USN	□□□□□□□□□□	CELEBRATING 30 YEARS CMRIT CMR INSTITUTE OF TECHNOLOGY, BENGALURU. ACCREDITED WITH A++ GRADE BY NAAC

Internal Assessment Test 1 – June 2024 – Scheme of Valuation

Sub:	Software Engineering and Project Management				Sub Code:	21CS61	Branch	ISE
Date:	04/06/2024	Duration:	90 min’s	Max Marks: 50	Sem/Sec:	VI / A, B & C		OBE

	Answer any FIVE questions	MARKS	CLO	RBT
1	(a) Define software Engineering. Definition – IEEE definition of software engineering. (b) Briefly define the attributes of a good software? List the attributes – 3 Brief one line explanation – 2	5 5	CLO1	L2
2	(a) What are the software myths? Define legacy software? List the myths given in the textbook – 3 Definition of legacy software – 2 (b) Briefly define the properties of a well-engineered software? Properties listed in the textbook – 2 Definition of the properties – 3	5 5	CLO1	L2
3	(a) Explain the waterfall model of the software development process with a neat diagram. Model diagram – 2 Explanation for the model – 3 (b) Describe the core advantages and unavoidable disadvantages. Advantages – 3, Disadvantages – 2	5 5	CLO1	L2
4	(a) Briefly explain the requirement elicitation and analysis process. Requirements elicitation process and explanation – 3 Analysis process with use case model – 2 (b) Explain the important differences between functional and non-functional requirements. Atleast 5 differences and suitable explanations – 5	5 5	CLO1	L2
5	(a) Explain prototype and V models for software development with suitable diagrams. Prototype Model – diagram – 1, explanation – 1.5 V Model – diagram – 1, explanation – 1.5 (b) Differentiate the above two models based on their core properties.	5 5	CLO1	L2

	Differences – 5 of them and relate to core properties			
6	(a) Explain software process framework with a neat diagram. Software Process Framework – diagram 2 marks Explanation for the framework – 2 marks (b) Draw the use case diagram for the following case study: A Modern Bazar Supermarket sells books and CDs using Online shopping. The customer adds items to the shopping cart. The customer may remove items or go to the checkout to make purchases at any time. The customer receives the purchased items by choosing a payment method. A sales employee at modern bazaar supermarket gets the order and purchase confirmation from the system and sends the electronic order to the warehouse. The warehouse employee updates the order status. The customer may check the order status. Use Case diagram – atleast 6 use cases with appropriate actors. System has to be represented in a rectangular box.	4 6	CLO2	L3

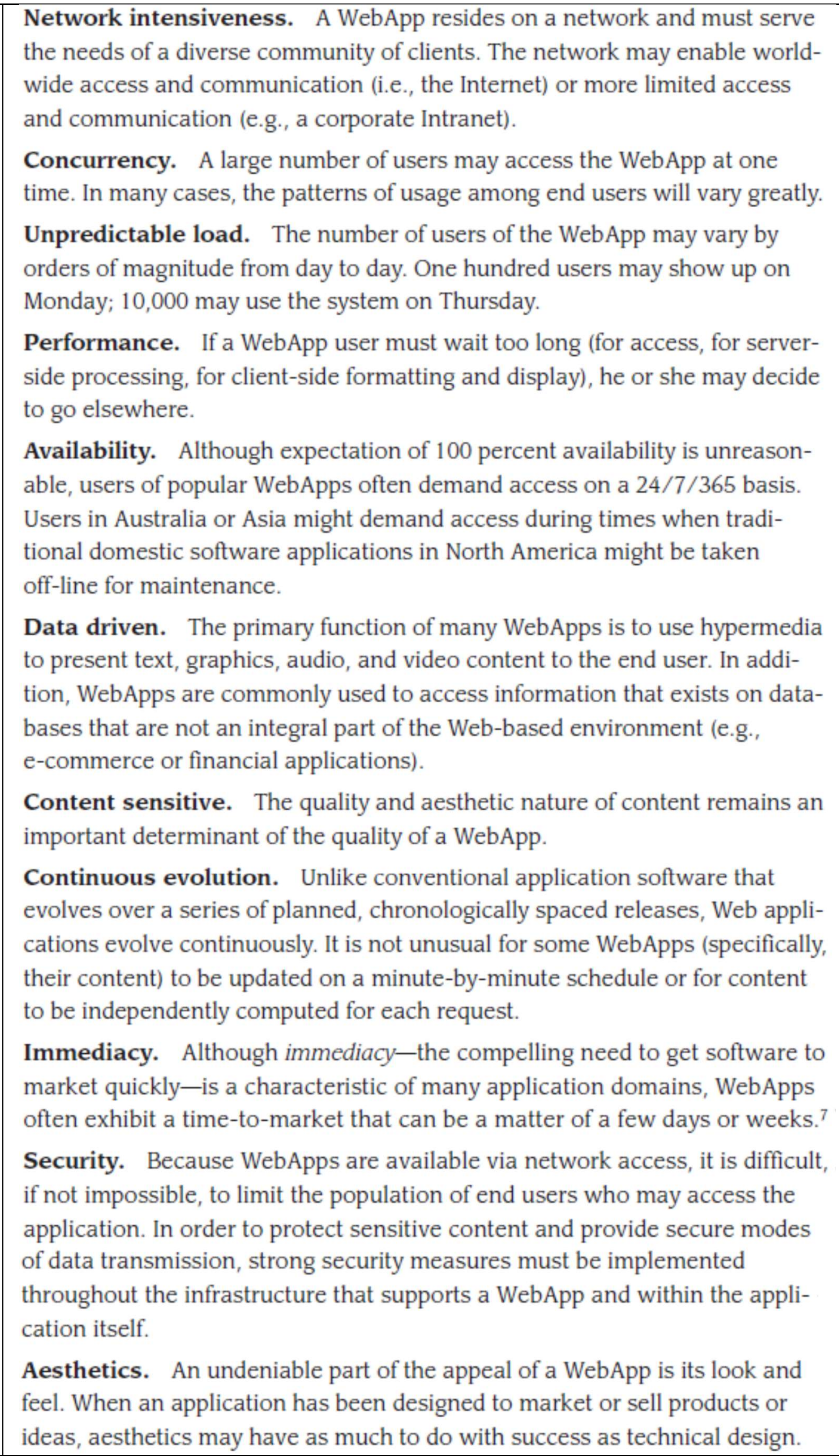
z

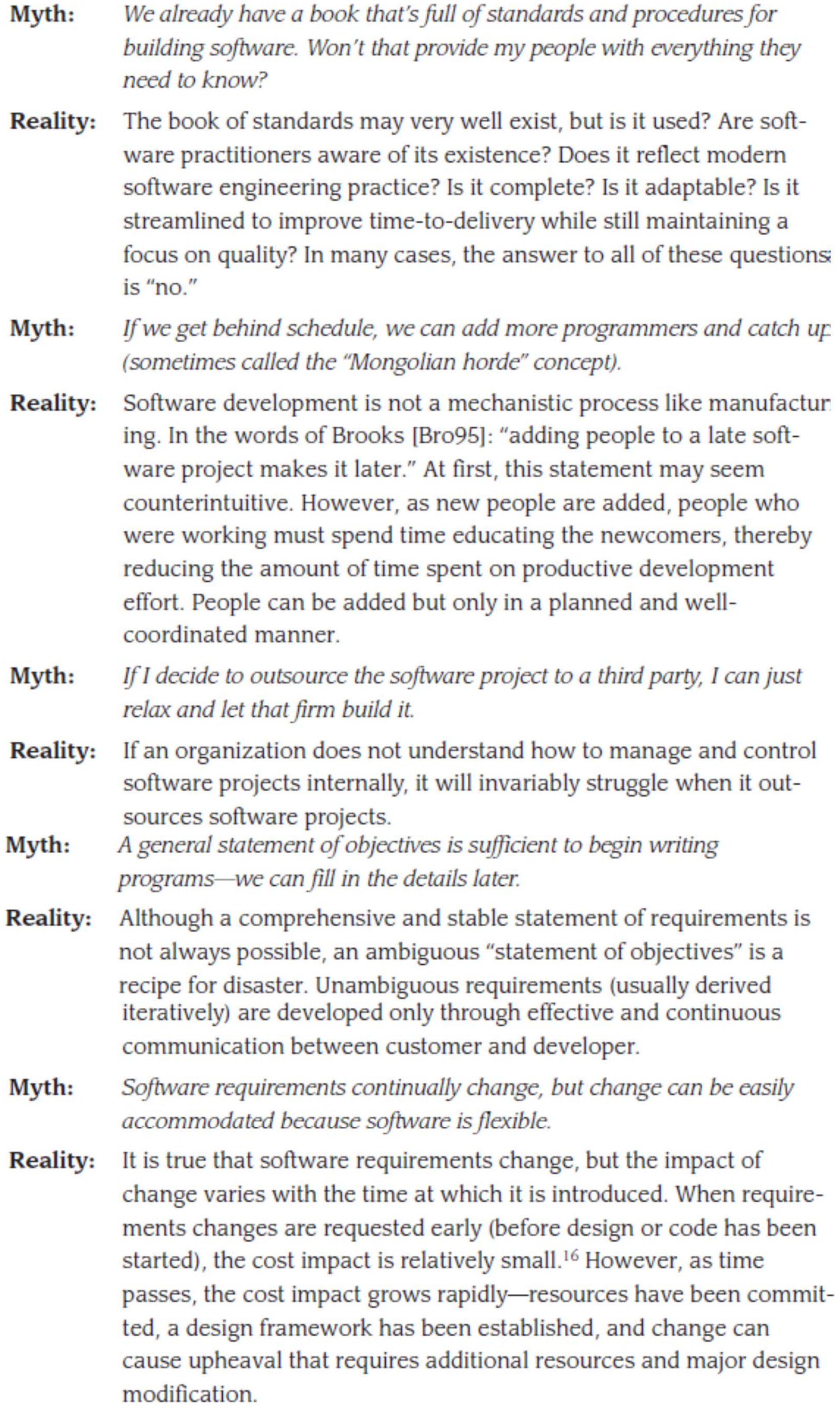
USN	□□□□□□□□□□

Internal Assessment Test 1 – June 2024 – Solution

Sub:	Software Engineering and Project Management					Sub Code:	21CS61	Branch	ISE
Date:	04/06/2024	Duration:	90 min's	Max Marks:	50	Sem/Sec:	VI / A, B & C		OBE

	Answer any FIVE questions	MARKS	CLO	RBT
1	(a) Define software Engineering. Definition – IEEE definition of software engineering. Software Engineering: (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. (2) The study of approaches as in (1). (b) Briefly define the attributes of a good software? List the attributes – 3 Brief one line explanation – 2	5 5	CLO1	L2

	 **Network intensiveness.** A WebApp resides on a network and must serve the needs of a diverse community of clients. The network may enable worldwide access and communication (i.e., the Internet) or more limited access and communication (e.g., a corporate Intranet). **Concurrency.** A large number of users may access the WebApp at one time. In many cases, the patterns of usage among end users will vary greatly. **Unpredictable load.** The number of users of the WebApp may vary by orders of magnitude from day to day. One hundred users may show up on Monday; 10,000 may use the system on Thursday. **Performance.** If a WebApp user must wait too long (for access, for server-side processing, for client-side formatting and display), he or she may decide to go elsewhere. **Availability.** Although expectation of 100 percent availability is unreasonable, users of popular WebApps often demand access on a 24/7/365 basis. Users in Australia or Asia might demand access during times when traditional domestic software applications in North America might be taken off-line for maintenance. **Data driven.** The primary function of many WebApps is to use hypermedia to present text, graphics, audio, and video content to the end user. In addition, WebApps are commonly used to access information that exists on databases that are not an integral part of the Web-based environment (e.g., e-commerce or financial applications). **Content sensitive.** The quality and aesthetic nature of content remains an important determinant of the quality of a WebApp. **Continuous evolution.** Unlike conventional application software that evolves over a series of planned, chronologically spaced releases, Web applications evolve continuously. It is not unusual for some WebApps (specifically, their content) to be updated on a minute-by-minute schedule or for content to be independently computed for each request. **Immediacy.** Although *immediacy*—the compelling need to get software to market quickly—is a characteristic of many application domains, WebApps often exhibit a time-to-market that can be a matter of a few days or weeks.[7] **Security.** Because WebApps are available via network access, it is difficult, if not impossible, to limit the population of end users who may access the application. In order to protect sensitive content and provide secure modes of data transmission, strong security measures must be implemented throughout the infrastructure that supports a WebApp and within the application itself. **Aesthetics.** An undeniable part of the appeal of a WebApp is its look and feel. When an application has been designed to market or sell products or ideas, aesthetics may have as much to do with success as technical design.			
2	(a) What are the software myths? Define legacy software? List the myths given in the textbook – 3 Definition of legacy software – 2	5	CLO1	L2



Myth: *We already have a book that's full of standards and procedures for building software. Won't that provide my people with everything they need to know?*

Reality: The book of standards may very well exist, but is it used? Are software practitioners aware of its existence? Does it reflect modern software engineering practice? Is it complete? Is it adaptable? Is it streamlined to improve time-to-delivery while still maintaining a focus on quality? In many cases, the answer to all of these questions is "no."

Myth: *If we get behind schedule, we can add more programmers and catch up (sometimes called the "Mongolian horde" concept).*

Reality: Software development is not a mechanistic process like manufacturing. In the words of Brooks [Bro95]: "adding people to a late software project makes it later." At first, this statement may seem counterintuitive. However, as new people are added, people who were working must spend time educating the newcomers, thereby reducing the amount of time spent on productive development effort. People can be added but only in a planned and well-coordinated manner.

Myth: *If I decide to outsource the software project to a third party, I can just relax and let that firm build it.*

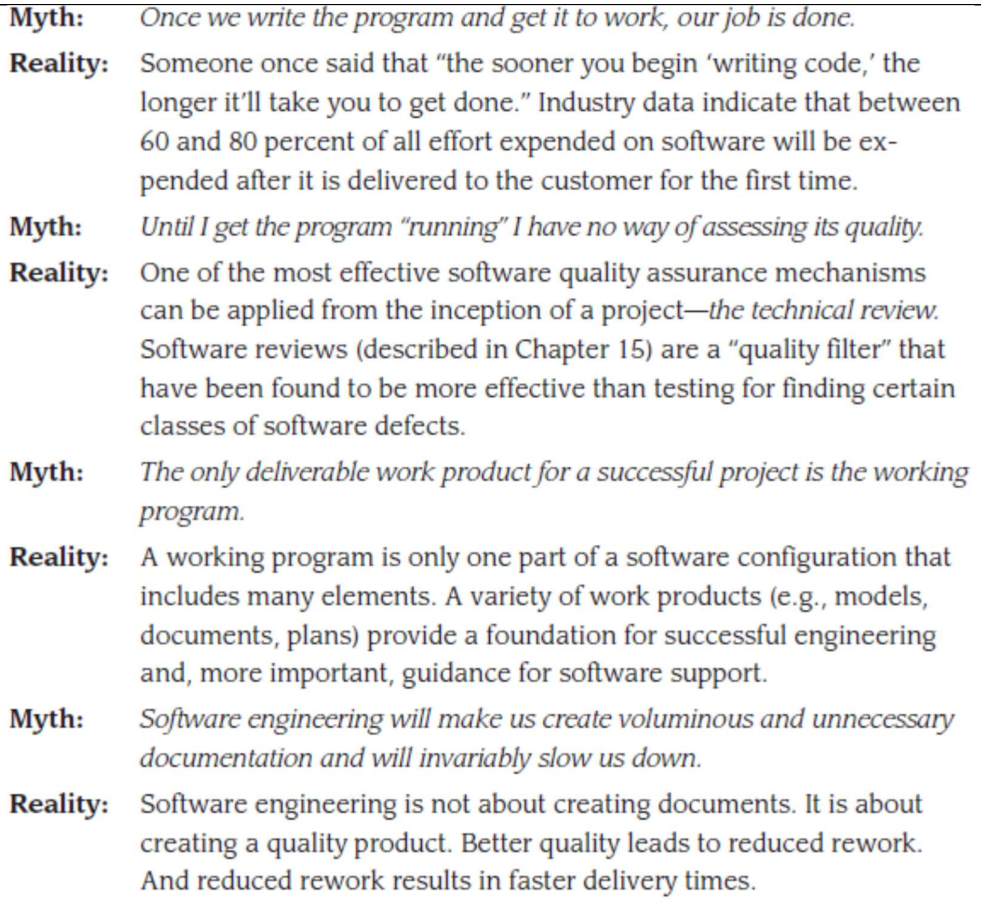
Reality: If an organization does not understand how to manage and control software projects internally, it will invariably struggle when it outsources software projects.

Myth: *A general statement of objectives is sufficient to begin writing programs—we can fill in the details later.*

Reality: Although a comprehensive and stable statement of requirements is not always possible, an ambiguous "statement of objectives" is a recipe for disaster. Unambiguous requirements (usually derived iteratively) are developed only through effective and continuous communication between customer and developer.

Myth: *Software requirements continually change, but change can be easily accommodated because software is flexible.*

Reality: It is true that software requirements change, but the impact of change varies with the time at which it is introduced. When requirements changes are requested early (before design or code has been started), the cost impact is relatively small.[16] However, as time passes, the cost impact grows rapidly—resources have been committed, a design framework has been established, and change can cause upheaval that requires additional resources and major design modification.



Myth: *Once we write the program and get it to work, our job is done.*

Reality: Someone once said that "the sooner you begin 'writing code,' the longer it'll take you to get done." Industry data indicate that between 60 and 80 percent of all effort expended on software will be expended after it is delivered to the customer for the first time.

Myth: *Until I get the program "running" I have no way of assessing its quality.*

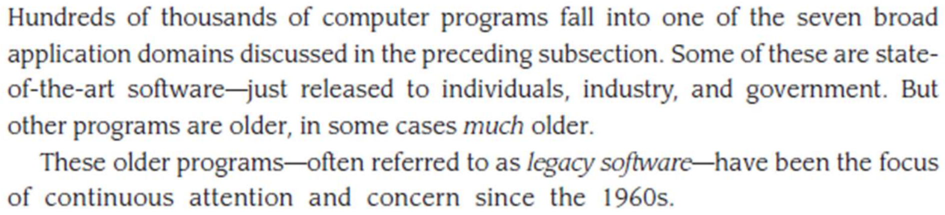
Reality: One of the most effective software quality assurance mechanisms can be applied from the inception of a project—*the technical review.* Software reviews (described in Chapter 15) are a "quality filter" that have been found to be more effective than testing for finding certain classes of software defects.

Myth: *The only deliverable work product for a successful project is the working program.*

Reality: A working program is only one part of a software configuration that includes many elements. A variety of work products (e.g., models, documents, plans) provide a foundation for successful engineering and, more important, guidance for software support.

Myth: *Software engineering will make us create voluminous and unnecessary documentation and will invariably slow us down.*

Reality: Software engineering is not about creating documents. It is about creating a quality product. Better quality leads to reduced rework. And reduced rework results in faster delivery times.



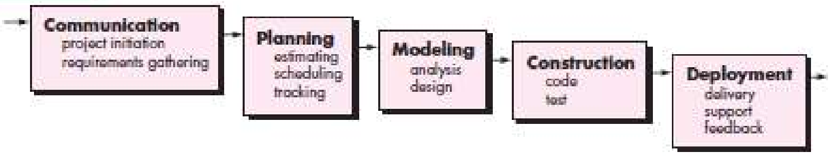
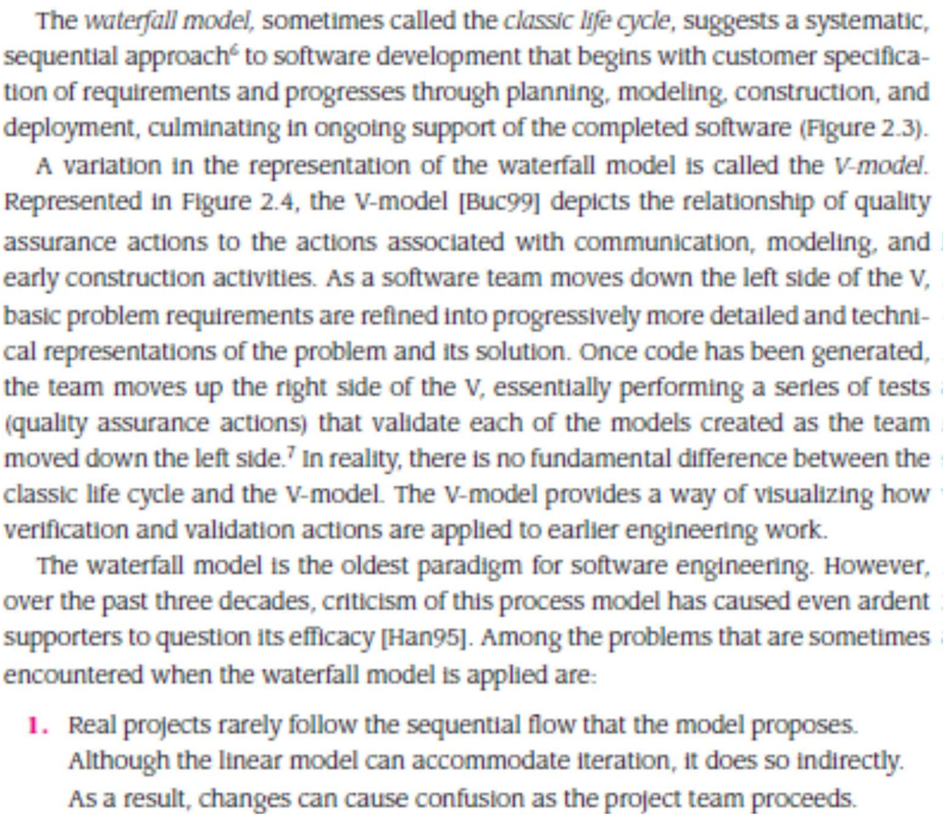
Hundreds of thousands of computer programs fall into one of the seven broad application domains discussed in the preceding subsection. Some of these are state-of-the-art software—just released to individuals, industry, and government. But other programs are older, in some cases *much* older.

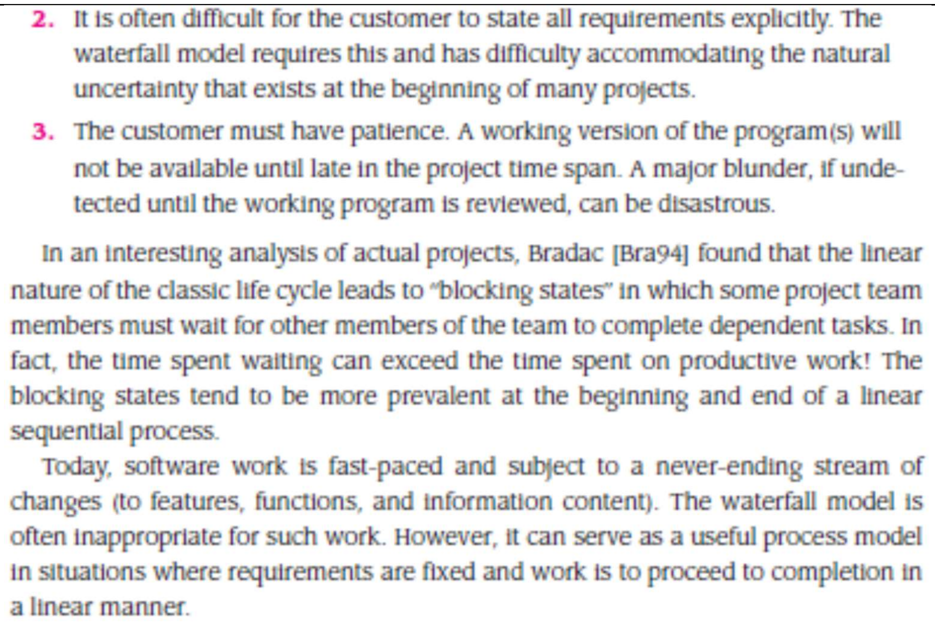
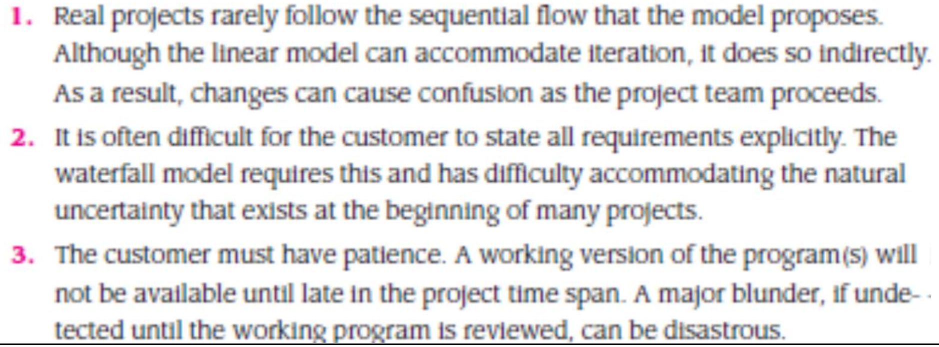
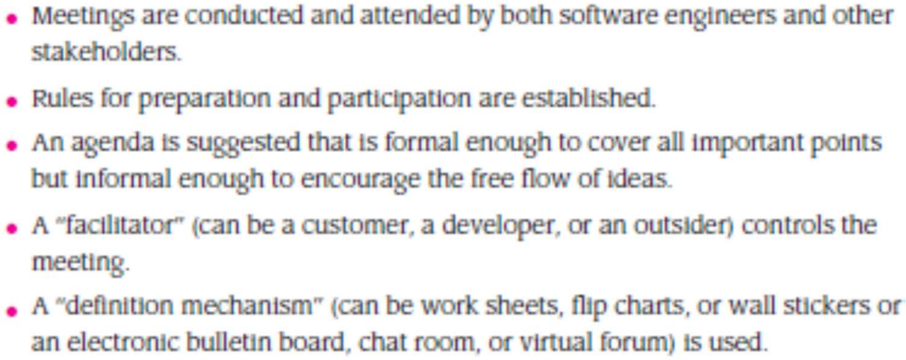
These older programs—often referred to as *legacy software*—have been the focus of continuous attention and concern since the 1960s.

(b) Briefly define the properties of a well-engineered software? 5

Properties listed in the textbook – 2
Definition of the properties – 3

1. Maintainability – Software maintainability is defined as the ease of finding and correcting errors in the software.
2. Dependability – It is often defined as the extent to which a program can be expected to perform intended functions which required precision over a given period of time.
3. Efficiency – Efficiency is the extent to which software uses minimum hardware resources to perform its functions.
4. Usability – This is the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.
5. Portability – It is defined as the ease of transporting a given set of software to a new hardware and/or operating system environment.
6. Availability of Documentation – System documentation includes all of the documents describing the implementation of the system from the requirements specification to the final acceptance test plan.

3	(a) Explain the waterfall model of the software development process with a neat diagram. Model diagram – 2 FIGURE 2.3 The waterfall model  Communication project initiation requirements gathering Planning estimating scheduling tracking Modeling analysis design Construction code test Deployment delivery support feedback Explanation for the model – 3  The *waterfall model,* sometimes called the *classic life cycle*, suggests a systematic, sequential approach[6] to software development that begins with customer specification of requirements and progresses through planning, modeling, construction, and deployment, culminating in ongoing support of the completed software (Figure 2.3). A variation in the representation of the waterfall model is called the *V-model*. Represented in Figure 2.4, the V-model [Buc99] depicts the relationship of quality assurance actions to the actions associated with communication, modeling, and early construction activities. As a software team moves down the left side of the V, basic problem requirements are refined into progressively more detailed and technical representations of the problem and its solution. Once code has been generated, the team moves up the right side of the V, essentially performing a series of tests (quality assurance actions) that validate each of the models created as the team moved down the left side.[7] In reality, there is no fundamental difference between the classic life cycle and the V-model. The V-model provides a way of visualizing how verification and validation actions are applied to earlier engineering work. The waterfall model is the oldest paradigm for software engineering. However, over the past three decades, criticism of this process model has caused even ardent supporters to question its efficacy [Han95]. Among the problems that are sometimes encountered when the waterfall model is applied are: 1. Real projects rarely follow the sequential flow that the model proposes. Although the linear model can accommodate iteration, it does so indirectly. As a result, changes can cause confusion as the project team proceeds.	5	CLO1	L2

	 2. It is often difficult for the customer to state all requirements explicitly. The waterfall model requires this and has difficulty accommodating the natural uncertainty that exists at the beginning of many projects. 3. The customer must have patience. A working version of the program(s) will not be available until late in the project time span. A major blunder, if undetected until the working program is reviewed, can be disastrous. In an interesting analysis of actual projects, Bradac [Bra94] found that the linear nature of the classic life cycle leads to "blocking states" in which some project team members must wait for other members of the team to complete dependent tasks. In fact, the time spent waiting can exceed the time spent on productive work! The blocking states tend to be more prevalent at the beginning and end of a linear sequential process. Today, software work is fast-paced and subject to a never-ending stream of changes (to features, functions, and information content). The waterfall model is often inappropriate for such work. However, it can serve as a useful process model in situations where requirements are fixed and work is to proceed to completion in a linear manner. (b) Describe the core advantages and unavoidable disadvantages. Advantages – 3, Disadvantages – 2 Advantages: 1. A systematic model can bring out a complete and stable software. 2. Can definitely produce a high-quality software. 3. Improve the life span of the software. Disadvantages:  1. Real projects rarely follow the sequential flow that the model proposes. Although the linear model can accommodate iteration, it does so indirectly. As a result, changes can cause confusion as the project team proceeds. 2. It is often difficult for the customer to state all requirements explicitly. The waterfall model requires this and has difficulty accommodating the natural uncertainty that exists at the beginning of many projects. 3. The customer must have patience. A working version of the program(s) will not be available until late in the project time span. A major blunder, if undetected until the working program is reviewed, can be disastrous.	5		
4	(a) Briefly explain the requirement elicitation and analysis process. Requirements elicitation process and explanation – 3 Analysis process with use case model – 2  • 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.	5	CLO1	L2

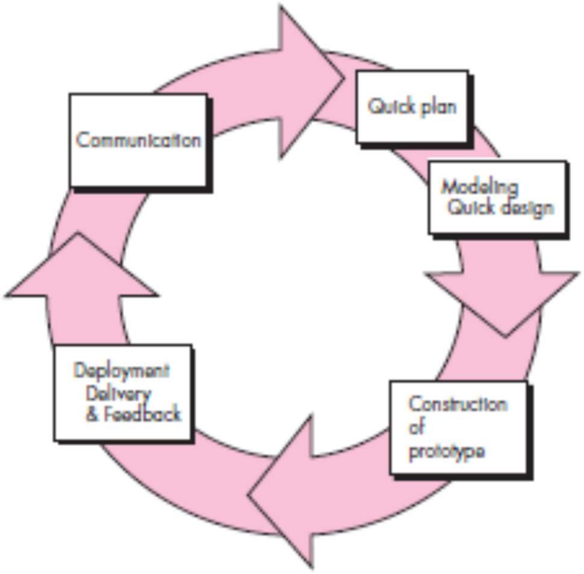
(b) Explain the important differences between functional and non-functional requirements. 5

Atleast 5 differences and suitable explanations – 5

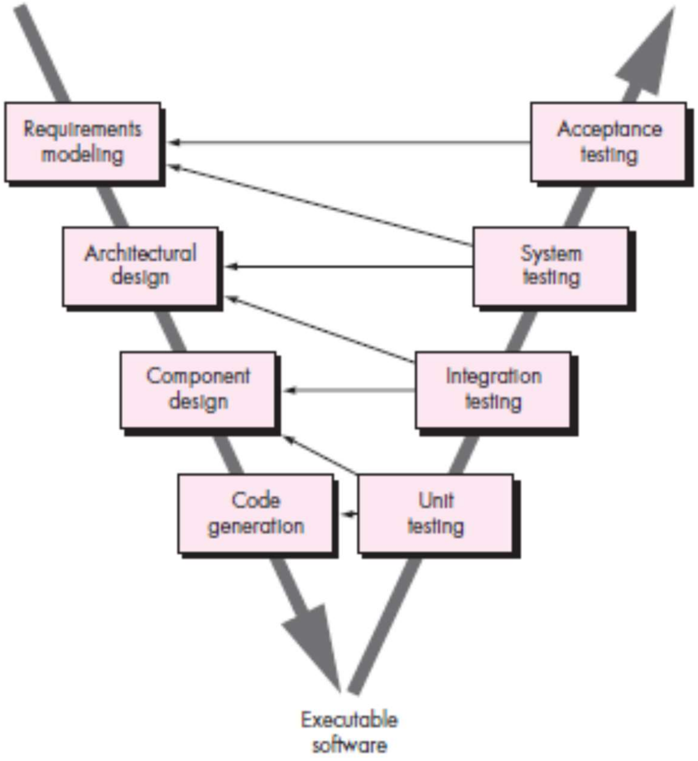
Functional Requirements	**Non Functional Requirements**
A functional requirement defines a system or its component.	A non-functional requirement defines the quality attribute of a software system.
It specifies “What should the software system do?”	It places constraints on “How should the software system fulfill the functional requirements?”
Functional requirement is specified by User.	Non-functional requirement is specified by technical peoples e.g. Architect, Technical leaders and software developers.
It is mandatory.	It is not mandatory.
It is captured in use case.	It is captured as a quality attribute.
Defined at a component level.	Applied to a system as a whole.
Helps you verify the functionality of the software.	Helps you to verify the performance of the software.
Functional Testing like System, Integration, End to End, API testing, etc are done.	Non-Functional Testing like Performance, Stress, Usability, Security testing, etc are done.
Usually easy to define.	Usually more difficult to define.

5	(a) Explain prototype and V models for software development with suitable diagrams. Prototype Model – diagram – 1, explanation – 1.5 V Model – diagram – 1, explanation – 1.5 Prototype Model:	5 5	CLO1	L2

<table>
<tr>
<td></td>
<td>



V Model:



(b) Differentiate the above two models based on their core properties.

Differences – 5 of them and relate to core properties

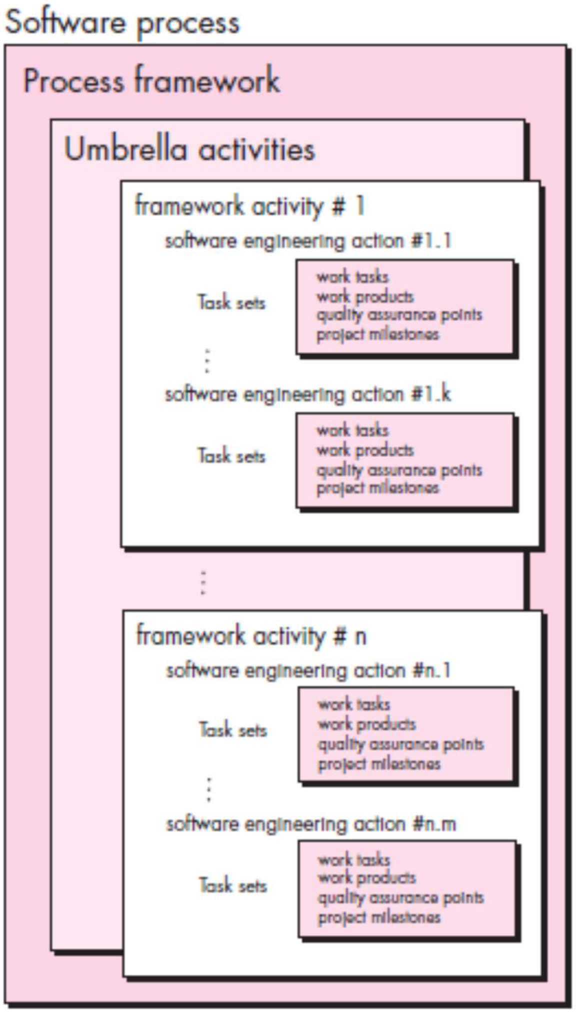
Differences:
1. Waterfall model is slow in delivering the software compared to the prototyping model.
2. …

</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>

(a) Explain software process framework with a neat diagram.

Software Process Framework – diagram 2 marks
Explanation for the framework – 2 marks

</td>
<td>4</td>
<td>CLO2</td>
<td>L3</td>
</tr>
</table>

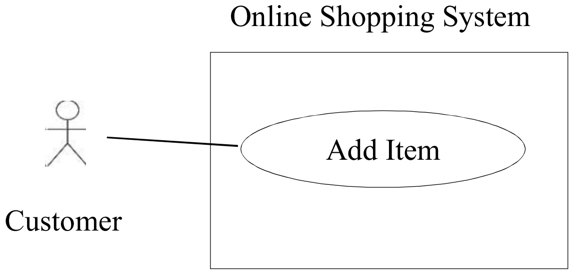


6

(b) Draw the use case diagram for the following case study:

A Modern Bazar Supermarket sells books and CDs using Online shopping. The customer adds items to the shopping cart. The customer may remove items or go to the checkout to make purchases at any time. The customer receives the purchased items by choosing a payment method. A sales employee at modern bazaar supermarket gets the order and purchase confirmation from the system and sends the electronic order to the warehouse. The warehouse employee updates the order status. The customer may check the order status.

Use Case diagram – atleast 6 use cases with appropriate actors. System has to be represented in a rectangular box.


Online Shopping System
Add Item
Customer