1 67437

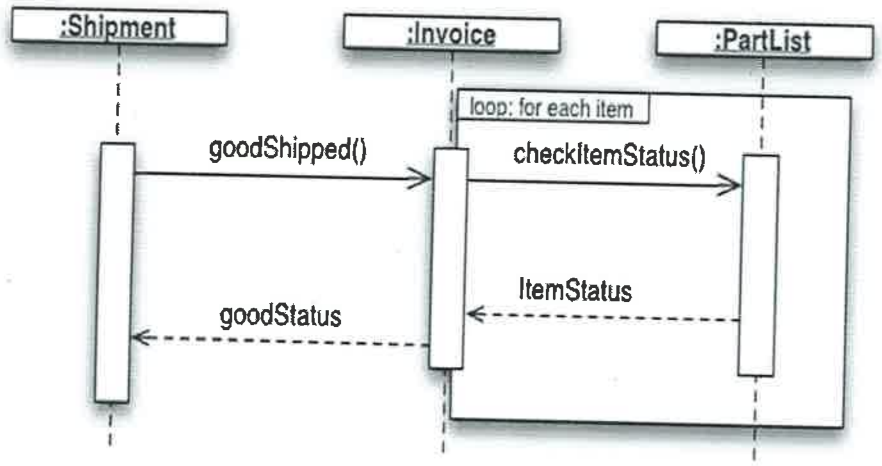
Q. 1	Attempt All (Each of 5Marks)
(a)	**1. Software Requirement Specification (SRS) is also known as specification of _______.** **a.** White box testing **b.** Acceptance testing **c.** Integrated testing **d.** Black box testing 2. **Which is the most desirable form of coupling?** **a. Control coupling** **b. Data coupling** **c. Common coupling** **d. Stamp coupling** **3. Kind of diagrams which** are used to show interactions between series of messages are classified as a. activity diagrams b. state chart diagrams c. collaboration diagrams d. object lifeline diagrams 4. Six Sigma methodology defines three core steps a. analyse, improve, control b. analyse , design , verify c. define , measure, analyse d. define , measure, control 5 Diagrams which are used to distribute files, libraries and tables across topology of hardware are called a. deployment diagrams b. use case diagrams c. sequence diagrams d. collaboration diagrams
(b)	1. HLD stands for High level design 2. SDP short for software development plan 3. KLOC stands for Thousands of line of code 4. RMMM stands for Risk Mitigation, Monitoring & Management 5. CMP stands for Configuration management plan
(c)	**1. Define time line charts in software engineering ?** A **timeline chart** is an effective way to visualize a process using **chronological**order. ... Often used for managing a project's schedule, **timeline charts** function as a sort of calendar of events within a specific **period** of **time**. **2.Define Software quality Assurance?** Quality Assurance monitors to check if proper process is followed while developing the software. **3.Define Validation?** Validation is the process of checking that a **software**system meets specifications and that it fulfills its intended purpose. **4. Define software engineering?** Software engineering is an engineering branch associated with software system development. 5. Define module cohesion? Cohesion is a measure that defines the degree of intra-dependability among the elements of the module
Q. 2	**Attempt the following (Any THREE)**
(a)	Define SRS? Write characteristics of SRS. SRS is a document created by system analyst after the requirements are collected from various

	stakeholders. Gathering software requirements is the foundation of the entire software development project. Hence they must be clear, correct and well-defined. A complete Software Requirement Specifications must be: • Clear • Correct • Consistent • Coherent • Comprehensible • Modifiable • Verifiable • Prioritized • Unambiguous • Traceable • Credible source Any Five points – 1 Mark Each
(b)	State advantages and disadvantages of waterfall model. Waterfall Model - Advantages The advantages of waterfall development are that it allows for departmentalization and control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one. Development moves from concept, through design, implementation, testing, installation, troubleshooting, and ends up at operation and maintenance. Each phase of development proceeds in strict order. Some of the major advantages of the Waterfall Model are as follows – • Simple and easy to understand and use • Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process. • Phases are processed and completed one at a time. • Works well for smaller projects where requirements are very well understood. • Clearly defined stages. • Well understood milestones. • Easy to arrange tasks. • Process and results are well documented. Waterfall Model - Disadvantages The disadvantage of waterfall development is that it does not allow much reflection or revision. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-documented or thought upon in the concept stage. The major disadvantages of the Waterfall Model are as follows – • 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. So, risk and uncertainty is high with this process model. • It is difficult to measure progress within stages. • Cannot accommodate changing requirements. • Adjusting scope during the life cycle can end a project. • Integration is done as a "big-bang. at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early. Advantages + disadvantages = 2.5 Marks Each

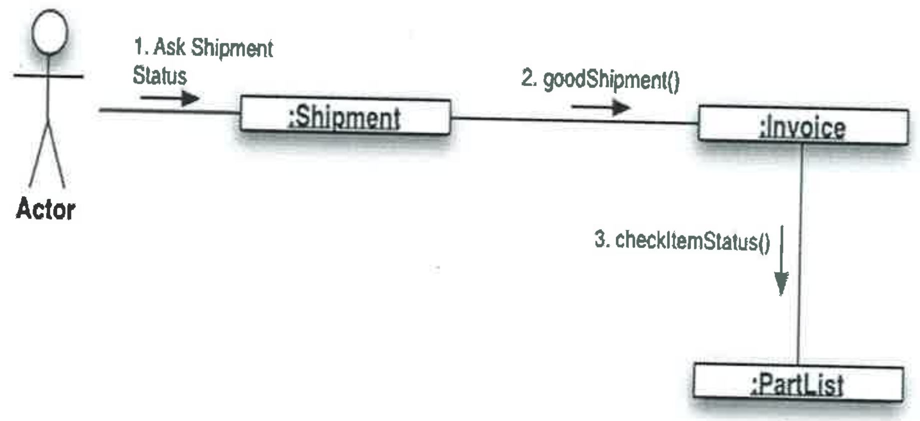
(c) Differentiate between sequence diagram and collaboration diagram.

Sequence diagrams highlight more the temporal aspect, by showing invocation and responses along a (vertical) timeline and by explicitly showing the activation time of objects. Sequence diagrams show how objects communicate with each other in terms of a temporal sequence of messages. The time flow is the most visible aspect in these diagrams, as messages are sequenced according to a vertical timeline and also the lifespan of objects associated to those messages is reported.

The figure below shows an example of a sequence diagram describing 3 objects (instances of classes Shipment, Invoice and PartList) and the messages exchanged between each other. Interaction diagrams describe execution scenarios of the system.



Collaboration diagrams aim at showing the communications that happen between objects, by defining messages that flow between each other. They basically consist of superimposing the communication actions upon an object diagram. The temporal aspect can be shown here too, by numbering the interactions with sequential labels. A collaboration diagram shows the interactions between objects or classes in terms of links (solid undirected lines connecting the elements that can interact) and messages that flow through the links. This describes at the same time some kind of static structure (links and nodes) and dynamic behavior (messages) of the system. An example is shown below.



2.5 Marks Each

(d) What are the attributes of good software?

Dependability: It should be secure, safe and reliable so: we can depend on it. In any case of system failure, it should not cause any case of physical or economic damage to the system.

Maintainability: It should be easily maintainable in any case of platform change or changing client's needs.

Efficiency: It should give us perfect possible efficiency without over using required memory and processor cycles.
Usability: It should be user friendly and easy to interact with customers through an appropriate user interface and adequate documentation.

Any Five Points = 1 Mark Each

(e) Explain Agility and write its advantages and disadvantages.

Agile model believes that every project needs to be handled differently and the existing methods need to be tailored to best suit the project requirements. In Agile, the tasks are divided to time boxes (small time frames) to deliver specific features for a release.

The advantages of the Agile Model are as follows –

- Is a very realistic approach to software development.
- Promotes teamwork and cross training.
- Functionality can be developed rapidly and demonstrated.
- Resource requirements are minimum.
- Suitable for fixed or changing requirements
- Delivers early partial working solutions.
- Good model for environments that change steadily.
- Minimal rules, documentation easily employed.
- Enables concurrent development and delivery within an overall planned context.
- Little or no planning required.
- Easy to manage.
- Gives flexibility to developers.

The disadvantages of the Agile Model are as follows –

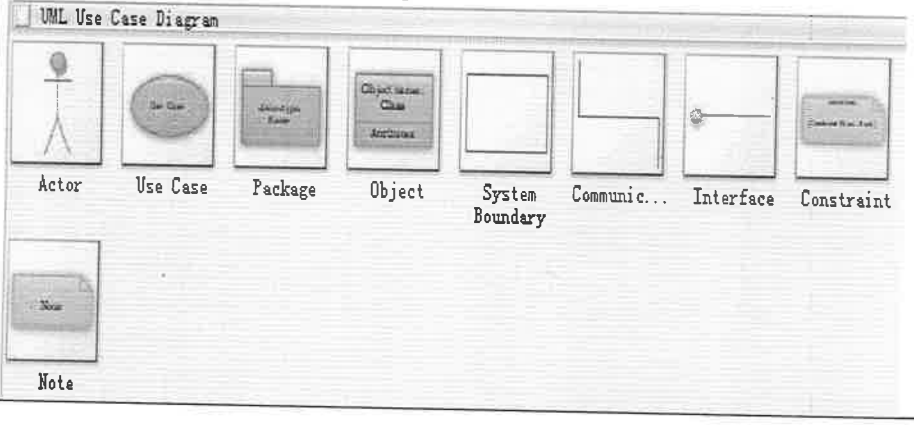
- Not suitable for handling complex dependencies.
- More risk of sustainability, maintainability and extensibility.
- An overall plan, an agile leader and agile PM practice is a must without which it will not work.
- Strict delivery management dictates the scope, functionality to be delivered, and adjustments to meet the deadlines.
- Depends heavily on customer interaction, so if customer is not clear, team can be driven in the wrong direction.
- There is a very high individual dependency, since there is minimum documentation generated.
- Transfer of technology to new team members may be quite challenging due to lack of documentation.

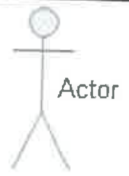
Explaination + Advantages+ disadvantages : 1M+2M+2M

(f) Define Use case diagram? Draw and explain symbols for the same

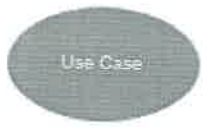
A **use case diagram** is a graphic depiction of the interactions among the elements of a system.

UML Use Case Diagram Symbols

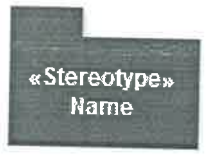




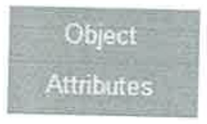
Actor specifies a role played by a user or any other system that interacts with the subject.



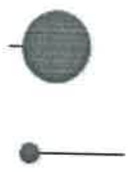
Use case is a list of steps, typically defining interactions between an actor and a system, to achieve a goal.



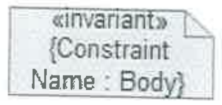
Package is used to group elements, and to provide a namespace for the grouped elements.



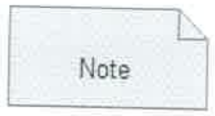
Objects are model elements that represent instances of a class or of classes.



Interfaces are model elements that define sets of operations that other model elements, such as classes, or components must implement.



Constraint is an extension mechanism that enables you to refine the semantics of a UML model element.

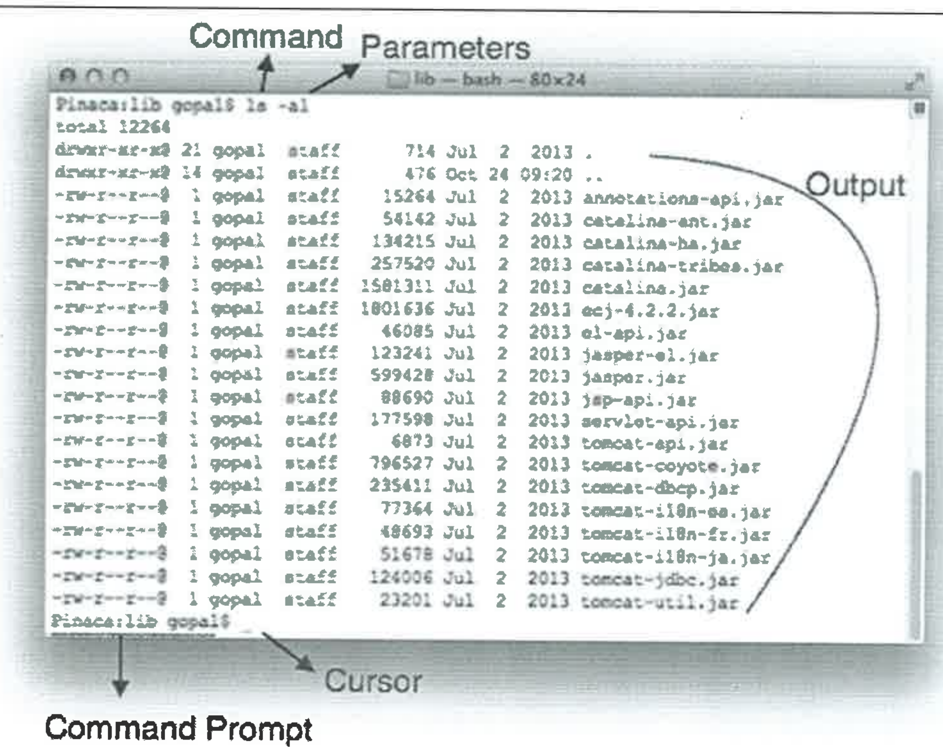


Note contains comments or textual information.

Def + Explain = 1M+ 4M

Q. 3	**Attempt the following (Any THREE)**
(a)	Define coupling what are the various levels of coupling. Coupling is a measure that defines the level of inter-dependability among modules of a program. It tells at what level the modules interfere and interact with each other. The lower the coupling, the better the

	program. There are five levels of coupling, namely - • **Content coupling** - When a module can directly access or modify or refer to the content of another module, it is called content level coupling. • **Common coupling-** When multiple modules have read and write access to some global data, it is called common or global coupling. • **Control coupling-** Two modules are called control-coupled if one of them decides the function of the other module or changes its flow of execution. • **Stamp coupling-** When multiple modules share common data structure and work on different part of it, it is called stamp coupling. • **Data coupling-** Data coupling is when two modules interact with each other by means of passing data (as parameter). If a module passes data structure as parameter, then the receiving module should use all its components. **Def+ levels = 1M + 4M**
(b)	**Expected results:** D is not > 0, read a, b, c again
(c)	Explain Software user interface design. UI can be graphical, text-based, audio-video based, depending upon the underlying hardware and software combination. UI can be hardware or software or a combination of both. The software becomes more popular if its user interface is: • Attractive • Simple to use • Responsive in short time • Clear to understand • Consistent on all interfacing screens UI is broadly divided into two categories: • Command Line Interface • Graphical User Interface **Command Line Interface (CLI)** CLI has been a great tool of interaction with computers until the video display monitors came into existence. CLI is first choice of many technical users and programmers. CLI is minimum interface a software can provide to its users. CLI provides a command prompt, the place where the user types the command and feeds to the system. The user needs to remember the syntax of command and its use. Earlier CLI were not programmed to handle the user errors effectively. A command is a text-based reference to set of instructions, which are expected to be executed by the system. There are methods like macros, scripts that make it easy for the user to operate. CLI uses less amount of computer resource as compared to GUI. **CLI Elements**



```
Pinaca:lib gopal$ ls -al
total 12264
drwxr-xr-x@ 21 gopal  staff      714 Jul  2  2013 .
drwxr-xr-x@ 14 gopal  staff      476 Oct 24 09:20 ..
-rw-r--r--@  1 gopal  staff    15264 Jul  2  2013 annotations-api.jar
-rw-r--r--@  1 gopal  staff    54142 Jul  2  2013 catalina-ant.jar
-rw-r--r--@  1 gopal  staff   134215 Jul  2  2013 catalina-ha.jar
-rw-r--r--@  1 gopal  staff   257520 Jul  2  2013 catalina-tribes.jar
-rw-r--r--@  1 gopal  staff  1581311 Jul  2  2013 catalina.jar
-rw-r--r--@  1 gopal  staff  1801636 Jul  2  2013 ecj-4.2.2.jar
-rw-r--r--@  1 gopal  staff    46085 Jul  2  2013 el-api.jar
-rw-r--r--@  1 gopal  staff   123241 Jul  2  2013 jasper-el.jar
-rw-r--r--@  1 gopal  staff   599428 Jul  2  2013 jasper.jar
-rw-r--r--@  1 gopal  staff    88690 Jul  2  2013 jsp-api.jar
-rw-r--r--@  1 gopal  staff   177598 Jul  2  2013 servlet-api.jar
-rw-r--r--@  1 gopal  staff     6873 Jul  2  2013 tomcat-api.jar
-rw-r--r--@  1 gopal  staff   796527 Jul  2  2013 tomcat-coyote.jar
-rw-r--r--@  1 gopal  staff   235411 Jul  2  2013 tomcat-dbcp.jar
-rw-r--r--@  1 gopal  staff    77364 Jul  2  2013 tomcat-i18n-es.jar
-rw-r--r--@  1 gopal  staff    48693 Jul  2  2013 tomcat-i18n-fr.jar
-rw-r--r--@  1 gopal  staff    51678 Jul  2  2013 tomcat-i18n-ja.jar
-rw-r--r--@  1 gopal  staff   124006 Jul  2  2013 tomcat-jdbc.jar
-rw-r--r--@  1 gopal  staff    23201 Jul  2  2013 tomcat-util.jar
Pinaca:lib gopal$ _
```

A text-based command line interface can have the following elements:

- **Command Prompt** - It is text-based notifier that is mostly shows the context in which the user is working. It is generated by the software system.
- **Cursor** - It is a small horizontal line or a vertical bar of the height of line, to represent position of character while typing. Cursor is mostly found in blinking state. It moves as the user writes or deletes something.
- **Command** - A command is an executable instruction. It may have one or more parameters. Output on command execution is shown inline on the screen. When output is produced, command prompt is displayed on the next line.

Graphical User Interface

Graphical User Interface provides the user graphical means to interact with the system. GUI can be combination of both hardware and software. Using GUI, user interprets the software.

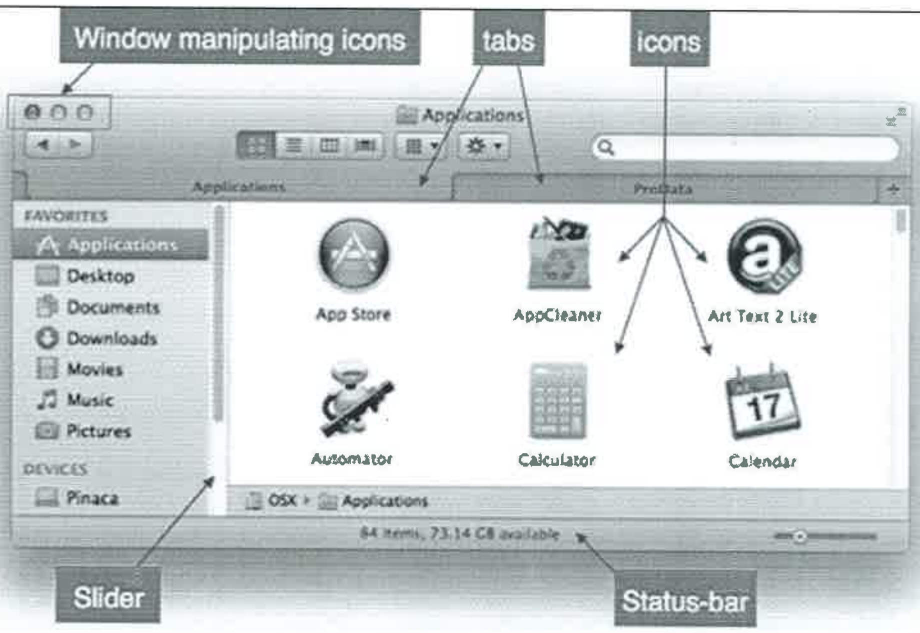
Typically, GUI is more resource consuming than that of CLI. With advancing technology, the programmers and designers create complex GUI designs that work with more efficiency, accuracy and speed.

GUI Elements

GUI provides a set of components to interact with software or hardware.

Every graphical component provides a way to work with the system. A GUI system has following elements such as:

⊘



- **Window** - An area where contents of application are displayed. Contents in a window can be displayed in the form of icons or lists, if the window represents file structure. It is easier for a user to navigate in the file system in an exploring window. Windows can be minimized, resized or maximized to the size of screen. They can be moved anywhere on the screen. A window may contain another window of the same application, called child window.
- **Tabs** - If an application allows executing multiple instances of itself, they appear on the screen as separate windows. **Tabbed Document Interface** has come up to open multiple documents in the same window. This interface also helps in viewing preference panel in application. All modern web-browsers use this feature.
- **Menu** - Menu is an array of standard commands, grouped together and placed at a visible place (usually top) inside the application window. The menu can be programmed to appear or hide on mouse clicks.
- **Icon** - An icon is small picture representing an associated application. When these icons are clicked or double clicked, the application window is opened. Icon displays application and programs installed on a system in the form of small pictures.
- **Cursor** - Interacting devices such as mouse, touch pad, digital pen are represented in GUI as cursors. On screen cursor follows the instructions from hardware in almost real-time. Cursors are also named pointers in GUI systems. They are used to select menus, windows and other application features.

CLI + GUI = 2.5 Marks Each

(d) Define Object-Oriented Programming and features of OOPs.

Object-oriented programming (OOP) is a programming paradigm based upon objects (having both data and methods) that aims to incorporate the advantages of modularity and reusability. Objects, which are usually instances of classes, are used to interact with one another to design applications and computer programs.

The important features of object–oriented programming are –

- Bottom–up approach in program design
- Programs organized around objects, grouped in classes
- Focus on data with methods to operate upon object's data
- Interaction between objects through functions
- Reusability of design through creation of new classes by adding features to existing classes

Some examples of object-oriented programming languages are C++, Java, Smalltalk, Delphi, C#, Perl, Python, Ruby, and PHP.

Grady Booch has defined object–oriented programming as *"a method of implementation in which programs are organized as cooperative collections of objects, each of which represents an instance of some class, and whose classes are all members of a hierarchy of classes united via inheritance relationships".*
Defn+ features = 1M+4M

(e) Write the scope of software metrics.

Scope of Software Metrics

Software metrics contains many activities which include the following –

- Cost and effort estimation
- Productivity measures and model
- Data collection
- Quantity models and measures
- Reliability models
- Performance and evaluation models
- Structural and complexity metrics
- Capability – maturity assessment
- Management by metrics
- Evaluation of methods and tools

Software measurement is a diverse collection of these activities that range from models predicting software project costs at a specific stage to measures of program structure.
Explain any Five points = 5 Marks

(f) Explain Halstead's metrics with an example.
Halstead introduced metrics to measure software complexity. Halstead's metrics depends upon the actual implementation of program and its measures, which are computed directly from the operators and operands from source code, in static manner. It allows to evaluate testing time, vocabulary, size, difficulty, errors, and efforts for C/C++/Java source code.
According to Halstead, "A computer program is an implementation of an algorithm considered to be a collection of tokens which can be classified as either operators or operands". Halstead metrics think a program as sequence of operators and their associated operands.
He defines various indicators to check complexity of module.

Parameter	**Meaning**
n1	Number of unique operators
n2	Number of unique operands
N1	Number of total occurrence of operators
N2	Number of total occurrence of operands

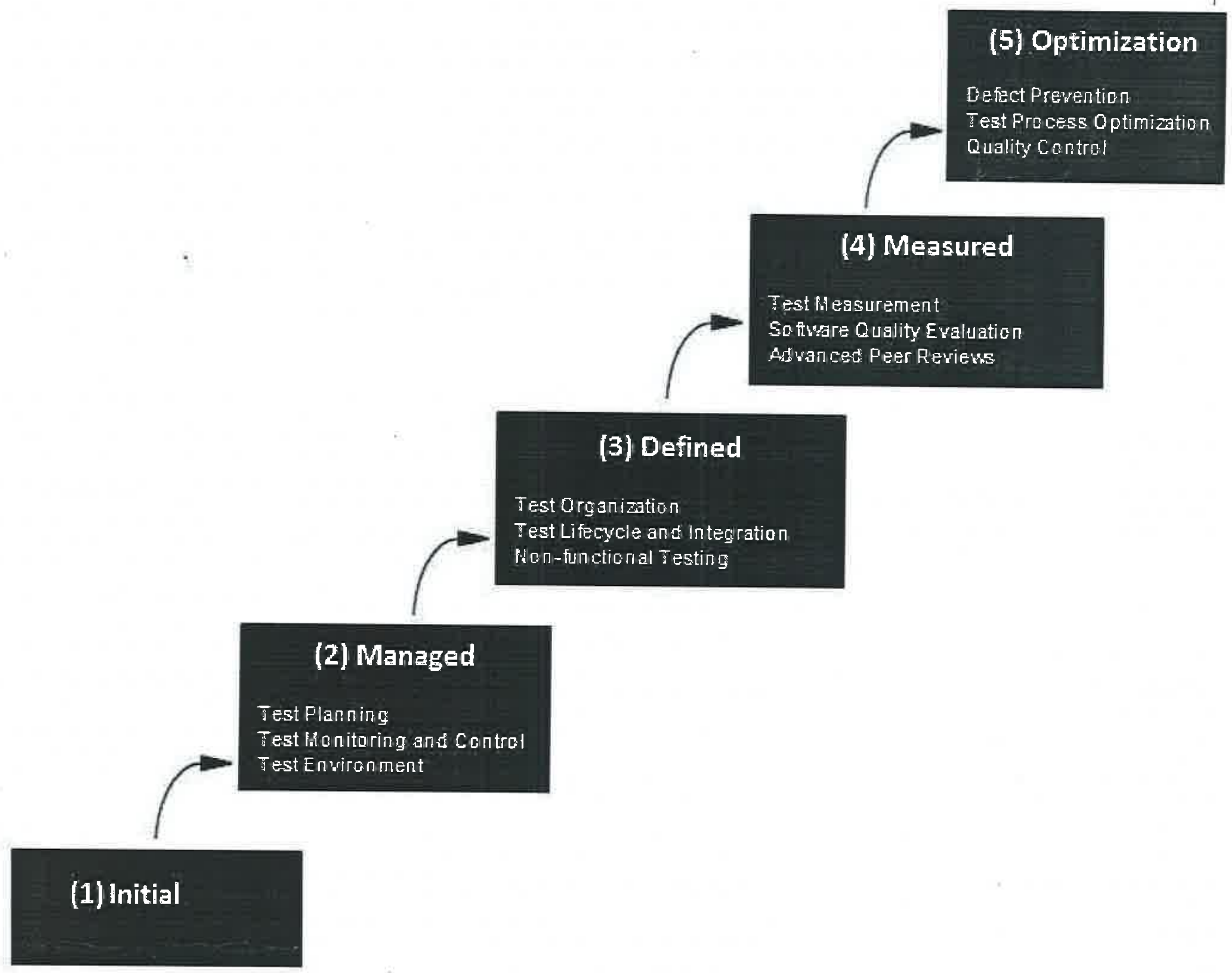
When we select source file to view its complexity details in Metric Viewer, the following result is seen in Metric Report:

Metric	**Meaning**	**Mathematical Representation**
n	Vocabulary	n1 + n2
N	Size	N1 + N2
V	Volume	Length * Log2 Vocabulary
D	Difficulty	(n1/2) * (N1/n2)
E	Efforts	Difficulty * Volume
B	Errors	Volume / 3000
T	Testing time	Time = Efforts / S, where S=18 seconds.

	Explaination + Example = 3M+ 2M
Q. 4	**Attempt the following (Any THREE)**
(a)	Explain Capability Maturity Model.

he Software Engineering Institute (SEI) Capability Maturity Model (CMM) specifies an increasing series of levels of a software development organization. The higher the level, the better the software development process, hence reaching each level is an expensive and time-consuming process.

Levels of CMM



- **Level One :Initial** - The software process is characterized as inconsistent, and occasionally even chaotic. Defined processes and standard practices that exist are abandoned during a crisis. Success of the organization majorly depends on an individual effort, talent, and heroics. The heroes eventually move on to other organizations taking their wealth of knowledge or lessons learnt with them.
- **Level Two: Repeatable** - This level of Software Development Organization has a basic and consistent project management processes to track cost, schedule, and functionality. The process is in place to repeat the earlier successes on projects with similar applications. Program management is a key characteristic of a level two organization.
- **Level Three: Defined** - The software process for both management and engineering activities are documented, standardized, and integrated into a standard software process for the entire organization and all projects across the organization use an

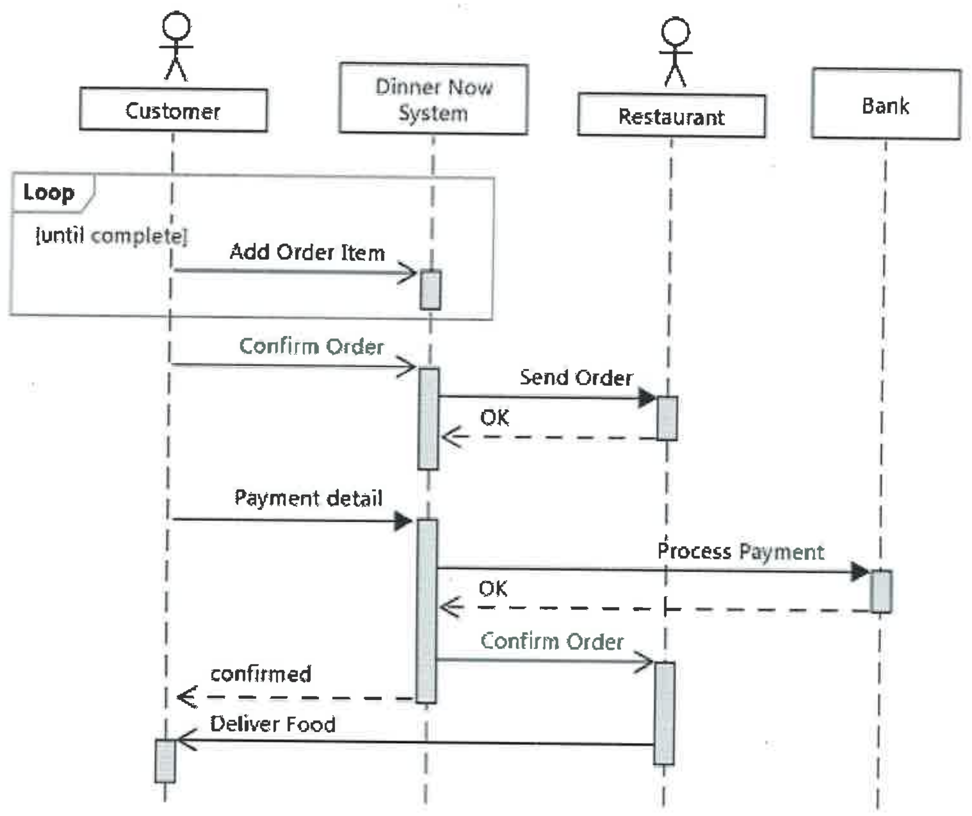
	approved, tailored version of the organization's standard software process for developing,testing and maintaining the application. • **Level Four: Managed** - Management can effectively control the software development effort using precise measurements. At this level, organization set a quantitative quality goal for both software process and software maintenance. At this maturity level, the performance of processes is controlled using statistical and other quantitative techniques, and is quantitatively predictable. • **Level Five: Optimizing** - The Key characteristic of this level is focusing on continually improving process performance through both incremental and innovative technological improvements. At this level, changes to the process are to improve the process performance and at the same time maintaining statistical probability to achieve the established quantitative process-improvement objectives. Per level = 1 Mark
(b)	What is Risk management ? Explain Software risk management process. Risk Mitigation, Monitoring and Management (RMMM) **There are three important issues considered in developing an effective strategy:** **Risk avoidance or mitigation** - It is the primary strategy which is fulfilled through a plan. **Risk monitoring** - The project manager monitors the factors and gives an indication whether the risk is becoming more or less. **Risk management and planning** - It assumes that the mitigation effort failed and the risk is a reality. **RMMM Plan:** It is a part of the software development plan or a separate document.The RMMM plan documents all work executed as a part of risk analysis and used by the project manager as a part of the overall project plan.The risk mitigation and monitoring starts after the project is started and the documentation of RMMM is completed. Definition + Plan = 2M+3M
(c)	**Explain the purpose of six sigma.** Six sigma's purpose is to identify the causes of defects and errors and then remove them in the manufacturing process. Any Five points = 1Mark each
(d)	**Explain any five software quality attributes.** I. Functionality II. Reliability III. Usability IV. Efficiency V. Maintainability VI. Portability
(e)	What is Structural testing? Write its advantages and disadvantages • The structural testing is the testing of the structure of the system or component. • Structural testing is often referred to as 'white box' or 'glass box' or 'clear-box testing' because in structural testing we are interested in what is happening 'inside the system/application'. • In structural testing the testers are required to have the knowledge of the internal implementations of the code. Here the testers require knowledge of how the software is implemented, how it works. Def + advantages+ disadvantages = 1+ 2+2 Marks
(f)	Explain McCall's Quality factors. McCall Quality Factors: Correctness: The extent to which a program satisfies its specs and fulfills the customer's mission objectives.

Reliability: The extent to which a program can be expected to perform its intended function with required precision.
Efficiency: The amount of computing resources and code required to perform is function.
Integrity: The extent to which access to S/W or data by unauthorized persons can be controlled.
Usability: The effort required to learn, operate, prepare input for, and interpret output of a program.
Maintainability: The effort required to locate and fix errors in a program.
Flexibility: The effort required to modify an operational program.
Testability: The effort required to test a program to ensure that it performs its intended function.
Portability: The effort required to transfer the program from one hardware and/or software system environment to another.
Reusability: The extent to which a program can be reused in other applications-related to the packaging and scope f the functions that the program performs.
Interoperability: The effort required to couple one system to another.

Any Five Quality Factors = 1 Mark Each

Q. 5 Attempt the following (Any THREE)

(a) Draw a Sequence diagram for online ordering of food delivery System.



(b) **State and Explain the Quality metrics**

The goal of software metrics is to determine the quality of the current product or process, improve that quality and predict the quality once the software development project is complete.
Software development managers are trying to :

- Increase return on investment (ROI)
- Identify areas of improvement
- Manage workloads
- Reduce overtime
- Reduce costs

13

	Any Five Point = 1 Mark Each
(c)	**State the difference between Black box testing and white-box testing?**

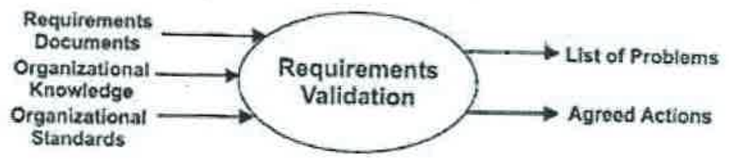
White-box testing not only checks for desired and valid output when valid input is provided but also it checks if the code is implemented correctly.
Black-box testing checks if the desired outputs are produced when valid input values are given. It does not verify the actual implementation of the program.

Criteria	**Black Box Testing**	**White Box Testing**
Knowledge of software program, design and structure essential	No	Yes
Knowledge of Software Implementation essential	No	Yes
Who conducts this test on software	Software Testing Employee	Software Developer
baseline reference for tester	Requirements specifications	Design and structure details

Defn WBT+BBT+ Criteria = 1M+1M+3M

(d) State all and write down a short note on any 3 fact finding techniques.
All fact finding techniques – 2 mark, any 3 explanation(observation, questionnaire, field visit etc) – 3 marks

(f) Explain requirement validation.
In the validation phase, the work products produced as a consequence of requirements engineering are examined for consistency, omissions, and ambiguity. The basic objective is to ensure that the SRS reflects the actual requirements accurately and clearly. Other objectives of the requirements document are listed below.
To certify that the SRS contains an acceptable description of the system to be implemented
To ensure that the actual requirements of the system are reflected in the SRS
To check the requirements document for completeness, accuracy, consistency, requirement conflict', conformance to standards and technical errors.
Requirements validation is similar to requirements analysis as both processes review the gathered requirements. Requirements validation studies the 'final draft' of the requirements document while requirements analysis studies the 'raw requirements' from the system stakeholders (users). Requirements validation and requirements analysis can be summarized as follows:
Requirements validation: Have we got the requirements right?
Requirements analysis: Have we got the right requirements?
Various inputs such as requirements document, organizational knowledge, and organizational standards are shown. The requirements document should be formulated and organized according to the standards of the organization.The **organizational knowledge** is used to estimate the realism of the requirements of the system. The **organizational standards are** specified standards followed by the organization according to which the system is to be developed.



Requirements Validation

	The output of requirements validation is a list of problems and agreed actions of the problems. The **lists of problems** indicate the problems encountered in the" requirements document of the requirements validation process. The **agreed actions** is a list that displays the actions to be performed to resolve the problems depicted in the problem list. Any 5 Points = 1 Mark each

****