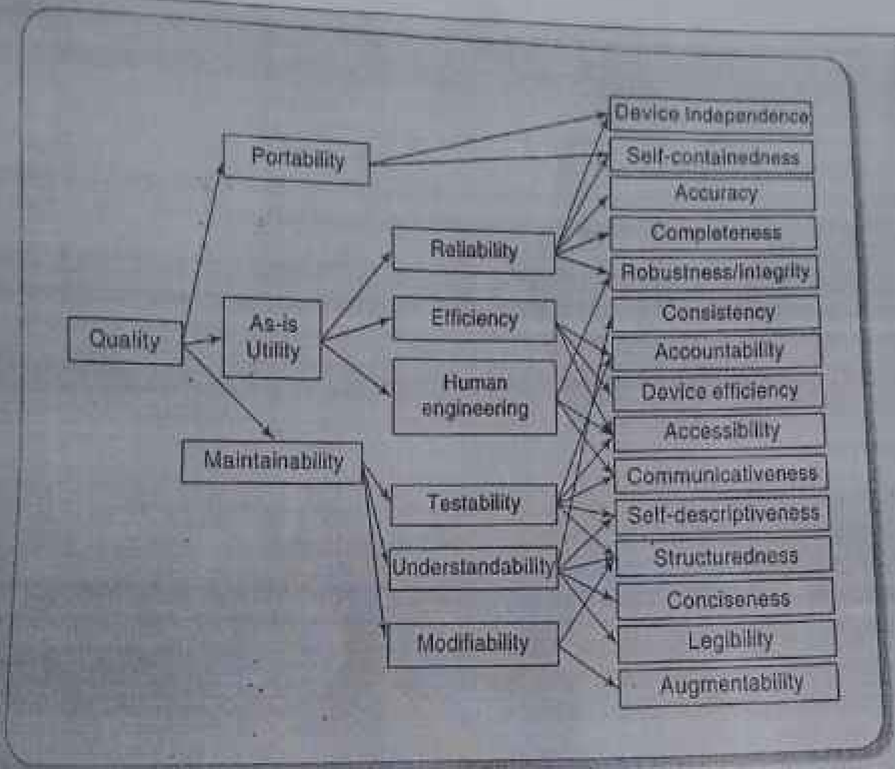


FIGURE 13.3 Boehm's quality model

Software Quality

1. Identify Project Scope and objective: Some objective could relate to the qualities of the application to be delivered.

2. Identify project infrastructure: Identify the installation standard and procedures. Some of these almost certainly be about quality

3. Analyze project characteristics: To identify the other qualities based requirement.

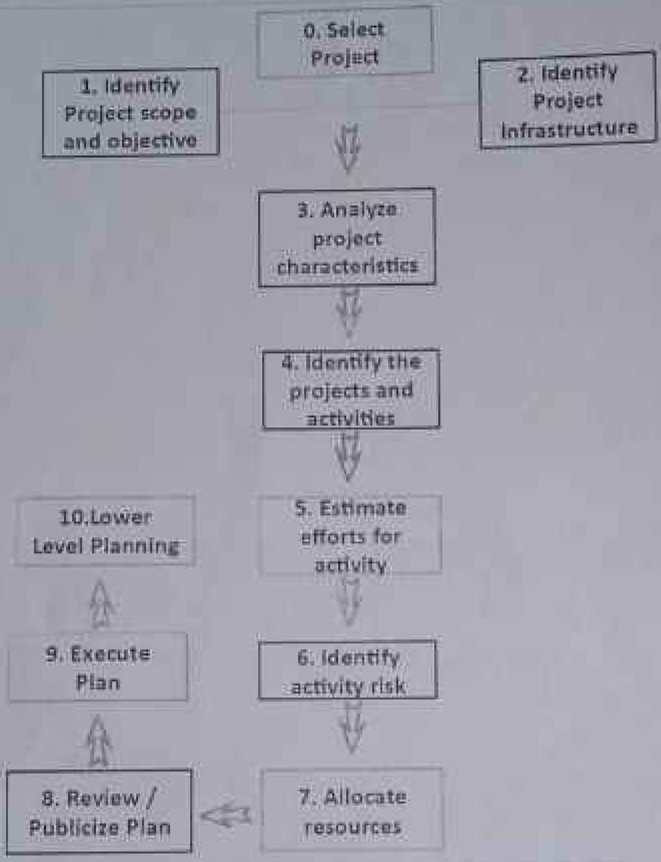
4. Identify the products and activities of the project: It is at this point the entry, exist and process requirement are identified for each activity

5. Review and publicize Plan: At his stage the overall quality aspects of the project plan are reviewed

Explaination

Diagram of Flow
of deliverables
& in each phase

Software Quality


0. Select Project
1. Identify Project scope and objective
2. Identify Project Infrastructure
3. Analyze project characteristics
4. Identify the projects and activities
5. Estimate efforts for activity
6. Identify activity risk
7. Allocate resources
8. Review / Publicize Plan
9. Execute Plan
10.Lower Level Planning

-Software Engineering

4

Explaination of Each phase is required

1.11 Setting Objectives

Among all these stakeholders are those who actually own the project. They control the financing of the project. They also set the objectives of the project. The objectives should define what the project team must achieve for project success. Although different stakeholders have different motivations, the project objectives identify the shared intentions for the project.

Objectives focus on the desired outcomes of the project rather than the tasks within it – they are the 'post-conditions' of the project. Informally the objectives could be written as a set of statements following the opening words '*the project will be a success if. . . .*' Thus one statement in a set of objectives might be '*customers can order our products online*' rather than '*to build an e-commerce website*'. There is often more than one way to meet an objective and the more possible routes to success the better.

There may be several stakeholders, including users in different business areas, who might have some claim to project ownership. In such a case, a *project authority* needs to be explicitly identified with overall authority over the project.

This authority is often a *project steering committee* (or *project board* or *project management board*) with overall responsibility for setting, monitoring and modifying objectives. The project manager runs the project on a day-to-day basis, but regularly reports to the steering committee.

This committee is likely to contain user, development and management representatives.

Sub-objectives and goals

An effective objective for an individual must be something that is within the control of that individual. An objective might be that the software application produced must pay for itself by reducing staff costs. As an overall business objective this might be reasonable. For software developers it would be unreasonable as any reduction in operational staff costs depends not just on them but on the operational management of the delivered system. A more appropriate *goal* or sub-objective for the software developers would be to keep development costs within a certain budget.

Defining sub-objectives requires assumptions about how the main objective is to be achieved.

We can say that in order to achieve the objective we must achieve certain goals or sub-objectives first. These are steps on the way to achieving an objective, just as goals scored in a football match are steps towards the objective of winning the match. Informally this can be expressed as a set of statements following the words '*To reach objective. . ., the following must be in place. . .*'.

The mnemonic SMART is sometimes used to describe well-defined objectives:

- *Specific* Effective objectives are concrete and well defined. Vague aspirations such as '*to improve customer relations*' are unsatisfactory. Objectives should be defined so that it is obvious to all whether the project has been successful.
- *Measurable* Ideally there should be *measures of effectiveness* which tell us how successful the project has been. For example, '*to reduce customer complaints*' would be more satisfactory as an objective than '*to improve customer relations*'. The measure can, in some cases, be an answer to simple yes/no question, e.g. '*Did we install the new software by 1 June?*'

This still leaves a problem about the level at which the target should be set, e.g. why, say, a 50% reduction in complaints and not 40% or 60%?

- *Achievable* It must be within the power of the individual or group to achieve the objective.

4(a) Traditional and Modern Project Management Practices

1. planning Increment Delivery
2. Quality Management
3. Change Management
4. Requirements Management
5. Release Management
6. Risk Management
7. Scope Management

Comparison of traditional & modern project management practices in Each of These phases — 4M

Explaination of Each phase — 4M

Total = 8M

[illegible] (SUSHMA N)
COURSE INCHARGE

[illegible]
HOD